Technical Report: Fall Retreat 2018

Christoph Meinel, Hasso Plattner, Jürgen Döllner,
Mathias Weske, Andreas Polze, Robert Hirschfeld,
Felix Naumann, Holger Giese, Patrick Baudisch,
Tobias Friedrich, Erwin Böttinger,
Christoph Lippert (Eds.)

Technische Berichte Nr. 129

des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam

Universität Potsdam

HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam | 129

Christoph Meinel | Hasso Plattner | Jürgen Döllner | Mathias Weske |
Andreas Polze | Robert Hirschfeld | Felix Naumann | Holger Giese |
Patrick Baudisch | Tobias Friedrich | Erwin Böttinger |
Christoph Lippert (Eds.)

Technical Report

Fall Retreat 2018

Universitätsverlag Potsdam

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de/ abrufbar.

Universitätsverlag Potsdam 2019
http://verlag.ub.uni-potsdam.de/

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 2533 / Fax: 2292
E-Mail: verlag@uni-potsdam.de

Die Schriftenreihe **Technische Berichte des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam** wird herausgegeben von den Professoren des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam.

ISSN (print) 1613-5652
ISSN (online) 2191-1665

Druck: docupoint GmbH Magdeburg

ISBN 978-3-86956-465-4

Zugleich online veröffentlicht auf dem Publikationsserver der Universität Potsdam:
https://doi.org/10.25932/publishup-42753
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-427535

Contents

Microtask Crowdsourcing as Means to Identify and Explain Software Failures

Christian Adriano

System Analysis and Modeling Research Group
Hasso-Plattner-Institut
christian.adriano@hpi.uni-potsdam.de

This report describes my research on the feasibility of a novel and fast method to identify and explain software failures, which I call "failure resolution". The method consists of partitioning complex work into small and independent tasks (microtasks) that can executed by a crowd of anonymous programmers recruited on a popular crowdsourcing platform (Mechanical Turk). My method automatically generates microtasks, distributes them, and aggregates their individual outcomes into a failure resolution result. This method was incrementally designed and evaluated through experiments with software programmers and real bugs from popular open source projects.

1 About this report

This is the second Fall report on my research. In the next two sections I review my previous results and summarize the results obtained through the year of 2018.

1.1 Previous report

In the previous report, I investigated how to enable a crowd of programmers to work within a feedback loop with mechanisms that generate, distribute, and aggregate microtasks aimed at resolving software failures (Figure 1).

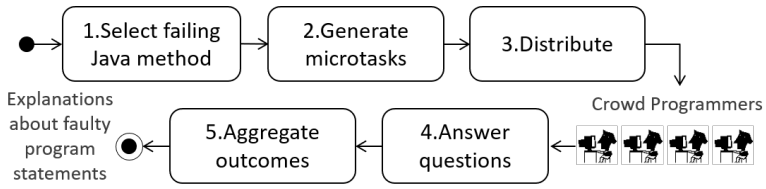


Figure 1: Failure resolution process

Feedback loop The goal of each loop cycle is to suggest the next microtasks to be performed by a group of programmers (subcrowd). For that, the feedback loop relies on five decision models, which I summarize below.

- *Feature selection model* predicts and ranks the attributes of answers and programmers which can be used to prioritize microtasks.
- *Sampling model* determines how to combine multiple results from the same task or from different tasks. The configurations available consist of under-sampling and over-sampling, which allow to trade-off between accuracy and variance in results.
- *Aggregation model* predicts the location of a root-cause by applying voting methods. We trained a set of machine learning models to predict the threshold for three different voting methods.
- *Subcrowd model* predicts which sub-groups of programmers (subcrowds) perform better at identifying root-causes for software failures. These subcrowds can be used to decide who should be prioritized to receive a certain microtask.
- *Task selection model* consists of determining which tasks should be executed next based on the result of previous tasks. The goal is to minimize the number of tasks needed. We report on this model in the current report.

1.2 Current report

My current report focuses on the last model (task selection) and two methodological concerns: the practical application of my method and the generalization of it to other domains distinct from software failure resolution. Moreover, this report also discusses the motivation for this type of research, the corresponding related work, and my research roadmap.

2 Motivation

Need for speed to resolve failures Quickly resolving software failures has been part of the practice of agile teams, which since the agile manifesto[1] became widespread by achieving the goldilocks of shorter software release cycles without sacrificing quality, cost, or overloading the development teams [2]. Nonetheless, software teams have come under renewed pressure by two external factors: the disruption from failures that happen only in later stages of testing and the need to deploy software multiple times a day. A recent industry report [3] shows that top performing businesses are deploying on average 32 times a day and are keeping lead times below one hour, i.e., the time between committing the source code and running it on production. Hence, while teams used to have days to resolve failures, now many teams have only a few hours. To alleviate this pressure on teams, many approaches have been investigated, e.g., improve continuous integration environments, mandatory

[1] http://agilemanifesto.org (last accessed 2018-10-18).

code reviews, and, finally, make failure resolution quicker to perform (my research focus).

Automated methods solutions In response to more agility, software engineering researchers developed various automated methods. Static analysis tools like PMC and FindBugs are executed during compilation time to quickly weed out errors that can become potential software failures. Continuous integration tools run unit tests to expose failures before they are moved into production. Spectra-based fault localization [11] methods run hundreds unit tests to identify the program statements that the most probable root-cause of a software failure.

Besides creating a safety net to catch software failures introduced during short development cycles, automated methods are also the pillar of many tools that automatically repair root-causes of failures [9]. Hence, agile teams, high frequency deployment, and automated program repair tools depend on the methods that quickly identify root-causes of software failures.

Perfect fault understanding assumptions However, research has shown that identifying a root-cause is not a sufficient condition to resolve a failure. Even when programmers are given the root-cause, there is no guarantee that programmers will unequivocally recognize the root-cause. Parnin and Orso [10] described this as the "perfect fault understanding assumption".

Difficulty to guarantee fault understanding is also an issue when fixes are automatically suggested. Research showed [12, 13] that programmers still need to be involved to evaluate if the suggested fixes are not over-fitting the unit tests ,i.e., making the failing test pass but not generalizing to other test input data. Research also confirmed the need for patches explanations [7], but this is hindered by the difficulty of using "understandability" as a metric to select of patches [8].

Nonetheless, in my research I depart from the assumption that fault understanding can be made explicit in written explanations. This assumption is corroborated by the studies on the performance of senior and junior programmers on debugging tasks. Senior programmers are shown to require fewer [4] hypotheses (i.e., explanations) to correctly identify a root-cause of a failure. Moreover, senior programmers' hypotheses have shown to present better quality [6] than the ones from junior peers.

Therefore, my research focuses on both the traditional problem of fault localization (identifying failure root-causes) and how to obtain the explanations for the corresponding software failures. I named this dual outcome as a failure resolution process, which I investigate with respect to its efficiency (speed) and efficacy (positive impact on bug fixing).

My investigation approach I proposed a method to speed up failure resolution by parallelizing its execution. My method obtains explanations to the failure root-causes by the execution of human-judgment tasks. The method was inspired on a MapReduce [1] model that has been successfully combined with crowdsourcing [5] to partition complex work into smaller and independent tasks (microtasks),

distribute them, and aggregate their individual outcomes into one failure resolution result. My method has many challenges that I discuss in detail in the next section.

3 Research problems

Problem.1 The choice for task design is constrained by trade-offs between the speed and quality of failure resolution. Trade-offs are present among the processes of partitioning, distributing, and executing tasks. Partitioning work in many tasks can increase failure resolution speed by increasing parallelization, which is achieved by making tasks as small as possible. However, as the number of tasks increase, their complete distribution might take longer because more capable programmers need to be recruited. When executing a task that is too small, quality might decrease because the programmer would have less information to make good judgments about the failure root-cause. Moreover, a programmer could be less motivated to do quality work, because each smaller tasks has a lower monetary pay. Fundamentally, the our problem is to discover what is a feasible task size regarding speed and quality of failure resolution. This might help understand whether it is better to have many smaller or larger tasks distributed to many or fewer programmers.

Problem.2 As the execution of tasks are completed, we face the problem of aggregating the outcomes of multiple tasks into a consolidated failure resolution. This can be challenging because programmers might have distinct beliefs about the root-cause of the software failure. Furthermore, even after agreeing on a root-cause, programmers might provide distinct explanations. This is particularly difficult when more than one root-cause and explanation are effective to suggest valid bug fixes. Ultimately, our problem is to learn how to identify valid root-causes and explanations by aggregating competing beliefs.

Problem.3 As we aggregate task outcomes, we have the problem of how to improve speed and quality from one failure resolution job to another. This is challenging because we cannot guarantee improvement by redesigning the tasks, i.e., making them larger or smaller depending on the current outcomes. The alternative is to learn to identify the programmers who produce faster and higher quality failure resolutions. This is particularly difficult when programmers have different levels of programming skill, experience, and motivation. More fundamentally, our problem is how to select groups of programmers (subcrowds) who consistently outperform the average programmer in terms of speed and quality of failure resolution.

Problem.4 After guaranteeing some level of efficacy in failure resolution, we need to solve the problem of speed. For that we need to minimize the number of tasks necessary to correctly identify and explain a software failure. This is difficult because at any given moment we have the option to obtain more results for the same task (exploit) or execute a different task (explore), or decide to stop executing certain tasks (expire). These decisions to exploit, explore,and expire are difficult because

one cannot know for certain which of the previous task outcomes are correct (e.g., true positive and true negatives).

4 Method

I investigated these problems with a method that automatically partitions work in small independent tasks (microtasks) and distributes them to programmers (crowd) recruited on a crowdsourcing platform (Mechanical Turk). Microtasks are automatically generated from template questions that cover the program statements of the source code that failed its unit test (Figure 2).

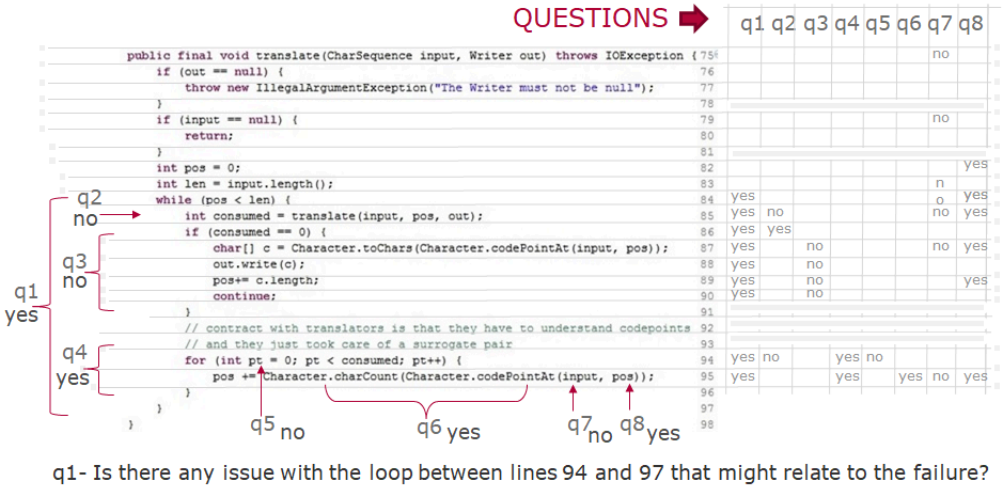


Figure 2: Questions and answers for each source code fragment

For each microtask, the programmers provide their beliefs on a root-cause of given a software failure and justify their beliefs with a written explanation. Their beliefs are expressed in an answer to a question about the possible relationship between a program statement and the software failure. The answers are either YES, NO, or I DON'T KNOW. Programmers are also required to provide their confidence for their answers. To answer these questions, programmers receive the following information: the unit test assertion, the failure message, the source code of the method that failed the test. This information is provided to programmers on web-based interface (Figure 3) that also collects programmers' answers.

The outcomes of microtasks are automatically aggregated by different voting methods. My method also selects the best performing subcrowds. This is done by prediction models that were trained with attributes of programmers and their corresponding answers.

Although I automated the processes of partitioning, distribution, aggregation, and subcrowd selection, I did not apply any automated debugging method. I purpose-

Figure 3: Web-based GUI for the microtask

fully did so as a first study because I wanted to evaluate a method that is independent of automated debugging tools. This way I sought to avoid possible tool biases and produce a baseline for future automation.

5 Contributions

I evaluated my method through three large scale experiments. The results of these experiments are summarized in the following contributions:

1. I showed the efficacy of my method for two different designs of microtasks and two sets of real software failures from 16 different popular open source software projects. In the first design, I performed an experiment with 777 programmers, who executed 5705 microtasks. The best group of programmers (subcrowd) located six out of the ten root-causes of software failures, but with high levels of false positives (57% precision and 70% recall). For the second microtask design, I performed an experiment with 654 programmers who performed 2580 microtask. The most effective subcrowd comprised 133 programmers, who answered 836 questions in 33 minutes to resolve all eight software failures with 94% precision and 65% recall at the program statement level. Therefore, my method achieved a competitive efficiency in the second experiment. Efficiency was measured in terms of the number of microtasks, programmers, cost per failure, and the time needed to locate the root-causes of all software failures.

2. I showed that models can be built to predict the quality of microtask outcomes. Models were built with features (attributes) from programmers and their answers to the microtasks. These predictive models were evaluated across different types of software failures and source code with different sizes and complexities.

3. The aggregation mechanisms based on cardinal voting method performs better than majority voting, proportional voting, and absolute threshold voting. We validated this for both microtask designs and across different software failures.

4. I showed that the root-cause explanations were meaningful and effective to help programmers to suggest bug fixes. Meaningfulness stems from explanations being categorized into distinct classes. We show efficacy by measuring accuracy of bug fixes when programmers where provided with explanations and when they were not.

6 Latest results

Since the last Fall report I have worked on three concrete topics. Below I discuss my preliminary results.

6.1 Task selection model

To increase the speed of failure resolution I investigated how to minimize the number of questions asked. My approach was to prioritize questions by the perceived utility to the programmer. I experimented with two formulations: user perceived difficulty and confidence. To evaluate these utility functions I designed an iterative algorithm that samples, aggregates, and ranks questions (Figure 4).

1: Start with one answer per question
2: Loop (**iterations**=10)
3: Aggregate answers
4: Compute question ranking (**classification frontier**)
5: Select top **R questions** as predictors of the fault location
6: Compute statistics (Precision, Recall, LOC)
7: Rank question by **utility value**
8: Select top **U questions to sample**
9: Sample **A** answers from the **U questions**
10: End loop

Figure 4: Algorithm to evaluate the utility-based selection of microtasks. Hyper-parameters of the algorithm in bold font

I explored the hyper-parameter space for a utility function based on the answer confidence. The best results were obtained for the following parameterization: sample one answer per question (A1), rank only the top question to be answered (U1), and select the top three questions (R3) as covering the failure root-causes. This enables to locate and explain all software failures with more than 90% precision, 100% recall (at program statement level) and requiring less than 20% of all questions asked (Figure 5).

As future work I plan to compute the expected utility, which would require to estimate a probability for each utility value. My approach for that will be to model the stream of answers for each question as a Markov chain. The stationary distribution of the Markov chain will provide me with the probabilities of the question receiving either a YES, NO, or I DON'T KNOW answer. Since I am interested in the questions with high chances of covering a root-cause, I can use the probability of a YES to compute the expected utility of each question at any given time. Note that as more answers are received, I will update these probability estimates.

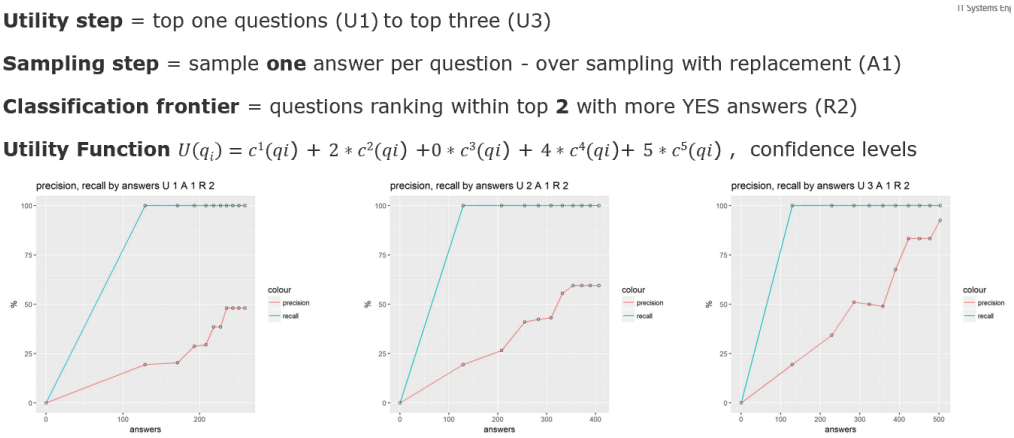


Figure 5: Precision and recall for various hyper-parameter values

6.2 Practical application

I performed a third experiment to evaluate the impact of the explanations on the perfect fault understanding assumption. The experiment involved 21 software programmers and three real software failures. These programmers were asked to suggest bug fixes based on the failure resolution produced in the second experiment by the crowd of programmers. I evaluated the correctness of the bug fixes in the presence and the absence of the explanations for the root-causes identified in the second experiment. I found that programmers who relied on explanations were more accurate in their bug fixes. The effect of explanations on bug fix accuracy was statistically significant (two-tailed t-test, p-value=0.0014) and had a large effect (Cohen's D = 1.13).

6.3 Generalization

The generalization aimed to isolate the characteristics that are independent of the failure resolution domain and to capture them in a software architecture. I proposed a software architecture that allows to instantiate my method as an online hybrid crowd-machine system. The goal of the system is to partition, distribute, aggregate, and prioritize work that can be performed as microtasks. My software architecture will be designed to satisfy two non-functional requirements: scalability (vertical and horizontal) and adaptability (at deployment and at run-time). My future work includes experiments to evaluate these two requirements.

7 Research roadmap

The roadmap is organized around paper writing and experiments.

- Perform a bug fixing experiment with the remaining 15 real software bugs from experiments one and two,
- Submit paper about the three experiments to ESE Journal (December 2018),
- Submit paper on task prioritization to SASO conference (February 2019),
- Perform experiments to evaluate the scalability and adaptability of the online hybrid crowd-machine system,
- Submit paper to ESEM conference (May 2019).

References

[1] J. Dean and S. Ghemawat. "MapReduce: simplified data processing on large clusters". In: *Communications of the ACM* 51.1 (2008), pages 107–113.

[2] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe. "A decade of agile methodologies: Towards explaining agile software development". In: *Journal of Systems and Software* 85.6 (2012), pages 1213–1221.

[3] N. Forsgren, G. Kim, J. Humble, A. Brown, and N. Kersten. *2017 State of DevOps Report*. Technical report. Puppet, 2017.

[4] L. Gugerty and G. M. Olson. "Comprehension differences in debugging by skilled and novice programmers". In: *Papers presented at the first workshop on empirical studies of programmers on Empirical studies of programmers*. 1986, pages 13–27.

[5] A. Kittur, B. Smus, S. Khamkar, and R. E. Kraut. "Crowdforge: Crowdsourcing complex work". In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 2011, pages 43–52.

[6] A. Ko and B. Myers. "Debugging reinvented". In: *ACM/IEEE 30th International Conference on Software Engineering (ICSE'08)*. 2008, pages 301–310.

[7] C. Le Goues, S. Forrest, and W. Weimer. "Current challenges in automatic software repair". In: *Software quality journal* 21.3 (2013), pages 421–443.

[8] M. Monperrus. "A critical review of automatic patch generation learned from human-written patches: essay on the problem statement and the evaluation of automatic software repair". In: *Proceedings of the 36th International Conference on Software Engineering*. 2014, pages 234–242.

[9] M. Monperrus. "Automatic software repair: a bibliography". In: *ACM Computing Surveys (CSUR)* 51.1 (2018), page 17.

[10] C. Parnin and A. Orso. "Are automated debugging techniques actually helping programmers?" In: *Proceedings of the 2011 international symposium on software testing and analysis*. 2011, pages 199–209.

[11] S. Pearson, J. Campos, R. Just, G. Fraser, R. Abreu, M. D. Ernst, D. Pang, and B. Keller. "Evaluating and improving fault localization". In: *Proceedings of the 39th International Conference on Software Engineering*. 2017, pages 609–620.

[12] Z. Qi, F. Long, S. Achour, and M. Rinard. "An analysis of patch plausibility and correctness for generate-and-validate patch generation systems". In: *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. 2015, pages 24–36.

[13] E. K. Smith, E. T. Barr, C. Le Goues, and Y. Brun. "Is the cure worse than the disease? Overfitting in automated program repair". In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. 2015, pages 532–543.

Understanding Change-Behavior of Data and Metadata

Tobias Bleifuß

Information Systems Group
Hasso-Plattner-Institut
tobias.bleifuss@hpi.de

The following report gives an overview of my research activities in the field of change exploration. It outlines our data model with its exploration primitives, transformations of datasets into this model and the classification of changes represented in this model. Finally, an outlook on possible applications of the gained insights presents ideas on how to predict future changes and build trust.

1 Change in Data and Metadata

Data change, all the time. This undeniable fact has motivated the development of database-management systems (DBMSs) in the first place. While DBMSs are good at recording this change, and while much technology has emerged to analyze this data, there has not been much research on exploring and understanding the change-behavior of data and metadata over time.

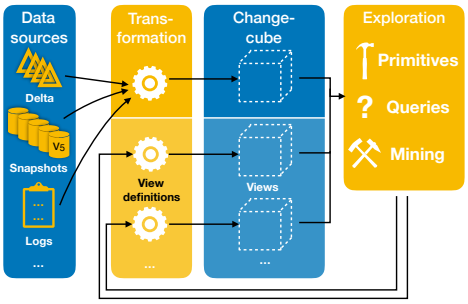
Also: schemata change, quite often. While such metadata-change happens less frequently, schemata are much less stable than what is alluded to in DBMS textbooks, and than what is desirable from an application developer's point of view, in our experience. In particular, modern "schemaless" DBMSs exacerbate the need to explore changing metadata.

The observed changes are manifold: values are inserted, deleted or updated; entities appear and disappear; properties are added or repurposed, etc. Explicitly recognizing, exploring, and evaluating such change can alert to changes in data ingestion procedures, can help assess data quality, and improve the general understanding of the dataset and its behavior over time.

With this work we address a common but less studied addition to the well-known "V"s of big data [3]: To the common notions of volume, velocity, and variety that characterize big data problems, we address *variability*. While variety is typically viewed as differences in format, semantics and other properties between multiple data sources, we introduce variability as describing such differences even *within* a data source over time.

2 Change Exploration: our Model, its Primitives, and a Tool

We propose a generic model to represent changes to a dataset. In particular, our model of *change-cubes* can reflect changes to both the data and its schema and integrate changes from various data sources. It includes the following four dimensions



Operator	Description
`sort`	sorts the changes within a change-cube
`slice`	filters the changes in a (set of) change-cube(s)
`split`	groups changes and splits the cube accordingly
`union`	unions multiple change-cubes into one
`rank`	sorts (a set of) change-cubes
`prune`	filters change-cubes by threshold
`top`	selects change-cubes by relative position

Figure 1: An overview of the DBChEx workflow and a short description of the devised exploration primitives on the change-cube

to represent when (time) what (entity) changed where (property) and how (new value):

Time A timestamp in the finest available granularity.

Entity The id of an entity represented in the dataset. An entity could correspond to a row in a relational database, a node in a graph, a subject of an RDF-triple, a schema element, etc. Entities can be grouped or belong to a hierarchy, modeled separately. In this way, we can recognize which rows belong to the same table, which RDF-subjects are of the same class, etc.

Property The property of the entity. Properties can be entities themselves and correspond to columns in a table, properties of a graph, predicates of an RDF-triple, schema associations, etc. Properties can be hierarchically organized, for instance grouped by semantic domain, such as person name, address, etc., or by data-type.

Value The new value introduced by the change (or the null-value ($\perp$) to represent a deletion). Values can be literals, which we denote in quotes, or ids of other entities. Furthermore, values need not be atomic, they can be sets or lists.

Without the time-dimension, the cube represents the traditional model-independent representation of facts as triples. By including time we can define an individual change and the change-cube as a set of changes:

Definition 1. *A* change *c is a quadruple of the form $\langle$timestamp, id, property, value$\rangle$ or in brief $\langle t, id, p, v\rangle$. We call a set of changes a* change-cube *$C = \{c_1, \ldots, c_n\}$. Among the changes, combinations of (t, id, p) are unique.*

A change $\langle \mathsf{t}, \mathsf{id}, \mathsf{p}, \mathsf{v}\rangle$ describes that at a certain point in time t (*When?*) the property p (*Where?*) of an entity with the stable identifier id (*What?*) was changed to the new value v (*How?*). Implicitly, a value of a change is valid from timestamp until the closest succeeding timestamp of a quadruple with same id and property but different value. Or it is valid until "now", when no succeeding quadruple exists.

On top of that data model, we enable *change exploration*, which we define as: for a given dynamic dataset, we aim to efficiently capture and summarize changes at value-, aggregate-, and schema-level, and enable users to effectively explore this change in an interactive and graphical fashion. We implemented a web-based exploration tool *DbChEx* that facilitates change exploration through a set of exploration primitives. Figure 1 shows how we envision the general workflow using DBChEx and a short summary of the implemented primitives. Because the defined operators are closed (operators are defined on a set of change-cubes and their result is also a set of change-cubes), we can compose them and we call such a composition of operators an *operator sequence*. The operator sequences are the primary navigation through the changes: they represent the current exploration path and show what the user currently focuses on (comparable to the navigation bar in a web browser). The user-interface is designed to keep the user engaged and reduce the amount of required text input. In our tool it usually takes the user just a click to add, edit, or remove new operators from the current operator sequence. A large portion of the user's findings based on exploration primitives are lead by serendipity, but also by a number of statistics, such as a *volatility* measure. However, our tool also integrates results from change mining (see Section 4). We define change mining as the automatic discovery of interesting change events and identification of underlying structures and constraints of change. Change mining can provide an overview of the vast amount of data, for example by clustering changes, which in turn can facilitate exploration.

Change exploration requires access not only to the current dataset, but also to its history. While all subsequent steps will profit by the unified change-cube model, it entails the upfront overhead of transforming the changes into the change-cube. The obvious problem is that there are many different ways of how the changes are stored in the real-world, which implies a high variety of different input formats. We transformed many datasets and their history into our model, for example the changes of Wikipedia infoboxes and tables (see Section 3) or IMDB data.

These transformations from data sources to change-cubes can be relatively generic in the beginning, because the user may not know and understand the previous schemata. Hence, we propose an iterative, closed-loop approach that improves the semantics of the change-cube over time. In this way, knowledge gained during exploration can be fed back into the transformation. For example, if users recognize that a property has been renamed, they can merge these two properties at the click of a button. Each of these transformations creates a view on the original change-cube. The view definitions can also be implemented with the help of operator sequences, in which the above operators are used to select the part of the cube to be transformed. The transformation on the selected quadruples is then achieved either via templates for frequent schema changes (such as renaming) or custom transformations defined in SQL or an external program.

Related work. Because data change is a fundamental concept of databases, many research areas are related, though none come close to covering what we envision. It is impossible for space reasons to discuss them all here, but there is an overview in

our vision paper [15], in which we for example compare our approach to temporal or sequence databases [2] and time series exploration techniques and tools [6].

3 Tracking Changes: Wikipedia Table History

Many Wikipedia tables have a long and eventful edit history. Until now users could only explore and understand the history of Wikipedia tables by manually browsing through the past revisions of a Wikipedia article. However, this is a cumbersome process as it is not obvious, which of the numerous revisions contain changes to a particular table. For an automatic extraction of the changes in tables, there is one big hurdle: given two consecutive revisions of the same Wikipedia article, the two versions of Wikitext (the markup language used in Wikipedia) do not immediately reveal what changes occurred.

Clearly, we can calculate a line-wise difference of two document revisions, but that approach completely ignores any semantic information. Therefore, in many cases the resulting diff-set does not represent the intentions the user had in mind when performing a certain change. The line-wise difference assumes that the (relative) position of an object stays constant over its lifetime and is incapable of identifying moving objects. In particular, given a new document revision and a set of previously seen tables in the same document, we want to determine which table is the successor of which other table. And, in order to get an even more fine-grained picture of table changes, we want to determine, which cell is the successor of which other cell within a table. This allows us to track tables and their cells over multiple document revisions in time. Our aim is *not* to calculate a minimal difference between two object revisions. Instead, we want to link multiple versions of the same element over multiple revisions in a semantically meaningful way.

We solve this task using the assumption that changes happen gradually, so that the context of changed elements can still be used to identify elements over time. This assumption allows us to use a rather simple similarity measure based on tokens. However, because it is difficult to find a similarity measure that works well universally, our method combines several similarity measures in multiple matching steps. The idea here is to first create obvious matches with as little computational effort as possible and linear scalability in the number of tables and only then to match the remaining candidates with more relaxed methods and potentially more expensive similarity metrics. These relaxed procedures are necessary, for example, to cope with the fact that tables can also grow or shrink, which has a negative impact on certain similarity measures.

Our manually generated gold standard shows that this procedure works very well on tables and our underlying assumption seems to hold. Our experiments also show that, despite this assumption, our algorithm remains robust at higher temporal resolutions. In this experiment, we simulated a scenario in which snapshots of the table are only available at longer intervals, such as a monthly crawl of web pages. In this case, too, our methods still achieve very good results. For the matching of cells it was more challenging to create a gold standard, because of the sheer number of cells

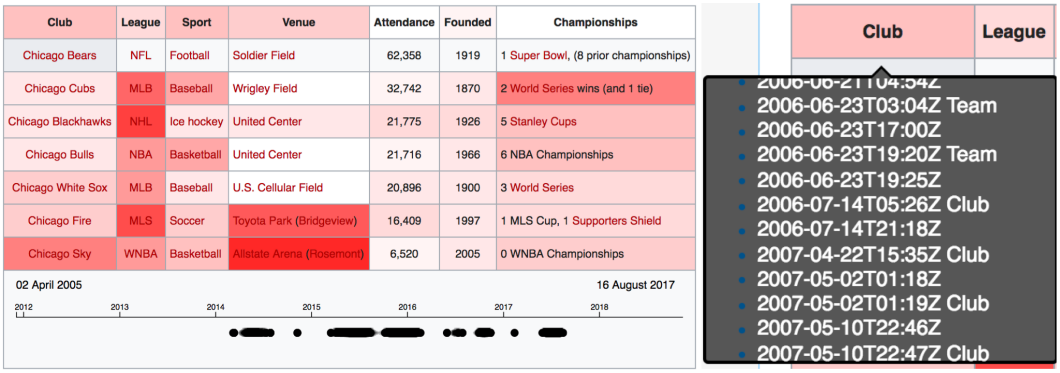


Figure 2: A browser plugin that displays the history of Wikipedia tables

even within one version of a table times the number of table versions. But we have now found a suitable solution that assists the user in picking a good cell matching and are confident to publish results soon.

We also developed a browser plugin that makes such histories accessible and explorable by the users. By tracking the history of tables and also their cells, we can enrich each table in a Wikipedia article by a timeline that displays revisions that contain changes to this particular table (see Figure 2). The user can examine the table in any of these revisions by simply clicking on that revision in the timeline. Furthermore, the history of individual cells allows us to draw a heat map that reflects certain cell metadata, such as the volatility of a cell or the age of the current value. The past values of a cell can be explored by hovering over a cell. These means allow users to gain insights on how the table developed over time, which can influence the trust in its content or give inspirations for future edits.

Related work. Related datasets have been extracted from the web and Wikipedia before. WebTables [4] are an extract of static versions of tables on the web. The infobox history [8] comprise an orthogonal subset of structured information on Wikipedia, namely However, the problem is different, as each property of an infobox has a unique identifier. Additionally, they ignore the fact that a Wikipedia article can contain multiple infoboxes. The tracking of individual rows has parallels with linking temporal records [7], but we do not assume a static schema.

4 Change Mining: Distinguish Different Change Types

Often it is necessary to distinguish different types of changes. For example, if you take a closer look at the changes in the change-cubes, you will notice that a large part of the changes will appear as noise. What exactly is considered noise is of course dependent on the use-case. While for some applications one user might want to ignore, e.g., vandalism in Wikipedia, another user might be particularly interested in vandalism. Besides vandalism, other types of noise exist, such as schema or format changes. In our change-cube format, these result in many changes and quickly

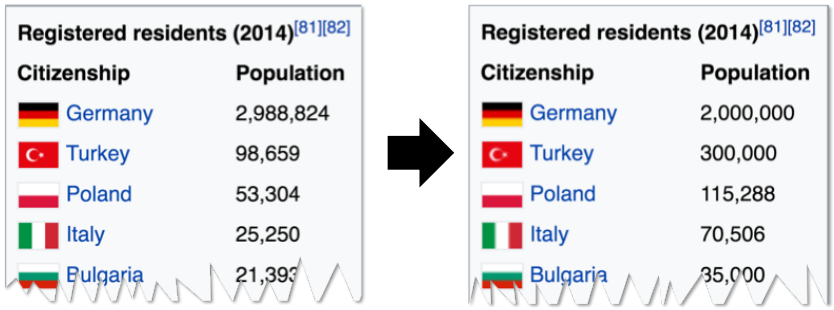


Figure 3: An example for subtle vandalism in Wikipedia, which tampered with the ratio of nationalities of Berlin citizens

dominate in number and size. Nevertheless, in both cases it is necessary to have a way of distinguishing between the different types of changes. If the different types are known in advance, a method method to distinguish them is called *classification*, but we also have published results on how to perform a *clustering* of change-cubes through mapping them to time series [16]. These clusterings are especially useful for exploration as also mentioned in Section 2.

Our master project in the last semester dealt with the detection of vandalism in Wikipedia tables. In particular, the three students examined whether the structure of tables could improve the results compared to purely text-based methods. Interestingly, we have found in tables a very subtle kind of vandalism which can only be detected with considerable background knowledge (see Figure 3). However, the results show that this is not an easy problem, because it is relatively difficult to find features that perform better than the already known, more general features from related work. Without external knowledge, it is even difficult for humans to distinguish between vandalism and non-vandalism in the case of the subtle type mentioned above.

Another classification problem is the following: When thinking of database changes, the first thing that comes to mind are changes that insert new elements or delete old ones or adjust values based on new factual information. However, because not only the information stored in the database changes over time, but also the requirements placed on the database, the representation of the information in the database also changes. These new requirements may have operational reasons, such as a new database system or performance optimizations, but also functional reasons, such as the need to store additional information about existing entities. I am currently working on distinguishing these two types of changes. I intend to use a machine-learning method, which also includes elements of rule-mining. For example, an approach that mines frequent itemsets of attributes that are added to or removed from an entity, can already find results like this one:

```
delete:placeofdeath, delete:born, delete:died,
delete:placeofbirth, add:birth_date, add:birth_place,
add:death_place, add:death_date
```

This result shows, for example, how four attributes were renamed. However, it also leads to new questions, such as which attribute pairs belong together, so which new attributes store the information that were stored in one particular deleted attribute. In addition, such changes will also change the format or domain of the values. So another question would be, whether we can derive automatic transformation rules for the values? If we had such rules, then we could also check whether the schema changes were also accompanied by simultaneous value changes. Furthermore, we could apply those rules to other entities for which that schema change was not performed yet and check for the other entities whether it was applied consistently.

Related work. There is related work on ontology change [5] and database schema evolution [9]. These works assume that you have multiple versions of an explicit ontology/schema, while we try to infer that an implicit schema changed based on observed data changes. There is also work on schema inference [13] that tries to infer schema on schemaless datasets, which, however, assume a static dataset and a static schema. If we apply this schema inference on multiple snapshots, though, and compare the inferred schemas, we might also recognize schema changes.

5 Future Work: Change Prediction and Trust

One step further than change mining is the change prediction: given a set of past changes can we predict which future changes are likely to happen and when they are going to happen? This problem is related to the problem of knowledge graph completion [12], which deals with the prediction of missing links between entities. The works in this field often assume a static knowledge graph, however there have been recent works that recognize that this assumption might be too strong for real-world datasets and therefore deal with dynamic knowledge graphs [10, 14], which are much more related to our problem than static graphs. There is also work that leverages schema information [11] for that task, but we are not aware of any work that tries to solve the problem on dynamic knowledge graph with a dynamic schema. Besides these probabilistic models, we also want to look into rule-based approaches. As an initial inspiration, we consider temporal association rules [1] and how we can adjust and apply them to the change-cube.

The insight we gain from our analysis will not only give the user a better understanding of their data and its behavior over time. In fact, it is possible to draw profit from these insights through application in, e.g., increasing data quality through notification of unusual change events and database tuning. For example, the discovered knowledge can support decisions on which indices and materialized views to choose, because it can serve as an input for predictions of their update costs. More importantly, it can help users decide which data to trust and which not to trust, which appears more important than ever today.

References

[1] J. M. Ale and G. H. Rossi. "An approach to discovering temporal association rules". In: *Proceedings of the ACM Symposium on Applied Computing*. 2000, pages 294–300.

[2] R. T. Snodgrass. *Developing Time-Oriented Database Applications in SQL*. Morgan Kaufmann, 2000.

[3] D. Laney. *3D Data Management: Controlling Data Volume, Velocity and Variety*. Technical report. Gartner, 2001.

[4] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. "WebTables: exploring the power of tables on the web". In: *Proceedings of the International Conference on Very Large Databases (VLDB)* 1.1 (2008), pages 538–549.

[5] G. Flouris, D. Manakanatas, H. Kondylakis, D. Plexousakis, and G. Antoniou. "Ontology change: classification and survey". In: *The Knowledge Engineering Review* 23.2 (2008), pages 117–152.

[6] J. Zhao, F. Chevalier, and R. Balakrishnan. "KronoMiner: using multi-foci navigation for the visual exploration of time-series data". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2011, pages 1737–1746.

[7] P. Li, X. L. Dong, A. Maurino, and D. Srivastava. "Linking temporal records". In: *Frontiers of Computer Science* 6.3 (2012), pages 293–312.

[8] E. Alfonseca, G. Garrido, J. Delort, and A. Peñas. "WHAD: Wikipedia historical attributes data - Historical structured data extraction and vandalism detection from the Wikipedia edit history". In: *Language Resources and Evaluation* 47.4 (2013), pages 1163–1190.

[9] A. Cleve, M. Gobert, L. Meurice, J. Maes, and J. Weber. "Understanding database schema evolution: A case study". In: *Science of Computer Programming* 97 (2015), pages 113–121.

[10] C. Esteban, V. Tresp, Y. Yang, S. Baier, and D. Krompaß. "Predicting the co-evolution of event and knowledge graphs". In: *International Conference on Information Fusion (FUSION)*. 2016, pages 98–105.

[11] P. Minervini, C. d'Amato, N. Fanizzi, and F. Esposito. "Leveraging the schema in latent factor models for knowledge graph completion". In: *Symposium on Applied Computing (SAC)*. 2016, pages 327–332.

[12] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. "A review of relational machine learning for knowledge graphs". In: *Proceedings of the IEEE* 104.1 (2016), pages 11–33.

[13] M.-A. Baazizi, H. B. Lahmar, D. Colazzo, G. Ghelli, and C. Sartiani. "Schema inference for massive JSON datasets". In: *Proceedings of the International Conference on Extending Database Technology (EDBT)*. 2017, pages 222–233.

[14] R. Trivedi, H. Dai, Y. Wang, and L. Song. "Know-Evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs". In: *International Conference on Machine Learning*. 2017, pages 3462–3471.

[15] T. Bleifuß, L. Bornemann, T. Johnson, D. V. Kalashnikov, F. Naumann, and D. Srivastava. "Exploring Change – A New Dimension of Data Analytics [Vision]". In: *Proceedings of the International Conference on Very Large Databases (PVLDB)* (2018). Accepted.

[16] L. Bornemann, T. Bleifuß, D. Kalashnikov, F. Naumann, and D. Srivastava. "Data Change Exploration using Time Series Clustering". In: *Datenbank Spektrum* 18.2 (2018), pages 1–9.

Mutual Human Actuation

Lung-Pan Cheng

Human Computer Interaction
Hasso-Plattner-Institut
lung-pan.cheng@hpi.uni-potsdam.de

Human actuation is the idea of using people to provide large-scale force feedback to users. While the experience of human actuators was decent, it was still inferior to the experience these people could have had, had they participated as a user. We address this issue by making everyone a user. The key idea is to run pairs of users at the same time and have them provide human actuation to each other. Our system, Mutual Turk, achieves this by (1) offering shared props through which users can exchange forces while obscuring the fact that there is a human on the other side, and (2) synchronizing the two users' timelines such that their way of manipulating the shared props is consistent across both virtual worlds. We demonstrate mutual human actuation with an example experience in which users pilot kites though storms, tug fish out of ponds, are pummeled by hail, battle monsters, hop across chasms, push loaded carts, and ride in moving vehicles.

1 Introduction

Many researchers argue that the next step in virtual reality is to allow users to not only see and hear, but also feel virtual worlds [10]. Researchers initially explored the use of mechanical machinery for that purpose, such as exoskeletons [1] and robotically actuated [7] props.

Unfortunately, the size and weight of such mechanical equipment tends to be proportional to what they actuate, often constraining such equipment to arcades and lab environments.

Researchers therefore proposed creating similar effects by replacing the mechanical actuators with human actuators. Haptic Turk, for example, uses four such human actuators to lift, bump, and shake a single human user [2]. TurkDeck brings human actuation to real walking [3]. It allows a single user to explore a virtual reality experience that is brought to life by ten human actuators that continuously rearrange physical props and apply forces to the user.

While both systems produced highly-rated experiences for their users during user testing, unfortunately (1) the need to recruit four to ten human actuators means that these systems require a non-trivial amount of preparation, and (2) unsurprisingly, human actuators rated their experience significantly lower than the user's experience.

In this paper, we address this issue by making everyone a user. We introduce mutual human actuation, a version of human actuation that works without dedicated human actuators.

2 Mutual Turk

Mutual Turk is a real walking virtual reality system that implements mutual human actuation. The key idea behind Mutual Turk is that it runs two users at the same time, synchronizing their experience so that every time one user is manipulating an object in her virtual world, the other user is subjected to forces presumably caused by something in his virtual world.

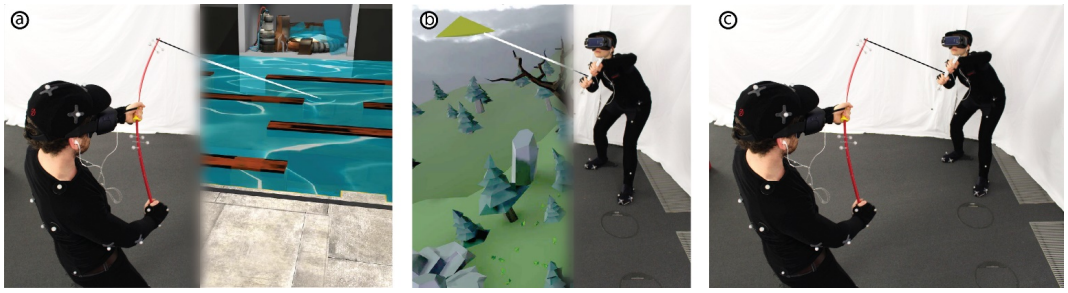


Figure 1: (a) This user, alone in his virtual world, is trying to pull a huge creature out of the water. (b) At the same time, this other user, also alone in her virtual world, is struggling to control her kite during a heavy storm. (c) While users' experiences of force might suggest the presence of a force feedback machine, Mutual Turk achieves force feedback instead by orchestrating users so as to actuate the shared prop at just the right moment and with just the right force to produce the correct experience for the other user.

Figure 1 shows an example. (a) One of the two users, alone in his virtual world, is trying to pull a huge creature out of the water. Through the fishing rod he feels how the creature is struggling. (b) At the same time, the other user, also alone in her virtual world, is struggling to control her kite during a heavy storm, which she feels pulling at her kite. (c) In reality, both users are connected by means of a shared prop, so that all forces they output become input to the other user. This is the main concept behind Mutual Turk. Mutual Turk's main functionality is that it orchestrates users so as to actuate props just at the right moment and just with the right force to produce the correct experience for the other user.

2.1 Shared Props

The key to enabling mutual human actuation is the use of *shared props* – props allow users to exchange forces without revealing that these forces are generated by another human (through, e.g., skin softness, temperature, moisture, shape of hands).

Different prop designs enable different levels of expressivity. We explored five: continuous force (most expressive), moving, impact, contactless sensations, and rearranging props (least expressive).

1. Continuous Exchange of Force Between Users' Hands The kite/fishing-rod prop from Figure 1 uses string to connect the two handles. This specific design allows the prop to eliminate much of the information about what is located at the other end of the prop – the only information that is transmitted is the direction and magnitude of the tension. This allows the system to re-envision the many dimensions that were filtered out, such as to render the kite at the end of a 100x longer tether.

2. Continuous motion In Figure 2a, the user pushes an empty cart under a faucet. He watches as the faucet drops water into the tank. (b) Meanwhile, in the other user's world, she hops onto the escape pod. (c) She then rides the automated pod down an evacuation tunnel – propelled by the first user pushing his cart, as the first user starts to push his (now much heavier) cart on to next destination.

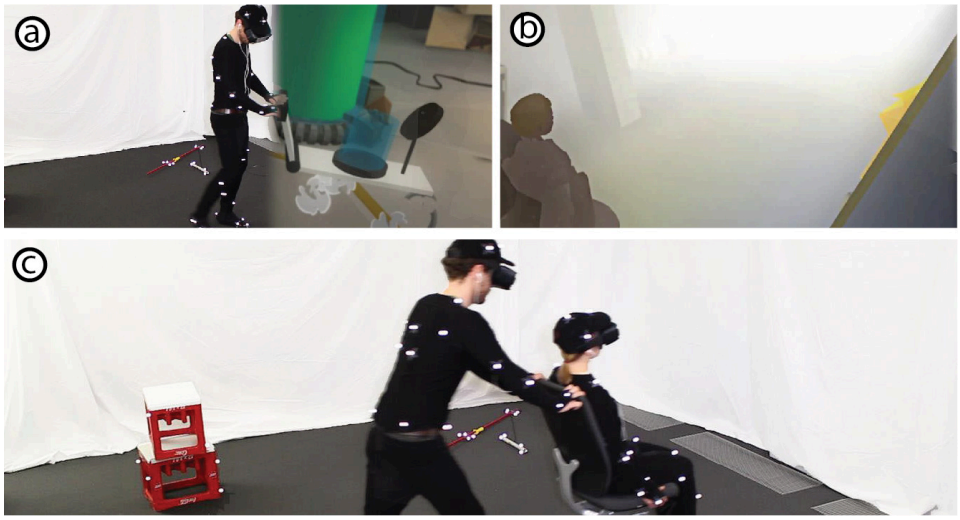


Figure 2: One user pushes cart around while the other enters and rides an escape pod

The office chair prop transmits movement and rotation in one direction. In return, the user sitting on the chair affects the chair's inertia. Unlike the interactions in the previous category, only one user's hands are involved in driving the office chair. This allowed us to drop the tether and use a rigid prop instead.

3. Impact The user in Figure 3 sees himself walking in stormy weather; he sees huge hailstones shooting down from the sky at an angle, hitting his body at various locations. (b) In the meanwhile, the other user is fighting back a monster using an improvised weapon made from a plastic tube she found at the lab.

The foam stick used in this scene touches the other user for only very brief periods of time, which properly obfuscates the origin of the force.

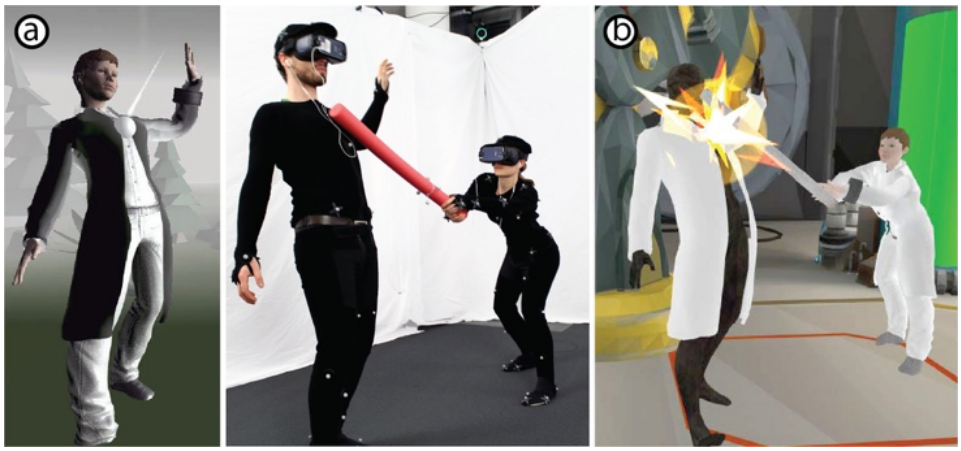


Figure 3: One user is getting bombarded by hail, as the other user is fighting a monster

4. Contactless sensations In Figure 4a, our user is trying to fight her way back to lab against very heavy wind. (b) Meanwhile, our user in the other world is trying to get a fire going to distill the emulsion created earlier.

The forces exchanged in this scene are obviously minimal. However, the interaction produces a strong tactile sensation (and certainly properly obfuscated).

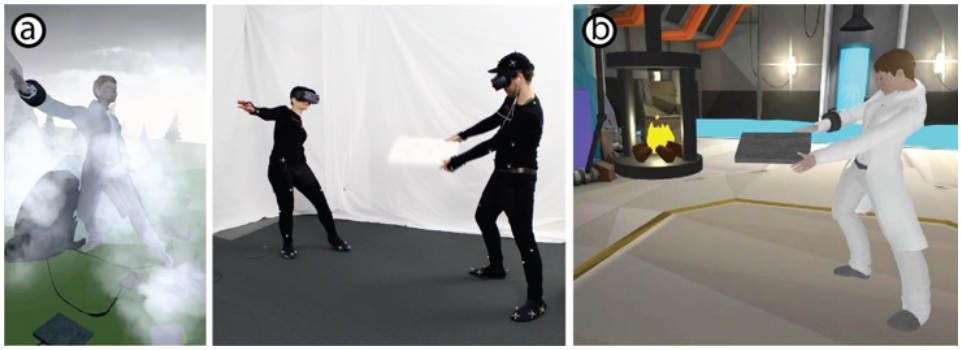


Figure 4: One user is trying to fight her way through heavy winds, while the other is trying to get a fire going

5. Rearranging props Finally, Figure 5a shows a user waiting for a series of pillars to rise in order to allow her to cross the pit ahead of her. As she lowers her right foot to probe the space below, she can feel the void. Once she sees that the pillar has fully risen, she can step on it. (b) In the meanwhile, the other user is solving a puzzle that requires him to place numbered boxes on matching tiles.

This is the least expressive type of exchange between two users as no physical contact between the two users is ever established. It thus is also the most obfuscated type of interaction.

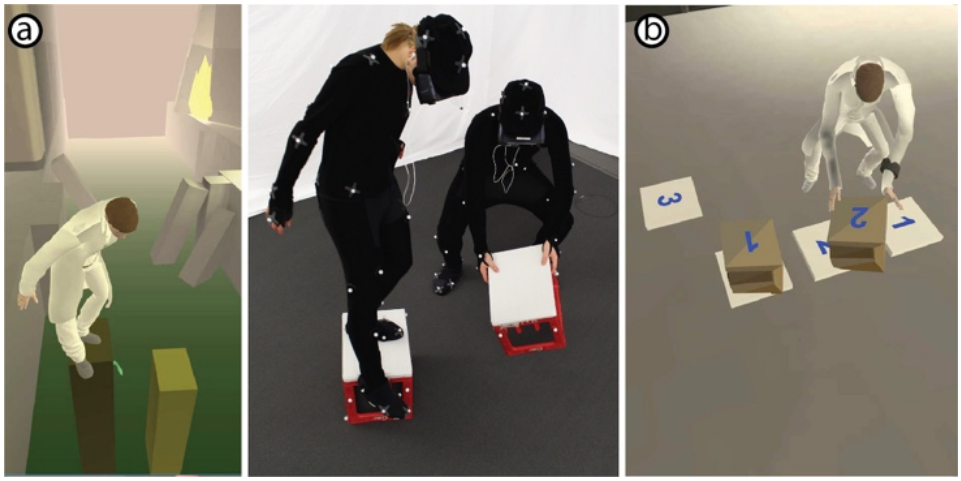


Figure 5: One user is waiting for the next pillar to rise, while the other user rearranges boxes to solve a puzzle

2.2 Synchronizing Users

As discussed earlier, the main function of the Mutual Turk system is to serve as scheduler, i.e., to orchestrate the two users in a way that their experiences are properly synchronized in time and space.

So far, we only looked at what we call action sequences, i.e., sequences during which the users already hold the shared prop and the subsequent interaction emerges largely from the use of this prop.

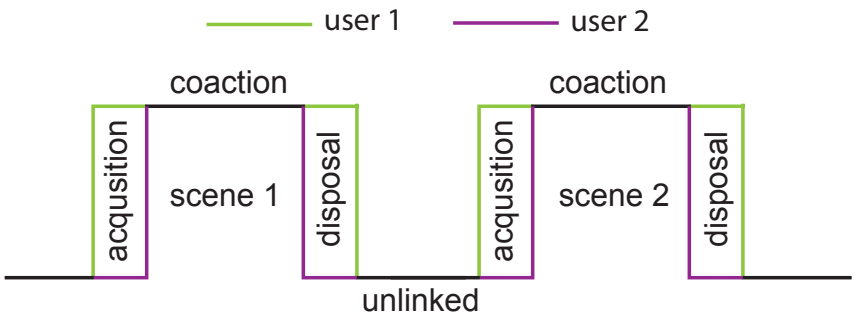


Figure 6: Mutual Turk experiences typically consist of multiple scenes, each of which consists of prop acquisition, use, and disposal

As illustrated by Figure 6, complete Mutual Turk experiences are more encompassing than this. Mutual Turk must not only synchronize the use of the shared props, but also their acquisition and disposal. A typical scene consists of a period of real walking within a designated area, the acquisition of a prop, the use of the prop forming an action sequence, the disposal of the prop, and return to unencumbered real walking. Experiences are then sequences of such scenes.

3 Implementation

As illustrated by Figure 7, Mutual Turk runs inside of Unity 3D and is written in C#. This includes (1) a native OptiTrack NatNet Unity plug-in that receives the tracking data directly from OptiTrack Motive, (2) Petri net server and client and (3) our demo experience called "*Edison, Jr.*" (in which users have to perform a series of experiments to help their ancestor regain physical form).

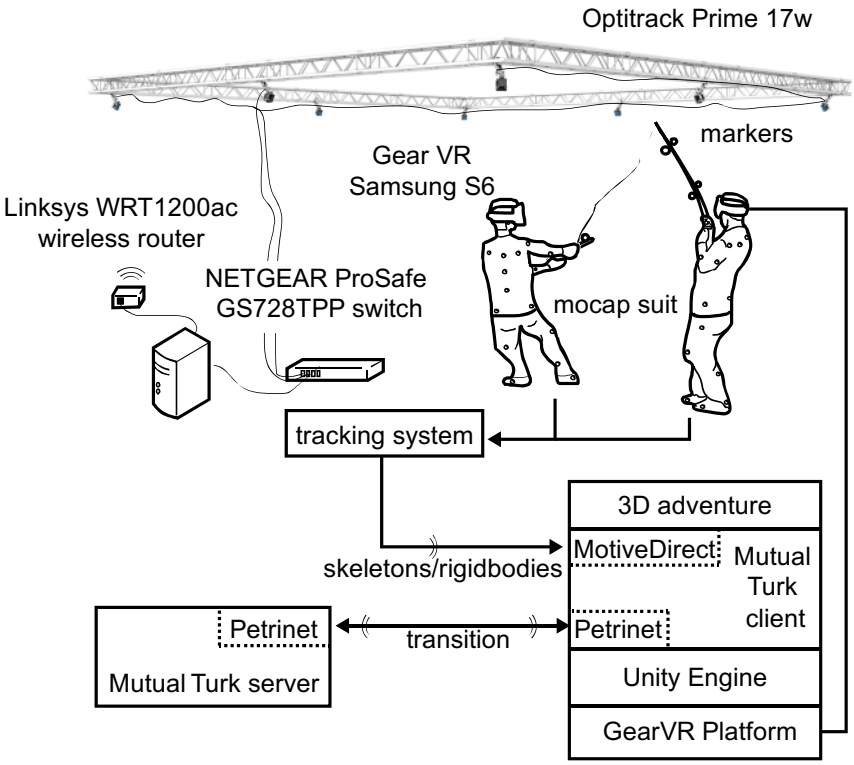


Figure 7: The Mutual Turk system diagram

Headsets to allow for unencumbered real walking, we used Samsung S6s mounted into GearVR headsets with earphones attached. Both headsets run their own Unity app where a Mutual Turk client and the adventure experience are embedded. Via our wireless network, the Mutual Turk client receives the tracking data and com-

municates with the Mutual Turk server to synchronize its Petri net with the other Mutual Turk clients.

For tracking, we use nine OptiTrack Prime 17w cameras to track a 5 m x 5 m tracking space, running the OptiTrack Motive 2.10 tracking software. Users wear motion capture suits. To make props trackable, we attached rigid body markers, 6.7 mm to 9.5 mm.

Mutual Turk runs in real time with two users. We achieved 40+ fps by making VR scenes low-poly, simple lights, etc. In addition, we enabled time warping to guarantee interactive rates. The maximum delay between visual and haptics was around 25ms. The devices receive tracking updates wirelessly at 120 Hz with ping interval 5ms in average.

3.1 Tracking Acquisition and Disposal

Mutual Turk determines when to advance the global timeline using simple rules, such as "fishing rod user is touching the fishing rod and the fishing rod prop has started to move".

Internally, Mutual Turk considers the two users and their acquisition and disposal of props as a concurrent state machine. It manages this state machine as a Petri net [8]. This allows Mutual Turk to ensure that the overall story arc does not progress until both users are ready for it. The Petri net is also useful for level designers to detect and avoid potential dead locks, i.e., situations where both users would be waiting for each other.

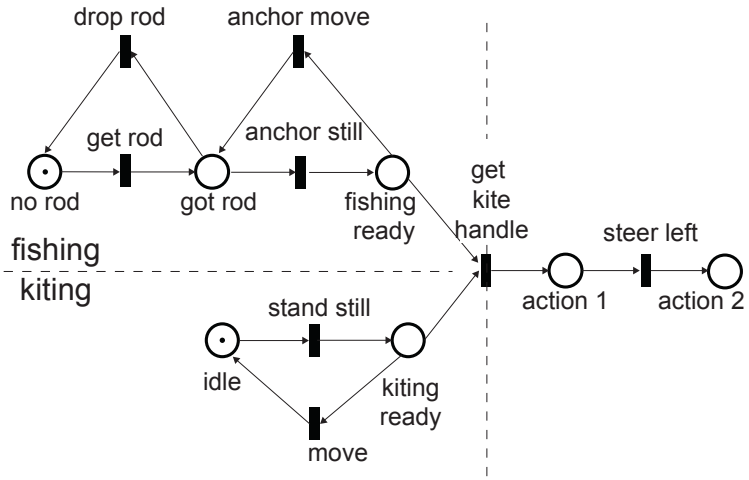


Figure 8: The Petri net diagram that governs the acquisition sequence of the fishing-vs.-kite scene

Figure 8 shows the Petri net of the fishing rod vs. kite acquisition sequence described earlier. As we see, in the first half of the Petri net, the fishing user and the

kite user have no influence on each other. The fishing user has the freedom to pick up or drop the fishing rod anytime and the kite user has the freedom to walk around as well. Only when they both are in their correct respective locations and the kite user has grabbed the handle, both users get to move on. One can extend the Petri net to continue the experience of the remaining user as a single-user without mutual haptics experience if one of the users refuse to progress.

3.2 Tracking and Extrapolation During Action Sequences

In the fishing rod vs. kite scene, Mutual Turk needs to know the amount of tension on the tether, e.g., in order to determine whether the user is pulling hard enough to reel in the creature, but also to render the kite and fishing line visuals properly. Figure 9 illustrates how the fishing rod/kite prop allows Mutual Turk to sense this tension. The key idea is that the prop bears two markers on the fishing rod side. The angle between the two markers indicates how much the rod is currently bent, which indicates the applied force.

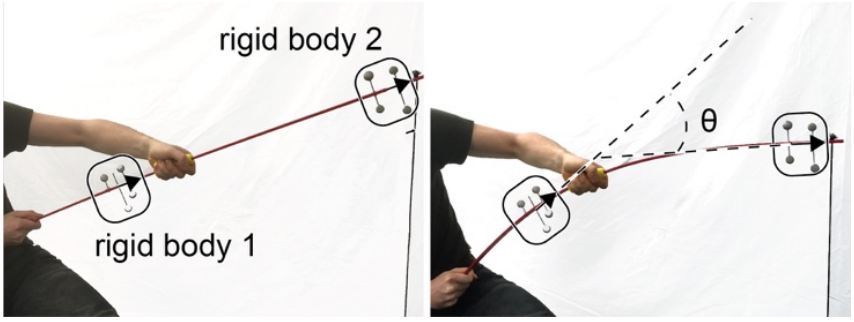


Figure 9: Mutual Turk computes the tension applied to the fishing line based on how much the prop is bent

4 Related Work

This work is based on haptics and motion experience devices, passive haptics, and in particular human actuation.

4.1 Haptics and Motion Experience Devices

A wide range of devices has been created in order to provide users with a sense of touch and motion. Many of the standard stationary platforms are based on the 6-DOF Stewart platform based on six hydraulic cylinders [9]. Force feedback can be realized through a variety of approaches. FlexTorque creates force feedback using an

arm exoskeleton using retractable belts [11]. Seminal work by McNeely introduced the idea of using a robotic arm to repositioning a single prop so as to simulate a surface wherever the user tries to touch [7].

4.2 Passive Haptics

Previous work shows that props, also known as passive haptics can enhance the sense of presence [4]. Several "passive haptic" systems use physical props in real walking environments. Low et al., for example, use Styrofoam walls onto which they project augmented reality experiences [6]. Similarly, mixed reality for military operations in urban terrain [5] uses passive haptics to add a haptic sense to otherwise virtual objects, terrain, and walls.

4.3 Human Actuation

Haptic Turk [2], for example, uses four such human actuators to lift, bump, and shake a single human user in the form of a human motion platform. By making human actuators perform movements according to timed motion instructions, Haptic Turk assures that users' physical experience matches their virtual experience. TurkDeck extends the concepts of human actuation to real walking [3]. It allows a single user to explore a virtual reality experience that is brought to life by ten human actuators that continuously rearrange physical props and apply forces to the user. Mutual Turk builds on Haptic Turk and TurkDeck, but eliminates the need for dedicated human actuators.

5 Conclusion and Future Work

We have introduced the concept of mutual human actuation, presented the implementation details and demonstrated this system at the example of a demo experience. The main benefit of our approach is that it eliminates the need for dedicated human actuators and instead allows everyone to enjoy their experience in the role of a user. The main limitation of Mutual Turk is that designing experiences for mutual human actuation requires additional care, as each scene is subject to at least twice the number of design requirements as regular scenes. As future work, we are planning to develop a design tool based on the Petri net model and extend mutual human actuation to more than two users.

References

[1] M. Bergamasco. "The GLAD-IN-ART Project". In: *Virtual Reality*. 1993, pages 251–258. ISBN: 978-3-642-88650-8.

[2] L.-P. Cheng, P. Luehne, P. Lopes, C. Sterz, and P. Baudisch. "Haptic Turk: A Motion Platform Based on People". In: *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*. 2014, pages 3463–3472. DOI: 10.1145/2556288.2557101.

[3] L.-P. Cheng, T. Roumen, H. Rantzsch, S. Koehler, P. Schmidt, R. Kovacs, J. Jasper, J. Kemper, and P. Baudisch. "TurkDeck: Physical Virtual Reality Based on People". In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology*. 2015, pages 417–426. DOI: 10.1145/2807442.2807463.

[4] H. G. Hoffman. "Physically Touching Virtual Objects Using Tactile Augmentation Enhances the Realism of Virtual Environments". In: *Proceedings of the Virtual Reality Annual International Symposium*. VRAIS '98. 1998, pages 59–64. ISBN: 0-8186-8362-7.

[5] C. E. Hughes, C. B. Stapleton, D. E. Hughes, and E. M. Smith. "Mixed Reality in Education, Entertainment, and Training". In: *IEEE Comput. Graph. Appl.* 25.6 (2005), pages 24–30. DOI: 10.1109/MCG.2005.139.

[6] K.-L. Low, G. Welch, A. Lastra, and H. Fuchs. "Life-sized Projector-based Dioramas". In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. 2001, pages 93–101. ISBN: 1-58113-427-4. DOI: 10.1145/505008.505026.

[7] W. A. McNeely. "Robotic Graphics: A New Approach to Force Feedback for Virtual Reality". In: *Proceedings of IEEE Virtual Reality Annual International Symposium*. 1993, pages 336–341. DOI: 10.1109/VRAIS.1993.380761.

[8] W. Reisig. "Petri Nets and Algebraic Specifications". In: *Theor. Comput. Sci.* 80.1 (1991), pages 1–34. DOI: 10.1016/0304-3975(91)90203-E.

[9] D. Stewart. "A Platform with Six Degrees of Freedom". In: *Proceedings of the Institution of Mechanical Engineers* 180.1 (1965), pages 371–386. DOI: 10.1243/PIME_PROC_1965_180_029_02.

[10] I. E. Sutherland. "The Ultimate Display". In: *Proceedings of the IFIP Congress*. 1965, pages 506–508.

[11] D. Tsetserukou, K. Sato, and S. Tachi. "FlexTorque: Exoskeleton Interface for Haptic Interaction with the Digital World". In: *Proceedings of the 2010 International Conference on Haptics - Generating and Perceiving Tangible Sensations: Part II*. 2010, pages 166–171. ISBN: 3-642-14074-2.

Process Mining Methodologies

Kiarash Diba

Business Process technology
Hasso-Plattner-Institut
Kiarash.diba@hpi.uni-potsdam.de

Process mining is a discipline that exploits real life data recorded in organizations' information systems to facilitate business process monitoring, analysis and improvement. As a relatively young research area there are a number of challenges unsolved in regard to process mining projects. My research will focus on methodologies to conduct a process mining project considering different requirements and objectives. My research is still in its beginning phases of exploration, and observation which will drive the research to the phases of building the theory, evaluation and deduction.

1 Overview

In every organization a variety of processes exist and are performed. These processes are a set of connected activities performed to realize a business goal, create or deliver value to customers. A few examples of such processes are sale processes, manufacturing processes, handling a purchase order, etc. Business process models are an abstract graphical representation of processes. Business Process Management (BPM) is a field concerning the design, modelling, execution, analysis, optimization and improvement of business processes and uses process models as its basis [4]. One important phase in BPM is the analysis and improvement of processes. On one hand, most classical process-oriented analyses in BPM (such as simulation) do not incorporate available data, while on the other hand, data-based analysis do not consider the end to end processes. Process mining however, bridges this gap between classical data analysis and process-oriented analysis [3]. Process mining incorporates data recorded in various information systems and ERP systems to analyze processes and generate insight on the process models and their performances. The starting point for process mining is an event log found in various information systems, usually consisting of a case id, the performed activity and a timestamp. Process mining takes the event data recorded by the software system and enables three main types of analysis: process discovery, conformance checking and process enhancement and performance analysis [2]. Process discovery is the automatic generation of process models using the event data. It answers the questions of what really happened and what the process actually looks like. Conformance checking takes both the event data and the process model and compares them by replaying reality on top of the model. It identifies differences, deviations and the desired paths. Conformance checking can be used for auditing and fraud detection among others. Enhancement takes the log and the model and enriches the model with performance related information such as waiting times, throughput times, and bottlenecks. It enables various

performance related analyses. However, process mining is not restricted to these types. Process mining can be used in online settings. Monitoring processes as they are being executed and incorporating data in real time to facilitate process monitoring, deviation detection, performance prediction (such as remaining flow time for a running case), and recommendation (e.g. identifying the next best activity to perform depending on the case). During the last few years, various process mining techniques, algorithms, and tools have been developed and have been used in different domains and scenarios. However, as e relatively new field, various challenges still exist. As identified in the process mining manifesto published by the members of IEEE task force on process mining, these challenges range from data collection, preparation and cleaning to dealing with concept drifts, providing operational support, etc. Various further challenges have been identified in different case studies when applying process mining techniques in different domains. Addressing these challenges has been the focus of process mining researchers in recent years. However, many challenges have remained unsolved and new challenges are emerging as a result of wider applications as use cases. This specifies the various potential research problems in the field of process mining. Currently, different case studies have been conducted in different ways and arising issues and challenges have been addressed in an ad-hoc manner. A more structured methodology for conducting case studies, tackling challenges and extracting valuable insights can have a significant impact on the quality of case studies and identifying process mining best practices. While, techniques and algorithms are reaching a relatively mature level, applications, use cases and a specific methodology to conduct process mining projects in different domains have not followed suit; while, for example, in the field of data mining a dominant methodology called Crisp-DM (short for Cross Industry Standard for Data Mining) has been suggested and successfully used for some years. Therefore, my future work will focus on process mining methodologies and application and use cases. Starting with a systematic literature review of use cases in different domains and structuring projects based on different characteristics such as project goals and objective, techniques, algorithms, tools etc. Further use cases will be conducted and an analysis of different process mining tools will be performed. This will tie to our projects in the Business Process Technology group at HPI. My research can contribute to these projects and can be positively influenced by the project, the collaboration and its result.

2 Approach

The starting point is a systematic literature review of available process mining use cases in different settings, domains and with various objectives. Structuring these case studies and identifying the best practices will contribute to the development of new methodologies to conduct process mining projects in real life. Furthermore, we will conduct case studies to develop, test and improve these methodologies through real life challenging projects.

3 Related Work

As mentioned before, there are not many works concerning methodologies for process mining. Three methodological frameworks have been published to support the execution of a process mining project: (i) the process diagnostic method [1], (ii) the L* life-cycle model and (iii) the PM²-methodology. Non of which are comprehensive and does not provide a generic approach and method to conduct process mining projects in real life setting with various requirements. The process diagnostic method (PDM) consists of 6 high level steps starting from log preparation and log inspection to control flow analysis, performance analysis, role analysis and transferring results. Their methodology is rather high level and do not specify further detail on each step. Furthermore, It does not consider one of the most important factors in process mining projects which is identifying project goals, objectives and requirements. It also neglects techniques and tool requirements. The L* life-cycle model encompasses steps such as plan and justify, extract, create control flow and connect it to the data, create integrated process model, and operational support. Although the L* life cycle model considers the process goal and objectives in the first two phases, it is rather too broad. It includes the whole spectrum of process mining capabilities from process discovery to operational support. Tailoring it to specific project is therefore, not a straightforward task and it can lead to ineffective use of resources for conducting the project and results which were not the main focus of the project. The third method PM²-methodology attempts to improve the two previously mentioned methodologies and includes six phases. Planning, extraction, data processing, mining and analysis, evaluation and process improvement and support. While this methodology is certainly an improvement of the previous methods and addresses some of their problems, there are still lacking areas. The method focuses on narrow perspectives of process mining (e.g. Control-flow), and more importantly neglects the domain in which the project is being conducted and its specific requirements. Therefore, there is still work that needs to be done in this area to drive projects more effectively and with less manual and unstructured effort.

4 Future Work

My future work will focus on process mining methodologies and application and use cases. Starting with a systematic literature review of use cases in different domains and structuring projects based on different characteristics such as project goals and objective, techniques, algorithms, tools and etc. Further use cases will be conducted and an analysis of different process mining tools will be performed. This will tie to our projects in the Business Process Technology group at HPI. My research can contribute to these projects and can be positively influence by the project, the collaboration and its result. During my Master thesis, I have been working on business process simulation tools and their capabilities, limitations and potential improvements. The use of real data through process mining for input to simulation scenarios has a significant impact on the reliability and precision of the final result of the analysis.

References

[1] M. Bozkaya, J. Gabriels, and J. M. van der Werf. "Process Diagnostics: A Method Based on Process Mining". In: *International Conference on Information, Process, and Knowledge Management (eKNOW'09)*. 2009, pages 22–27.

[2] W. M. Van der Aalst. "Process Discovery: An Introduction". In: *Process Mining*. 2011, pages 125–156.

[3] W. M. Van der Aalst and A. Weijters. *Process Mining: A Research Agenda*. 2004.

[4] M. Weske. "Business Process Management Architectures". In: *Business Process Management*. 2012, pages 333–371.

Preparing for a Virtual City Model as a Digital Twin of an Urban Environment

Andreas Fricke

Computer Graphics Systems
Hasso Plattner Institute for Digital Engineering
Andreas.Fricke@hpi.uni-potsdam.de

This report describes my recent activities on Service-oriented Systems Engineering in the HPI Research School as a continuation of my Spring Report 2018. Furthermore it summarizes my research and teaching activities since my joining the graduate school in mid-October 2017.

1 Overview

Because of its transdisciplinary nature, geoinformation science integrates research and development of different disciplines to generate solutions for complex research and application problems with a spatial component. Current R&D, however, largely disregards the complexity to model multidimensional geodata from various sources in order to build – in this case – more than a visually represented city model for various applications (such as maps, data, analysis, apps). Such complexity is required to cater for different domains, and heterogeneous user groups (universality). This development is paralleled by the ubiquity of geodata and geodata processing systems [8], facing experts and users alike with modern data and software engineering concepts, such as SOA/SOC or open source software [6].

To enable efficient utilisation of heterogeneous geospatial software systems and data pools, key challenges to build (transmit, harmonise, transform, combine) smart, integrated virtual city model as a digital twin of the urban environment – regardless of potential users' domains or expertise. Dedicated web-services are required to provide non-professional users with the necessary knowledge to make full use of potential to present such higher dimensional information spaces [6, 15].

1.1 About the previous report

The previous report in spring 2018 dealed with the simple integration of initial services to real-world problems. Thereby a simple seen but more complex task is two combine heterogeneous geodata from different origins, but identical area coverage. Concept and procedures were aligned with the framework of the Open Geospatial Consortium (OGC). An issue in this context was to assess if existing standards could be applied here or a specific service standard will have to be defined. Another issue was related to how services can be linked in an intelligent way. The previous main research focus was how intelligent services are transferable to different application

fields in a generic way and what kind of granulation therefore is needed. The aspect of domain-less services was under consideration due to a generic conceptualisation.

1.2 About this report

This report will give a summary about the application of the concept based on a real-world scenario of a virtual city model. This city model is the preparation for an smart digital twin of an urban environment. At the outset a fundamental view on the terms Digital Twin and Digital Model will be pointed out, while this outset highlights the access to the subject matter in general. Afterwards the case of application is described while important aspects are emphasised. Finally the past research activities were summarised and a short outlook will been given.

1.3 Digital Twin or Digital Model?

The scientific disciplines Computer Graphics and Geoinformatics differ. In contrast to geoinformatics, the Digital Twin in Computer Graphics is seen as the most accurate image of a real-world object. However, in geoinformation science the focus is always on the digital model, whose level of abstraction serves as a basis for a wide variety of applications. They differ in the degree of similarity defined and required by a particular application [4].

The Digital Gemini, on the other hand, is usually the most accurate image of reality, although it represents only a model of it, due to possible fine granular differences to the reference, the reality. Therefore, Digital Twins are also called “Look-a-Like”, “Feel-a-Like”, “Work-a-Like”, since the focus is often not solving an application problem, but much more expressing reality in certain forms [18].

The functions of a digital twin are usually limited to a few individuals [17]. While the spatial reference and application context usually plays a minor role. The Computer Graphics has a technical application reference, as an example the calculation of lighting in a rendered scene of different objects or as a technical limitation factor in the process flow. Geoinformation science is application-oriented due to its domain. Thus, each digital modelling, the degree of similarity, and abstraction grade is defined by the application itself, research question and target group, as an example a spatial planning application.

Against Digital Twin as a model, on the other hand, has a very low modelling depth in terms of classical spatial applications, because its abstraction level is very low. The realization level is very high, therefore it is realistic, although the focus still is on the technical processing and presentation [1]. Nevertheless, this is a limiting factor to that degree as the limit of perception in terms of similarity to reality is undershot [4]. However, for a large part of the applications this is technically conceivable but irrelevant, as it does not help to represent or solve a problem in a better way [7].

At this point, the different approaches of the mentioned scientific disciplines becomes apparent, although both use the example of the virtual city model to work on the same application. Viewed on a continuum that reflects iconicity [10]. Both disciplines approach each other bipolarly, but always with different foci of modelling

Figure 1: An urban agglomeration. Source Wikimedia: 2018.

and "look-a-like". However, there are points of contact that constitute the basis of this research. But without further effort, those views on the same object or application cannot be combined. Certainly, by combining various concepts, techniques, and processes in a complementary way, the application that is based inter alia on spatial data can benefit [6].

2 Approach

As mentioned the use case scenario for the adaptation of web-services in a smart process chain is a virtual city model for a Middle Eastern metropolitan city. The conceptual work will be applied to a current R & D project base. In this project, different user groups with different geospatial expertise collaborate to acquire and process geodata to build a smart 3D model of an urban agglomeration for exploration, analysis and decision making. Figure 1 gives an impression of potential granular object categories a smart 3D model has to cope with. Benefits and limitations of servicification [22] in the geospatial domain will be demonstrated by discussing, firstly, the service-based generation of a geospatial database from a variety of heterogeneous sources [6], secondly, any further analytics and provisioning [20], and thirdly, visualisation modules as well [9].

2.1 Use case objectives

In this context, the objectives discussed here are to build an integrated database for a Middle Eastern urban agglomeration. Modelling such agglomerations, the city centres of which constitute a highly developed form of densely integrated urban structures, requires a variety of geospatial data with different geometric, semantic and temporal features and granularity [3, 12]. Dealing with such complex spatial structures calls for efficient and effective handling systems [13, 14]. The architecture, following the concerns of service-oriented systems, is designed to address requirements, focusing on processing scalability (performance) and interoperability (parametric polymorphism) [5]. A modular process chain implements generic processing strategies to speed up individual tasks and facilitate the distribution of those tasks alongside corresponding datasets within a distributed infrastructure for data storage [6]. The integrated geospatial database mentioned above serves as a basis for the generation of a smart 3D city model with GIS functionalities. The database is therefore populated with a range of data from heterogeneous sources, such as remote sensing imagery, geotopographic base data, building-related data, various thematic data stocks, etc. The city model is conceived as a flexible, service-based, modular structure, allowing for a variety of applications and analyses, such as map generation, locational and network analysis or planning and citizen participation instrument. In short, the 3D city model provides an intuitive, transparent, and informative decision support system for local communities and stakeholders, i.e. heterogenous user groups. Proper use of the system requires considerable knowledge and skills on the part of local administrators, planners, decision makers, interested citizens and their associations. However, most of the targeted users as well as a number of the collaborating partners do not have a domain-specific background or professional expertise. Besides the characterisation of users it is important to highlight the characteristics of existing heterogeneous data pools as well.

2.2 Multidimensional Geodata

What does multidimensional mean? In many disciplines, two-dimensional data sets are the ones to chose. While two- and higher-dimensional data sets are multidimensional, the term multidimensional tends to be applied only to data sets with three or more dimensions. Normally when thinking about multi-dimensional data they are characterised by time or multiple phenomena obtained [2]. Typically the geographical dimension is limited to 3 + 1 D (spatially and temporally) [21]. In fact combining time series and cross-sectional data can be thought of as daily issues in the geospatial domain. But often these analyses or standard tasks are limited to one dimension only. As an example, weather forecast data sets provide forecasts for multiple target periods, conducted by multiple forecasters or automated models, and made at multiple horizons [21]. The multiple dimensions provide more information than can be gleaned from two-dimensional data sets. Therefore, multidimensionality has two meanings which both are important but require different strategies. The tricky task is to combine both in an flexible process chain to take advantage of

different perspectives of one and the same object. The solution are partly reverse engineering techniques where key facts are:

1. Multidimensional database based on various resources
2. Reverse engineering
   find identical features in different scales
   harmonise geodata
   homogenise geodata
3. Enabling geoprocessing at runtime

2.3 Requirements for Digital Modelling and Data Integration

The geographic dimension or the scale of geographic research includes a specific spatial order of the objects studied and a certain scale-adequate intensity in the analysis and synthesis. The equality of methods used and the same intensity of the exploration are reflected in the selection of parameters for characterization of objects (natural space or landscape space and their components [20]. Following this theory, within a geographical dimension, the objects under investigation are strictly comparable, the methods used have the same degree of exactitude about the nature of objects and these are described at the same level of abstraction [16]. Indeed models, as simplified representations, come in many different "flavours" and are defined as a schematisation of reality with operational potentials [11]. This points to the intimate relationship between the type and degree of schematisation and the objectives for the call on the operational potentials. It is targeting to scale down and convert reality to a form which is comprehensible. Hence, a model is a manageable, comprehensible and schematic representation of a piece of reality [19].

Technically the modelling can be utilised "rule-based" and "constraint-based". Whereas "rule-based" includes an expert systems legacy ("if something then action") with a blind end. The generalisation has a high level of complexity and it is not easy to identify rules because of its rigidity. This is a not exactly generalised modus ponens, and therefore the reverse engineering approach is called, because there are no generic non-inventoried objects. In contrast the "constraint-based" approach can be broken down to a set of numerical conditions (size, distance, ...) but there it is impossible to satisfy all. It is a kind of an optimization as a synthesis of conditions. Flexibility is granted due to the method choice. But this approach is more result-oriented in case of the application domain. A combination of both approaches is the result due to holistic software-agents. Each software-agent is represented as a generic service and exemplified in 2. To take advantage of geodata's multidimensionality it is needed to explicitly define objects and it relations. Thus, it follows an incremental model and is therefore complex as well as complicated to implement in a generic way. This is described as different services of "amplified intelligence" or geospatial intelligence [6] and illustrated as follows:

1. Data Distribution

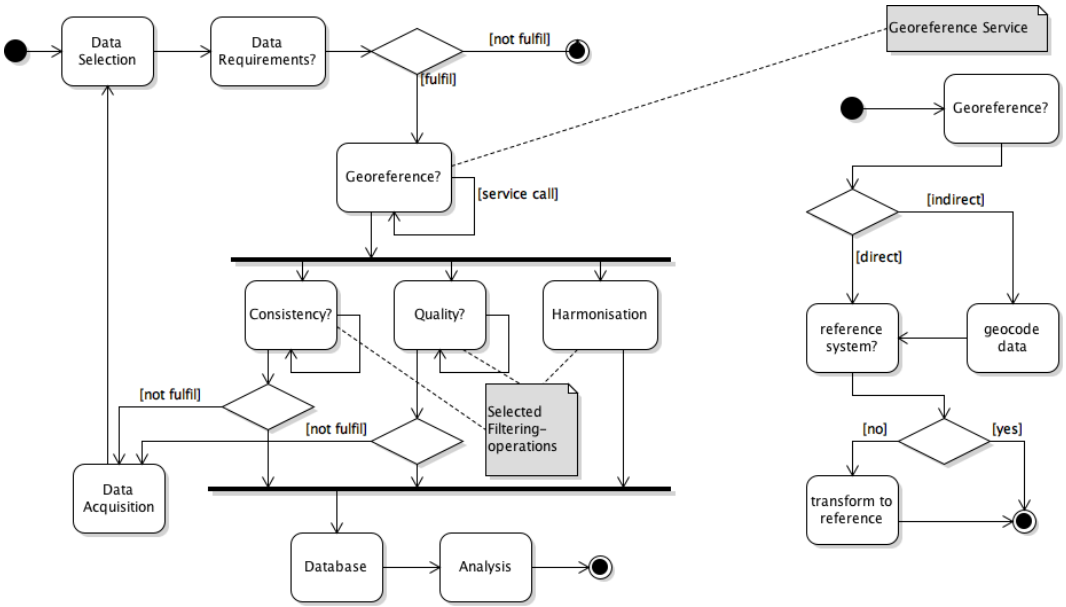


Figure 2: Schematic overview of the concept structure

Access, Shaping, Characteristics

2. Data Integration

   Structure Recognition

   Congestion, Coalescence, Complication, Inconsistency, Imperceptibility

3. Data Modification

   Spatial Transformation and Semantic Enrichment

   Incremental Adaptation or Generalisation

4. Data Analytics

5. Data and Service Provision

   Rule- or Constraint-based

   Concatenating Applications

6. Quality Assurance

   Maintain Accuracy, Consistency, Logical Hierarchy, Cartographic Quality

   Computational Issues, Algorithm Costs, Maximum Data Reduction

2.4 Generalisation of multi-criteria geodata as Data Integration

Generalisation is known and needed for many different tasks, domains and applications unless its fundamentally equal meaning could be described as simplification or abstraction to get in the end a more universal insight/evidence for an inspected

behaviour. The process encompasses in general the reduction of complexity, the emphasis of essential and suppress unimportance, the maintenance of relationship among objects while preserving a defined quality [4]. Not least it is a process of semantic and numeric resolution reduction. Quality is seen as a selection and simplified representation of a detail, appropriate scale and/or purpose of an application, different processes deriving from data source through the application of transformations, whereas derivation objectives are to reduce data in scope of amount, type and cartographic portrayal with maintenance of consistency and clarity of the presentation. Thus, the term generalisation is separated into two main parts [20]. **Model Generalisation** as a new conceptualisation of phenomena (mostly extending qualitative) and **Visual Generalisation** as a new structural phenomenon representation (mostly quantitative). However, both generalisation types aim to isolate the characteristics that are independent to both feature and domain resolution. Furthermore it defines the capture in a software architecture as well. The proposed software architecture allows to instantiate this method as a web-based service system. That means users – regardless of their expertise level or domain affiliation – could adapt the system to their specific (micro-)task or workflo [6]. It is designed to enable a complete process chain split as micro-tasks. The architecture will be designed to satisfy two non-functional requirements: scalability (vertical and horizontal) and adaptability (at deployment and at run-time). The future work includes experiments to evaluate these two requirements in terms of quality assurance and performance issues.

3 Research Activities

Since joining the Research School in October 2017 I have been able to benefit from the versatility of specialist groups at HPI and regular events, such as the last fall retreat in 2017 and the Symposium on Future Trends in Service-Oriented Computing in spring 2018. The interdisciplinary perspective is very important and helpful especially in the geodomain, because of its mentioned interdisciplinary character. To conceptualize and sharpen my dissertation project, I can already count on a contribution in the German geo-community (DGFK, DGPF, Runder Tisch GIS) and the international branch of growing Geo-IT Experts in the International Cartographic Association (ICA) in terms of cartographic relevance and the informatics community of service-oriented systems engineering. It has been shown that geo-servicification in general is on the rise, but not yet a research topic, because of its continuum sphere demonstrated within the IT-disciplines. The next steps are to highlight the domainless aspect of the approach described to cover up the mentioned continuum and to bridge the application focus of this conceptualisation. It is planned to consider methods and techniques of this approach in the community as well at international conferences and paper contributions.

References

[1] T. Akenine-Moller, E. Haines, and N. Hoffman. *Real-time rendering*. AK Peters/CRC Press, 2018.

[2] D. Asimov. "The grand tour: a tool for viewing multidimensional data". In: *SIAM journal on scientific and statistical computing* 6.1 (1985), pages 128–143.

[3] I. R. M. Association et al. "Geospatial research: concepts, methodologies, tools and applications". In: *Information Science Reference, Hershey, PA* (2016).

[4] S. Boschert and R. Rosen. "Digital twin – the simulation aspect". In: *Mechatronic Futures*. 2016, pages 59–74.

[5] T. Erl, P. Merson, and R. Stoffers. *Service-oriented architecture: analysis and design for services and microservices*. Prentice Hall, 2017.

[6] A. Fricke, J. Döllner, and H. Asche. "Servicification – Trend or Paradigm Shift in Geospatial Data Processing?" In: *Computational Science and Its Applications – ICCSA 2018*. 2018, pages 339–350. ISBN: 978-3-319-95168-3.

[7] M. F. Goodchild. "A GIScience Perspective on the Uncertainty of Context". In: *Annals of the American Association of Geographers* (2018), pages 1–6.

[8] M. F. Goodchild. "Twenty years of progress: GIScience in 2010". In: *Journal of spatial information science* 2010.1 (2010), pages 3–20.

[9] D. Hildebrandt and J. Döllner. "Service-oriented, standards-based 3D geovisualization: Potential and challenges". In: *Computers, Environment and Urban Systems* 34.6 (2010), pages 484–495.

[10] I. Hinterwaldner. *Das systemische Bild: Ikonizität im Rahmen computerbasierter Echtzeitsimulationen*. Wilhelm Fink, 2010.

[11] Y. Iwasaki and H. A. Simon. "Causality and model abstraction". In: *Artificial intelligence* 67.1 (1994), pages 143–194.

[12] H. A. Karimi. *Big Data: techniques and technologies in geoinformatics*. CRC Press, 2014.

[13] R. Kitchin. "Big Data, new epistemologies and paradigm shifts". In: *Big Data & Society* 1.1 (2014), pages 1–12.

[14] J.-G. Lee and M. Kang. "Geospatial big data: challenges and opportunities". In: *Big Data Research* 2.2 (2015), pages 74–81.

[15] A. M. MacEachren and M.-J. Kraak. "Research challenges in geovisualization". In: *Cartography and geographic information science* 28.1 (2001), pages 3–12.

[16] J. F. Roddick, K. Hornsby, and D. de Vries. "A unifying semantic distance model for determining the similarity of attribute values". In: *Proceedings of the 26th Australasian computer science conference*. 2003, pages 111–118.

[17] R. Rosen and S. Boschert. *Modellbildung und simulation*. 2017.

[18] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui. "Digital twin-driven product design, manufacturing and service with big data". In: *The International Journal of Advanced Manufacturing Technology* 94.9-12 (2018), pages 3563–3576.

[19] J. Thatcher, A. Shears, and J. Eckert. *Thinking big data in geography: New regimes, new research*. Univ. of Nebraska Press, 2018.

[20] V. Tolpekin and A. Stein. *The core of GIScience: a systems-based approach*. University of Twente, Faculty of Geo-Information Science and Earth Observation (ITC), 2012.

[21] P. Van Oosterom and J. Stoter. "5D data modelling: full integration of 2D/3D space, time and scale dimensions". In: *International Conference on Geographic Information Science*. 2010, pages 310–324.

[22] S. Vandermerwe and J. Rada. "Servitization of business: adding value by adding services". In: *European management journal* 6.4 (1988), pages 314–324.

Understanding Sources of Heterogeneity in SMP Systems

Andreas Grapentin

Operating Systems and Middleware
Hasso-Plattner-Institut
andreas.grapentin@hpi.uni-potsdam.de

There is a disconnect between the behavior of modern SMP systems and the expectations of application programmers and system software routines about the behavior of these systems. One of the more notable manifestations of this disconnect are the punishing effects of the non-uniform memory access (NUMA) architecture on the performance of unaware applications, that operating systems are still not able to mitigate satisfyingly. Another example is the introduction of the *big.LITTLE* architecture on mobile devices, in order to balance the requirements of efficiently handling compute load spikes while maintaining low power consumption. Although the big.LITTLE architecture has shown to be a success, unmodified schedulers of general purpose operating systems are not yet capable of utilizing the strengths of the heterogeneous compute resources.

At the core of this phenomenon seems to be the notion that SMP systems are homogeneous structures of memory resources and compute resources, which leads to numerous assumptions that are made by system software and application developers that do not hold on real hardware. I have identified four sources of heterogeneity in SMP systems, that are discussed below in terms of their purpose, as well as their impact on the behavior of applications, and what utilization or mitigation strategies exist in systems software and operating systems.

1 Symmetric Multiprocessing

The term *Symmetric Multiprocessing* (SMP) describes a type of multiprocessing system in which concurrent units of work (*Processes*) are executed in parallel, distributed on a homogeneous set of independent processing units. The processes are assigned slices of runtime on the processing units by the *scheduler*, which is a component of the operating system that balances and orders the workload in a way that is optimized for certain criteria such as throughput, responsiveness and fairness.

Virtually the only types of multiprocessing system supported by todays generation of general purpose operating systems – such as Windows or GNU/Linux – are SMP systems. This is due in part to the constant and continued ubiquitousness of SMP systems and their ability to scale up and get a lot of work done quickly, and also in part to the three decades these operating systems and their predecessors have had to be adapted and optimized for efficient operation in multiprocessing environments.

Specialized versions of UNIX supported SMP Systems as early as 1984, motivated by the arrival of the *Balance 8000* and *Balance 21000* machines [4], and the necessity for the pivotal turn towards ubiquitous multiprocessing and SMP has been described by Herb Sutter in 2005 in his article "The Free Lunch is Over" [6]. Today, while

SMP is deeply rooted in all areas of computing, from IoT devices to server blades, the further advancement of technology has introduced aspects of heterogeneity to SMP systems, that have proven to be detrimental to the performance and stability of unaware software.

2 Sources of Heterogeneity

The homogeneity assumptions of SMP – and the decades of adaptation and optimization of SMP-capable scheduling algorithms to this assumption – are disconnected from the actual structure and the behaviour of real hardware, where technological advancements or specific problems have necessitated the introduction of heterogeneity in the memory- and compute resources of the system.

In following, as outlined in Figure 1, are described four common sources of heterogeneity in SMP systems, two of which focus on the systems main memory latency and throughput characteristics, and two of which outline ways in which the compute resources in a system can be heterogeneous.

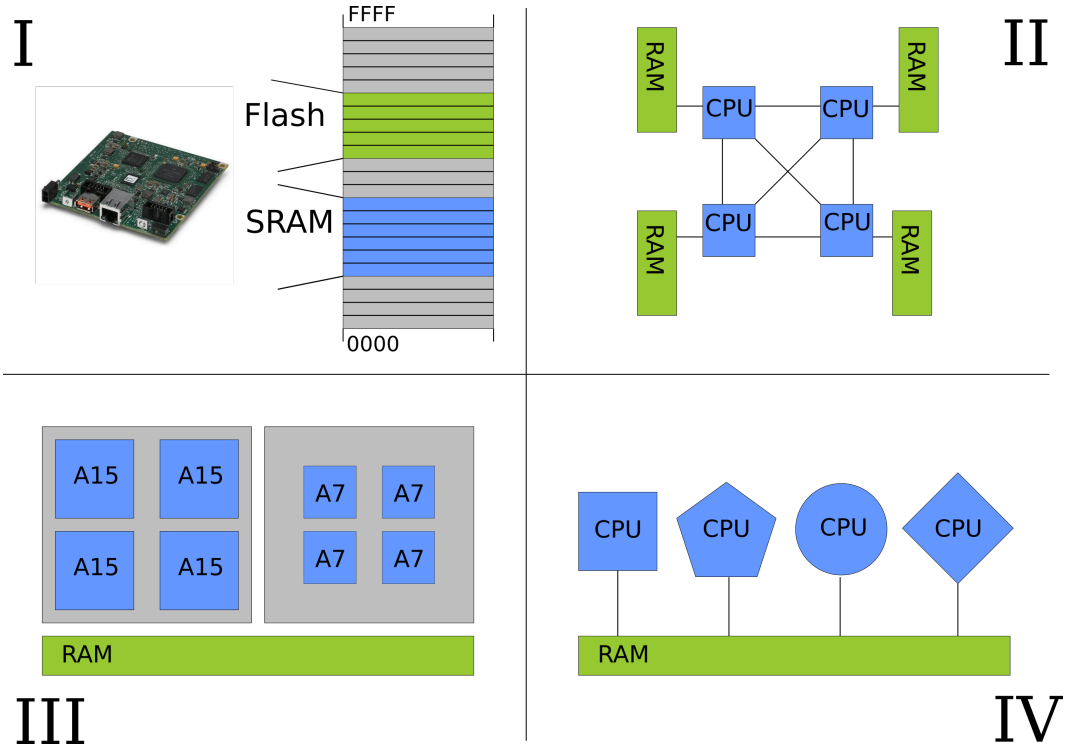


Figure 1: Four sources of heterogeneity in an SMP system. I – non-uniform memory segments caused by hardware properties. II – non-uniform memory access topologies. III – non-uniform CPU capabilities and big.LITTLE systems. IV – non-uniform CPU ISA.

2.1 Non-Uniform Memory Segments

To the application, the main memory available in the system appears as a continuous, byte-addressable array of numbers that is homogeneous in regards to access latency and throughput. However, in reality the access latency is hard to predict due to the effects of caching. Additionally, the physical address space of a system is only partially backed by physical memory modules and hardware registers.

If memory modules of different technologies were used to provide different segments of physical memory, then the latency and throughput of memory access in the system would depend on the physical properties of the backing module of the given target address, which would introduce heterogeneity in the memory access.

This effect, while benign, is especially common in embedded devices, where different memory technologies, such as *Flash* and *SRAM* are utilized in significant amounts and generally available in the same address space.

On devices that support an operating system, this problem is typically solved through a virtual memory abstraction, where the different memory regions are used for different purposes and only the *SRAM* is typically utilized by the operating system for dynamic memory allocation by the application, partially restoring homogeneity of the address space available to the application. On deeply embedded devices without an operating system, virtual memory abstractions and dynamic memory allocation methods typically do not exist and the application operates on physical addresses. In this case, the developer is responsible for placing variables in a suitable memory region.

2.2 Non-Uniform Memory Access Topologies

Assuming a set of uniform memory modules in a given system, heterogeneity is still not impossible. In fact, the majority of multi core SMP systems adhere to the *Non-Uniform Memory Access* (NUMA) Architecture to combat the effects of the *von-Noumann bottleneck*.

In the NUMA Architecture, the processing units in a system are linked through a network of interconnects, and the physical memory modules are each attached to a separate processing unit, as opposed to a global shared memory bus. This has the advantage that a processing unit has full and undisturbed access to its local memory, and is not limited by the congestion of a global shared bus, meaning systems can scale up further.

This scale up is in turn limited by the overhead of maintaining *cache coherence* among the processing units, which requires a constant communication on the interconnect network. Additionally, accessing memory not local to the executing processing unit invokes a communication through the interconnect network and is orders of magnitude more expensive than a local memory access.

As a result, the latency and throughput of a memory access depends on the distance of the originating processing unit to the memory module containing the target address in hops, as well as the congestion of the interconnect network.

This is a challenge for operating systems, since the scheduler as well as the virtual memory manager need to be aware of the NUMA topology of the underlying hardware, as well as the projected memory access and compute resource utilization behavior of the application, to be able to make optimal work and data placement decisions in the system. This information is not known a-priori, and may be difficult to gather at runtime, depending on the actual system. Estimating application behavior to control future placement strategies is not straightforward, and optimal placement might not even be possible, or may be futile if a later task migration destroys the maintained locality.

The field of NUMA penalty mitigation is a vast area of active research, with varying strategies being implemented and evaluated, and constantly improved upon. One strategy to mitigate the performance adverse effects of the NUMA architecture is implemented by `numad` on GNU/Linux. `numad` is a system daemon that will monitor task behavior and data access to incrementally improve the placement of tasks and core affinity. Other approaches try to dictate new programming models to help developers create NUMA aware applications through explicit language constructs [3], and many more techniques and approaches exist.

In the context of the *Shared Something* architecture, an effort has been made to generalize the properties of the NUMA Architecture and to allow the specification of arbitrary interconnect networks between compute and memory components in the system through introduction of the *Generation Z* architecture specification [2], as illustrated in Figure 2.

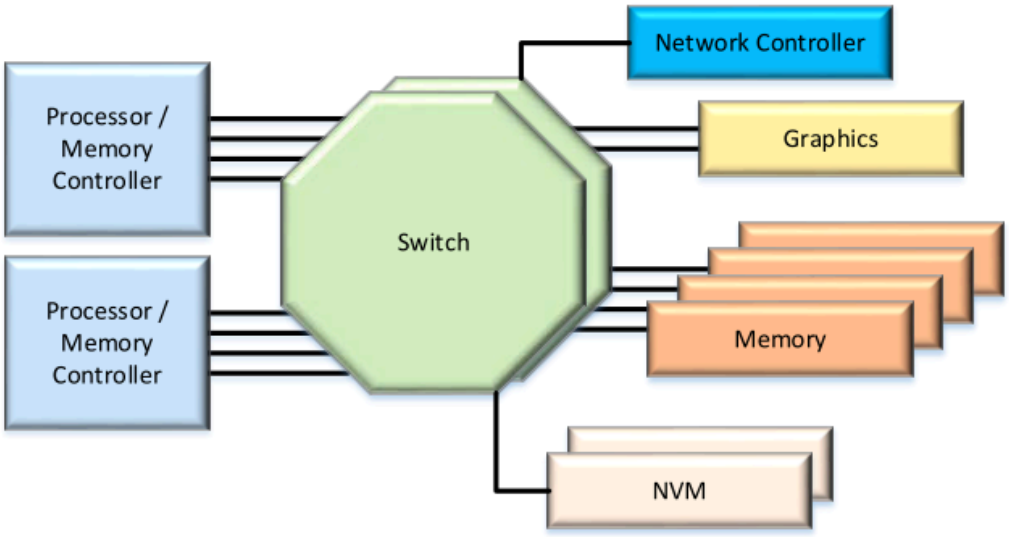


Figure 2: The *GenZ* Architecture specification generalizes the NUMA Architecture and provides a means to describe memory access topologies of arbitrary complexity.

2.3 Non-Uniform CPU Capabilities

The processing units in an SMP system can be heterogeneous in ways beyond their connection to segments of the main memory. While still maintaining binary compatibility, they can differ in their functional and non-functional capabilities, such as power efficiency, instructions per second, or the availability of functional units such as vector- or floating-point units. This type of heterogeneity is less common, but such systems do exist.

For example, handheld devices such as mobile phones need to conserve energy during phases of low activity to extend their battery life, while also being able to process short to medium bursts of high compute load during user activity. These scenarios pose conflicting requirements to the hardware, where smaller, more power-efficient processing units would benefit battery life through the long phases of low activity, while bigger, more performance focused processing units would improve the peak performance of the device during the short bursts of high activity.

To meet these conflicting requirements, ARM has introduced the *big.LITTLE* architecture in 2011, where different types of processing units – a cluster of energy efficient cores, and a cluster of performance optimized cores – are combined on the same chip, allowing the system to switch between energy efficiency and peak performance at will, in three possible modes of operation.

The first two modes of operation only ever utilize half of the available cores.

Clustered Switching In *Clustered Switching* the operating system can choose to utilize either the complete cluster of energy efficient cores, or the complete cluster of performance optimized cores. The active cluster can be switched at any time, resulting in all active tasks to be migrated to the activated cluster.

Regardless of which cluster is active, within the active cores the homogeneity is maintained and SMP scheduling can proceed as usual. Switching between the clusters only switches the entire system to power-saving, or to high-performance mode.

In-Kernel Switching In *In-Kernel Switching* or *CPU Migration*, the clusters are split up into pairs of one power-saving core and one performance optimized core each. At runtime, in each pair only one core is active and the operating system can switch between them, resulting in the active task on that pair of cores to be migrated. This allows to selectively enable a process that is compute heavy to complete faster, while still keeping the majority of the system in a power-saving state and vice versa, allowing for more fine grained control.

In this mode, the active cores are not necessarily homogeneous, and a combination of power-saving and performance optimized cores may be active simultaneously. However, while the active cores may be heterogeneous, the available compute resources are homogeneous since every pair of cores is equal. This means that placement decisions do not matter, since the scheduler can simply choose to enable a high performance core for a task that needs it.

Table 1: Specifications of the Odroid-XU4 test system

	Odroid-XU4
Processor	Samsung Exynos5422 Cortex-A15 2Ghz and Cortex-A7
Memory	2GB LPDDR3 RAM PoP stacked
Operating system	modified Ubuntu Linux 16.04

Heterogeneous Switching In the *Heterogeneous Switching* mode, the operating system has access to all cores simultaneously and no cores are switched off, resulting in an SMP configuration with truly heterogeneous compute resources.

This poses a challenge to the systems scheduler, since tasks will behave differently on the different cores. A compute intensive task will take longer to complete on a power-saving core, while a memory intensive task will waste cycles waiting for data when running on a performance optimized core. Additionally, activating a performance optimized core in addition to its corresponding power-saving core puts stress on the system, and should be done sparingly. In this scenario, in order for the scheduler to fulfil its goals of optimizing throughput and system efficiency, it needs to take into account the projected behavior of the tasks to be scheduled when making a placement decision. Additionally the scheduler becomes responsible to answer a new dimension of questions, that is which cores to keep active, and which cores to turn off, if any.

In the mainline Linux kernel, there is currently, as of version 4.18, no support for these scenarios, and the scheduler will treat all cores equally, resulting in a suboptimal schedule and high stress on the system. As a result, attempting to run all cores under high load on the mainline kernel will lock up the system reliably. This has been shown by running a set of virtual machines on an *Odroid-XU4* as a testbed for the *Fabric-Attached Memory* emulation [1] as shown in Figure 3 and the details of the testbed are listed in Table 1.

The virtual machines were pinned to individual cores to investigate the behavior of heterogeneous shared-something computing, where otherwise independent virtualized systems running on heterogeneous hardware have access to a shared memory segment.

While there is currently no consensus on how to proceed with these problems in the mainline kernel, there are several out-of-tree attempts at implementing support, one of which is headed by Qualcomm, and another by Linaro, but both still have shortcomings and are not likely to be included in the mainline linux kernel soon. Both approaches try to sample the behavior of running tasks to distinguish between compute dominant and memory access dominant tasks, and also provide a concept of big and small tasks, and important and unimportant tasks, that should allow for more informed scheduling decisions. However, in practice these sampling approaches are expensive to perform and do not adapt well to tasks with switching runtime profile. Both approaches also include a load balancing algorithm to try and reduce the system load if a cluster is overloaded and in danger of overheating.

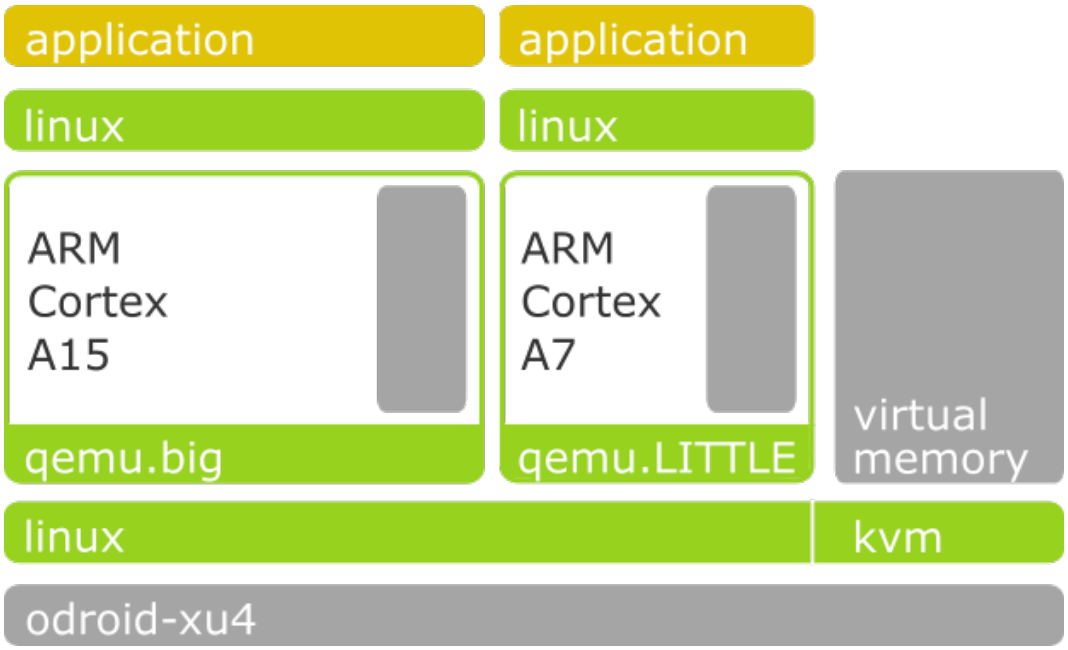


Figure 3: A set of virtual machines on an Odroid-XU4 demonstrating the properties of the fabric-attached memory on the big.LITTLE heterogeneous platform

However, these types of scheduling shortcomings might be typical to monolithic operating system kernels and other types operating systems, such as the *multikernel operating system* Barrelfish might be better suited for heterogeneous SMP systems due to their inherently distributed nature [5].

In 2017, ARM have announced the successor of the big.LITTLE architecture, called *DynamIQ*. DynamIQ relaxes the homogeneous cluster requirement imposed by the big.LITTLE platform and allows any combination of up to 8 power-saving and up to 4 performance optimized cores per cluster, and an arbitrary amount of clusters per chip. It remains to be seen what configurations of this architecture are being implemented by hardware manufacturers and how scheduling approaches will be able to evolve to accommodate for these configurations, while still being universal, and imposing a manageable overhead.

2.4 Non-Uniform CPU Instruction Set Architecture

The processing units in a system can be even more diverse than described before, if the requirement of binary compatibility is dropped. Binary compatibility between a set of processing units and a *program image* is a property that describes that the program image compiled once from a program can be executed on either of the processing units without change. This requires the processing units to understand the same bytecode, or to have the same *Instruction Set Architecture* (ISA).

Dropping this requirement means that the processing units in a system can be truly heterogeneous, arbitrarily connected to the systems main memory, and capable

Table 2: Specifications of the parallella test system

	parallella
Processor	ARM Cortex-A9
FPGA	ZYNC Artix-7
Accelerator	16-core Epiphany RISC SOC
Memory	1GB SDRAM
Operating system	modified Ubuntu Linux 16.04

of executing arbitrary program images. This type of heterogeneity is very common, albeit not usually associated with SMP scheduling. All kinds of accelerators, such as *General-Purpose GPUs* (GPGPUs), FPGAs and others can be counted into this category. Generally, these sorts of devices are not controlled by the system scheduler, but by applications acquiring dedicated access, and provisioning special program images made for the accelerator for execution, sometimes even written in special programming languages. In the context of a GPGPU, this program image would be a *shader*, or an *OpenCL Kernel*, deployed by an application directly to the accelerator.

Scheduling tasks on a heterogeneous cross-ISA SMP system is challenging. The program images could function only on a subset of the available cores, and the available cores could only process a subset of the available workload. To limit the possibility of starvation due to overload of a certain type of core, *fat binaries* can be utilized, effectively allowing a single binary to function on more than one type of the available cores. This results in a possible n-to-m mapping of available tasks to available cores that a scheduler would need to accommodate at runtime. A *task broker* would be needed to match the capabilities of the available compute resources to the requirements of a given task.

In order to investigate the behavior of this type of SMP system, I created a prototypical system by deploying a *RocketCore riscv64* softcore CPU to the FPGA of a *parallella* SoC board, as shown in Figure 4 and the specifications of the parallella board are listed in Table 2. For SMP scheduling to function on such a platform, the task model of the Linux kernel needs to be extended to accommodate for the differences in ISA support of a program image and its process. Cross-ISA task migration could also be conceivable.

3 Conclusions

Heterogeneity in SMP systems is unavoidable and can cause major problems in system performance and stability, if the system software and the application are unaware of it. For example, managing the NUMA characteristics of the hardware in a transparent and portable manner is still challenging, and newly emerging systems supporting heterogeneous processing unit configurations have yet to be recognized and fully supported by leading general purpose operating systems. With the growing

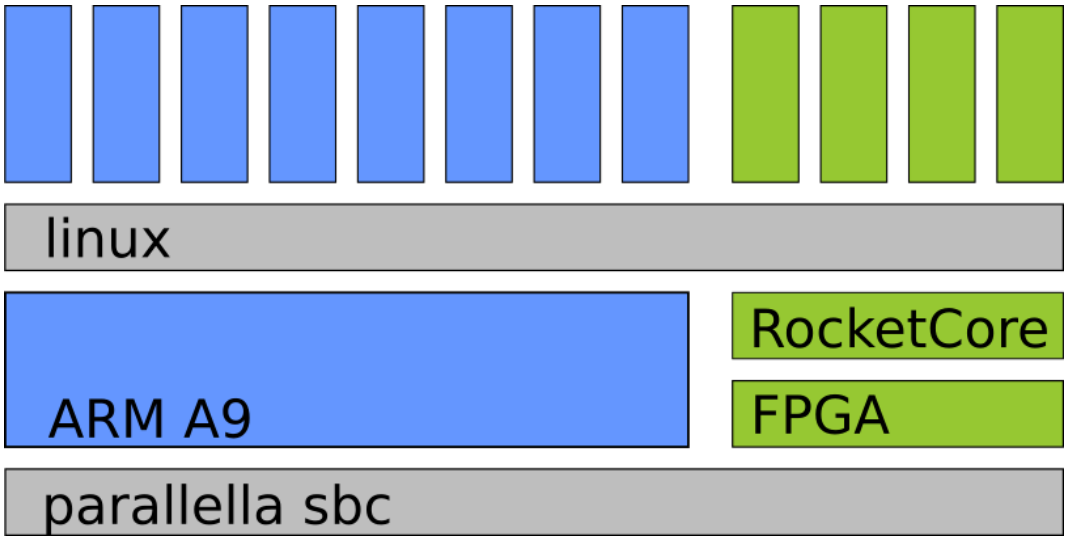


Figure 4: A RocketCore deployed as softcore CPU onto on FPGA in a parallella SoC configured as an SMP system

complexity of heterogeneous characteristics of announced future architectures, such as DynamIQ, these effects are only going to get stronger.

Heterogeneity is embraced by hardware manufacturers as a solution to scalability problems, or energy efficiency problems, or as a means to solve one very specific set of tasks, but this introduces new challenges in the system software or in the application that are not sufficiently addressed.

I have identified four major sources of heterogeneity in SMP systems, and outlined their impact on the systems they are a part of, as well as some of the mitigation strategies that currently exist to reduce the problems they introduce.

4 Future Work

I would like to revisit the big.LITTLE architecture and propose an efficient scheduling algorithm for SMP systems, and extend the linux kernel with an implementation of this algorithm. I would also like to investigate how multikernel operating systems such as Barrelfish handle the pitfalls of heterogeneous compute resources.

I also want to extend the cross-ISA prototype to successfully schedule tasks of different binary architectures and study their behavior in order to try to implement a task broker and a concept of fat binaries on a single system.

Additionally, I want to further my understanding of the Generation Z architecture specification and look into the specifics of the DynamIQ architecture proposed by ARM, and see how the heterogeneous SMP scheduling algorithms proposed for big.LITTLE perform on DynamIQ.

References

[1] R. Craig. *Emulation of Fabric-Attached Memory for The Machine*. 2017. URL: https://github.com/FabricAttachedMemory/Emulation (last accessed 2018-09-03).

[2] Gen-Z Consortium. *Gen-Z Draft Core Specification*. 2017.

[3] W. Hagen, M. Plauth, F. Eberhardt, F. Feinbube, and A. Polze. "PGASUS: a framework for C++ application development on NUMA architectures". In: *Computing and Networking (CANDAR), 2016 Fourth International Symposium on*. 2016, pages 368–374.

[4] R. W. Hockney and C. R. Jesshope. *Parallel Computers 2: architecture, programming and algorithms*. Volume 2. CRC Press, 1988.

[5] A. Schüpbach, S. Peter, A. Baumann, T. Roscoe, P. Barham, T. Harris, and R. Isaacs. "Embracing diversity in the Barrelfish manycore operating system". In: *Proceedings of the Workshop on Managed Many-Core Systems*. 2008, page 27.

[6] H. Sutter. "The free lunch is over: A fundamental turn toward concurrency in software". In: *Dr. Dobb's journal* 30.3 (2005), pages 202–210.

Data-Knoller: A Framework for Systematic Data Preparation

Lan Jiang

Information Systems
Hasso-Plattner-Institut
lan.jiang@hpi.uni-potsdam.de

Raw data comes in many shapes and forms, most of which are not what a data engineer, data scientist, or an analytics tool desires. The tedious task of preparing data dominates time and effort spent in data science and on data analytics projects. Its ad-hoc nature also impedes transparency and reproducibility of analyses and experiments. We lay out a path for a systematic framework to specify and execute a comprehensive set of operations related to data preparation. To this end, we propose a formal and comprehensive taxonomy of preparation operators based on metadata extracted from data at every stage of the preparation process. On top of the extensible framework, we propose many useful components, such as optimizing the pipeline of preparators, suggesting useful preparation steps, or even automatically inferring an entire preparation process given some input data and a desired output format as prescribed by subsequent analytics or other processing tasks.

1 Overview

1.1 Motivation

Data preparation is the process of obtaining, cleaning, transforming and storing raw data before serving them to downstream applications, such as data analysis, machine learning, and visualization. Coming from various sources, raw data do not always meet the requirements of the following applications, leading users, including both expert data scientists and novice data workers, to frequently conduct data preparation tasks. It is reported that preparing data is a labor-intensive yet tedious work for users, and this phase accounts for 50 %–80 % of the time spent in the whole data analysis lifecycle [9, 19, 24]. In business scenarios, users spend a lot of time and budget to support their data preparation work, reducing resources that should be devoted to interesting downstream applications. In academia, scientists massage data in various ways and usually omit to document each step, leading to the non-repeatability of experiments and evaluations.

To create prepared data that are valid input for further applications, a common practice is to conduct many trivial but critical transformations on raw data, such as tuning field delimiter, reformatting strings, and conducting arithmetic calculation on specific fields. Building ad-hoc scripts to prepare such kind of datasets is a common practice. The drawbacks of these approaches are obvious. First, writing ad-hoc scripts requires certain programming skills, which many analysts do not have, and implementing these solutions possibly produces large time expenses, impairing

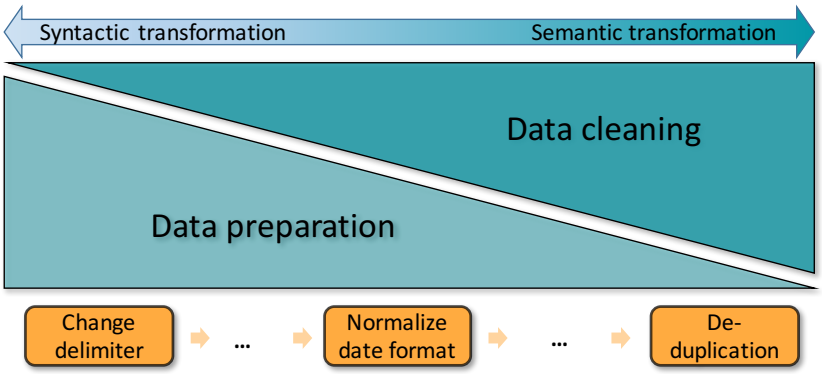


Figure 1: Data preparation versus data cleaning

the overall efficiency of data-driven businesses. Second, these scripts are specific preparation task oriented, and thus reusing them may be hard. Furthermore, ad-hoc scripts may not support the management of critical information, such as metadata, provenances, and preparation errors, which all contribute to execution robuesness and preparation repeatability.

1.2 Data cleaning vs data preparation

Data cleaning is a frequently used term for the activity to handle these data quality problems in the research community. However, it differs from data preparation in our context, although the border between them is blurry. Figure 1 indicates the explanation of their difference. Both are data transformation activities aiming at ensuring data quality. However, in this work, data cleaning refers to transformations that are more semantic (e.g., de-duplication), whereas data preparation comprises more primitive transformations (e.g., changing delimiter of a delimiter-separated file). Both types of transformations may appear at any position of a data transformation pipeline. Nevertheless, we expect data preparations to be applied more towards the beginning while data cleaning towards the ending stages.

2 Approach

To solve the above issues, we propose a framework for systematic data preparation, named Data-Knoller.[1] The framework aims to provide an easy-to-use platform for robust, efficient, and repeatable data preparation. It accepts input data in any supported format and data model, creates a data preparation task as a *preparation pipeline* of transformations on data, and eventually produces prepared data that are valid

[1]Knolling is the art-form of neatly arranging related objects in parallel or 90-degree angles as a method of organization.

Figure 2: A pipeline describing a data preparation task

input for further applications. We give the name *preparator* to each transformation operator in our context, and the pipeline considers each transformation as an individual preparation on data. Figure 2 illustrates an example of a preparation pipeline with seven individual preparators performed frequently by analysts.

We propose a comprehensive taxonomy of preparators to formalize the specifications of transformations, as well as a taxonomy of metadata that specifies the metadata employed in the system. Due to the space limitation, we do not display them in this report, but have placed them on our web page[2] Data-Knoller incorporates six components: a metadata manager, an error manager, a provenance manager, and a preparation suggester, a pipline optimizer, and a pipeline adapter. These components contribute to the robustness of results and efficiency of data preparation. The metadata manager is in charge of collecting, validating, and maintaining metadata during preparation processes. The correctness and consistency of metadata play a critical role for executability of preparators and generation of expected results. The error manager deals with preparation pipeline errors and data errors that possibly appear before and during the execution of data preparation tasks. In order to enable effective data preparation, the system shall be capable of telling users the positions of the exact errors occurred while executing preparators. Analysts may be interested in not only the data results produced by executing a pipeline, but also how the results were produced and where they originated. The provenance manager captures and stores provenance information produced by preparation pipelines while they are running. As we notice that users spend a lot of time figuring out what is the next transformation on data, we propose a suggestion component, which is based on available metadata and a specification of the expected result. The pipeline optimizer re-schedules the executing order of the preparators of a pipeline, ensuring that the execution is most efficient in terms of time and memory expenses. We propose a pipeline adapter to trace the evolution of data schema, and adapting the pipeline used to prepare data to the new schema accordingly.

2.1 Architecture

Figure 3 illustrates the architecture of Data-Knoller. It composes of a preparation pipeline that comprises a couple of preparators, along with six components that assist data processing, i.e. metadata manager, error manager, provenance manager,

[2] https://hpi.de/naumann/projects/data-quality-and-cleansing/data-preparation.html (last accessed 2018-10-18).

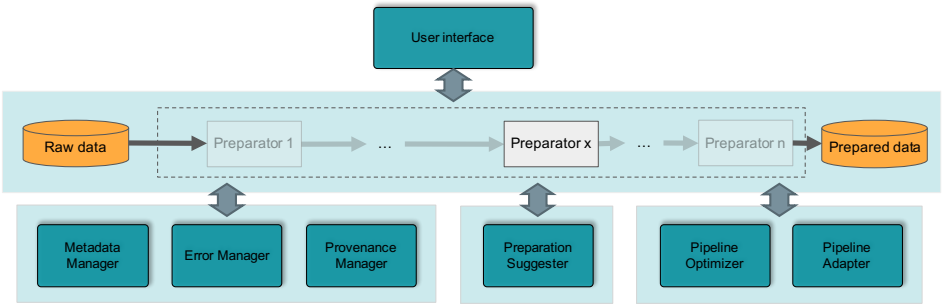


Figure 3: The architecture of data preparation framework

preparator suggester, pipeline optimizer, and pipeline adapter. In the rest of this section, we introduce the principles and details of these components.

2.2 Data preparation pipeline

The preparation pipeline is the main component of Data-Knoller. It composes of a chain of preparators, each of which performs a specific data transformation, such as changing the data type of a column or filling the empty slots of a column. Starting from the input data, each preparator consumes the output data from the last preparator as its input data, producing the output data for the next preparator. The pipeline eventually produces the data that meet the requirements of the downstream application.

2.3 Metadata manager

Metadata are inevitable and important for a data preparation pipeline, because these metadata maintain the properties of data that determine the executability of the preparators. Applying preparators without validating corresponding metadata may fail the transformations, whereas well-managed metadata ensure preparation pipelines produce expected results. For example, misinterpreting the delimiter of a data file may cause the truncation of values in some columns.

The metadata manager serves to manage the metadata and dynamically maintain them during the execution of a data preparation pipeline. It serves multiple functionalities, including generating, validating, and updating metadata as requested by preparators in pipelines. To this end, the metadata manager maintains a metadata coordinator, which communicates with the metadata generator, the metadata validator, and the metadata updater that perform the aforementioned actions, respectively. In order to maintain runtime metadata, the metadata manager additionally holds a metadata repository to store the created or updated metadata.

It is worth mentioning that metadata generator is extendable, because the performance of generating a single metadata could be largely diverse among different

implementations. The system provides users with default solutions to generate metadata, as well as the space to let users bring their own implementations.

2.4 Error manager

Data preparation tasks may fail in various ways, usually because preconditions of the preparator are not met by its input data. In many cases, locating which part of the data results in the failure of the data preparation task is overwhelmingly expensive. For example, when changing the data type of a column from string to double fails, users possibly need to spend a lot of time locating and analyzing the value on which and the reason why the preparation does not work. Therefore, In order to reduce this time overhead, an efficient and robust data preparation system needs to incorporate an error handler that can tell users the what and where is the error occurred while executing the preparation.

It can be assumed that the more fine-grained reason of an error the system gives users, the faster they can figure out how to fix it. Therefore, we define three levels of data preparation errors: *pipeline-level*, *metadata-level*, and *data-level* errors. Pipeline-level errors occur *during specification* of the preparation pipeline and are thus independent of the actual data. Metadata-level errors occur *before execution* of a preparator by identifying a mismatch between the known properties (metadata) of the data and the preconditions of the preparator. Finally, data-level errors occur *during execution* of a preparator. The error manager captures the incurred errors during the executions of preparation pipeline, and report them to users while the executions terminate.

2.5 Preparation suggester

At any time during data preparation, analysts need to persistently ask themselves two questions: 1) which transformation do I need to perform; 2) which parameters do I choose for this particular transformation. To obtain the answers for these two questions, they need to carefully inspect the data, come up with some hypotheses for the statistics and structure of the data, and possibly make a few trial and error to realize the specific hidden features of the data, for which they choose a proper preparator or parameters. However, these processes may take an unpredictably long time and become the bottleneck of efficient data preparation [15]. The condition becomes deteriorated in business scenarios where data has large capacity and complex structure.

Realizing that most of the time expenses are essentially donated to inspecting and trying out preparators and parameters, which can be easily delegated to machines, we propose a preparation suggester to search for ideal preparator candidates and their corresponding parameters. Taking the previous preparator, the data at the current stage, the metadata context as input, the preparation suggester traverses the search space of the preparators and produce a list of parameterized preparators as suggestions, from which analysts can then choose the most suitable preparator. It is

notable that traversing the whole search space is in most cases not feasible, due to the large amount of parameters that could be assigned to each preparator.

2.6 Provenance management

Provenance refers to arbitrary information that explain the origin of and the processes applied to data [17]. This information is of wide usage in many cases, such as enabling scientific experiments repeatability [8], data processing and debugging [5, 18]. In the context of data preparation, depending on whether we consider preparator mechanisms as black boxes, provenances fall into two groups, i.e. pipeline provenance (a.k.a. workflow provenance in some literature [3, 11]) and data provenance. Pipeline provenance captures the processes applied to data, as well as the run-time setup of specific executions. It represents coarse-grained provenance, interpreting that the full output of a preparator come from its input data as a whole. Such information could be useful in the cases where users want to repeat the same processes with the same setups. Data provenance, on the other hand, produces more fine-grained information. It determines which part of the input data contributes to a specific output *data item*. A data item is shaped as a particular forms in different data models, such as a tuple of a relational data model, or an element or subtree of an XML file. Data provenance is capable of giving an explanation of how data are created from particular sources throughout data preparation pipelines.

Data-Knoller maintains a *provenance repository* to store and manage provenance of the aforementioned types. The repository records heterogeneous types of provenance compose of different pieces of information. For example, pipeline provenance only tracks the steps of a preparation pipeline along with run-time environment setup, whereas data provenance tracks the input and output of a preparator to a particular degree of concreteness, depending on the semantics of data provenance required to record. The implementation of this component provides a storage of provenance of various types, as well as an convenient query to this information.

2.7 Pipeline optimization

So far, we discussed the architecture of Data-Knoller based on the assumption that the preparators in a pipeline are executed in the order they are specified. However, this order, which follows the preparation as designed by the user is not necessarily the most efficient solution. In fact, the execution order of some preparators can be exchanged without modifying the result of the whole pipeline, whereas the time and memory cost of the two orders might be largely different. For example, suppose a user specifies the following four preparators in a row in the preparation pipeline,

```
1. removeExactDuplicates();
2. select(predicate: orderPrice>20);
3. normalizeDateFormat(field: shipDate,
targetFormat: YYYY-MM-DD);
4. deleteField(field: telephone)
```

Removing exact duplicates can be exchanged with selecting all the records whose 'orderPrice' is larger than twenty, and the same for normalizing the value format of field 'shipDate' with geocoding 'shipAddress', because the results of the preparators in each pair are independent from each other. However, removing duplicates first means to examine also the records that do not fulfill the predicate of the next selecting preparator, which could be omitted by executing the select first. In the second aforementioned case, the two preparators operate on different data fields, leaving the space to reduce execution cost by switching their execution order as well.

Some preparators must be executed in an certain order strictly, for not changing the execution result. For example, removing duplicates cannot be executed after normalizing the format of 'shipDate', because the latter one may reconcile the different format of date for the same entity, and thus producing exact duplicates afterward.

The goal of the pipeline optimizer is to find out the optimal pipeline execution order that obtain the same output with the least time expense. Given a preparation pipeline with n preparators specified in a certain order, there are possibly up to $n!$ different orders, meaning that a brute-force approach is not feasible. The pipeline optimizer is further obliged to evaluate the costs of different execution orders by estimating the overall cost of individual preparators in a certain order, and thus produce the optimal execution order with the least cost. Currently we are designing the mechanism to discover the most efficient executable pipeline order.

2.8 Pipeline evolution

In many real-world cases, users face a large number of datasets defined with schemata similar to one another. Such data might come from different sources that define the essentially same schema in similar ways or the same source with evolving schema definitions. An extreme case of such a situation is the preparation of web tables: The emergence of collaborative online knowledge services, such as Wikipedia, enables individual users to revise contents on web pages with significant freedom. Therefore, web tables are subject to more frequent changes than traditional relational database tables in terms of both values and schema. In order to re-use existing pipelines on data with evolved schema, users need to revise the existing pipeline so as to fit the new schema. However, manually doing this can be time-consuming and error-prone. We are currently exploring the approach to automate the adaption of preparation pipeline based on the evolution specification and the pipeline logic.

3 Related Work

Each of these components incoporated in Data-Knoller is backed by an interesting research area and we have seen numerous academic efforts on each of them. Parallel to research activities, the industry has turned the fruits of academia into actual products. In this section, we summarize representative work in each relevant field and highlight popular data preparation tools.

Data quality and data cleaning. Data quality issues arise in different forms and shapes. For example, there might be inconsistencies of date or phone number formats, extra or missing characters, duplicate records, etc. Compiling the variety of error types helps the research community recognize the complete family of data quality issues and possible challenges of cleaning them.

Some endeavors have been made to collect various forms of data errors to certain extents. Researchers have proposed various taxonomies of dirty data types to describe possible data quality issues [21, 22, 23, 25]. The preparator taxonomy we build is inspired by these previous work. Note that some preparators in our taxonomy reflect an interesting and challenging research topic themselves, e.g., automatic string transformations, such as automating string transformation [2, 14, 27]. In the implementation of preparators, we employ the techniques from previous work.

Metadata management. Regarding data preparation, a metadata standard is needed to interchange and manage metadata within the system. A number of metadata have been recognized and categorized to different extents [1], facilitating the creation of our metadata taxonomy. Capturing and storing metadata are two of the traditional research concerns of metadata management [10, 28], which are of importance for our data preparation framework as well. In a typical data preparation pipeline, data are modified frequently, leading to numerous metadata changes. Thus, keeping metadata up-to-date and consistent is yet another issue.

Preparation suggestion. In many cases, users might not know what preparation shall be done on data. This is a common situation where preparation suggestion comes into play. Wrangler [20] is a research prototype based on the prior interactive data cleaning system Potter's Wheel [26]. It introduces an inference engine that takes data context, transformation context and the selected data units in the interface as input, producing a list of ranked candidate transformations accordingly. Wrangler later became the foundation of Trifacta [16], a commerical data preparation system. To promise an agile self-service data preparation, our system incorporates preparation suggestion as well.

Provenance management. Provenance is useful in the context of a data preparation system: by tracking the origins of data, the system enables users to debug preparation pipelines. Provenance provides an insight of the transformations that have been done on data, thus enabling repeatability of preparations. A lot of efforts have been undertaken to capture [6, 13], store [4, 7, 12], and query [12] provenance, inspiring us to build a data-preparation-specific provenance management system based on them.

Data preparation tools. Seeing the importance of a systematic data preparation system for the day-to-day data preparation effort, several industrial products have

been built, such as Open-Refine,[3] Tableau Prep,[4] Trifacta,[5] and Talend.[6] These tools provide users with interactive interfaces to conduct direct manipulation for data preparation and visualize data properties, such as statistical data characteristics, data type, and data histogram.

4 Future Work

There are three obvious things that we are going to explore in future work. First, we will look for a solution to prune the search space of preparators and different paramters so as to enable more efficient preparation suggestion. Second, we will explore the exchangeability of different preparators under particular conditions and employ it to prune the search space of execution order. Last but not least, we will explore the approach to enable preparation pipeline evolution with the evolution specification of schema.

References

[1] Z. Abedjan, L. Golab, and F. Naumann. "Profiling Relational Data: A Survey". In: *Proceedings of the VLDB Endowment* 24.4 (2015), pages 557–581.

[2] Z. Abedjan, J. Morcos, M. N. Gubanov, I. F. Ilyas, M. Stonebraker, P. Papotti, and M. Ouzzani. "Dataxformer: Leveraging the Web for Semantic Transformations". In: *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*. 2015.

[3] P. Alper, K. Belhajjame, C. A. Goble, and P. Karagoz. "Enhancing and Abstracting Scientific Workflow Provenance for Data Publishing". In: *EDBT/ICDT Workshops*. 2013, pages 313–318.

[4] Z. Bao, H. Köhler, L. Wang, X. Zhou, and S. W. Sadiq. "Efficient Provenance Storage for Relational Queries". In: *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*. 2012, pages 1352–1361.

[5] P. Bourhis, D. Deutch, and Y. Moskovitch. "POLYTICS: Provenance-Based Analytics of Data-Centric Applications". In: *Proceedings of the International Conference on Data Engineering (ICDE)*. 2017, pages 1373–1374.

[6] P. Buneman, S. Khanna, and W. C. Tan. "Why and Where: A Characterization of Data Provenance". In: *Proceedings of the International Conference on Database Theory (ICDT)*. 2001, pages 316–330.

[3] http://openrefine.org/ (last accessed 2018-10-18).
[4] https://www.tableau.com/products/prep (last accessed 2018-10-18).
[5] https://www.trifacta.com/ (last accessed 2018-10-18).
[6] https://www.talend.com/ (last accessed 2018-10-18).

[7] A. Chapman, H. V. Jagadish, and P. Ramanan. "Efficient Provenance Storage". In: *Proceedings of the International Conference on Management of Data (SIGMOD)*. 2008, pages 993–1006.

[8] K. Cranmer, L. Heinrich, R. Jones, and D. M. South. "Analysis Preservation in ATLAS". In: *Journal of Physics: Conference Series*. Volume 664. 3. 2015.

[9] T. Dasu and T. Johnson. *Exploratory Data Mining and Data Cleaning*. John Wiley, 2003.

[10] R. Ferreira, J. Moura-Pires, R. Martins, and M. Pantoquilho. "XML Based Metadata Repository for Information Systems". In: *Portuguese Conference on Artificial intelligence (EPIA)*. 2005, pages 205–213.

[11] B. Glavic. "Big Data Provenance: Challenges and Implications for Benchmarking". In: *Proceeding of the Workshop on Specifying Big Data Benchmarks (WBDB)*. 2012, pages 72–80.

[12] B. Glavic and G. Alonso. "The Perm Provenance Management System in Action". In: *Proceedings of the International Conference on Management of Data (SIGMOD)*. 2009, pages 1055–1058.

[13] T. J. Green, G. Karvounarakis, and V. Tannen. "Provenance Semirings". In: *Proceedings of the Symposium on Principles of Database Systems (PODS)*. 2007, pages 31–40.

[14] S. Gulwani. "Automating String Processing in Spreadsheets Using Input-Output Examples". In: *Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*. 2011, pages 317–330.

[15] J. Heer, J. M. Hellerstein, and S. Kandel. "Predictive Interaction for Data Transformation". In: *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*. 2015.

[16] J. M. Hellerstein, J. Heer, and S. Kandel. "Self-Service Data Preparation: Research to Practice". In: *IEEE Data Engineering Bulletin* 41.2 (2018), pages 23–34.

[17] M. Herschel, R. Diestelkämper, and H. Ben Lahmar. "A Survey on Provenance: What for? What Form? What from?" In: *Proceedings of the International Conference on Very Large Databases (VLDB)* 26.6 (2017), pages 881–906.

[18] M. Herschel and T. Grust. "Transformation Lifecycle Management With Nautilus". In: *VLDB Workshop on the Quality of Data (QDB)*. 2011.

[19] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. "Enterprise Data Analysis and Visualization: An Interview Study". In: *IEEE Visual Analytics Science and Technology (VAST)* 18 (2012), pages 2917–2926.

[20] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. "Wrangler: Interactive Visual Specification of Data Transformation scripts". In: *Proceedings of the International Conference on Human Factors in Computing Systems (CHI)*. 2011, pages 3363–3372.

[21] W. Y. Kim, B. Choi, E. K. Hong, S. Kim, and D. Lee. "A Taxonomy of Dirty Data". In: *Data Min. Knowl. Discov.* 7.1 (2003), pages 81–99.

[22] L. Li, T. Peng, and J. Kennedy. "A Rule Based Taxonomy of Dirty Data". In: *GSTF Journal on Computing (JoC)* 1.2 (2018).

[23] P. Oliveira, F. Rodrigues, P. Henriques, and H. Galhardas. "A Taxonomy of Data Quality Problems". In: *2nd Int. Workshop on Data and Information Quality*. 2005, pages 219–233.

[24] G. Press. "Cleaning Data: Most Time-Consuming, Least Enjoyable Data Science Task". In: *Forbes* (2016).

[25] E. Rahm and H. H. Do. "Data Cleaning: Problems and Current Approaches". In: *IEEE Data Eng. Bull.* 23.4 (2000), pages 3–13.

[26] V. Raman and J. M. Hellerstein. "Potter's Wheel: An Interactive Data Cleaning System". In: *Proceedings of the International Conference on Very Large Databases (VLDB)*. 2001, pages 381–390.

[27] R. Singh and S. Gulwani. "Learning Semantic String Transformations from Examples". In: *Proceedings of the VLDB Endowment* 5.8 (2012), pages 740–751.

[28] D. Srivastava and Y. Velegrakis. "Using Queries to Associate Metadata with Data". In: *Proceedings of the International Conference on Data Engineering (ICDE)*. 2007, pages 1451–1453.

TrussFormer: 3D Printing Large Kinetic Structures

Robert Kovacs

Human Computer Interaction group
Hasso Plattner Institute
robert.kovacs@hpi.uni-potsdam.de

TrussFormer is an integrated end-to-end system that allows users to 3D print large-scale kinetic structures, i.e., structures that involve motion and deal with dynamic forces. TrussFormer builds on TrussFab, from which it inherits the ability to create static large-scale truss structures from 3D printed connectors and PET bottles. TrussFormer adds movement to these structures by placing linear actuators into them: either manually, wrapped in reusable components called assets, or by demonstrating the intended movement. TrussFormer verifies that the resulting structure is mechanically sound and will withstand the dynamic forces resulting from the motion. To fabricate the design, TrussFormer generates the underlying hinge system that can be printed on standard desktop 3D printers. We demonstrate TrussFormer with several example objects, including a 6 legged walking robot and a 4m tall animatronics dinosaur with 5 degrees of freedom.

1 Introduction

Personal fabrication tools, such as 3D printers, afford rapid prototyping [10] as well as to fabricate interactive objects [7]. The latter includes animated objects, such as kinematic animals [4] or actuated paper origami [14], and simple machines [18].

More recently, HCI researchers have started to explore how to enable non-expert users to fabricate large-scale structures. While professional users may have access to large-scale 3D printing equipment, non-experts are generally limited to the use of desktop 3D printers, causing these systems to achieve scale by combining 3D print with ready-made objects, such as empty plastic bottles [8]. The resulting systems also support users in creating structures capable of dealing with the substantial forces such structures are subject to. TrussFab [8], for example, achieves this by allowing users to combine already sturdy primitives and by checking stability during editing.

While large-scale fabrication systems like TrussFab have been shown to support a wide range of applications, from furniture to trade-show pavilions, such systems are limited to creating static structures.

In this paper, we present a system that allows users to create large kinetic structures, i.e., structures that involve motion and deal with dynamic forces, as they occur as part of animatronics devices, such as the animated Tyrannosaurus Rex, illustrated by Figure 1, and other large-scale machinery. TrussFormer embodies the required engineering knowledge from creating the appropriate mechanism, verifying its structural soundness, and generating the underlying hinge system printable on desktop 3D printers.

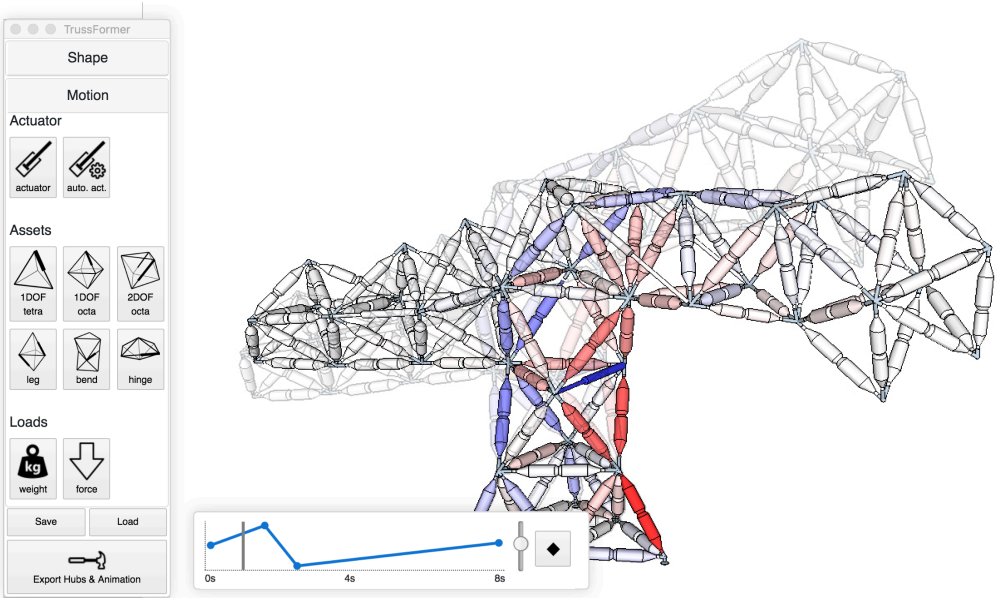


Figure 1: TrussFormer is an end-to-end system that allows users to design and 3D print large-scale kinetic truss structures. TrussFormer verifies that the designed structure can handle the forces resulting from its motion, as shown on this animatronics 4m tall T-Rex.

2 Related Work

TrussFormer builds on previous efforts in animatronics, robotics, software tools for creating mechanisms in HCI and graphics, and creating variable geometry truss mechanisms.

2.1 Software tools for Animatronics

Many HCI researchers have built software tools to empower users to animate robots [17]. This is especially challenging when the users are novices and the intended results are expressive movements, such as imitating animal (organic) movements, i.e., animatronics. Animatronics interfaces follow several designs, from manual control [9] to puppeteering using skeletal tracking [19]. Marti et al. designed an early example of an animatronics software tool for a small (puppet sized) phone call handling agent, demonstrating two methods: manual control (user directly controls each single actuator using one GUI fader) and programming motion patterns using a sequencer [9].

2.2 Software tools for designing mechanisms and dealing with forces

TrussFormer draws from work on systems that assist users with creating mechanisms that involve motion or forces. Algorithmic tools can help users create moving

mechanisms. For example, kinematic synthesis of mechanisms [22], or generation of personalized walking toys from a library of predefined template mechanisms [1]. These can be embedded in design support systems, for example, generating moving toys from motion input [24].

Researchers in the domain of personal fabrication have started to investigate the effects of dynamic forces in the resulting models, such as balancing rotating objects [13] and interactively designing model airplanes [23]. TrussFormer extends these approaches by using physical simulation interactively in its editor. This combination is necessary to provide an editor that embodies the domain knowledge needed to produce large scale animated truss structures.

2.3 Variable geometry truss mechanisms

TrussFormer's mechanisms are based on variable geometry trusses (VGT) [15, 16, 21]. VGTs have been used extensively in robotics. Tetrobot [5] is built by chaining the tetrahedron edges with linear actuators, which unite at a vertex in a spherical joint. The design and mechanics behind this type of spherical joints have been extensively analyzed [2]. Tetrobot was designed to enable robots to reconfigure into different usages by reusing the same basic primitives. Researchers and engineers have explored variations of this VGT design in different contexts, for example: space applications [15], reconfigurable robotic manipulators [5], and shape morphing trusses [20]. Other researchers introduced design variations in this basic cell, allowing the resulting structure to afford new qualities. For instance, the Spiral Zipper [3] is an extendable edge, based on extending a cylinder that allows for extreme expansion ratios (e.g., 14:1). Similarly, Pneumatic Reel Actuator [6] is based on a mechanism that extrudes and retracts a plastic (tape-like) tubing, to act as an actuator. The mechanism is designed to be lightweight and low-cost while being limited in its robustness.

TrussFormer takes inspiration from VGTs and builds on the conceptual design of Tetrobot. To this work, TrussFormer contributes a spherical joint design that is automatically generated based on the designed truss geometry.

3 Creating kinetic structures using TrussFormer

TrussFormer helps users to create the shape and design the motion of large-scale kinetic structures. It does this by incorporating linear actuators into rigid truss structures in a way that they move "organically", i.e., hinge around multiple points at the same time. These structures are also known as variable geometry trusses [15]. Figure 2 illustrates this on the smallest elementary truss, (a) the rigid tetrahedron. (b) We swap one of the edges with a linear actuator, (c) resulting in a variable geometry truss. The only required change for this is to introduce hubs that enable rotation at the nodes. We call these hinging hubs.

This simple approach to create variable geometry truss mechanisms scales well to arbitrary larger structures. Our T-Rex model from Figure 1 contains five linear

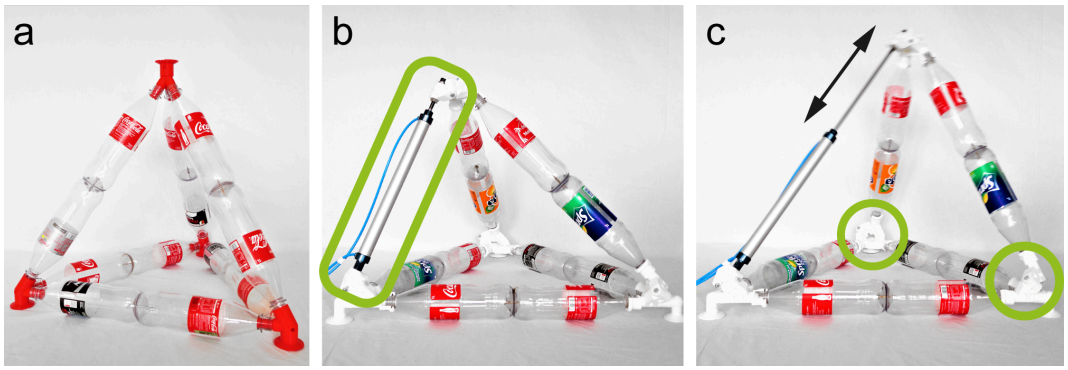


Figure 2: (a) The static tetrahedron (b-c) is converted into a moving structure by swapping one edge with a linear actuator. The only required change is to introduce connectors that enable rotation.

actuators and thus offers five degrees of freedom (DoF). It can (a) lift or lower its neck (1 DoF), (b) turn its head left and right (1 DoF), (c) sweep its tail (2 DoF), and (d) open its mouth (1 DoF), as shown in Figure 3.

In the following, we demonstrate how TrussFormer allows non-expert users to create such structures in six steps.

3.1 Step 1: Creating the static structure

As shown in Figure 4, this particular model was created by first modeling the T-Rex as a static structure in TrussFormer. Our editor's ability to create static structures is based on TrussFab [8] : users design the shape of their T-Rex using structurally stable primitives (tetrahedra and octahedra).

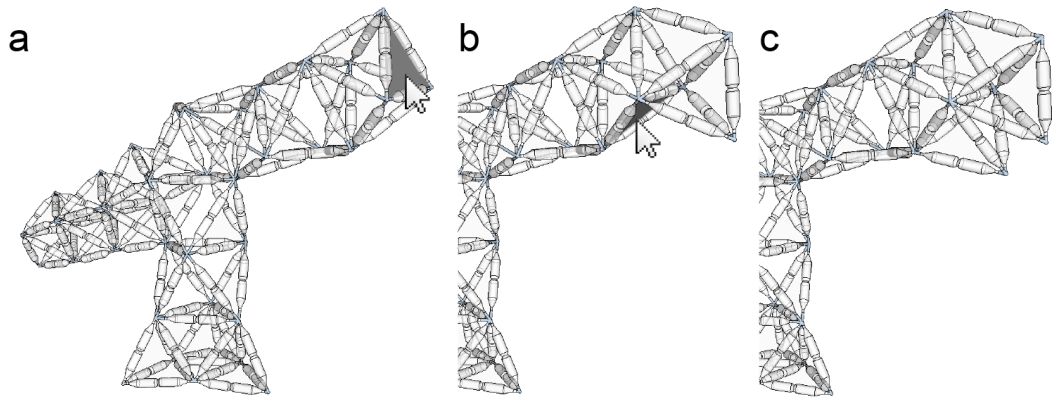


Figure 3: Modeling the static shape of the T-Rex. Here, the user creates the jaws of the T Rex by attaching tetrahedron primitives through the steps (a, b, c).

3.2 Step 2: Adding movement

To add movement to the static structure, users select the demonstrate movement tool and pull the T-Rex head downwards, as shown in Figure 5. TrussFormer responds by placing an actuator that turns the T-Rex body into a structure that organically moves and bends down. Together with the Demonstrate movement tool, TrussFormer provides three different approaches to animating structures, ranging from this (1) automated placement (for novice users), through (2) placing elements with predefined motion, called assets, to (3) manual placement (as users acquire engineering knowledge). We discuss these in section "Adding motion to the structure".

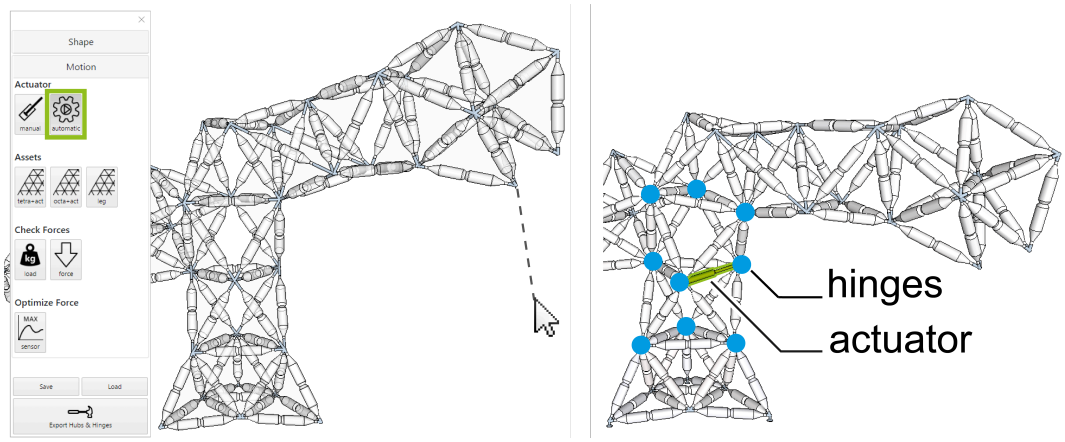


Figure 4: (a) The user selects the demonstrate movement tool and pulls the T-Rex head downwards. (b) TrussFormer responds by adding an actuator to the T-Rex body so that it is capable of performing this type of motion. At this point the system also places 9 hinging hubs to enable this motion (marked with blue dots).

3.3 Step 3: Stability check across poses

During this step, TrussFormer also verifies that the mechanism is structurally sound. In the background, TrussFormer finds the safe range of expansion and contraction of the placed actuator by simulating the occurring forces in a range of positions. If there is a pose where the forces exceed the pre-determined breaking limits or the structure would tip over, TrussFormer sets the limits for the actuator so it will not extend beyond them. This check prevents users from producing invalid configurations.

3.4 Step 4: Animation

To animate the structure users open the animation pane in the toolbar, as shown in Figure 6. First, they control the movement of the structure manually using slid-

ers, to try out the movement. When they find the desired pose, they simply add it as a keyframe to the animation timeline. With this TrussFormer allows users to orchestrate the movement of all actuators using a simple timeline/keyframe editor. In Figure 6 we program a "feeding" behaviour, where the T-Rex opens its mouth while reaching down and waving its tail.

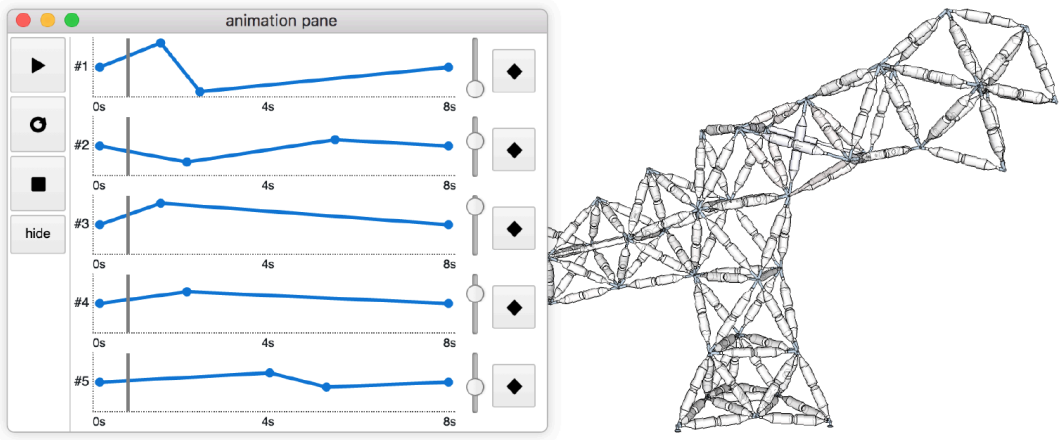


Figure 5: Animating the structure. Users sets the desired pose using the sliders in the animation pane and orchestrates the movement by placing key-frames on the timeline.

3.5 Step 5: Checking forces during the motion

Once a movement has been defined, TrussFormer computes the dynamic forces. As shown in Figure 7a, the user creates an animation that moves the T-Rex body up and down. (b) TrussFormer computes the forces while T-Rex's body comes back up quickly after dipping down; the large acceleration of the long neck leads to very high inertial forces, exceeding the breaking limit of the construction, (c) causing the structure to fail at the indicated time point. These situations are hard to foresee, because the inertial forces can be multiple times higher than the static load in the structure. (d) TrussFormer addresses this by automatically correcting the animation sequence by either limiting the acceleration or the range of the movement, assuring that the structure will now withstand the movement.

3.6 Step 6: Fabrication

When users are satisfied with their design (structure, movement and animation), they click the fabricate button, shown in Figure 8a. This invokes (1) TrussFormer's hinge generation algorithm, which analyzes the structure's motion and generates the appropriate 3D printable hinge and hub geometries, annotated with imprinted

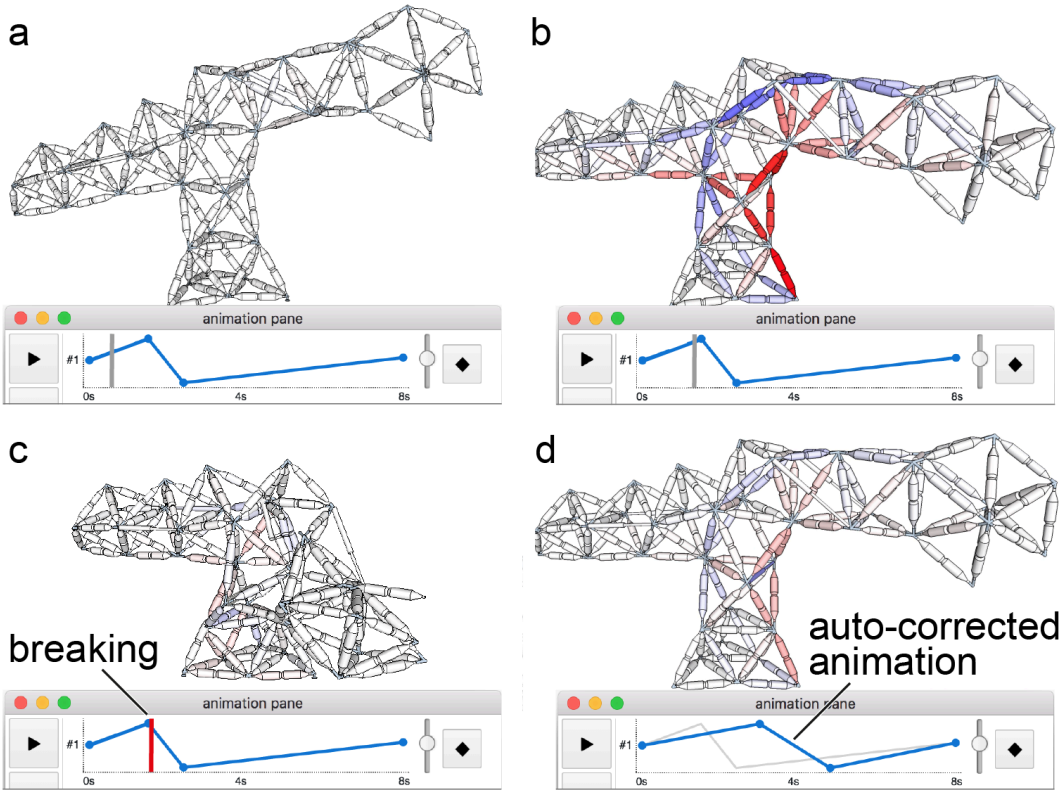


Figure 6: Verifying the inertial forces: (a-b) The forces are increasing with the acceleration of the structure. (c) The structure breaks when the direction of the movement changes rapidly. (d) TrussFormer resolves this by making the movement slower.

IDs for assembly. In the case of the T-Rex, the system exports 42 3D printed hubs, consisting of 135 unique hinging pieces. (2) Next, TrussFormer exports the created animation patterns as Arduino code that users upload to their microcontroller. (3) Lastly, it outputs a specification, containing the force, speed, and motion range of the actuators, in order to achieve the desired animation pattern. Users find these actuators as standardized components.

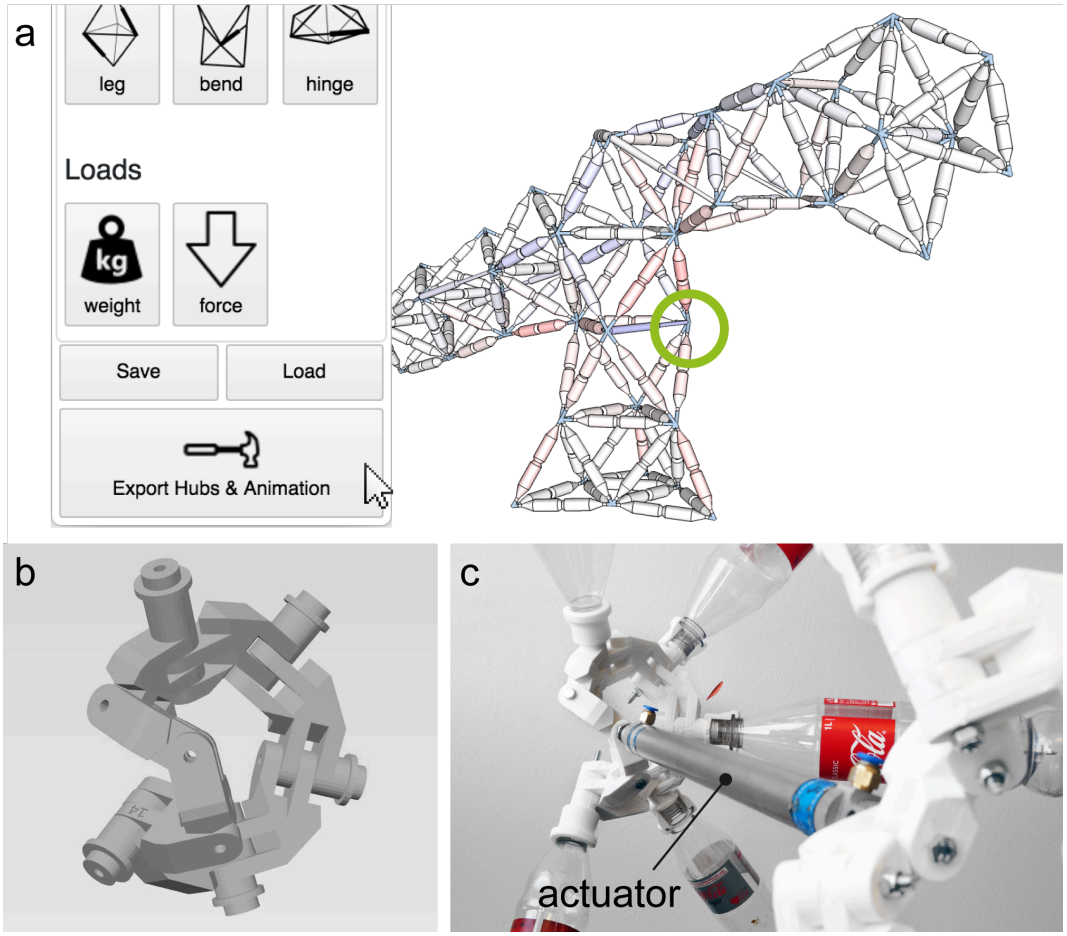


Figure 7: (a) To fabricate our T-Rex model, TrussFormer exports: (b) the appropriate 3D printable hinging-hubs, (c) and the specifications for the actuators that inform the users which one to buy. TrussFormer also exports the animation sequence for an Arduino.

3.7 Implementation

3.7.1 Software system

We extend the TrussFab editor [8], which provides the core functionality to create, save, load, and export static structures. TrussFormer further allows users to

add movement and animate these structures. Like TrussFab, TrussFormer is also implemented as a plugin for the 3D modelling software SketchUp. The native programming language for SketchUp plugins is Ruby, which most of the features that we described throughout the paper, such as the hinge placement algorithm, are written in. To simulate the movement and the force distribution in the 3D model, we use the physical simulation engine MSPhysics, a Ruby wrapper for the C++ physics library Newton Dynamics [11]. To achieve interactive performance, the only simulated components are the hubs, the edges are just animated on the scene. The hubs contain all the necessary information, such as weight, breaking force, and the stiffness determining how much hubs can move in relation to their neighbors. User interface elements (e.g., the control or the animation pane) are displayed in a SketchUp-integrated Web Browser View. We implemented the UI in HTML and JavaScript to take advantage of UI frameworks such as React. To generate the 3D printable hinge, we use the parametric 3D modeling tool OpenSCAD [12]. When users export their kinetic structure, TrussFormer determines the hinges and static hubs and calls the pre-defined OpenSCAD scripts with the relevant parameters (e.g., angle, connection type, or length of the connection). These scripts describe the resulting parametrized 3D model, which are rendered in OpenSCAD as .stl files.

3.7.2 Control system and actuators

Figure 24 shows the hardware we use to actuate our T-Rex example. We use pneumatic actuators with interfaced with proportional valves (Festo VPPE and MPYE series) that are controlled by an Arduino Nano. The pneumatic cylinders have diameters from 25 to 35 mm and produce forces between 390 N and 770 N. We use an Airpress HL 360 compressor that can provide up to 8 bar of pressure.

4 Conclusion

We presented TrussFormer, an end-to-end system that enables novice users to design and build large animated structures. Such structures are usually a privilege of industry such as theme parks. TrussFormer encapsulates domain knowledge about the occurring dynamic forces so that even novice users can build such animated structures. We showed how TrussFormer enables users to add motion to static structures in three ways, including simply pulling on the virtual model and letting the system find the placement of an actuator to enable this motion. Furthermore, we showed how TrussFormer finds valid motion and force ranges for actuators to realize user-defined animations. TrussFormer detects and automatically suggests corrections for animations that would break the simulated structure, thereby ensuring that the physical structure will function as desired. As a last step, TrussFormer generates all connectors and hinges that users print on their desktop 3D printer and exports the actuator specifications.

References

[1] G. Bharaj, S. Coros, B. Thomaszewski, J. Tompkin, B. Bickel, and H. Pfister. "Computational design of walking automata". In: *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2015, pages 93–100.

[2] P. Bosscher and I. Ebert-Uphoff. "A novel mechanism for implementing multiple collocated spherical joints". In: *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*. Volume 1. 2003, pages 336–341.

[3] F. Collins and M. Yim. "Design of a spherical robot arm with the spiral zipper prismatic joint". In: *IEEE International Conference on Robotics and Automation (ICRA'16)*. 2016, pages 2137–2143.

[4] S. Coros, B. Thomaszewski, G. Noris, S. Sueda, M. Forberg, R. W. Sumner, W. Matusik, and B. Bickel. "Computational design of mechanical characters". In: *ACM Trans. Graph.* 32.4 (2013), 83:1–83:12. DOI: 10.1145/2461912.2461953.

[5] G. J. Hamlin and A. C. Sanderson. "Tetrobot: A modular approach to parallel robotics". In: *IEEE Robotics & Automation Magazine* 4.1 (1997), pages 42–50.

[6] Z. M. Hammond, N. S. Usevitch, E. W. Hawkes, and S. Follmer. "Pneumatic reel actuator: Design, modeling, and implementation". In: *IEEE International Conference on Robotics and Automation (ICRA'17)*. 2017, pages 626–633.

[7] S. E. Hudson. "Printing teddy bears: a technique for 3D printing of soft interactive objects". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2014, pages 459–468. DOI: 10.1145/2556288.2557338.

[8] R. Kovacs, A. Seufert, L. Wall, H.-T. Chen, F. Meinel, Müller, S. You, M. Brehm, J. Striebel, Y. Kommana, A. Popiak, T. Bläsius, and P. Baudisch. "TrussFab: Fabricating sturdy large-scale structures on desktop 3D printers". In: *Proceedings of the ACM Conference on Human Factors in Computing Systems*. 2017. DOI: 10.1145/3025453.3025624.

[9] S. Marti and C. Schmandt. "Physical embodiments for mobile communication agents". In: *Proceedings of the 18th annual ACM symposium on User interface software and technology*. 2005, pages 231–240.

[10] S. Mueller, S. Im, S. Gurevich, A. Teibrich, L. Pfisterer, F. Guimbretière, and P. Baudisch. "WirePrint: 3D printed previews for fast prototyping". In: *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 2014, pages 273–280. DOI: 10.1145/2642918.2647359.

[11] *Newton Dynamics*. URL: http://newtondynamics.com/forum/newton.php (last accessed 2018-09-25).

[12] *OpendSCAD*. URL: http://openscad.org (last accessed 2018-09-25).

[13] R. Prévost, E. Whiting, S. Lefebvre, and O. Sorkine-Hornung. "Make it stand: balancing shapes for 3D fabrication". In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), page 81.

[14] J. Qi and L. Buechley. "Animating paper using shape memory alloys". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2012, pages 749–752.

[15] V. A. Reinholtz and L. T. Watson. *Enumeration and analysis of variable geometry truss manipulators*. Technical report. Virginia Polytechnic Institute and State University; Blacksburg, VA, United States, 1990.

[16] M. D. Rhodes and M. Mikulas Jr. *Deployable controllable geometry truss beam*. Technical report. NASA Langley Research Center; Hampton, VA, United States, 1985.

[17] T. Ribeiro and A. Paiva. "The illusion of robotic life: principles and practices of animation for robots". In: *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*. 2012, pages 383–390.

[18] T. J. Roumen, W. Müller, and P. Baudisch. "Grafter: Remixing 3D-Printed Machines". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, page 63.

[19] M. Sakashita, T. Minagawa, A. Koike, I. Suzuki, K. Kawahara, and Y. Ochiai. "You as a puppet: Evaluation of telepresence user interface for puppetry". In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 2017, pages 217–228.

[20] A. Sofla, D. Elzey, and H. Wadley. "Shape morphing hinged truss structures". In: *Smart Materials and Structures* 18.6 (2009), pages 1–8.

[21] A. Spinos, D. Carroll, T. Kientz, and M. Yim. "Variable topology truss: Design and analysis". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'17)*. 2017, pages 2717–2722.

[22] D. Subramanian et al. "Kinematic synthesis with configuration spaces". In: *Research in Engineering Design* 7.3 (1995), pages 193–213.

[23] N. Umetani, Y. Koyama, R. Schmidt, and T. Igarashi. "Pteromys: interactive design and optimization of free-formed free-flight model airplanes". In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), page 65.

[24] L. Zhu, W. Xu, J. Snyder, Y. Liu, G. Wang, and B. Guo. "Motion-guided mechanical toy modeling." In: *ACM Trans. Graph.* 31.6 (2012), pages 127–136.

Theory of Estimation-of-Distribution Algorithms

Martin Krejca

Algorithm Engineering
Hasso-Plattner-Institut
martin.krejca@hpi.de

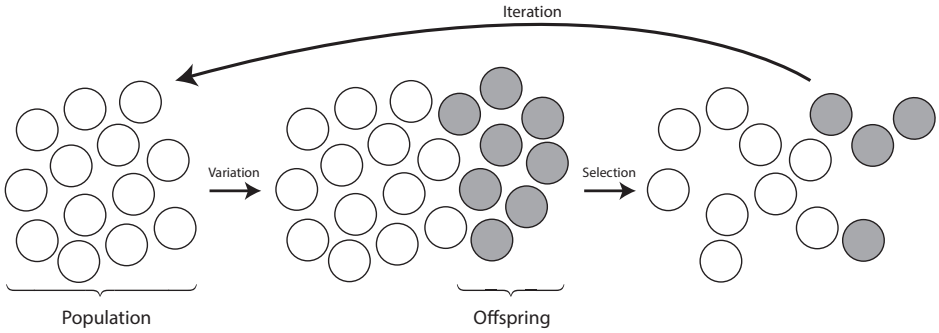
Estimation-of-distribution Algorithms (EDAs) are randomized search heuristics and general-purpose optimizers that create a probabilistic model of the problem's domain and iteratively refine it using samples. Due to their flexibility, they are most often used when there is little problem-specific knowledge or when there is too little budget to implement or set up highly specific solvers.

Our focus lies on theoretically analyzing EDAs in order to gain insights into their key features, with the aim of answering important questions, such as how to overcome noise, how fast these algorithms can optimize at all, and how they compare to classical evolutionary algorithms (EA). We present our latest results, which are a new type of EDA that is fast on different classes of functions – a feat that, up to now, no other EDA or EA has achieved – and better tools for analyzing random processes theoretically.

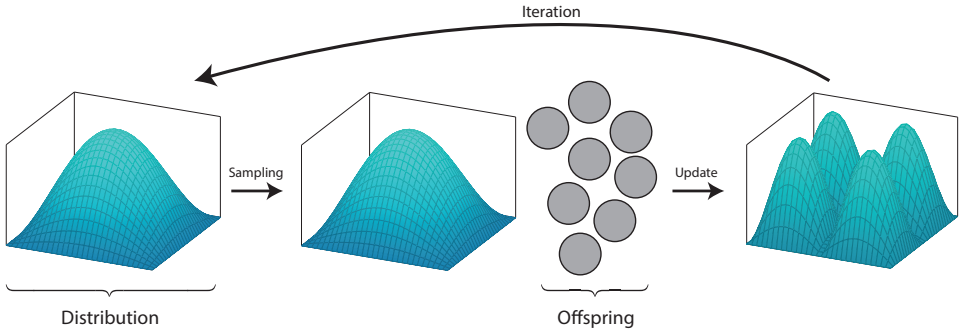
1 Introduction

When facing real-world optimization problems, it is sensible to use state-of-the-art solvers. Due to the NP-hard nature of many of these problems, one does not necessarily expect to get optimal results but a fair approximation. However, if the problem cannot (easily) be transformed into the form that the solver accepts, if there is no good solver for the problem, or if one is interested in a fast (and hopefully good) preliminary solution, one may consider using alternative approaches like randomized search heuristics. One class of such heuristics are estimation-of-distribution algorithms (EDAs) [20], which can produce good solutions in little time. They have been applied to a variety of different problems, where they proved to be superior to competing algorithms, oftentimes yielding better solutions or any solution at all [20].

In a broader sense, EDAs belong to the class of evolutionary algorithms (EAs) [12], which are optimization algorithms that mimic behavior seen in natural evolution, such as mutation or crossover. Classical EAs maintain a (multi)set (the *population*) of candidate solutions (the *individuals*) over the space of all potential solutions of the optimization problem and iteratively create new individuals by modifying the existing ones and then discarding bad individuals, creating a new population (see Figure 1a). In contrast to that, EDAs store a probabilistic model (which implies a probability distribution) of the problem space, which they evolve by learning from samples of the model (see Figure 1b). Thus, EDAs do not only show which individuals are good but, additionally, show which areas of the solution space are more beneficial. From this point of view, EDAs can be considered more insightful



(**a**) The schematic view of a classical EA. The population first increases through variation (that is, mutation or crossover) and then gets reduced by selection.



(**b**) The schematic view of an EDA. The algorithm samples offspring from a distribution and then performs an update.

Figure 1: A comparison of the main differences between classical EAs and EDAs. A classical EA (Figure 1a) works with an explicit population, whereas an EDA (Figure 1b) uses a probability distribution instead. Both algorithms create offspring and perform an update afterward.

than classical EAs. Therefore, it is interesting to study the differences between EDAs and classical EAs.

EAs in the broad sense are general-purpose heuristics that only need a problem-specific quality measure (called a *fitness function*) that says how good or bad each individual is. Thus, from a theoretical point of view, these algorithms fall into the black-box model. Theoretical analyses then follow black-box complexity, which means that only the number of function evaluations – the supposedly most costly operation – is considered.

When performing such analyses, the function class is usually limited to certain benchmarks that exhibit a desired property. In this context, pseudo-Boolean functions are the most commonly analyzed class, that is, functions $f\colon \{0,1\}^n \to \mathbf{R}$. The function analyzed the most in this class is OneMax (equation (1)): a function that returns the number of 1s of a bit string, thus, each bit string (but the optimum) has a neighbor (of Hamming distance 1) that is better. When considering this function, one is interested in how well an algorithm is able to follow a simple gradient in

fitness (also called *hill climbing*). Another common function is LeadingOnes (equation (2)), which returns the number of consecutive 1s starting from the left. This function resembles the process of finding a hidden permutation, and it shows how well an algorithm can handle dependency in the bits, as information about bits more to the right of a bit string is only revealed once all bits to the left are 1. In Section 3, we go more into detail about the run times of many EAs and EDAs on these two functions (Table 1).

Our first result introduces a new EDA, the *sig-cGA* (Algorithm 1), that is able to optimize both OneMax and LeadingOnes in $O(n \log n)$ in expectation – a feat that is up to now unique for any EA or EDA. The algorithm works by saving a (condensed) history of samples and then searches for significances, that is, a far higher number of 1s or 0s than expected. If such a significance is found, the probabilistic model of the algorithm is adjusted accordingly.

Our second result is concerned with *drift theory* – the essential toolbox of theorems used when analyzing EAs and other random processes. We improve the applicability of the most basic theorem (the *additive drift theorem*; Theorem 1). As it is used to prove many other drift theorems, our result entails that these theorems are improved as well.

We now proceed by stating some notation and terms in Section 2 that we need to explain our results presented in Section 3. We conclude with an overview on future work in Section 4.

2 Preliminaries

Recall that we consider pseudo-Boolean optimization, that is, our fitness functions $f\colon \{0,1\}^n \to \mathbf{R}$ are defined over the discrete hypercube. Let $n \in \mathbf{N}^+$ always denote the dimension of the search space. For an individual $x \in \{0,1\}^n$, that is, a bit string of length n, we call its respective value $f(x)$ *fitness*. Further, we call every component x_i of x a *bit* and index i the *position* of that bit. Last, we denote string concatenation with the $\circ$ symbol.

In Section 3, we investigate the following two functions, which we already discussed in Section 1:

$$\text{OneMax}(x) = \sum_{i=1}^{n} x_i \quad \text{and} \tag{1}$$

$$\text{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_i \,. \tag{2}$$

Our algorithm of interest is the sig-cGA (short for *significance-based compact genetic algorithm*; Algorithm 1), which we recently introduced [3]. It works with a very simple probabilistic model that assumes independence between all bits of the hypercube, that is, its model is univariate. This model is represented by a vector τ (the *frequency vector*) of n probabilities. Each component τ_i (a *frequency*) denotes the probability to sample a 1 at position i (and a 0 with probability $1 - \tau_i$) independently from all

Algorithm 1: The sig-cGA [3] with parameter ε and significance function sig (equation (3)) optimizing f

$t \leftarrow 0$
for $i \in \{1, \dots, n\}$ **do** $\tau_i^{(t)} \leftarrow \frac{1}{2}$ and $H_i \leftarrow \emptyset$
repeat
  $x, y \leftarrow$ offspring sampled with respect to $\tau^{(t)}$
  $x \leftarrow$ winner of x and y with respect to f
  for $i \in [n]$ **do**
    $H_i \leftarrow H_i \circ x_i$
    if $\mathrm{sig}(\tau_i^{(t)}, H_i) = \mathrm{up}$ **then** $\tau_i^{(t+1)} \leftarrow 1 - 1/n$
    else if $\mathrm{sig}(\tau_i^{(t)}, H_i) = \mathrm{down}$ **then** $\tau_i^{(t+1)} \leftarrow 1/n$
    else $\tau_i^{(t+1)} \leftarrow \tau_i^{(t)}$
    if $\tau_i^{(t+1)} \neq \tau_i^{(t)}$ **then** $H_i \leftarrow \emptyset$
  end
  $t \leftarrow t + 1$
until termination criterion met

other positions. Every iteration, two individuals are sampled and compared via their fitness. We call the better individual the *winner* and the other the *loser* (breaking ties uniformly at random). Additionally, the algorithm keeps a history of bit values for each position and only performs an update when a statistical significance within a history occurs. This approach aligns with the intuitive reasoning that an update should only be performed if there is valid evidence for a different frequency being better suited for sampling good individuals.

In more detail, for each position i, the sig-cGA keeps a history $H_i \in \{0,1\}^*$ of all the bits sampled by the winner of each iteration since the last time τ_i changed – the last bit denoting the latest entry. Observe that if there is no bias in selection at position i, that is, each individual has the same fitness, the bits sampled by τ_i follow a binomial distribution with a success probability of τ_i and $|H_i|$ tries. We call this our *hypothesis*. Now, if we happen to find a sequence (starting from the latest entry) in H_i that significantly deviates from the hypothesis, we update τ_i with respect to the bit value that occurred significantly, and we reset the history. We only use the following three frequency values:

- $1/2$: starting value;
- $1/n$: significance for 0s was detected;
- $1 - 1/n$: significance for 1s was detected.

We formalize *significance* by defining the threshold as follows for all $\varepsilon, \mu \in \mathbf{R}^+$, where μ is the expected value of our hypothesis and ε is an algorithm-specific parameter: $s(\varepsilon, \mu) = \varepsilon \max\{\sqrt{\mu \ln n}, \ln n\}$.

For an $\varepsilon \in \mathbf{R}^+$, we say that a binomially distributed random variable X deviates significantly from a hypothesis $Y \sim \mathrm{Bin}(k, \tau)$, where $k \in \mathbf{N}^+$ and $\tau \in [0,1]$, if there exists a $c = \Omega(1)$ such that $\Pr\big[|X - \mathrm{E}[Y]| \leq s(\varepsilon, \mathrm{E}[Y])\big] \leq n^{-c}$.

We now state our significance function $\mathrm{sig}\colon \{\frac{1}{n}, \frac{1}{2}, 1-\frac{1}{n}\} \times \{0,1\}^* \to \{\mathrm{up}, \mathrm{stay}, \mathrm{down}\}$, which scans a history for a significance. However, it does not scan the entire history but multiple subsequences of a history (always starting from the latest entry). This is done in order to quickly notice a change from an insignificant history to a significant one. Further, we only check in steps of powers of 2, as this is faster than checking each subsequence and we can be off from any length of a subsequence by a constant factor of at most 2. More formally, for all $H \in \{0,1\}^*$, we define, with ε being a parameter of the sig-cGA, where $H[k]$ denotes the last k bits of H,

$$\begin{aligned}
\mathrm{sig}\Big(\frac{1}{2}, H\Big) &= \begin{cases} \mathrm{up} & \text{if } \exists m \in \mathrm{N}\colon \|H[2^m]\|_1 \geq \frac{2^m}{2} + s\big(\varepsilon, \frac{2^m}{2}\big), \\ \mathrm{down} & \text{if } \exists m \in \mathrm{N}\colon \|H[2^m]\|_0 \geq \frac{2^m}{2} + s\big(\varepsilon, \frac{2^m}{2}\big), \\ \mathrm{stay} & \text{else.} \end{cases} \\
\mathrm{sig}\Big(1-\frac{1}{n}, H\Big) &= \begin{cases} \mathrm{down} & \text{if } \exists m \in \mathrm{N}\colon \|H[2^m]\|_0 \geq \frac{2^m}{n} + s\big(\varepsilon, \frac{2^m}{n}\big), \\ \mathrm{stay} & \text{else.} \end{cases} \qquad (3) \\
\mathrm{sig}\Big(\frac{1}{n}, H\Big) &= \begin{cases} \mathrm{up} & \text{if } \exists m \in \mathrm{N}\colon \|H[2^m]\|_1 \geq \frac{2^m}{n} + s\big(\varepsilon, \frac{2^m}{n}\big), \\ \mathrm{stay} & \text{else.} \end{cases}
\end{aligned}$$

We stop at the first (minimum) length 2^m that yields a significance. Thus, we check a history H in each iteration at most $\log_2 |H|$ times.

Observe that in line 14, the termination criterion is not specified. When we consider the run time of the sig-cGA, we mean the number of iterations the algorithm needs to sample an *optimum* for the first time. This means that the termination criterion could also say *until optimum found*. However, since this is a piece of information the algorithm does not have, we assume that it keeps iterating forever, and we just stop all further considerations once the optimum is found.

Note that the first-hitting time is a random variable because the individuals in each iteration are random variables, due to the random nature of the sampling process. The most valuable piece of information about a random variable is of course its distribution. However, determining the exact distribution of such a complex random variable is basically impossible. Therefore, the standard approach is to calculate its expectation. We call this value the expected run time of a randomized algorithm. The prevalent way of determining this expected value when analyzing EAs theoretically is to use drift theory.

Drift theory is a toolbox of various theorems that bound the expected first-hitting times of random processes. The core idea of these drift theorems is to bound the expected progress of a random process during a single step. They then yield a bound on the overall first-hitting time. Hence, drift theory transforms information on local changes to global information about the process.

The initial drift theorem was proven by Hajek [8] and later introduced to the theory community analyzing EAs and restated in a different fashion by He and Yao [9, 10].

It is called the *additive drift theorem* because it bounds the expected change of the process by a global constant. We state it in its most general form.

Theorem 1 (Additive Drift Theorem [13])**.** *Let $(\mathcal{F}_t)_{t\in\mathbf{N}}$ be a filtration, let $(X_t)_{t\in\mathbf{N}}$ be a random process over $\mathbf{R}$ adapted to $\mathcal{F}$, and let $T = \inf\{t \mid X_t \leq 0\}$. Furthermore, suppose that,*

1. *(non-negativity) for all $t \in \mathbf{N}$, it holds that $X_t \cdot \mathbf{1}_{\{t\leq T\}} \geq 0$, and that*
2. *(drift condition) there is some value $\delta > 0$ such that, for all $t \in \mathbf{N}$, it holds that $(X_t - \mathrm{E}[X_{t+1} \mid \mathcal{F}_t]) \cdot \mathbf{1}_{\{t<T\}} \geq \delta \cdot \mathbf{1}_{\{t<T\}}$.*

Then

$$\mathrm{E}[T \mid X_0] \leq \frac{X_0}{\delta}\,.$$

Note that $\mathbf{1}_{\{t<T\}}$ is the indicator random variable for the event that the random process did not reach 0 yet. This effectively means that we are only interested in the process as long as it did not reach 0. What happens afterward is irrelevant for T.

3 Results

We are not going to discuss our prior results that we already discussed in great detail in the last reports [21, 22]. Here, we discuss our run time results of the sig-cGA [3] as well as our results on drift theory [13]. We start with the former.

As already discussed in Section 1, two common benchmark functions for the theory of EAs are OneMax and LeadingOnes. We show various run time results of different algorithms on these functions in Table 1. The usual expected run time for OneMax is $\Theta(n \log n)$, as many algorithms perform a process known as *coupon collector,* which has an expected time of $\Theta(n \log n)$ [18]. The reasoning behind this is that, intuitively speaking, at the beginning, the optimization process progresses quickly and finds many of the n correct bits. However, the process slows down over time, as it is harder to find incorrect bits, adding a factor of $\log n$ to the run time in total. In contrast, the usual expected run time for LeadingOnes is $\Theta(n^2)$, as the bits have to be found sequentially and each position takes, on average, a time of $\Theta(n)$ to optimize.

The main point for the slower run time on LeadingOnes when compared to OneMax for EAs is that the mutation probability is commonly very low (at $1/n$) in order to have a constant number of changes per iteration. Hence, when only a single position is relevant, the waiting time for a correct mutation to occur is in the order of n. If the mutation probability is increased, the algorithm performs more and more of an unbiased random walk, which is detrimental for the overall optimization process. We propose the sig-cGA (Algorithm 1), which starts, so to speak, with a constant mutation probability. However, it circumvents the aforementioned problem in two ways: first, once a bit value seems to be significant, the respective frequency is set to an extreme value (of either $1 - 1/n$ or $1/n$), making it very likely to sample the same bit every iteration without being stuck with this decision. Second, the sig-cGA only performs an update once a significance (see function (3)) is detected. This way, the

Table 1: Expected run times (number of fitness evaluations) of various algorithms until they first find an optimum for the two functions OneMax and LeadingOnes (equations (1) and (2), respectively). For optimal parameter settings, many algorithms have a run time of $\Theta(n \log n)$ for OneMax and of $\Theta(n^2)$ for LeadingOnes.

Algorithm	OneMax	constraints	LeadingOnes	constraints
$(1+1)$ EA	$\Theta(n \log n)$ [6]	none	$\Theta(n^2)$ [6]	none
$(\mu+1)$ EA	$\Theta(\mu n + n \log n)$ [24]	$\mu = O(\text{poly}(n))$	$\Theta(\mu n \log n + n^2)$ [24]	$\mu = O(\text{poly}(n))$
$(1+\lambda)$ EA	$\Theta\left(n \log n + \frac{\lambda n \log\log\lambda}{\log\lambda}\right)$ [4, 11]	$\lambda = O(n^{1-\varepsilon})$	$\Theta(n^2 + \lambda n)$ [11]	$\lambda = O(\text{poly}(n))$
$(1+(\lambda,\lambda))$ GA	$\Theta\left(\max\left\{\frac{n \log n}{\lambda}, \frac{n\lambda \log\log\lambda}{\log\lambda}\right\}\right)$ [2]	$p = \frac{\lambda}{n}, c = \frac{1}{\lambda}$	unknown	–
CSA	unknown	–	$O(n \log n)$ [17]	$\mu \geq 8 \ln\big((4n+6)n\big)$, restarts
UMDA/PBIL	$\Omega(\lambda\sqrt{n} + n \log n)$ [14]	$\mu = \Theta(\lambda)$	$O(n\lambda \log\lambda + n^2)$ [1, 16]	$\lambda = \Omega(\log n), \mu = \Theta(\lambda)$
	$O(\lambda n)$ [15, 25]	$\mu = \Omega(\log n) \cap O(\sqrt{n})$, $\lambda = \Omega(\mu)$ or $\mu = \Omega(\sqrt{n} \log n)$, $\mu = \Theta(\lambda)$ or $\mu = \Omega(\log n) \cap o(n)$, $\mu = \Theta(\lambda)$		
cGA/2-MMAS$_{\text{ib}}$	$\Omega\left(\frac{\sqrt{n}}{\rho} + n \log n\right)$ [23]	$\frac{1}{\rho} = O(\text{poly}(n))$	unknown	–
	$O\left(\frac{\sqrt{n}}{\rho}\right)$ [23]	$\frac{1}{\rho} = \Omega(\sqrt{n} \log n) \cap O(\text{poly}(n))$		
1-ANT	$\Theta(n \log n)$ [19]	$\rho = \Theta(1)$	$O(n^2 \cdot 2^{5/(n\rho)})$ [5]	none
			$2^{\Omega(\min\{n, 1/(n\rho)\})}$ [5]	none
scGA	$\Omega\big(\min\{2^{\Theta(n)}, 2^{c/\rho}\}\big)$ [3]	$1/\rho = \Omega(\log n)$, $a = \Theta(\rho)$, $d = \Theta(1)$, $c > 0$	$O(n \log n)$ [7]	$1/\rho = \Theta(\log n)$, $a = O(\rho)$, $d = \Theta(1)$
sig-cGA (Alg. 1)	$O(n \log n)$ [3]	$\varepsilon > 12$	$O(n \log n)$ [3]	$\varepsilon > 12$

decision of setting a frequency to an extreme value is not arbitrary but well-founded. Together, these properties achieve that the search process does not degenerate into a random walk but into a somewhat directed search process.

Our main result is that the sig-cGA has an expected run time of $O(n \log n)$ on both OneMax and LeadingOnes [3]. This is the first time this has been proven for any EA or EDA. Although the run time is the same on both functions, the reasons are quite different. For LeadingOnes, the leftmost position whose frequency is not at $1 - 1/n$ samples drastically more 1s than 0s (by a constant factor larger than expected). Hence, it only takes $O(\log n)$ iterations in expectation to optimize such a position instead of the usual $O(n)$ iterations. Note that this is the minimum number of iterations necessary in order to perform an update, due to how we define a significance. Once a frequency is increased, it will not decrease with high probability. After setting a frequency to $1 - 1/n$, the next position gets a significant boost in saving 1s and gets increased after $O(\log n)$ iterations in expectation. Thus, the optimization process advanced sequentially. Consequently, since n frequencies need to be optimized, the expected run time is $O(n \log n)$.

For OneMax, a single frequency is increased during $O(n \log n)$ iterations in expectation. This is by a factor of n longer when compared to LeadingOnes. The reason is that all of the bits influence the fitness of an individual drastically at the same time – the probability to save a 1 in the history is only by a factor of $1/\sqrt{n}$ larger than expected. Thus, it takes some time in order to collect enough samples before a significance occurs. However, all of these frequencies are optimized at the same time, that is, in parallel. Thus, the overall run time is still $O(n \log n)$.

These results show a stark contrast in how the sig-cGA optimizes different functions: LeadingOnes is optimized fully sequentially, whereas OneMax is optimized fully in parallel. Nonetheless, in the end, the same run times result.

Another result of ours is the improvement of the classical additive drift theorem [13]. Our main contribution is to remove all prior assumptions that are not necessary for proving the theorem. Such unnecessary assumptions included the state space to be finite, discrete, or bounded, which all imply (albeit indirectly) that the expected first-hitting time $E[T]$ of the analyzed process is finite – a property that is necessary, as the theorem would not hold otherwise since it yields a finite upper bound. Our new theorem works for *any* process over $\mathbf{R}_{\geq 0}$ (Theorem 1). This means that we show that the finiteness of $E[T]$ already follows mainly from the drift condition (condition 2). This makes the theorem easier to use and less error-prone, as fewer requirements have to be checked.

Further, we provide examples that show that the theorem does not hold anymore if one of the two conditions are not met. For condition 2 this is trivial, as the bound provided does not make any sense without it. However, condition 1 is also necessary, as it provides (indirectly) a bound on how close the process can get to 0. If the process can get up to a value of $a \in \mathbf{R}^+$, then the drift can be at most a for condition 2 to hold. This implies a lower bound on the bound provided by the theorem, assuring that it cannot get below the actual expected first-hitting time of the process. These insights are very helpful when coming up with new drift theorems or trying to improve

existing ones, as one has a better understanding of what properties are important and how they interact with each other.

4 Future Work

Over the last three and a half years with the Research School, I co-authored many publications on EDAs, mainly analyzing the run time behavior of these algorithms. Hence, my current focus is on writing my PhD thesis.

Independently of my current activities, the introduction of the sig-cGA provides plenty of new research possibilities. One important goal are more lower bounds (especially on LeadingOnes) of EDAs in order to see if the sig-cGA is strictly better than other EDAs on certain functions. Further, the sig-cGA should be analyzed on more functions, for example, the Jump function or linear functions. In addition to that, the algorithm can be thought of as a regular EA (as briefly discussed in Section 3). Hence, it would be interesting to see if run time results of EAs carry over to the sig-cGA in the spirit of that the sig-cGA is only by a constant slower than most EAs. Since many run time results are known for EAs, this would increase the results for the sig-cGA tremendously.

References

[1] D.-C. Dang and P. K. Lehre. "Simplified runtime analysis of estimation of distribution algorithms". In: *Proc. of GECCO*. 2015, pages 513–518.

[2] B. Doerr and C. Doerr. "A tight runtime analysis of the $(1+(\lambda, \lambda))$ genetic algorithm on OneMax". In: *Proc. of GECCO*. 2015, pages 1423–1430.

[3] B. Doerr and M. S. Krejca. "Significance-based Estimation-of-Distribution Algorithms". In: *Proc. of GECCO*. 2018, pages 1483–1490.

[4] B. Doerr and M. Künnemann. "Optimizing linear functions with the $(1+\lambda)$ evolutionary algorithm – different asymptotic runtimes for different instances". In: *Theoretical Computer Science* 561 (2015), pages 3–23.

[5] B. Doerr, F. Neumann, D. Sudholt, and C. Witt. "On the runtime analysis of the 1-ANT ACO algorithm". In: *Proc. of GECCO*. 2007, pages 33–40.

[6] S. Droste, T. Jansen, and I. Wegener. "On the analysis of the $(1+1)$ evolutionary algorithm". In: *Theoretical Computer Science* 276.1-2 (2002), pages 51–81.

[7] T. Friedrich, T. Kötzing, and M. S. Krejca. "EDAs cannot be Balanced and Stable". In: *Proc. of GECCO*. 2016, pages 1139–1146.

[8] B. Hajek. "Hitting-time and occupation-time bounds implied by drift analysis with applications". In: *Advances in Applied probability* 14.3 (1982), pages 502–525.

[9] J. He and X. Yao. "A study of drift analysis for estimating computation time of evolutionary algorithms". In: *Natural Computing* 3.1 (2004), pages 21–35.

[10] J. He and X. Yao. "Drift analysis and average time complexity of evolutionary algorithms". In: *Artificial Intelligence* 127.1 (2001), pages 57–85.

[11] T. Jansen, K. A. D. Jong, and I. Wegener. "On the choice of the offspring population size in evolutionary algorithms". In: *Evolutionary Computation* 13 (2005), pages 413–440.

[12] J. Kacprzyk and W. Pedrycz, editors. *Springer Handbook of Computational Intelligence*. Springer, 2015. ISBN: 978-3-662-43504-5.

[13] T. Kötzing and M. S. Krejca. "First-Hitting Times Under Additive Drift". In: *Proc. of PPSN*. 2018, pages 92–104.

[14] M. S. Krejca and C. Witt. "Lower Bounds on the Run Time of the Univariate Marginal Distribution Algorithm on OneMax". In: *Proc. of FOGA*. 2017, pages 65–79.

[15] P. K. Lehre and P. T. H. Nguyen. "Improved Runtime Bounds for the Univariate Marginal Distribution Algorithm via Anti-Concentration". In: *Proc. of GECCO*. 2017, pages 1383–1390.

[16] P. K. Lehre and P. T. H. Nguyen. "Level-Based Analysis of the Population-Based Incremental Learning Algorithm". In: *Proc. of PPSN*. 2018, pages 105–116.

[17] A. Moraglio and D. Sudholt. "Principled design and runtime analysis of abstract convex evolutionary search". In: *Evolutionary Computation* 25.2 (2017), pages 205–236.

[18] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. ISBN: 978-0-521-47465-8.

[19] F. Neumann and C. Witt. "Runtime analysis of a simple ant colony optimization algorithm". In: *Algorithmica* 54.2 (2009), pages 243–255.

[20] M. Pelikan, M. Hauschild, and F. G. Lobo. "Estimation of Distribution Algorithms". In: *Springer Handbook of Computational Intelligence*. Edited by J. Kacprzyk and W. Pedrycz. Springer, 2015, pages 899–928. ISBN: 978-3-662-43504-5.

[21] *Proceedings of the 10th Ph.D. retreat of the HPI Research School on service-oriented systems engineering*. Universitätsverlag Potsdam, 2016. ISBN: 978-3-86956-390-9.

[22] *Proceedings of the 9th Ph.D. retreat of the HPI Research School on service-oriented systems engineering*. Universitätsverlag Potsdam, 2015. ISBN: 978-3-86956-345-9.

[23] D. Sudholt and C. Witt. "Update Strength in EDAs and ACO: How to Avoid Genetic Drift". In: *Proc. of GECCO*. 2016, pages 61–68.

[24] C. Witt. "Runtime analysis of the $(\mu + 1)$ EA on simple pseudo-Boolean functions". In: *Evolutionary Computation* 14 (2006), pages 65–86.

[25] C. Witt. "Upper bounds on the runtime of the univariate marginal distribution algorithm on OneMax". In: *Proc. of GECCO*. 2017, pages 1415–1422.

Event Handling in Business Process Enactment

Sankalita Mandal

Business Process Technology
Hasso-Plattner-Institut
Sankalita.Mandal@hpi.uni-potsdam.de

Business process management (BPM) enables modeling, executing and monitoring organizational processes to achieve certain business goals. Organizations continue to strive for agility and take advantage of the digital era to bring flexibility in their processes, for example by integrating complex event processing (CEP) techniques. Event handling specifies how a process interacts with its environment and how the environmental occurrences influence the execution of the process. Though highly expressive and feature-rich languages like BPMN exist for process specification, they still lack the flexibility required for event handling in different real-life scenarios. In this work, an event handling model is proposed that take into account the possibilities of event subscription at different points in time with respect to process execution. The model is grounded formally and provides mapping to Petri Nets as implementation semantics. Further, trace analysis ensures correct execution of process behavior while maintaining the temporal dependencies intact among event subscription, event occurrence, event consumption and event unsubscription.

1 Introduction and Motivation

Process implementation in an organizational context is model driven, where a process model defines a set of activities to achieve certain business goal(s) [16]. The activities are connected with causal and temporal dependencies for their execution using control-flow. The process model also specifies how a process is supposed to interact with its environment. The environmental occurrences are represented as events [7]. Process engines subscribe to event sources and react to events emitted by them following process specification. Often, a separate complex event processing engine is used to abstract from the complexity of connecting to different event sources, parsing events in different formats and aggregating events from multiple event streams [8]. The CEP engine or an event publisher registers subscription for specific event(s) and notifies the process engine when matching events occur [11].

Business Process Model and Notation (BPMN) [13] is the de facto standard for modeling and executing processes. BPMN includes explicit event constructs to depict the production and consumption of events, message flows to link external process participants, and control-flow routing based on events such as event-based gateways or boundary events. However, the event handling semantics in BPMN is quite limited while it comes to specify when to subscribe to an event source or when to stop listening to an event stream [12]. BPMN specification [13] states:

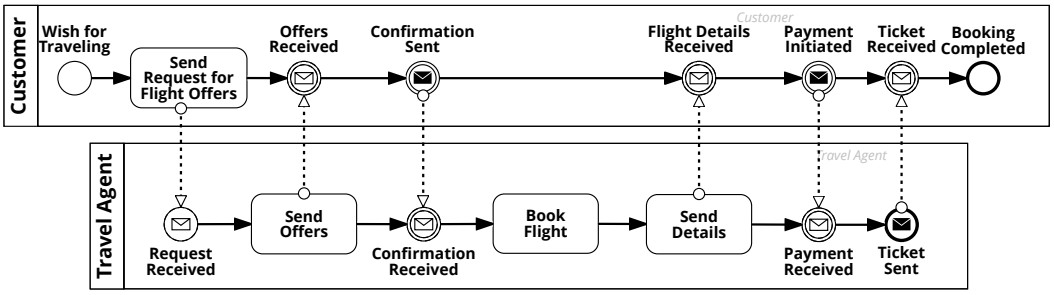


Figure 1: Process collaboration involving travel agent and customer.

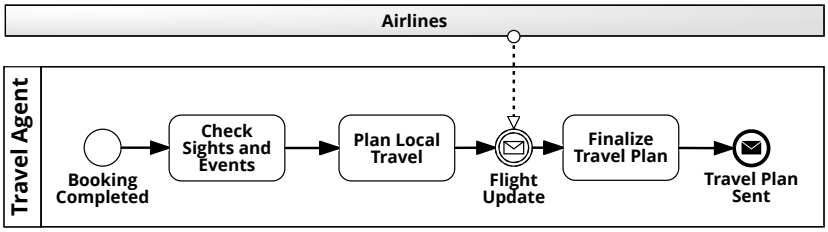


Figure 2: Process showing travel plan creation

'For Intermediate Events, the handling consists of waiting for the Event to occur. Waiting starts when the Intermediate Event is reached. Once the Event occurs, it is consumed. Sequence Flows leaving the Event are followed as usual.' [*13*] (*Sect. 13.4.2*)

According to BPMN, when the control-flow reaches the event construct, the node is enabled and the process instance waits for the event occurrence. Once it happens, the control-flow is passed to downstream activities. As a result, a process instance may not react to an event that occurred before the control-flow enabled the respective event node. The above semantics is a severe limitation as the event sources might be unaware of the process execution status and therefore the events can occur anytime irrespective of the process being ready to consume it. Figure 1 shows a process collaboration diagram for booking flights with several message exchanges between a travel agent and a customer. The event `ConfirmationReceived` can not occur before the previous activity `Send Offers` has been completed, as `Confirmation-Sent` has to be produced for receiving the confirmation, which in turn depends on the intermediate catching message event `OffersReceived` as a result of the activity `Send Offers`. Thus, the causal dependencies bound the order of event occurrences.

Additionally, the travel agent sends some local travel plan once the customer books the flight, as shown in Figure 2. The research about the sights and ongoing events at the destination and planning the travel accordingly take some time. After having the initial plan, the flight update is considered to make sure the time of arrival and departure and based on that, the plan is finalized. Here, the `FlightUpdate` can be released before or during the planning as the airlines (shown as a grey box as the internal processes are not visible to travel agent) is independent of the processes the travel agent might execute. In these situations, BPMN semantics restrict the process to access the events that have already occurred and are still relevant for the process execution.

The major share of adopting complex event processing concepts from a BPM perspective is application oriented, such as using external events to monitor business processes [1, 3, 8], predicting deviations [4], checking compliance to the process model [15] and so on. The CASU framework proposed in [5] talks about the (un)-subscription of start events to instantiate a process. Though our work is hugely inspired by this research, the work in this paper addresses the shortcomings of event handling semantics and proposes a formal operational semantics to BPMN events supporting high degree of flexibility. Possible usages of an intermediate event in process are defined as event constructs. The occurrence and consumption possibilities of events are considered based on a timeline starting from the initiation of process engine and ending at the termination of engine. To this end, a formal framework is proposed that specifies the semantics for event subscription, occurrence, consumption and unsubscription with the help of Petri Nets. Trace analysis shows correct execution behavior complying with the new semantics.

The remainder of the paper is structured as follows. section 2 introduces the fundamentals of business process model and Petri Nets later used for the event handling model. The conceptual framework for event handling is introduced in section 3. Next, the Petri Net mappings and trace analysis in section 4 show the correct execution behavior for each scenario derived from the framework. section 5 discusses the application of the concepts to two different use cases and finally, section 6 gives concluding remarks and outlines future research directions.

2 Foundations

This section presents a brief introduction to the concepts and formalism based on which the event handling model has been built up. Namely, we introduce business process management (BPM) and Petri Nets in this section.

Business Process Management. A process model consists of nodes and edges [16] where nodes can be activities, gateways and events. A process model is a blue print for a set of process instances which are the individual executions of this process. Each process instance consists of several activity instances which traverse through different life cycle states such as *ready*, *running*, *terminated* among others.

Business processes modeled with BPMN may include start, intermediate or end event constructs to represent the interaction with other processes as well as with the environment. These are different from transitional events logged during process execution. Rather, these event constructs represent something that has happened in a particular system or context [7] and can influence process execution to a great extent [10]. BPMN offers several event types to represent the interaction between a process and its environment. Start events are needed to instantiate a process model. An intermediate event can be produced by the process (throwing event) as well as it can be received by the process (catching event). A start event or an intermediate catching event can be engine generated, such as timer event or received from outside of the process pool, e.g., a message from another process, a sensor update, or simply

an email from a user. An end event is always a throwing event. Based on the above discussion, we formally define the relevant concepts for our work in the following. For simplicity, we consider only structurally sound processes.

A *Process Model* is a tuple $\mathfrak{M} = (N, cf)$ with

- a finite non-empty set of nodes $N = N_A \cup N_E \cup N_G$ where N_A, N_E and N_G are pairwise disjoint sets of the activities, the events and the gateways, respectively, and
- a control flow relation $cf \subseteq N \times N$.
- $N_E = \{e_s\} \cup E_I \cup \{e_e\}$, where e_s is the start event, E_I is the set of intermediate events, and e_e is the end event.
- $E_I = E_{IC} \cup E_{IT}$ where E_{IC} and E_{IT} are pairwise disjoint sets of intermediate catching events and intermediate throwing events, resp.
- The function $\mathcal{B} : N_A \rightarrow 2^{E_{IC}}$ maps the activities to the associated boundary event(s).
- $N_G = G_A \cup G_X \cup G_E$, where G_A, G_X, and G_E are pairwise disjoint sets of AND gateways, XOR gateways, and event-based gateways.

For an activity $A \in N_A$, let A_b, A_t, A_c be the beginning, termination and cancellation of A, respectively. A start event is always a catching event, whereas an end event is always a throwing event. The preset of a node $n \in N$ is defined as $\bullet n = \{x \in N \mid (x, n) \in cf\}$. The postset of a node $n \in N$ is defined as $n\bullet = \{x \in N \mid (n, x) \in cf\}$.

Petri Nets. Petri Nets are one of the most popular and standard techniques to represent the behavior of concurrent systems [9]. The net is composed of places and transitions, connected with directed arcs between them in a bipartite manner. The transitions represent active components of a system, such as activities, events or gateways in a process. On the other hand, the places are used to model the passive components, e.g., the input place models the precondition and the output place models the postcondition of a transition. We chose Petri Nets for our mapping since it gives clearer implementation semantics than BPMN. The Petri Net semantics used here follow the definitions proposed in [9] and [14].

A marking of a Petri Net signifies the system state. A marking is calculated using the distribution of tokens over the places of the net. The firing of a transition can change the marking, i.e., the state of the system. Firing of transitions are considered as atomic step. The behavior of the system is described by all firing sequences of a net that start with an initial marking. A single firing sequence is named as a *trace*. The relevant definitions are quoted in the following.

A *Petri net* is a tuple $\mathfrak{N} = (P, T, F)$ with

- a finite set P of places,

- a non-empty, finite set T of transitions, such that $T \cap P = \emptyset$, and
- a flow relation $F \subseteq (P \times T) \cup (T \times P)$.

A marking of $\mathfrak{N}$ is a function $M : P \rightarrow \mathbb{N}_0$, that maps the set of places to the natural numbers including 0. $M(p)$ returns the number of tokens on the place $p \in P$. Let $\mathbb{M}$ be the set of all markings of $\mathfrak{N}$. A Petri Net system is a pair $S = (\mathfrak{N}, M_0)$, where $\mathfrak{N}$ is a Petri net and $M_0 \in \mathbb{M}$ is the initial marking. A sequence of transitions $\sigma = t_1, t_2, \ldots, t_n, n \in \mathbb{N}_0$, is a firing sequence, if and only if there exist markings $M_0, \ldots, M_n \in \mathbb{M}$, such that for $1 \leq i \leq n$, transition t_i changes the system from $(\mathfrak{N}, M_{i-1})$ to $(\mathfrak{N}, M_i)$. The set of *traces* $\Im$ of S contains all firing sequences σ, such that σ is enabled in M_0.

3 Flexible Event Handling Model

The proposed event handling model supports flexible event handling notions introduced in [12] and extends it. We first classify the intermediate catching message events into four types of event constructs depending on their behavior. Next, we define the possible points in time when the subscription for events can be done. We assume that early subscriptions do not have any unresolved data dependency, i.e. subscription is independent of any data that must be generated during process execution and therefore, feasible even if it is done earlier than process instantiation. The BPMN to Petri Net formal mapping proposed by [6] is followed in our work for mapping the semantics of a process model to a Petri Net, added with the event handling semantics enhancing flexibility. The Petri Net mapping is discussed in detail in section 4.

3.1 Event Constructs

We define the following event constructs with respect to the position and behavioral semantics of an intermediate event.

Mandatory Event. An event $e \in E_{IC}$ is a mandatory event iff $\forall \sigma \in \Im$, $\exists j$ such that $t_j = e$. Let $E^M \subseteq E_{IC}$ be the set of mandatory events in $\mathfrak{M}$.

These are the events that have to occur in order to complete the process execution. If an event is located after an XOR gateway, then it is not mandatory for all instances of the process, but if an instance follows that particular branch, then the event becomes mandatory to complete the instance execution. Note that, this is not what we refer to as mandatory events. According to our definition, a mandatory event can be identified directly from the model, for example, an event in the main process flow or an event inside a parallel branch that is in the main process flow.

Boundary Event. An event $e \in E_{IC}$ is a boundary event iff $e \in image(\mathcal{B})$ where the function $\mathcal{B}$ maps the activities to its associated boundary event(s). Therefore, $image(\mathcal{B})$ is the set of events associated with activities. Let $E^B \subseteq E_{IC}$ be the set of boundary events in $\mathfrak{M}$.

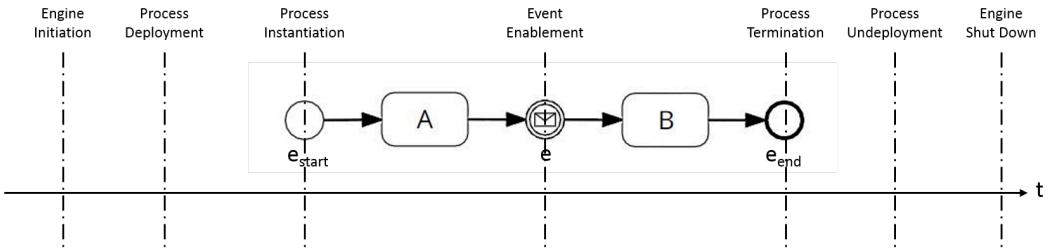


Figure 3: Process execution timeline

The interrupting boundary events in BPMN, once occurred, cancel the associated activity and trigger an exceptional branch. The relevance of the event occurrence here is the duration of the associated activity being in *running* state, i.e. the event must happen after the activity begins and before it terminates in order to follow the exceptional path. Since we are considering only sound processes, we do not include process patterns with non-interrupting boundary event.

Racing Event. If there exists a gateway $g \in G_E$ such that $\forall e_i, 1 \leq i \leq n \in \mathbb{N}, \bullet e_i = \{g\}$, then the events $e_1, e_2, ..., e_n \in E_{IC}$ are racing events. Let $E^R \subseteq E_{IC}$ be the set of racing events in $\mathfrak{M}$.

The event-based gateway in BPMN is a special gateway where instead of data, the event occurrence decides which branch to follow further. The gateway is immediately followed by the events and whichever event occurs first, the process takes the branch lead by that event. Therefore, the events after an event-based gateway are in a race with each other.

Exclusive Event. An event $e \in E_{IC}$ is an exclusive event iff e is in $\sigma = g_i, \ldots, e, \ldots, g_j$ such that $g_i, g_j \in G_X \wedge \nexists g \in G_X$ such that $g_i < g < e$. Let $E^X \subseteq E_{IC}$ be the set of exclusive events in $\mathfrak{M}$.

A specific process execution follows only one of the paths after an exclusive gateway. This makes the events on the branches after an XOR split and before an XOR join exclusive, i.e. when one of the branches is chosen, the events in other branches are not needed any more.

3.2 Point of Subscription

The milestones of process execution starting from the engine initiation until the engine is shut down is shown in Figure 3. An event independent of process causality can occur before, after or during the whole process execution, basically anytime throughout the timeline and beyond that. Since we assumed that there is no unresolved data dependency for the event subscriptions, we can subscribe to the events at several milestones. The notion of *Point of Subscription* (POS) listed below controls at which point in time events start to become relevant for a process.

POS1: At Event Enablement. A subscription is made only when the event construct is enabled by the control-flow. This is completely in line with the BPMN semantics and should be implemented when subscription for an event can be done only af-

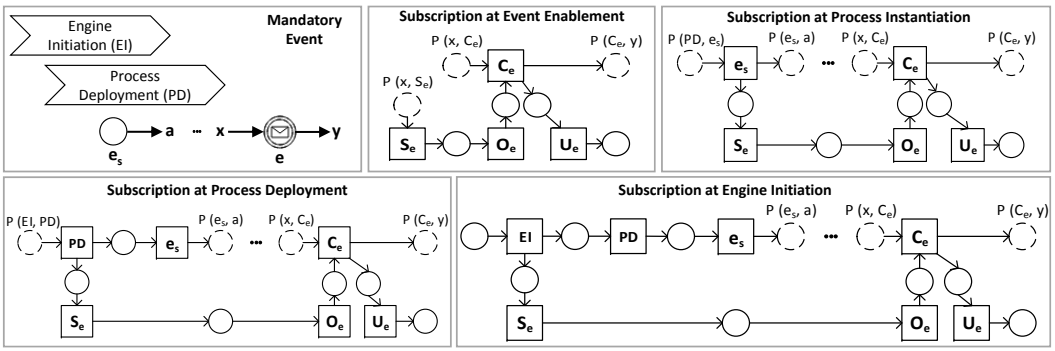


Figure 4: Petri Net modules for mandatory event construct

ter completing the previous activity. Subscription at event enablement means the following for the event constructs described before:
Given $e \in E_{IC} \setminus E^B$ and $x \in N_A \cup N_E$ such that $x\bullet = \{e\}$,
subscribe to e when x terminates.
Given $e \in E^B$ and $A \in N_A$ such that $A \rightarrow e$,
subscribe to e when A begins (A_b).

POS2: At Process Instantiation. A subscription is made as soon as the process is instantiated, i.e., $\forall e \in E_{IC}$, subscribe to e when e_s occurs. This is required when the subscription is done separately for each process instance, but the event can occur earlier than scheduled in the process.

POS3: At Process Deployment. According to subscription at process deployment, $\forall e \in E_{IC}$, subscribe to e at process deployment (`PD`). Here, the subscription for all the intermediate catching events are created as soon as one of the process patterns is deployed. This is almost always true for subscription to a start event, as it is needed for process instantiation. For an intermediate event, this should be implemented when all instances of a process model might need the event.

POS4: At Engine Initiation. A subscription is made by the process engine itself at the time when the engine starts running. So, following subscription at engine initiation, $\forall e \in E_{IC}$, subscribe to e at engine initiation (`EI`).

This is helpful in a situation where the engine already knows which events might be needed by the processes to be executed and already subscribes to the events. In such scenarios, an event information is often shared by several processes. When one process starts executing, it can then immediately access the events already occurred and stored by the engine.

4 Petri Net Mapping

We have defined the *Event Constructs* and *Points of Subscription* in the previous section. In this section we map each set of event construct and point of subscription to corresponding Petri Net modules. We take the BPMN to Petri Net mapping prescribed by [6] as basis and extend it to enable flexible event handling.

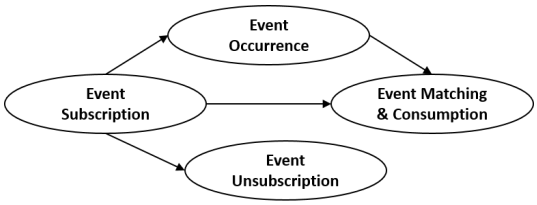


Figure 5: Dependencies among event subscription, event occurrence, consumption, and unsubscription

Figure 5 is an extended version of the causality proposed by [2] that said an event can only be consumed if both, a subscription has been issued earlier and the event actually occurred already. We argued that this provides only a partial order [12] and proposed that to make an event relevant for a process, the subscription should be done before the event occurrence. Yet, the unsubscription was not part of the model. As obvious, the subscription should exist before an unsubscription can take place. But an unsubscription is independent of the event consumption, or even event occurrence, e.g., if at certain point of process execution the event becomes irrelevant, unsubscription can be done. Formally, the temporal dependencies can be expressed as $S_e < O_e < C_e \wedge S_e < U_e$ where S_e denotes the subscription of event e, O_e denotes the occurrence of e relevant for the consumer process, C_e denotes the consumption of e, and U_e denotes the unsubscription of e.

In the current paper, we abstract from the details included in CEP engine or event buffering discussed in [12]. Also, we do not show the activity life cycle phases as separate transitions, unless they are needed explicitly (e.g., for boundary event). As our focus is on intermediate catching events, for start and end events, we only show the occurrence and abstract from other details. In general, the subscription for start events has to happen at the latest at process deployment, otherwise the process does not get initiated. On the other hand, the end event is a throwing event produced by the process, therefore we do not need any subscription for it. Additionally, we have the following assumptions:

- For any event, occurrence time coincides detection time.
- Unsubscription is done either at event consumption or at a point when the event becomes irrelevant to complete the process execution.

As explained in [6], x denotes the predecessor node of event e, y denotes the successor node of e and places with dashed borders mean they are not unique to one module. Note that, we use Petri Net semantics in our work to capture the behavior of a process engine with respect to one process model. To clarify, we show the milestones such as engine initiation `(EI)` or process deployment `(PD)` (represented in Figure 3) as Petri Net transitions. Therefore, the initial marking of the Petri Net has one token in the input place of `(EI)`. Process instantiation is denoted by start event e_s, followed by node a. A process ends with an end event e_e. For each intermediate catching event, the temporal order $S_e < O_e < C_e \wedge S_e < U_e$ holds. Additional restrictions on execution are discussed at the corresponding trace analysis sections. To summarize, we use

BPMN process model excerpts along with the engine execution milestones as our representation level and specify the corresponding Petri Net as the implementation level.

4.1 Petri Nets for Mandatory Event

Following *subscription at event enablement*, these are the steps to map a mandatory event construct to corresponding Petri Net module:

1. Event `e` is mapped to four separate transitions, subscription to `e` (S_e), occurrence of `e` (O_e), consumption of `e` (C_e) and unsubscription to `e` (U_e).
2. S_e has one input place to link with `x`, the predecessor node of `e`.
3. C_e has one input place to link with `x`.
4. C_e has one output place to link it to `y`, the successor node of `e`.
5. A flow is added from S_e to O_e.
6. A flow is added from O_e to C_e.
7. A flow is added from C_e to U_e.

For other points of subscription, *Step 2* is replaced as indicated in the following. The event construct and resulting Petri Nets are shown in Figure 4.

- *Subscription at process instantiation:* A flow is added from the transition for start event e_s to S_e.
- *Subscription at process deployment:* A flow is added from the transition for process deployment `(PD)` to S_e.
- *Subscription at engine initiation:* A flow is added from the transition for engine initiation `(EI)` to S_e.

Trace Analysis for Mandatory Event The temporal constraints complying with correct execution behavior for a process containing mandatory event e are given for each *POS* in the following.

- *POS1:* Subscription should be done only after the previous transition x is executed, i.e. $x < S_e$ must hold.
- *POS2:* Subscription should be done immediately after start event has occurred, the event is consumed after x is executed, i.e., $e_s < S_e < x < C_e$ must hold.
- *POS3:* Here, the subscription is done after process deployment but before process instantiation. Thus, $PD < S_e < e_s < x < C_e$ must hold.
- *POS4:* Subscription is done after engine initiation, before process deployment, i.e., $EI < S_e < PD < e_s < x < C_e$ must hold.

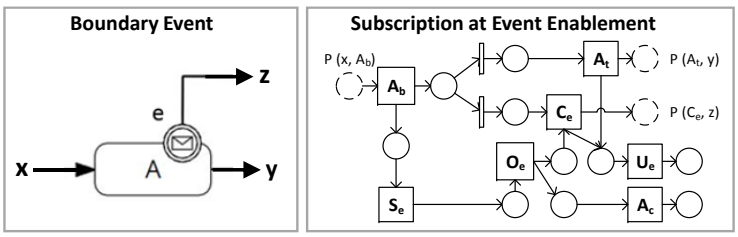


Figure 6: Petri Net module for boundary event construct

4.2 Petri Nets for Boundary Event

For a boundary interrupting event, even if the subscription is created earlier, the event occurrence is relevant only during the running phase of the associated activity. Therefore, for this event construct, subscription is recommended at event enablement. For the same reason, the unsubscription should be done either after consuming the event or as soon as the associated activity terminates. Figure 6 shows the resulting Petri Net module. The steps for mapping boundary event construct with subscription at event enablement are given below.

1. The event e is mapped to S_e, O_e, C_e and U_e.
2. The associated activity A is mapped to three transitions, A_b depicting the beginning of A, A_t depicting the termination of A and A_c depicting the cancellation of A.
3. A_b has one input place to link with x, the predecessor node of A.
4. A_t has one output place to link with y, successor of A (normal flow).
5. C_e has one output place to link with z, the successor node of e (exception branch).
6. A flow is added from A_b to S_e.
7. A flow is added from S_e to O_e.
8. A flow is added from O_e to C_e.
9. Another flow is added from O_e to A_c.
10. A flow is added from C_e to U_e.

Trace Analysis for Boundary Event $x < A_b < S_e \wedge O_e < A_c \wedge A_t < U_e$ should hold for a boundary event.

4.3 Petri Nets for Racing Event

For subscription at event enablement, once the control-flow reaches the gateway, all the events following the gateway are subscribed. Immediately after one of the events occurs, the other events are unsubscribed and the process takes the path lead by the

event occurred. To ensure the correct order of event consumption for other *POS*, the occurrence of an event passes the token to the input place for event consumption and at the same time, it also enables the unsubscription of all other events that were in race. The methodology for mapping a racing event construct to Petri Net following *subscription at event enablement* is as following:

1. The event e_i is mapped to two separate transitions, occurrence of e_i (O_{e_i}), and consumption of `e` (C_{e_i}).
2. Two additional transitions are introduced to represent the combined subscription to all the racing events $e_1, e_2, \ldots, e_n$ ($S_{e_1,e_2,\ldots,e_n}$), and the combined unsubscription to all the racing events ($U_{e_1,e_2,\ldots,e_n}$).
3. The subscription transition has one input place to link with g, the event based gateway.
4. Each consumption transition C_{e_i} has one output place to link with y_i, the successor node of e_i.
5. A flow is added from subscription to each occurrence transition O_{e_i}.
6. A flow is added from O_{e_i} to C_{e_i}.
7. A flow is added from each O_{e_i} to the unsubscription transition.

For other points of subscription, instead of the subscription transition, each consumption transition C_{e_i} is connected with an input place `P(g,`C_{e_i}`)` to link it to g, the event based gateway. Additionally, *Step 3* is replaced as indicated in the following.

- *Subscription at process instantiation:* A flow is added from the transition for start event e_s to the subscription transition ($S_{e_1,e_2,\ldots,e_n}$).
- *Subscription at process deployment:* A flow is added from the transition for process deployment `(PD)` to to the subscription transition.
- *Subscription at engine initiation:* A flow is added from the transition for engine initiation `(EI)` to the subscription transition.

Trace Analysis for Racing Event The temporal constraints for each *POS* are:

- *POS1:* $g < S_{e_1,e_2,\ldots,e_n}$
- *POS2:* $e_s < S_{e_1,e_2,\ldots,e_n} < g < C_i$
- *POS3:* $PD < S_{e_1,e_2,\ldots,e_n} < e_s < g < C_i$
- *POS4:* $EI < S_{e_1,e_2,\ldots,e_n} < PD < e_s < g < C_i$

4.4 Petri Nets for Exclusive Event

In essence, the exclusive event behaves like a mandatory event once the associated branch is enabled by control-flow. Therefore, the mapping of exclusive event construct to Petri Net realizing *subscription at event enablement* coincides with the Petri Net module for mandatory event construct for the same POS (see Figure 4). However, if subscription is done earlier, then all the events situated in different optional branches can occur before the control flow reaches the XOR gateway. In this case, as soon as the decision is taken at the gateway and one branch is selected, the event(s) in other branch(es) should be excluded, i.e., the unsubscription should be done for them. The steps for the mapping with *subscription at process instantiation* are given below:

1. The event e_j is mapped to S_{e_j}, O_{e_j}, C_{e_j} and U_{e_j}.
2. C_{e_j} has one output place to link it to y_j, the successor node of e_j.
3. A flow is added from start event e_s to S_{e_j}.
4. A flow is added from S_{e_j} to O_{e_j}.
5. A flow is added from O_{e_j} to C_{e_j}.
6. A flow is added from C_{e_j} to U_{e_j}.
7. An input place $\texttt{P}(d_i, U_{e_j})$ where $i \neq j$ is added to trigger the unsubscription of e_j when the first transition in any other branch after the gateway is fired.

For *POS3* and *POS4*, *Step 3* is replaced as following:

- *Subscription at process deployment:* A flow is added from the transition for process deployment `(PD)` to S_{e_j}.
- *Subscription at engine initiation:* A flow is added from the transition for engine initiation `(EI)` to S_{e_j}.

Trace Analysis for Exclusive Event The temporal constraints for each *POS* are:

- *POS1:* $g < S_{e_j}$
- *POS2:* $e_s < S_{e_j} < g < C_{e_j} \wedge d_i < U_{e_j}$
- *POS3:* $PD < S_{e_j} < e_s < g < C_{e_j} \wedge d_i < U_{e_j}$
- *POS4:* $EI < S_{e_j} < PD < e_s < g < C_{e_j} \wedge d_i < U_{e_j}$

To enable flexible event handling for any process, the events should be mapped to Petri Net in accordance with the associated methodology for that event construct and chosen point of subscription. The rest of the process should be mapped to Petri Net modules according to the methodology provided in [6]. Combining the modules along with process flow will result in the Petri Net for implementation.

5 Application to Use Case

In this section, we apply the concepts introduced so far to the travel agent's processes of booking a flight ticket (refer to Figure 1) and creating local travel plan (refer in Figure 2) introduced in section 1). We identify the intermediate catching events as *mandatory event construct*s.

The correct traces for the booking process are:
Sub. at Event Enablement: $EI, PD, RR, SO, S_{CR}, O_{CR}, C_{CR}, U_{CR}, BF, SD, S_{PR}, O_{PR}, C_{PR}, U_{PR}, TS$
Sub. at Process Instantiation: $EI, PD, RR, S_{CR,PR}, SO, O_{CR}, C_{CR}, U_{CR}, BF, SD, O_{PR}, C_{PR}, U_{PR}, TS$
Sub. at Process Deployment: $EI, PD, S_{CR,PR}, RR, SO, O_{CR}, C_{CR}, U_{CR}, BF, SD, O_{PR}, C_{PR}, U_{PR}, TS$
Sub. at Engine Initiation: $EI, S_{CR,PR}, PD, RR, SO, O_{CR}, C_{CR}, U_{CR}, BF, SD, O_{PR}, C_{PR}, U_{PR}, TS$

For the travel plan process, the correct traces are:
Sub. at Event Enablement:
$EI, PD, BC, CSE, PLT, S_{FU}, O_{FU}, C_{FU}, U_{FU}, FTP, TPS$
Sub. at Process Instantiation:
$EI, PD, BC, S_{FU}, CSE, PLT, O_{FU}, C_{FU}, U_{FU}, FTP, TPS$
$EI, PD, BC, S_{FU}, CSE, O_{FU}, PLT, C_{FU}, U_{FU}, FTP, TPS$
$EI, PD, BC, S_{FU}, O_{FU}, CSE, PLT, C_{FU}, U_{FU}, FTP, TPS$
Sub. at Process Deployment:
$EI, PD, S_{FU}, BC, CSE, PLT, O_{FU}, C_{FU}, U_{FU}, FTP, TPS$
$EI, PD, S_{FU}, BC, CSE, O_{FU}, PLT, C_{FU}, U_{FU}, FTP, TPS$
$EI, PD, S_{FU}, BC, O_{FU}, CSE, PLT, C_{FU}, U_{FU}, FTP, TPS$
$EI, PD, S_{FU}, O_{FU}, BC, CSE, PLT, C_{FU}, U_{FU}, FTP, TPS$
Sub. at Engine Initiation:
$EI, S_{FU}, PD, BC, CSE, PLT, O_{FU}, C_{FU}, U_{FU}, FTP, TPS$
$EI, S_{FU}, PD, BC, CSE, O_{FU}, PLT, C_{FU}, U_{FU}, FTP, TPS$
$EI, S_{FU}, PD, BC, O_{FU}, CSE, PLT, C_{FU}, U_{FU}, FTP, TPS$
$EI, S_{FU}, PD, O_{FU}, BC, CSE, PLT, C_{FU}, U_{FU}, FTP, TPS$
$EI, S_{FU}, O_{FU}, PD, BC, CSE, PLT, C_{FU}, U_{FU}, FTP, TPS$

Discussion. From the above presented traces it is evident that for the booking process, even if the subscription is done at different points, the event `CR` will always take place after the previous activity `SO` is executed and the event `PR` will always occur after `SD` is executed. The reason is the process causality explained in section 1 which restrict the temporal ordering of an event occurrence based on the completion of predecessor tasks. Existing BPMN semantics is perfectly applicable for these scenarios. However, for the local travel planning process, the traces show more variety. Depending on the time of subscription, the event integration gets more accommodating here,

i.e., even if the event `FlightUpdate` is published by the `Airlines` much earlier than the `Travel Agent` is ready to consume it, the information carried by the event can still be used when needed. In this case, the use of flexible event handling model decreases the probability of the process execution getting delayed or stuck forever due to the lag between event occurrence and consumption.

6 Conclusion and Future Work

Business processes extensively interact with its environment, represented as events, to be informed about the context and react to it. BPMN and such expressive modeling notations include event constructs to model and realize these interactions. However, for a communication model in a distributed setup, the existing semantics do not give the necessary flexibility. The BPMN assumption of *an event occurrence only after the event construct is enabled* restricts the communication possibilities between event producers and consumers where the separate entities are not necessarily informed about each others internal status. This can lead to missing out on a still relevant event and waiting for an already occurred event, resulting in process delay, even deadlock.

In this work, intermediate catching events are classified as four event constructs, namely, mandatory event, boundary event, racing event, and exclusive event. Next, four points of subscription are identified with respect to the process execution timeline – at event enablement, at process instantiation, at process deployment, and at engine initiation. The dependencies among event subscription, occurrence, consumption and unsubscription are analyzed and Petri Net mapping for each pair of event construct and point of subscription are described. Further, execution trace analysis on two use cases show that the proposed event handling model gives detailed semantics for a more flexible interaction between business processes and external events required for real world scenarios.

The formally grounded event handling model opens many significant research directions. Implementing the semantics will help to discover the technical challenges, if any. Next steps will include evaluating the model by exploring the behavior of a process in the light of the interactions with environmental events. The newly defined semantics along with the trace analysis is also planned to be used to verify the correctness of process specification.

References

[1] M. Backmann, A. Baumgrass, N. Herzberg, A. Meyer, and M. Weske. “Model-Driven Event Query Generation for Business Process Monitoring”. In: *Service-Oriented Computing – ICSOC 2013 Workshops*. 2013, pages 406–418. DOI: 10.1007/978-3-319-06859-6_36.

[2] A. Barros, G. Decker, and A. Grosskopf. “Complex Events in Business Processes”. In: *BIS*. 2007.

[3] A. Baumgrass, N. Herzberg, A. Meyer, and M. Weske. "BPMN Extension for Business Process Monitoring". In: *EMISA*. Lecture Notes in Informatics. Gesellschaft für Informatik (GI), 2014.

[4] C. Cabanillas, C. D. Ciccio, J. Mendling, and A. Baumgrass. "Predictive Task Monitoring for Business Processes". In: *BPM*. 8659. 2014, pages 424–432. DOI: 10.1007/978-3-319-10172-9_31.

[5] G. Decker and J. Mendling. "Process Instantiation". In: *Data Knowledge Engineering* 68.9 (2009), pages 777–792. DOI: 10.1016/j.datak.2009.02.013.

[6] R. M. Dijkman, M. Dumas, and C. Ouyang. "Semantics and Analysis of Business Process Models in BPMN". In: *Information and Software Technology* 50.12 (2008), pages 1281–1294. DOI: 10.1016/j.infsof.2008.02.006.

[7] O. Etzion and P. Niblett. *Event Processing in Action*. Manning Publications, 2010. ISBN: 978-1-935182-21-4.

[8] N. Herzberg, A. Meyer, and M. Weske. "An Event Processing Platform for Business Process Management". In: *EDOC*. 2013.

[9] M. Kunze and M. Weske. *Behavioural Models - From Modelling Finite Automata to Analysing Business Processes*. Springer, 2016. DOI: 10.1007/978-3-319-44960-9.

[10] D. C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley, 2010. ISBN: 0-201-72789-7.

[11] S. Mandal, M. Hewelt, and M. Weske. "A Framework for Integrating Real-World Events and Processes in an IoT Environment". In: *CoopIS*. 2017.

[12] S. Mandal, M. Weidlich, and M. Weske. "Events in Business Process Implementation: Early Subscription and Event Buffering". In: *Business Process Management*. 2017.

[13] OMG. *Business Process Model and Notation (BPMN), Version 2.0*. 2011.

[14] M. Weidlich. "Behavioural Profiles: A Relational Approach to Behaviour Consistency". PhD thesis. University of Potsdam, 2011.

[15] M. Weidlich, H. Ziekow, J. Mendling, O. Günther, M. Weske, and N. Desai. "Event-based Monitoring of Process Execution Violations". In: *BPM*. 2011, pages 182–198.

[16] M. Weske. *Business Process Management - Concepts, Languages, Architectures, 2nd Edition*. Springer, 2012. DOI: 10.1007/978-3-642-28616-2.

Scenograph: Fitting Real-Walking VR Experiences into Various Tracking Volumes

Sebastian Marwecki

Human Computer Interaction
Hasso-Plattner-Institut
sebastian.marwecki@hpi.uni-potsdam.de

When developing a real-walking virtual reality experience, creators generally design virtual locations to fit a specific tracking volume. Unfortunately, this prevents the resulting experience from running on a smaller or differently shaped tracking volume. To address this, we present a software system called Scenograph. The core of Scenograph is a tracking volume-independent representation of real-walking experiences. Scenograph instantiates the experience to a tracking volume of given size and shape by splitting the locations into smaller ones while maintaining narrative structure.

1 Introduction

The most immersive approach to experiencing virtual reality is to allow users to walk around in the tracking volume in a way that maps the virtual world one-to-one to the tracking volume. This approach, known as real-walking [13], can lead to higher immersion than in-situ walking (e.g., treadmills [3], walking in place [12]) and related VR navigation techniques (e.g., teleportation [2]).

The typical workflow for designing real-walking experiences is to initially determine the size and shape of the available tracking volume, such as the designer's research lab or some standardized installation like The Void, and then design the virtual world for that volume accordingly. Because of this, real-walking experiences tend to be specific to the size and shape of the tracking volume they were designed for.

Recent tracking technologies such as Oculus or Vive and online platform such as Steam allow users to bring VR experiences into their preferred environments (i.e., their living room), instead of having to go to the VR setup. Since designers of VR experiences cannot anticipate the amount or shape of users preferred space, the current approach of designing experiences that are tailored to the tracking space becomes impossible.

Creators of VR experiences at home are thus faced with a choice: (1) to artificially narrow down their market by designing for niche tracking volumes or (2) abandon real-walking – creators have picked the latter. We find ourselves in a situation where users have paid for a VR system capable of real-walking but have essentially no real-walking contents – they miss out on the extra immersion potentially available.

While researchers investigated how to reduce space demands for real-walking [15]), their proposed techniques are still based on tailoring the experiences to a specific tracking space. Techniques like redirected walking [7], flexible spaces [15]),

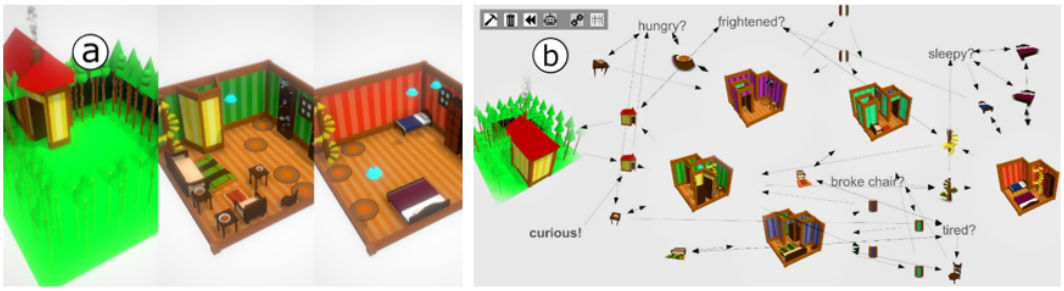


Figure 1: (a) This rendition of the fairy tale 'Goldilocks' consists of three 25m^2 locations filled with interactive assets. Unfortunately, specifying the tracking volume prevents the experience from running on smaller tracking volumes. (b) Here we used Scenograph to map 'Goldilocks' to an L-shaped 8m^2 space. While maintaining the narrative structure, it splits the three locations into six smaller ones, each fitting the new tracking volume.

or VirtualSpace [**XYZ**] considerably reduce space requirements, but ultimately do not address the problem that applications still assume a tracking volume with well-defined size and shape (e.g., VirtualSpace requires 16m^2 to let individual requirements sink to 4m^2). A substantial step in the right direction is Oasis [10], which enables customization of virtual worlds. Users scan their tracking volume with a depth camera, and from this data, Oasis creates a static virtual location that fits into the space.

In this paper, we present a system, Scenograph, that pushes this idea further, but instead of mapping static location to arbitrary tracking volumes, we map experiences. By making real-walking experiences independent of any particular tracking volume, Scenograph provides a crucial component for making real-walking experiences available to consumers.

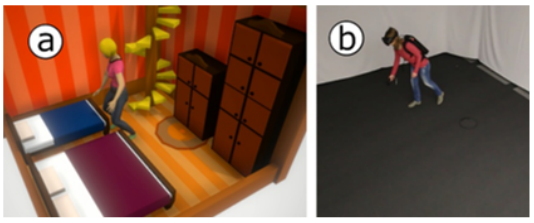


Figure 2: Figure 2: (a) Our adaptation of the fairy tale 'Goldilocks and the Three Bears', in which Goldilocks maliciously enters the home of the three bears, eats their porridges, sits on their chairs and sleeps in their beds. (b) The user in our tracking space of 5m x 5m.

2 SCENOGRAPH

Scenograph is a software system that offers a tracking volume-independent representation of real-walking experiences. Instead of designing for a tracking volume of specific size and shape, Scenograph lets designers specify an experience independent of the tracking volume. The virtual world is then automatically generated by Scenograph (while applying space compression techniques like [11], so that users can run the experience in their individual tracking volume. We demonstrate this process through an example application based on the 19th-century fairytale 'Goldilocks and the Three Bears' (see Figure 1 for our design). Naturally, Scenograph allows for the design of any application that can be procedurally generated, 'Goldilocks' is a good example as the narrative unfolds within one connected environment.

The interface to Scenograph is an editor, in which application designers define the unfolding of their real-walking experience. An experience is the designed arrangement of possible interaction sequences, where users switch between different virtual locations, or scenes to be more general. Users experience those scenes by real-walking as each scene has a designed arrangement of objects that users walk between. Scenograph encodes that experience in a bipartite graph (specifically, a petri-net). The instantiated graph maintains the arrangement of virtual scenes and objects. Goldilocks contains three scenes, a 'dark forest', the 'three bears' home', and 'upstairs bedroom' (see Figure 2). The 'home' scene is connected to three 'porridges' and three 'chairs'. Corridors are used as portals to switch between scenes making the switch unperceivable (like in [11]). Logical elements enable story progression when the user interacts with certain objects, e.g., after eating 'little bear's porridge', the user changes into the 'tired' state, so that users can now interact with the 'chairs'.

Internally, Scenograph represents the experience as a petri-net, as such it has transitions and nodes. Nodes are either spatial or logical. The spatial nodes are the scenes of the experience. The logical nodes are the states that enable story progression. The two kinds of nodes are connected by transitions, the virtual objects in the scenes. Transitions pass tokens between input and output nodes. The first 'door' for example, passes tokens from the 'forest' (spatial) and 'curious' (logical) nodes to the 'home' (spatial) and 'hungry' (logical) node. In Figure 3 we see that the 'porridge' then takes these tokens away from the 'hungry' state, but keeps it in the 'house', as the user remains there. A petri-net is suitable for encoding any direction a narrative may take (see Figure 4). This data structure can be expressed in any environment the application is developed in, here we used Unity3D and C# (see implementation section).

Scenograph adapts the scenes to the available space by splitting the nodes into multiple instances – this is the core value of the system. As seen in Figure 2, limiting the tracking volume from $25m^2$ to $8m^2$ results in splitting the 'home' node into four nodes. Figure 5 shows in detail how transitions get re-linked to maintain narrative structure. Scenograph takes the petri-net and the available tracking volume as input and transforms them into the new layout.

The end-user has no knowledge of Scenograph's data structure. The system merely requires a specification of the user's tracking volume in the form of a polygon. This

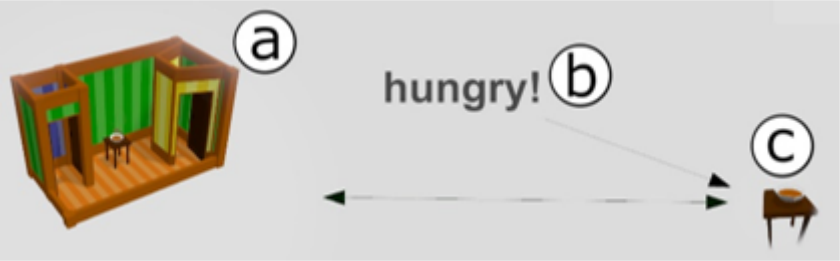


Figure 3: Scenograph encodes all possible interaction sequences in a graph. (a) Spatial nodes define the virtual scenes, here a part of the 'three bears' home'. (b) Logical nodes express states of the user and of objects, here whether the main character Goldilocks is 'hungry'. (c) Transitions, here a 'porridge', let users switch states and scenes, they populate the virtual scenes as objects. The 'porridge' can be interacted with, as both its requirements, 'home' and 'hungry', are satisfied.

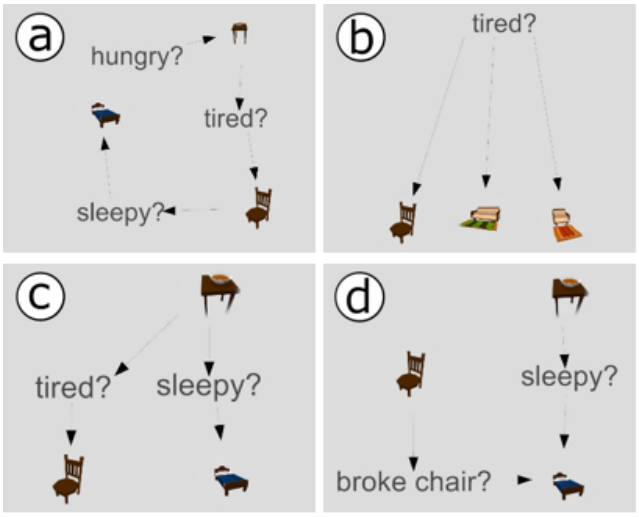


Figure 4: Here are examples of different narrative arrangements, which the petri-net representation allows. (a) Sequence: the 'porridge', 'chair' and 'bed' are accessed in a set order. (b) Conflict: the user needs to decide between one of the 'chairs'. (c) Concurrency: eating the 'porridge' allows using the 'chair' and the 'bed'. (e) Synchronization: sitting on the 'chair' and eating the 'porridge' is necessary before sleeping in the 'bed'. Other progressions are also possible ('confusion', 'merging', etc.).

Figure 5: Scenograph splits the 'home' node into two as the designed for 25m^2 get reduced to an L-shaped 12m^2. (a) This node has six transitions (three porridges followed by three chairs). (b) The porridges are placed in the first (upper) node, the chairs in the second, as they are interacted with after the porridges.

specification can be provided by a range of tracking technologies. The designer provides the volume-independent representation of the experience.

To create an experience, the designer must define all possible interaction sequences, i.e., the connections between spatial nodes, logical nodes, and transitions. This specification follows a bipartite graph structure, as nodes and transitions can only connect to each other, not to themselves.

Scenograph can also attribute multiple transitions to the same virtual object. In Figure 6, for example, one side of the 3D model of the big bed is considered 'mama bear's bed', the other side 'papa bear's bed'. A staircase can be used for 'going upwards' and 'going downwards', etc. The classic Goldilocks fairy tale is sequential (like most stories). Goldilocks eats the porridges, sits on the chairs, then lies on the beds. She always starts with the item of papa bear, then mama bear and finally the small bear. In our rendition of Goldilocks, we instead chose to allow for user decisions: any porridge is edible until small bear's porridge is eaten, sitting on any chair is possible until the user sat on small bear's chair, etc. A sequentially told story in Scenograph would provide an easy to solve problem (cut off the story when we run out of space, then split the location node). Scenograph usefulness increases with the complexity of the narrative, for example, when user decisions are involved, since the problem of where to split the nodes is then non-trivial. On the other end of the spectrum, if there is no logical connection at all (all objects can be interacted with anytime without consequences or story progression), then the decision where to split the node would be arbitrary.

Scenograph requires its applications to declare the space requirements for its virtual objects the transitions are paired with. Space requirements entail the objects length and width as hard constraints, and placement preferences (close to a wall, middle of the room, etc.) as soft constraints with a cost function.

Scenograph requires a specification of the given tracking volume in the form of a polygon, as well as the resolution into which the space gets virtualized, which is provided by the application. In our lab setup we have 5m x 5m available, and our

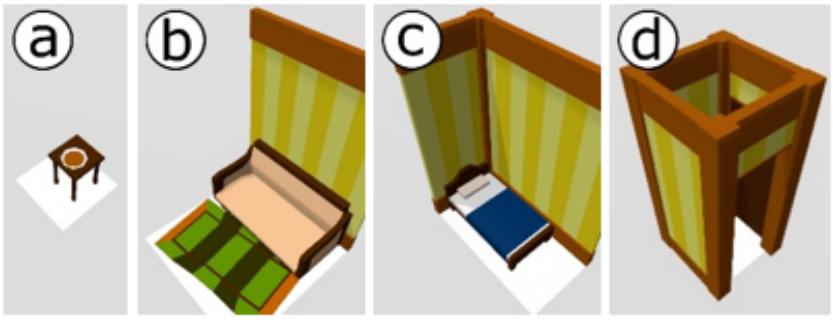


Figure 6: (a) A 'porridge' is a virtual object that requires $1m^2$, (b) 'papa chair' needs $4m^2$ and a wall, (c) 'little bed' 1m x 2m and a corner, (d) a generic corridor-portal connecting scenes takes $1m^2$ and a corner.

example applications is designed for a resolution of 1m x 1m so that Scenograph tessellates the space into 5 x 5 tiles.

The generated chunks of tiles are allocated to the system for the generation of the virtual scenes. Different physical setups will thus result in different scenes. Note that each experience requires a minimum size based on its largest virtual object, e.g., for Goldilocks this is the sofa with 2m x 2m. Scenograph allows to switch setups at runtime, for example if space gets occupied or freed up suddenly. In this case Scenograph reinstantiates the experience, however, it maintains the current logical and spatial nodes (e.g., 'frightened' and 'upstairs bedroom').

Scenograph needs to determine if all nodes can support their transitions, i.e., if the virtual objects' can be packed together onto the space the scene is given. The system offers different packing algorithms, ("best-fit", using simulated annealing), or random placement ("first-fit", using random placement). Given a smaller or differently shaped tracking volume, the packing algorithm might not find a solution and the node is then not able to support its transitions. In Figure 5 we cannot pack three 'porridges' and three 'chairs' into L-shaped $12m^2$, thus Scenograph splits the 'home' node into two.

To determine the number of splits per node and the distribution of transition onto split nodes Scenograph uses divisive hierarchical clustering. The distance computation between transition pairs, required for our hierarchical clustering, uses a simple semi-decision technique (distance of 1 if two transitions can be interacted with in any order, 2 if one transition needs to be interacted with after the other, 3 for neither). Scenograph now needs to evaluate which clustering to take for each node. Scenograph cannot linearly iterate through each node separately to find the right clustering, as some virtual objects need to be instantiated on the same tiles in more than one scene (transitions with different spatial nodes as input and output, such as a door). This means that Scenograph needs to consider all possible clusterings for all nodes in parallel, making the packing problem 3-dimensional (width, depth, occurrence in nodes). Scenograph iterates through all possible clustering in an informed manner. The number of clusters and therefore the potential connected scenes to consider is exponential in the number of virtual objects. Our hierarchical

clustering does not reduce this amount, it merely sorts all potential clusterings based on the conceptual distance between transitions. The number of virtual objects and thus of potential clusters is different for each scene. For example, our three nodes have one transition ('forest' has a door leading to 'home'), eight transitions ('home' has 3 'porridges', 3 'chairs', 1 'door', 1 'stairway') and three transitions ('upstairs bedroom' has 2 'beds', 1 'stairway'), leaving $20 + 27 + 22 = 512$ possibilities. Each possibility corresponds to a certain clustering depth per node, which we represent using a mixed radix numeric system (e.g., 112613 corresponds to splitting the second node in two). We iterate through this numeric system linearly first based on the checksum of this clusterings number (to reduce the number of nodes) and then on its order within the hierarchical clustering (maximizing proximity of virtual objects that are also conceptually close).

The virtual scenes are now generated. Each virtual object is placed onto the space the packing algorithms allocated for it. Afterwards, the rest of the scene is generated. While an application may create the visuals for each scene itself, Scenograph offers some default procedural generation algorithms to create the scene automatically. The application just provides the decorative objects, walls, floors, etc., together with placement constraints (e.g., a wall element with a window cannot be used next to occupied tiles). Scenograph loads and unloads scenes dynamically depending on the users' interaction with the virtual objects or where they walk. Scenograph uses corridors similar to "impossible spaces" to connect scenes [11], which serve as portals or locks. The L-shaped $12m^2$ space in Figure 5 can thus be used twice to fit the 'porridges' as well as the 'chairs'. Less overlap makes the technique less perceptible. However, since we focus on small spaces, Scenograph here fully overlaps the scenes.

The process of our system can be summarized as seen in Figure 7.

```
s_n = Application.GetSpatialNodes()
for all spatial nodes s_n
    d_n = GenerateDendrogram (s_n)
r = Application.GetResolution()
t = GetAvailableSpace(r)
threshold = Application.PackingThreshold
iterate through cluster possibilities
    c_m = CurrentClustering
    s_n c_m = SplitNodesForClustering(d_n, c_m)
    if(not PackingPossible(s_n c_m))
        reject s_n c_m
    if(PackingValue(s_n c_m) >= threshold)
        pick s_n c_m
for all s_n c_m
    e_n= GenerateScene(s_n c_m)
LinkScenes(e_n, Application.Preference)
```

Figure 7: The process of our system.

The system was implemented in C#, the example application and editor interface in Unity3D. ALGLIB [1] was used for clustering. To allow researchers to replicate our work, we provided the full source code online [8].

3 Contribution

Scenograph allows designers of real-walking virtual reality experiences to reach users with tracking volumes of arbitrary size and shape. The core of Scenograph is a tracking volume-independent representation of real-walking experiences. This representation is not spatial until instantiated into tracking volumes of specific size and shape. Scenograph thereby lays the groundwork for real-walking experiences to reach the consumer market.

4 Related Work

While walking in VR can be enabled by treadmills [3], it is more often simulated with techniques such as walking in place [12] or teleportation [2]. However, real-walking, a one-to-one mapping of physical to virtual motion, leads to the highest user satisfaction [13]. Real-walking has a high space demand, which researchers have tried to reduce with several techniques (for an overview see [14]). Resetting [16] rotates and virtually repositions users once they hit the tracking volume's borders. Seven league boots [5] scales the virtual motion. In one way or another, these and related techniques perceptibly interrupt or alter the one-to-one mapping of physical to virtual motion, which leads to a reduction of the immersive quality of walking.

With redirected walking Razzaque et al. [7] break this one-to-one mapping unperceivably and fold long walking paths into limited tracking space, thus lowering the required amount of space for real-walking. Redirected walking benefits from the a priori knowledge of the virtual environment, onto which the walking paths can be mapped. Even with the constraint of a priori setting of walking paths, this approach is subperceptible only for large tracking volumes (e.g., 4m x 10m [7]). For smaller spaces, complementary fallback techniques such as resetting [16] are needed, which disrupt the walking experience.

Instead of reducing space demand by breaking the mapping of physical to virtual motion, designers can alter or influence the virtual world. VirtualSpace [6] packs more people into limited space, reducing the space demand to $4m^2$ per user. Collisions are avoided by design; VirtualSpace requires its applications to adhere to an API to dynamically set virtual obstacles and goals that redirect their respective users. However, this technique still assumes a tracking volume of specific size for its applications (in this case $16m^2$). In impossible spaces, Suma et al. [11] lets virtual rooms unnoticeably overlap to compress space. The layout can be dynamically geerated, such as in Vasylevska et al.'s [15] flexible spaces. As impossible spaces uses change blindness, it requires large amounts of space to be sub-perceptible (9m x 9m). However, even when the overlap is noticeable this technique still enables enjoyable experiences, like the commercially available game "unseen diplomacy", a game with relative success that uses only a relatively small space (4m x 3m). The major drawback here is that this game, like VirtualSpace, was designed with a fixed tracking volume in mind. As argued in the introduction, this makes the game unsuitable for

a lot of setups; either it does not make use of possible surplus space of the user or it does not even run at all for tracking volumes that do not fit the required 4m x 3m.

An important step towards altogether circumventing the problem of a virtual scene not fitting into the tracking volume has been taken in Oasis [10]. Sra and colleagues adapted virtual scenes to fit rooms of arbitrary shape. Conceptually, this widens the possibilities for space setups, also to mobile scenarios. This concept is also known as procedural content generation (PCG).

Techniques using PCG for real-walking, however, do not take the creation or maintenance of a coherent narrative into account. This points to a problem of PCG in general – it is designed to quickly generate variance in the virtual scenes and is usually not used for story focused experiences. Translating this problem into real-walking applications for VR: when PCG is applied to generate a virtual scene there are two constraints, tracking volume as well as maintaining the virtual narrative.

Our system conceptually borrows from the field of operating systems; the software, here our 'Goldilocks' application, is abstracted so that it can run on arbitrary physical hardware, in this case, arbitrary physical tracking volume. Specifically, it links compiled and assembled objects together and loads them onto physical space, the hardware. This abstraction of hardware and software allows for hardware changes (space variance) and code survivability (preservation of the experience). In this regard, Scenograph performs as an integral part of operating system for real-walking in VR and enables development of software independent of the hardware.

5 Discussion

Our main finding is that Scenograph can create real-walking experiences in VR for tracking volumes of any size and shape. When comparing Scenograph's experiences to two commonly used locomotion techniques, namely instant teleportation [2] and scaling the mapping of physical to virtual motion (e.g., [5]), experiences were rated to be more realistic, leading to the conclusion that Scenograph degrades more gracefully with limitations of tracking volume. (Study omitted in this report, please see UIST'18 publication for further details). Space compression impacts realism differently than enjoyment. Both teleportation and motion scaling still provided enjoyable experiences to the participants. Also, the size of the tracking volume did not measurably impact enjoyment (comparing $9m^2$ to $25m^2$). The number of participants might have played into not discovering the hypothesized differences, but it seems that even when space compression is high and becomes more perceivable to the user (teleport, stronger motion scaling, higher overlap of impossible spaces) it affects realism first. Only when the compression becomes very high (motion scaling for L-shaped $8m^2$) does it also impact enjoyment. Based on our results, we can thus only make claims about Scenograph's impact on realism.

For future work, we imagine integrating not only variance in tracking volume, but variance in the number of users within the tracking volume. While VirtualSpace [6] already addressed this this issue, the arcade-style applications did not contain a narrative. Building on Supple++ [4], we also consider binding in user capabilities

and preferences for achieving tighter or looser packing of virtual objects or speeding up narrative and altering story arcs. While binding in physical props (e.g., "substitutional reality" [9]) and automatic layouting (e.g., "flexible spaces" [15]) has already been addressed with regard to real-walking, future work should address this also with regard to narrative structure.

6 Conclusion

We presented Scenograph, a software system that supports the design of real-walking experiences, which can adapt to any tracking volume's size and shape. Scenograph achieves this by representing the experience in a petri-net. Based on this representation and a given tracking volume the system generates a set of virtual scenes that guarantees the preservation of the experience for each user's individually available space. This strategy provides more realistic real-walking experiences than commonly applied techniques for variance in tracking volume. Real-walking experiences are typically designed with a specific tracking volume in mind. Scenograph allows users with constrained tracking volume to also have these experiences.

References

[1] *Alglib, Cross-platform Numerical Analysis and Data Processing Library*. URL: http://www.alglib.net/ (last accessed 2018-09-30).

[2] E. Bozgeyikli, A. Raij, S. Katkoori, and R. Dubey. "Point & Teleport Locomotion Technique for Virtual Reality". In: *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play – CHI PLAY '16*. 2016, pages 205–216. DOI: 10.1145/2967934.2968105.

[3] R. P. Darken, W. R. Cockayne, and D. Carmein. "The Omni-directional Treadmill: A Locomotion Device for Virtual Worlds". In: *Proceedings of the 10th annual ACM symposium on User interface software and technology – UIST '97* (1997), pages 213–221. DOI: 10.1145/263407.263550.

[4] K. Z. Gajos, J. O. Wobbrock, and D. S. Weld. "Automatically Generating User Interfaces Adapted to Users' Motor and Vision Capabilities". In: *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*. 2007, pages 231–240. DOI: 10.1145/1294211.1294253.

[5] V. Interrante, B. Ries, and L. Anderson. "Seven League Boots: A New Metaphor for Augmented Locomotion through Moderately Large Scale Immersive Virtual Environments". In: *2007 IEEE Symposium on 3D User Interfaces*. 2007. DOI: 10.1109/3DUI.2007.340791.

[6] S. Marwecki, M. Brehm, L. Wagner, L.-P. Cheng, F. '. Mueller, and P. Baudisch. "VirtualSpace – Overloading Physical Space with Multiple Virtual Reality

Users". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, 241:1–241:10. DOI: 10.1145/3173574.3173815.

[7] S. Razzaque, Z. Kohn, and M. C. Whitton. "Redirected Walking". In: *Proceedings of EUROGRAPHICS*. 2001, pages 105–106.

[8] *Scenograph, Online Repository*. URL: https://github.com/sebastianmarwecki/Scenograph (last accessed 2018-09-30).

[9] A. L. Simeone, E. Velloso, and H. Gellersen. "Substitutional Reality: Using the Physical Environment to Design Virtual Reality Experiences". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015, pages 3307–3316. DOI: 10.1145/2702123.2702389.

[10] M. Sra, S. Garrido-Jurado, and C. Schmandt. "Procedurally Generated Virtual Reality From 3D Reconstructed Physical Space". In: *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology - VRST '16*. 2016, pages 191–200. DOI: 10.1145/2993369.2993372.

[11] E. A. Suma, Z. Lipps, S. Finkelstein, D. M. Krum, and M. Bolas. "Impossible Spaces: Maximizing Natural Walking in Virtual Environments with Self-Overlapping Architecture". In: *IEEE Transactions on Visualization and Computer Graphics* 18.4 (2012), pages 555–564. DOI: 10.1109/TVCG.2012.47.

[12] J. N. Templeman, P. S. Denbrook, and L. E. Sibert. "Virtual Locomotion: Walking in Place through Virtual Environments". In: *Presence: Teleoperators and Virtual Environments* 8.6 (1999), pages 598–617. DOI: 10.1162/105474699566512.

[13] M. Usoh, K. Arthur, M. C. Whitton, R. Bastos, A. Steed, M. Slater, and F. P. Brooks. "Walking > Walking-in-place > Flying, in Virtual Environments". In: *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH '99* (1999), pages 359–364. DOI: 10.1145/311535.311589.

[14] K. Vasylevska and H. Kaufmann. "Compressing VR: Fitting Large Virtual Environments within Limited Physical Space". In: *IEEE Computer Graphics and Applications* 37.5 (2017), pages 85–91. DOI: 10.1109/MCG.2017.3621226.

[15] K. Vasylevska, H. Kaufmann, M. Bolas, and E. A. Suma. "Flexible Spaces: Dynamic Layout Generation for Infinite Walking in Virtual Environments". In: *IEEE Symposium on 3D User Interface 2013, 3DUI 2013 - Proceedings*. 2013, pages 39–42. DOI: 10.1109/3DUI.2013.6550194.

[16] B. Williams, G. Narasimham, B. Rump, T. P. McNamara, T. H. Carr, J. Rieser, and B. Bodenheimer. "Exploring Large Virtual Environments With an HMD When Physical Space is Limited". In: *Proceedings of the 4th symposium on Applied perception in graphics and visualization - APGV '07*. 2007, page 41. DOI: 10.1145/1272582.1272590.

Employing Software Development Data to Drive Process Change

Christoph Matthies

Enterprise Platform and Integration Concepts
Hasso-Plattner-Institut
christoph.matthies@hpi.de

Modern software is not produced by single individuals, but by teams. The form that collaboration in these software development teams takes, how team members work together, organize tasks and responsibilities and coordinate themselves impacts the quality of the product being produced. Therefore, practicing and upholding an effective development process is crucial for team success. These processes need maintenance, requiring study and continuous improvement. This is true both for education settings, where students are taught processes and best practices, as well as in professional software development contexts. The ideas of iterative improvement and self-reflection are at the core of agile software development methodologies, which have become widespread in industry. However, these methods do not specify exactly how improvement should be achieved, how it should be tracked or measured. In order to find improvement possibilities or weaknesses, teams rely on their personal impressions of past development iterations or mentoring by a knowledgeable third party. However, these methods mostly depend on subjective perceptions of team members and their results are hard to quantify. Therefore, after process changes have been adopted, it may be difficult to assess their impact and whether the action taken resolved the identified issue. In this research, we propose tackling these challenges by supplementing existing process improvement techniques with approaches leveraging the wealth of information contained within the development data created by development teams. This data, such as commits, test runs, issues or results of static analyses are already being produced during the regular work of modern software development teams, be they students or professionals. By collecting, aggregating, linking and analyzing this mostly untouched treasure trove of team data, insights into the development process of teams can be generated. Teams can be enabled to better reflect on their own executed processes and opportunities for discussions on the basis of data can be fostered.

1 Introduction

How the methods used to construct software by development teams can be studied and analyzed in order to improve them is one of the core questions of the Software Process Improvement (SPI) field [18]. Over the decades since the research in the field started, many frameworks have been proposed, success factors were studied and experiences were reported. A mapping study of publications over the last 25 years by Kuhrmann et al. found that the field was still shaped by experience reports and solution proposals, which make up about two-thirds of studied publications. The authors identify that much of the research suffers from missing evidence. Recent

work has also focused on the application of SPI in the context of agile software development, pointing out that a main challenge is to define the role of SPI in an agile context where people and interactions are more valued than process and tools [21]. In this research, we focus on how theses identified issues of basing analysis on data as well as applying SPI to agile contexts can be tackled. We chose an educational context for our preliminary studies, as this allows rapid changes in experimental design, easy iteration and control of the context that studies are performed in. In this paper, we describe a case study on how software development artifacts, created during students' project work in a software engineering course, can be used to check educators' assumptions on student behavior. The course's setting of collaborative software engineering in a simulated real-world scenario is ideally suited for collecting development data. During the project work, students use common development tools such as version control systems (VCS), issue trackers and Continuous Integration services. The artifacts produced using these systems, i.e. commits in a VCS containing code changes and descriptions, contain a large amount of information on how students work and collaborate in their groups [20, 22].

1.1 Research Questions

The following research questions (RQ) guide our work:

RQ1 How can surveys be used to gauge students perceptions of changes in course design over time?

RQ2 What data can be collected from software development artifacts created by students during a classroom course?

RQ3 What metrics can be applied to student development data to gauge changes in student behavior during project work?

2 Case Study Context

The undergraduate software development course *Softwaretechnik II* described in this case study is repeatedly run in the winter semester with a length of 15 weeks and was most recently taught in the winter semester of 2017/18. Course installments prior to the winter semester of 2014/15 taught exclusively the Scrum methodology [23]. However, Lean development approaches, such as Kanban [27], have gained popularity in industry [10, 25]. Therefore, in an ongoing effort to keep the course as relevant and closely related to real-world scenarios as possible, the practice of Kanban was included. Students employed the Scrum methodology at the beginning of the course, before switching to Kanban. In comparison to Scrum, Kanban is considered less authoritative and prescriptive, having fewer rituals and rules than Scrum [8]. In order to improve learning results, it is therefore advisable to introduce Kanban after students have already gained experience with the more structured Scrum method [11]. Iterations of the course prior to the winter term 2015/16 did not include Kanban and

focused solely on the application of Scrum. The inclusion of Kanban in the course is a major change as it impacts how students collaborate, plan their work and organize their team structures and meetings. After course participants have become familiar with the Scrum process and their teams, i.e. after they have reached the *norming* stage of group development [4], and have developed a cohesive group, Kanban and its practices are introduced in a lecture. The concepts of Kanban such as the Kanban board, the idea of workflow visualization and the guiding principle of limiting work in progress (WIP) [1] are introduced. We encourage students to try out and apply these new ideas in their teams. Participants employ Kanban for the last iteration of the project, instead of a final Scrum sprint.

3 Surveys

When trying to assess the impact of changes in curriculum design and whether the expected changes to student behavior took place, students' perceptions can be collected through the use of surveys.

3.1 End-of-term Survey

As part of an ongoing effort to collect feedback from students to improve teaching and university courses, standardized end-of-term surveys were conducted in all iterations of our software engineering courses in the years 2013 up to 2018. This has become standard practice for educational institutions to evaluate teaching quality [5]. The survey is administered online before students receive their final course grades to prevent interference. The survey collects perceptions of students on a range of topics, including satisfaction with the course in general, perceived importance of course contents and satisfaction with mentoring. An extract of the questions relevant to student satisfaction with the project work and the course over the iterations of the course is shown in Table 1.

Table 1: Questions and statements of the end-of-term survey

#	Survey item
1	The course was fun
2	The course motivated me to delve deeper into the discussed topics
3	I learned a lot in the course
4	The course is important to my course of studies
5	The course was well structured
6	The topics of the course were well chosen
7	How would you rate the course overall?

Questions and statements could be rated on a scale of "fully agree"/"great" to "totally disagree"/"bad". Results of the anonymous survey are presented to course instructors in aggregate form, with ratings mapped to German school grades. The grade 1 ("very good") is the best, with the grade 5 ("inadequate") signifying a fail.

Results Average ratings for all installments of the course showed overwhelmingly positive perceptions of the course and its content, see Figure 1. Very few questions

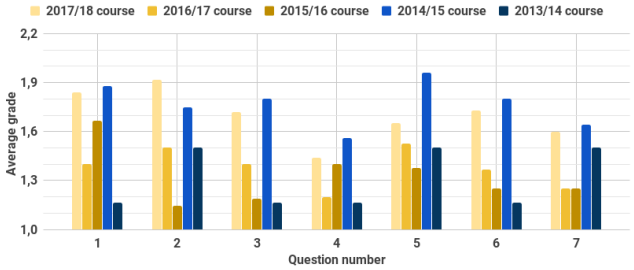


Figure 1: Mean grades given to course installments by students after the course's end. German school grades: 1 is the best grade, i.e. the lower the better. Courses in 2013/14 and 2014/15 (blue) employed solely Scrum, the others (yellow) employed both Scrum and Kanban.

were answered with mean scores larger than 2 ("good"). While this is satisfying to see as far as student satisfaction goes it also means no significant change can be detected in student satisfaction between the courses employing Kanban and those that did not. The variance in answers is very low, as students rated aspects of the course overwhelmingly as very good (1) to good (2). Therefore, we devised a more specific survey.

3.2 Kanban Survey

In the second installment of the software engineering course that used Kanban (in 2016/17), we conducted a voluntary, anonymous online survey among all students after course completion, in addition to the regular end-of-term surveys. The first course that introduced Kanban (in 2015/16) using newly created teaching materials had received critical comments from students in oral feedback sessions, which we addressed in the following course. The Kanban survey focused on students' perceptions of Kanban as well as the advantages and drawbacks of the introduced practices and methods. While the survey was designed to elicit responses to the details of how Kanban was introduced in the course, it also explicitly included questions on whether and how students' workflows were adapted when changing from Scrum to Kanban processes. These questions were designed to better understand whether the expected changes in process had actually taken place. The survey questions related to process change are listed in Table 2.

Table 2: Questions related to Kanban adoption of the anonymous online student survey performed at the end of the 2016/17 software engineering course

#	Type	Question
1	5-point scale	Was the Kanban week at project end more useful and productive then a last week of Scrum?
2	5-point scale	Did you have to adapt your workflow for the Kanban week?
3	free text	What were the biggest advantages and disadvantages of using Kanban in your team?
4	multiple choice	How did user stories change from using Scrum to Kanban?
5	5-point scale	Would you recommend using Kanban to the participants of next year's course?

Table 3: Summarized answers of participants to the 5-point Likert scale questions of the survey. Answer possibilities: 1 (strong no) to 5 (strong yes).

#	Question Topic	Mean	Std. Dev.	10% Trim. Mean	Median	Range
1	Kanban week preferred over another Scrum week?	4.08	1.38	4.30	5.00	4.00
2	Was the workflow adapted?	3.83	1.11	4.00	4.00	4.00
5	Recommended for next year?	4.33	0.98	4.50	5.00	3.00

The survey consisted mainly of questions that could be answered using a 5-point Likert scale, ranging from 1 (strong no) to 5 (strong yes), with 3 being neutral. Additionally, the survey included free text questions as well as a multiple choice question to gather more detailed insights. It was possible to submit the survey with missing answers.

Results Overall, 18 students, 17 men and 1 woman, answered the questionnaire. All questions featuring the Likert scale were answered by all participants. Table 3 contains a summary of the collected answers. Concerning the change of Scrum to Kanban methods (question 2), students on average stated that they had adapted their workflows, see Figure 2. The high mean value (4.08), as well as the median of 5 (highest agreement), point to students having adapted their workflow in a reflected manner. The overall positive student attitude towards group software development methodologies was also reflected in the answers to questions 1 and 6, regarding the preference of Kanban over Scrum for the last iteration as well as recommending the course for next year's students. Surveys participants indicated that they would strongly recommend the usage of Kanban to the next cohort of students of the soft-

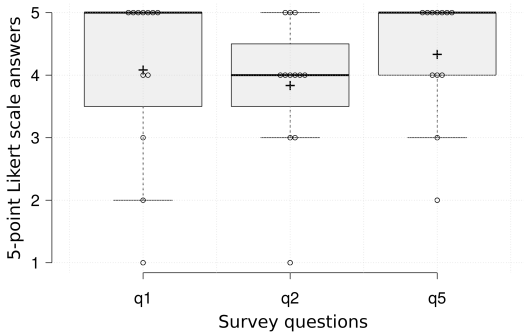


Figure 2: Summary of answers to questions 1, 2 and 5 as a box plot. Center lines show the medians, box limits indicate the 25th and 75th percentiles. N = 12.

ware engineering course (question 6), indicating that, even though this question is not a measure of learning success, applying Kanban was most likely at least fun.

The free text answers to question 3, see Figure 2, regarding the (dis)advantages of Kanban, were manually labeled with the mentioned topics. The list of topics was refined repeatedly after evaluating every question. Survey participants identified the following topics as advantages of Kanban (N=11): Efficiency (7 mentions), Autonomy (4 mentions). Three other other topics were mentioned only twice or fewer times. As concepts, efficiency and autonomy are closely related to Lean Software's guiding principles of *Eliminate Waste* and *Empowering the Team*, respectively [17]. As Kanban is heavily inspired by these ideas, it is reassuring to see that these ideas transferred. Regarding the disadvantages of Kanban usage, students mentioned the following topics (N=9): Only worked on small user stories (3 mentions), item Uneven task distribution (2 mentions). Another six disadvantages only received single mentions. Solely working on small user stories, i.e. work items in an agile process, may be a consequence of team member autonomy. Developers may choose to work on small items that can be moved through the columns of the Kanban board quickly, instead of picking larger, more time-consuming tasks to work on. Tackling uneven task distribution between team members is an ongoing challenge in educational settings and especially in self-organizing teams of students. It can be seen as a negative consequence of the autonomy identified by survey participants. Developers are free to handle their workload, with some developers choosing to do more and others choosing to work on fewer items.

User stories are one of the core means of communication, both in Kanban and Scrum, between the Product Owner, who receives input from stakeholders, and developers [19]. As such, we included a question on the perceived change of user stories when switching from Scrum to Kanban (question 4). In order to make answering easier, this question was a multiple choice question that provided a range of answer possibilities of which any number could be chosen. The choices, as well as the summarized answers of survey participants, are shown in Table 4.

Students classified the user stories that were written by Product Owners and developers during the Kanban iteration as shorter and more bug-oriented than in

Table 4: Answers of survey participants to question #4, regarding attributes of user stories when changing from Scrum to Kanban processes. N=12.

Topic	Answer choice and count			
User story focus	bug-oriented	11	feature-oriented	0
User story length	Shorter	11	Longer	0
Requirements	More detailed	8	More general	0
Interaction with PO	More	3	Less	0
Prioritization of stories	Better	3	Worse	2

the previous Scrum iterations. While an influx of small fixes to a software product is expected shortly before the final deadline, a time in which Kanban was used, smaller user stories can also help move tickets through the Kanban board more quickly. This reduced the *cycle time*, the time from when work begins on an item until it is ready for delivery [16].

However, students also answered that the requirements, i.e. part of the *acceptance criteria* [24] within user stories, had gotten more detailed during Kanban usage. This allows work on a user story to be started without having to clarify open questions beforehand and can help efficiency by decreasing the cycle time. Small user stories that contain enough detail to be immediately implementable are ideal for usage in the Kanban process [15].

3.3 Discussion

Both surveys, the more general end-of-term survey as well as the more specialized survey on Kanban, revealed positive attitudes towards our approach of teaching agile processes in a hands-on fashion as well as the shift from Scrum to Kanban at the end of the project ($RQ2$). Students indicated that they would recommend using Kanban to participants of the following year's course. These results are in line with related, similar studies [12]. In particular, Melnik et al. state that students, in general, were "very enthusiastic about core agile practices" and accepted and liked them [14]. The authors also point out that this observation held for a broad range of students, regardless of educational program, age or industry experience. While the findings of this and similar studies are encouraging for educators teaching software project courses, they also pose challenges. If students are generally content in agile project courses, simply due to the course setting and the fact that teamwork and building software together is fulfilling, how can improvements to curriculum design and changes in student behavior be evaluated?

Student opinions and attitudes towards course contents are an important part of any assessment plan. However, evaluations of teaching methods should primarily rely on the assessment of learning outcomes [5], which surveys only partly capture as they are geared towards collecting perceptions and attitudes. Furthermore, if not every survey participant answered the more reflective, time-consuming free text

answers, data on the learning outcomes of these participants is missing completely. To help tackle these challenges, the outcomes and artifacts produced during the process transition from Scrum to Kanban can be analyzed to gain additional insights into development teams.

4 Development Artifact Analysis

While surveys are effective tools for capturing the attitudes of participants, they do not allow insights into whether the perceived change in workflow or in user story quality actually took place during the project or how severe the change was. In order to provide another dimension of analysis based on real project data, we evaluated the software development artifacts produced by course participants. In particular, we compared students' development artifacts of the last five installments of our software engineering course, the earliest two of which did not include Kanban and the three most recent ones that did.

4.1 Data Collection

The development artifacts we collected from course repositories included commits into the version control system *git*, containing code changes, timestamps and commit messages describing the change, as well as tickets, acting as user stories, in an issue tracker. Both of these data sources were collected from the collaboration and hosting service GitHub, where all projects were hosted. GitHub features extensive application programming interfaces (APIs) that allow programmatically extracting the data stored by the service.

For every repository of the last five course iterations, user stories/issues and commits from the last seven days of project work were collected. This is the time frame that Kanban was employed in the more recent course iterations. Using this data, both the assumptions on student behavior, i.e. educators hypothesis of how artifacts would change from using Scrum to Kanban, as well as the accurateness of student perceptions could be tested.

4.2 Discussion

The collected data shows that the length of user stories did not significantly differ from when Kanban or Scrum was used in the last iteration of the course, see Table 5. This differs from the reported perceptions of students in the survey performed in the 2016/17 course installment. There, students reported that user stories were perceived to be shorter when using Kanban when compared to Scrum. While these two measures are not necessarily directly comparable, further study into the content differences between user stories in Kanban and Scrum is required. However, the analysis was able to uncover this discrepancy and provides a starting point for further investigation. Most other measures calculated from commits, such as the mean

Table 5: Issue body and title length of issues for the last week of projects. Courses marked with * employed Kanban.

	Issue body length			Issue title length		
Course year	**Mean**	**Stdev**	**Median**	**Mean**	**Stdev**	**Median**
2013/14	274.8	295.2	169.0	35.3	15.3	32.0
2014/15	420.8	327.3	361.0	50.7	15.5	50.0
2015/16*	360.9	339.4	253.0	36.9	16.9	32.5
2016/17*	505.5	556.9	378.0	36.9	16.7	35.0
2017/18*	579.8	393.3	526.0	35.9	14.3	37.5

amount of touched files, did not differ significantly between the two processes in different course years, see Table 6. However, it is encouraging to see that the amount

Table 6: Comparison of commit attributes for the last week of projects. Courses marked with * employed Kanban. All values stated normalized by course participant count.

Course year	**Commit amount**	**Touched files**	**Last-minute commits**	**Line changes per commit**	**Unique issues referenced**
2013/14	12.1	13.3	1.4	590.5	2.8
2014/15	7.2	6.9	1.2	408.0	1.0
2015/16*	6.1	8.0	8.1	466.0	0.1
2016/17*	3.4	5.4	2.2	163.5	2.2
2017/18*	8.6	5.3	1.7	195.0	1.5

of *last-minute commits*, i.e. commits made close to the end of the iteration [13] tended to be higher in the course installments in which Kanban was employed. Scrum's iteration plan, the Sprint Backlog, contains all the work items a team intends to address in a sprint, i.e. the amount of work that can be performed by a team in an iteration. Ideally, these items are worked on in a continuous manner, so that towards the end of the sprint the last user story is finished [23]. In this manner, work intensity and commit frequency should be uniformly distributed during an iteration. In contrast, Kanban does not explicitly call for iteration planning and so work is more likely to be assigned more dynamically: new work items can easily be added to the work queue, especially towards the end of the project, when the deadline approaches.

The more dynamic nature of Kanban is also reflected in the fact that the mean line change per commit, as well as the mean number of touched files, were smaller in the last two years of the course, see Table 6, when educators had already gathered some experience teaching the new methodology. This is in line with the Kanban survey

results, where students stated that their user stories when employing Kanban were more bug-oriented than feature-oriented. Fixing a bug usually requires changing fewer lines touching fewer files than implementing an entirely new feature. Furthermore, bugs are usually noticed during regular development activities or during testing and can easily be added to a Kanban board [6], whereas they might only end up in the next Scrum sprint [9].

Interactions with user stories, i.e. issues on GitHub, did not differ significantly between those courses that employed kanban and those that used Scrum, see Table 7.

Table 7: Comparison of issues and their attributes for the last week of projects. Courses marked with * employed Kanban.

	Mean per contributor			% issues opened & closed by same person
Course year	**Issue amount**	**Issue events**	**Issue comments**	
2013/14	4.9	27.5	17.1	43
2014/15	2.6	47.8	4.1	58
2015/16*	3.9	35.5	4.1	20
2016/17*	2.8	64.4	6.4	46
2017/18*	2.1	68.9	4.8	65

Normalized by participant count, similar mean numbers of issues were closed in the studied time frame and a similar amount of comments were attached to issues. However, the two most recent courses using Kanban showed higher mean amounts of non-comment events, such as labeling or assignments, a sign that the issue tracker was used more heavily. While Scrum explicitly calls for a role that mainly writes and prioritizes user stories, the Product Owner [23], Kanban does not. We had thus hypothesized that introducing Kanban would result in higher engagement by the entire team with the list of outstanding work items, instead of teams mostly relying on the PO to maintain it. However, the percentage of issues opened and closed by the same person, see the last column of Table 7, was surprisingly low even in the Scrum courses, with only 43% and 58% in course installments 2013/14 and 2014/15. These numbers did not differ significantly for the Kanban courses. While these results do not support our original hypothesis on team engagement with user stories, they represent opportunities for future work on how agile student teams interact with user stories and work item backlogs in collaboration with a Product Owner role.

By analyzing software development data created by students, which is already produced during regular development activities, we were able to uncover areas where some of our assumptions on student behavior regarding the adoption of different agile software development methodologies were confirmed and some were refuted. These areas will serve as a basis for future improvements to the course.

5 Related Work

A variety of specialized data sources have been analyzed in previous work in order to gain insights into student behaviors in computer science courses.

Wilson and Shrock [26] employed a survey to determine factors that promote success in an introductory computer science course. They examined twelve factors of which comfort level, math, and attribution to luck for success/failure were the most important for predicting student scores. Similarly, Bennedsen and Caspersen [3] attempted to help improve students' learning premises by collecting data on the emotional and social factors of students in a survey. They collected data on the factors of perfectionism, self-esteem, coping tactics, affective states, and optimism.

While surveys can be used to collect data on arbitrary factors, depending on which questions are used, entirely different ways of collecting student data have also been explored. Keen and Etzkorn [7] analyzed the complexity of teachers' lecture notes, the *buzzword density*, i.e. the amount of Computer Science domain-specific words divided by the total number of words in the lecture, to predict grades. Ashenafi et al. used data created by students during course activities, namely the results of several semi-automated peer-assessment tasks throughout the semester, to build a linear regression model for predicting final grades [2].

6 Conclusion

Employed processes need to be maintained and monitored to ensure that teams employ a process that works for every team member and the team's context. This is also true in university, where courses need to be continuously adapted to changes in requirements, such as industry shifts and technology advancements. Feedback to educators on curriculum design usually takes the form of end-of-term surveys or questionnaires on specific course aspects. However, techniques from the field of Educational Data Mining can be employed to gain insights into student teams and their reactions to changes in curricula that go beyond those possible with surveys. This research presents our approach to analyze development artifacts to achieve this goal. We analyzed the software development artifacts of student teams from five university undergraduate software development courses, which teach different agile development methodologies. Surveys with participants revealed positive attitudes towards the course and changing the employed development methodology during the course from Scrum to Kanban. However, surveys were not able to ascertain the degree to which students had adapted their workflows accurately. Therefore, we propose an approach of analyzing the software development data created by students during regular project work activities, specifically user stories and commits to a version control system, as an additional dimension of analysis. While this data serves a primary purpose in communication between team members, it also represents information that can be collected and analyzed to generate insights into a team's behavior during project work.

References

[1] M. O. Ahmad, J. Markkula, and M. Oivo. "Kanban in software development: A systematic literature review". In: *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*. 2013, pages 9–16. DOI: 10.1109/SEAA.2013.28.

[2] M. M. Ashenafi, G. Riccardi, and M. Ronchetti. "Predicting students' final exam scores from their course activities". In: *2015 IEEE Frontiers in Education Conference (FIE)*. 2015, pages 1–9. DOI: 10.1109/FIE.2015.7344081.

[3] J. Bennedsen and M. E. Caspersen. "Optimists have more fun, but do they learn better? On the influence of emotional and social factors on learning introductory computer science". In: *Computer Science Education* 18.1 (2008), pages 1–16.

[4] D. A. Bonebright. "40 years of storming: A historical review of Tuckman's model of small group development". In: *Human Resource Development International* 13.1 (2010), pages 111–120. DOI: 10.1080/13678861003589099.

[5] R. M. Felder and R. Brent. "How to improve teaching quality". In: *Quality Management Journal* 6.2 (2009), pages 9–21.

[6] M. Ikonen, E. Pirinen, F. Fagerholm, P. Kettunen, and P. Abrahamsson. "On the impact of Kanban on software project work: An empirical case study investigation". In: *16th IEEE International Conference on Engineering of Complex Computer Systems*. 2011, pages 305–314. DOI: 10.1109/ICECCS.2011.37.

[7] K. J. Keen and L. Etzkorn. "Predicting students' grades in computer science courses based on complexity measures of teacher's lecture notes". In: *Journal of Computing Sciences in Colleges* 24 (2009), pages 44–48. ISSN: 1937-4771.

[8] H. Kniberg. *Kanban and Scrum – Making the most of both*. Lulu.com, 2009, pages 1–49. ISBN: 978-0-557-13832-6.

[9] H. Kniberg. *Scrum and XP from the trenches*. C4Media, 2007, pages 1–105. DOI: 978-1-4303-2264-1.

[10] A. Komus and M. Kuberg. *Abschlussbericht : Status quo Agile 2016/2017*. Technical report. Hochschule Koblenz, University of Applied Sciences, 2017.

[11] V. Mahnic. "From Scrum to Kanban: Introducing lean principles to a software engineering capstone course". In: *International Journal of Engineering Education* 31.4 (2015), pages 1106–1116. ISSN: 0949-149X.

[12] V. Mahnič. "Scrum in software engineering courses: An outline of the literature". In: *Global Journal of Engineering Education* 17.2 (2015), pages 77–83. ISSN: 1328-3154.

[13] C. Matthies, T. Kowark, M. Uflacker, and H. Plattner. "Agile metrics for a university software engineering course". In: *IEEE Frontiers in Education Conference (FIE)*. 2016, pages 1–5. DOI: 10.1109/FIE.2016.7757684.

[14] G. Melnik and F. Maurer. "A cross-program investigation of students' perceptions of agile methods". In: *Proceedings of the 27th International Conference on Software Engineering (ICSE'05)*. 2005, pages 481–488. DOI: 10.1109/ICSE.2005.1553593.

[15] N. Nikitina and M. Kajko-Mattsson. "Developer-driven big-bang process transition from Scrum to Kanban". In: *Proceeding of the 2nd workshop on Software engineering for sensor network applications (SESENA'11)*. 2011, page 159. DOI: 10.1145/1987875.1987901.

[16] R. Polk. "Agile and Kanban in Coordination". In: *2011 AGILE Conference*. IEEE, 2011, pages 263–268. DOI: 10.1109/AGILE.2011.10.

[17] M. Poppendieck and T. Poppendieck. *Lean software development: an agile toolkit*. Addison-Wesley, 2003.

[18] J. Pries-Heje and J. Johansen. "SPI manifesto". In: *European System & Software Process Improvement and Innovation* (2010).

[19] M. Rees. "A feasible user story tool for agile software development?" In: *Ninth Asia-Pacific Software Engineering Conference, 2002.* Volume 2002-Janua. 2002, pages 22–30. DOI: 10.1109/APSEC.2002.1182972.

[20] C. Rosen, B. Grawi, and E. Shihab. "Commit guru: analytics and risk prediction of software commits". In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE'15)*. 2015, pages 966–969. DOI: 10.1145/2786805.2803183.

[21] C. Santana, F. Queiroz, A. Vasconcelos, and C. Gusmao. "Software process improvement in agile software development: A systematic literature review". In: *41st Euromicro Conference on Software Engineering and Advanced Applications*. 2015, pages 325–332. DOI: 10.1109/SEAA.2015.82.

[22] E. A. Santos and A. Hindle. "Judging a commit by its cover". In: *Proceedings of the 13th International Workshop on Mining Software Repositories (MSR'16)*. 2016, pages 504–507. DOI: 10.1145/2901739.2903493.

[23] K. Schwaber and J. Sutherland. *The Scrum guide – The definitive guide to Scrum: The rules of the game*. Technical report. 2017, page 19.

[24] A. Silva, T. Araújo, J. Nunes, M. Perkusich, E. Dilorenzo, H. Almeida, and A. Perkusich. "A systematic review on the use of Definition of Done on agile software development projects". In: *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (EASE'17)*. 2017, pages 364–373. DOI: 10.1145/3084226.3084262.

[25] VersionOne Inc. *The 11th Annual State of Agile Report*. Technical report. VersionOne Inc., 2017.

[26] B. C. Wilson and S. Shrock. "Contributing to success in an introductory computer science course". In: *ACM SIGCSE Bulletin* 33.1 (2001), pages 184–188. DOI: 10.1145/366413.364581.

[27] J. P. Womack, D. T. Jones, and D. Roos. *The machine that changed the world: the story of lean production*. Harper Collins, 1991.

Mining Concepts from Code to Support Program Comprehension and Software Modularity

Toni Mattis

Software Architecture Group
Hasso-Plattner-Institut
toni.mattis@hpi.uni-potsdam.de

Software modularity is a quality that determines how fluently individual parts of a system can vary and be understood if taken by themselves. Modularity tends to degrade during program evolution – old concepts get entangled with code introduced into their modules, while new concepts end up being scattered over many existing modules. This process, which we call *architectural drift*, makes programs more difficult to understand and more expensive to maintain.

Our research aims at creating tools to overcome these obstacles and helping programmers understand and effectively manipulate a system in the presence of architectural drift. We research probabilistic models that recover and relate the underlying concepts from source code repositories and evaluate them on a large-scale dataset to assess how well they perform in clustering, recommendation, and information retrieval tasks. We proceed to explore how programming environments can be improved by a concept-oriented representation of a system. We argue that better understanding and continuous feedback on modularity eventually helps to improve software quality in the long run.

1 Introduction

Software modularity is a quality that determines how fluently individual parts (modules) of a system can vary and be understood if taken by themselves. It contributes to lower maintenance costs by preventing requirement changes from propagating through the system, helping programmers understand a system, increasing testability, helping teams work independently on different parts, and a wide range of other effects.

Problem statement Modularity tends to degrade during program evolution – old concepts get entangled with code introduced into their modules, while new concepts end up being scattered over many existing modules. Several factors contribute to this so called *architectural drift*: The programming language may lack mechanisms to express a concept modularly, the existing architecture cannot accommodate a new requirement, or management may prioritize short-term goals over long-term maintainability.

Even in reasonably modularized systems, the inherent complexity increases the risk of misunderstanding the underlying conceptual model or making design decisions with incomplete information.

Architectural drift creates a reinforcing feedback loop in which a lack of understanding and bad maintainability causes programmers to add more scattered and tangled code over time. It is an instance of *technical debt* that accumulates "interest".

Architectural drift is often approached from a language design perspective, where the objective is to make underlying programming languages more expressive to better separate concepts. Examples include Aspect-oriented Programming (AOP) or type systems. However, when architectural drift already manifests itself by driving up the costs of change, retroactive approaches that improve program comprehension and assist refactoring or re-modularization move into focus.

Goal Our research approaches the feedback loop at three different leverage points:

1. *Metrics:* Measure architectural drift to provide programmers with feedback where and when modularity is degrading and to which degree and if countermeasures are effective

2. *Reverse engineering:* Help programmers discover which concepts are distributed across which modules, how they interact in the system, and how they evolved throughout the version history

3. *Forward engineering:* Support programmers in their choice of names, code locations to edit, refactorings, and other activities that can directly improve modularity

Approach Our approach is centered around finding a latent decomposition of a software system into *concepts* through a machine learning process. Our notion of a *concept* is derived from that of cognitive psychology, in which it constitutes a unit of comprehension or – especially in software engineering – a *unit of communication* and usually has one or more names used to refer to that concept and its constituents.

We propose a knowledge representation, referred to as *concept model*, that encodes which concepts are in a system, where each individual concept is expressed in source code, how they interact with each other, and which distinguishing names and features identify each concept. Possible extensions of such a model can also attribute expert knowledge about certain concepts to individual programmers or teams, and associate external artifacts, such as issues in a bug tracker, to concepts.

Integrating such a *concept model* into the programming toolchain can support all three of the above objectives through continuous feedback:

- Modularity metrics can be obtained from measuring how much the current system deviates from the inferred *concept model*, or how much an individual change to the software contributes to that deviation

- The model, when properly visualized, can be explored by programmers to gain a high-level understanding of the system, or queried for specific concepts to see which code locations are affected from a change to that concept

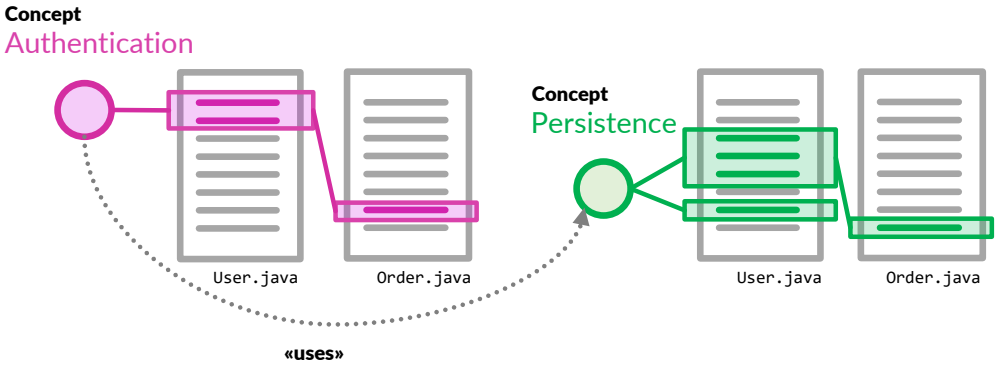


Figure 1: Illustration of two concepts (circles) that cut across multiple modules, their locations in source code, and a relation between them

- The model can provide immediate support as programmers work, for example in the form of refactoring recommendations that would improve modularity and names that better fit the current conceptual context

Each type of feedback is *explainable* and *actionable*, e.g., placing code at the wrong location does not only quantitatively impact a metric, but the model can always *explain* which concepts have been confused or mixed, and *recommend* better locations. When presented with a possibly misguiding explanation, expert programmers can intervene, correct a mistake made by the model, and thus refine it on the fly.

We explored different types of probabilistic generative models, such as topic models, and their explanatory power with regard to a software system. We eventually developed a graph-based model in which edges constitute observable relations in source code, edit history, and at run-time, and a stochastic process that generates such a graph from a set of interacting concepts. As with any probabilistic model, our objective is to find a decomposition of the system into concepts that most likely explains the observed graph.

Evaluation and Tooling To explore uses and evaluate effectiveness, we follow both a qualitative and quantitative approach. The quantitative part focuses on evaluating different versions of the concept model itself and comparing it with related work on multi-view learning, topic modeling, and software recommender systems. Metrics to compare how well a concept model captures the underlying structure of a system are recently being researched, and one of our contributions will be a representative evaluation data set extracted from more than 10 billion program modifications on GitHub for change prediction and software artifact recommendation.

A future qualitative evaluation will be based on a modified concept-aware programming environment as described in [9] and study the effects of new tools on program comprehension and modification tasks in a few case studies.

2 Mining Concepts from Code

2.1 Background: Concepts and Names

Origins In cognitive sciences and philosophy, a *concept* is often described as a mental *representation* (or placeholder) for a set of instances and expectations. For example, the concept of a "tree" can represent the set of real-world plants with certain shape, creates the ability to imagine a protoypical tree, allows us to recognize a tree if we see one, and evokes expectations, e.g., branches of certain size support a person's weight. Most concepts carry a specific vocabulary, e.g., the tree concept is associated with words like *branch*, *root*, and *leaf*.

Concepts in programming In programming, concepts appear in two different kinds: First, there are *computational concepts* provided by the programming and execution environment. They include types, operations, memory and process abstractions, etc. They also dictate how programmers can express the second kind: *Domain concepts* that need to be engineered in a program to achieve its purpose. Domain concepts originate from requirements (e.g., business processes, users, documents, goods, other systems, etc.) and appear in a program as *named* instances or combinations of computational concepts.

The mapping between domain concepts and computational concepts is described in literature as *theory building* [11] or *concept assignment problem* [4], and often the only recognizable links are *names*. For example, the computational concept of a string of characters has no meaning in any domain, but can be linked to the concept of a *phone number* by assigning it to a variable named `phoneNumber`. Seeing this name and having the associated concept in mind, programmers can have some expectations how the string might be formatted and can imagine examples for such a string. Similarly, seeing the names `root` and `leaf` invokes the tree concept again, this time as a *metaphor* for a recursive data structure analogous to real-world trees.

High-level concepts In the context of this work, we are interested in larger high-level concepts that have the potential to span multiple modules and, thereby, impede program comprehension. Such concepts can include concrete domain subjects (*user*, *order*, ...), other subsystems (*database*, *services*, ...), processes (*user registration*, *booking*, ...), and cross-cutting concerns (*authorization*, *logging*, ...). Their vocabularies span dozens up to hundreds of distinct words, such that full-text search usually does not suffice to find all participants of a complex system function and not every word can be immediately associated with a concept.

2.2 Inferring Concept Models

Finding names Our first objective is to find the *names* that programmers have chosen. Typical places include the names of variables, arguments, classes, methods, fields, and types. Moreover, short strings or, in case the language has syntax for it, symbols, carry meaning. They can be found as dictionary keys, in metaprogramming,

error messages, etc. In Smalltalk and related languages there are category names used to organize class members. Comments are of interest in future work, but excluded for the moment, since we do not want to deal with the noise introduced by unrestricted natural language.

Each of these lexical tokens can be constructed from multiple names, which justifies the decision to attach multiple concept labels to a single lexical token. Typical examples are camel-case and underscore identifiers like `isEmpty` or `open_file`, or Smalltalk's multi-part messages `at:put:`, where each part can be camel-cased again. Our analysis is not constrained to nouns, or even actual words, but builds an exhaustive vocabulary of potential names used in the source code.

Distributional Hypothesis In semantics, the *distributional hypothesis* states that two lexical tokens that share meaning also share the same statistical distribution in a text corpus. That means, they are more likely co-occurring (and co-missing) than two unrelated tokens.

If the lexical tokens in a program are derived from concepts similarly as we refer to concepts in text, the distributional hypothesis should hold for program source code as well. For example, when a structure resembles a graph and programmers decide to name the parts *vertex* and *edge*, they will be more frequently occurring together than, e.g., *vertex* and *password*.

Abstraction and limitations of the Distributional Hypothesis A method or other unit of modularity is typically not self-contained, i.e., it makes use of other units and is being used by units building on top of it. This naturally limits the coherence of names within a unit, since it tries to express one concept while internally depending on vocabulary exposed by other concept's interfaces, e.g., a *stack* using a *list* internally.

Such abstraction barriers still manifest themselves as highly correlated names, and force different sides of the abstraction barrier into being labeled with the same concept if we solely rely on the distributional hypothesis.

Abelson and Sussman [1] identified that programming environments typically have three constituents: primitives, *means of combination*, and *means of abstraction*.

Means of combination include all operations on data (e.g. mathematical operators) as well as operations on operations (e.g. sequencing two statements, conditionally executing or repeating operations). Common means of abstraction are constant and variable assignments, function/method definitions, and class/interface definitions. All abstractions *define* one or more (new) names in terms of (a *combination* of) implementation-specific primitives or existing names, which gives us a new way of how two names can be related.

We refine the distributional hypothesis in the context of a programming language to mean that two names that share the same concept tend to be frequently *combined*, and, if a concept A makes use of a different concept B, names from A will be more frequently *defined* in terms of names from B.

LDA-Type Topic Modeling In a first step, we used topic modeling, a technique that represents a set of text documents as mixture of high-level concepts which

in turn consist of words. Probabilistically, topic models factorize the histogram of words w in each document d, considered as a sample from a distribution $P(w|d) \propto \sum_t P(w|t)P(t|d)$ into a mixture of topics t for each document $P(t|d)$ and a distribution of words within each topic $P(w|t)$ such that individual words which are correlated by their meaning become conditionally independent given their topic. Latent Dirichlet Allocation (LDA) treats both conditional distributions as multinomial distributions with a Dirichlet prior [6] and each distribution can be easily approximated using Gibbs Sampling [7].

By treating methods in source code as documents in the LDA model, we could recover high-level concepts reasonably well, reproducing prior research of applying LDA to Java classes [8]. We can also assign single concepts to each name, for example by choosing the most likely topic as concept $\text{argmax}_t P(t|w,d)$ or by taking the most frequent topic from the Gibbs sampling distribution. User feedback can be incorporated by fixing specific names to a topic and re-running inference for the unmodified names. However, several types of user feedback (e.g. re-ranking a name inside a concept) are difficult to represent in LDA.

Limits of LDA-Type models and Abstraction-aware LDA We observed several weaknesses in classic topic modeling emerging from independence assumptions between documents, and also between topics. Units of modularity in source code systematically introduce new vocabulary by defining it in terms of already known vocabulary. We try to capture this abstraction and composition process with our concept model. In an extension to LDA, we refined the generative model to consider concept transitions, i.e., defining a term of one concept using a term from another concept, resulting in a more expressive model. Probabilistically, we introduce the topic or concept transition probability $P(t_i|t_a)$ to reflect the chance that a term of abstraction concept t_a is defined through or implemented by a term in implementation concept t_i, and model the distribution of right-hand side identifiers (i.e., those which appear in the body of a variable or method definition) as $P(w_i|d) = \sum_{t_i}\sum_{t_a} P(w_i|t_i)P(t_i|t_a)P(t_a|d)$, while left-hand side identifiers (i.e., method signatures or names being currently defined) in the same document are treated like they belong to the abstraction, $P(w_a|d) = \sum_{t_a} P(w_a|t_a)P(t_a|d)$. The inference procedure used for LDA is extended to accommodate for the new model parameters $P(t_i|t_a)$. The model can now summarize large-scale interactions between concepts through its transition matrix formed by all pairs in $P(t_i|t_a)$, but may collapse into the LDA model with all $P(t_i|t_a) \approx 1.0$ if $t_i = t_a$ else $P(t_i|t_a) \approx 0.0$ on a small code base. An additional limitation is a slowdown of factor 2 to 3 compared to LDA since the number of latent variables has doubled.

Graph-based Topic Modeling To better address the underlying structure of code repositories, we designed a graph-based model, in which nodes are *names* and co-occurrence is represented by multi-*edges* between these names. By defining a stochastic process that generates a graph $G = (V, E)$ in a way that two nodes get connected when they play a role in the same concept, we can recover the most likely concept that explains an edge and propagate this information to the adjacent nodes. Prob-

abilistically, we add an edge $n_1 \leftrightarrow n_2$ between two nodes (names) $n_1, n_2 \in V$ with probability $P(n_1 \leftrightarrow n_2) = \sum_c P(n_1|c)P(n_2|c)P(c)$, that means our concepts c represent, analogously to topics, a probability distribution over nodes. $P(c)$ represents the global proportions of concepts, there is no document-specific concept mixture. We can infer the concept assignments c_e for every edge $e \in E$ (E is a multi-set) using Gibbs sampling from a previously constructed co-occurrence graph. We can obtain a single concept for every name n by looking for the concept that most likely generated the majority of adjacent edges.

In our implementation, we define co-occurrence, and thus whether we add an edge to the graph, by two names being combined in the sense of [1], e.g., if a statement applies a mathematical operator the value of two variables, then both variables are co-occurring.

Graph-based Topic Modeling with Abstraction We extended this model by a directed edge type $n_1 \rightarrow n_2$ to encode abstraction; a directed edge is explained by the first concept making use of the second one, analogously to our extended distributional hypothesis:
$P(n_1 \rightarrow n_2) = \sum_{c_a} P(c_a) \sum_{c_i} P(c_i|c_a)P(n_1|c_a)P(n_2|c_i)$. Hence, we obtain additional model parameters $P(c_i|c_a)$ that quantify how an abstraction c_a makes use of concept c_i for its implementation. The observation on which the model is trained includes those directed edges whenever an abstract name n_a is defined (from left-hand side tokens or method signatures) in terms of another implementation-specific (from right-hand side tokens or the method body) name n_i.

A large benefit of this model is that it does not require the code to be sliced into documents like in LDA-type models. New edge types can be added easily that encode new concept relations, e.g. a directed edge type that models data flow, or edges connecting program parts to other artifacts (e.g. comments or documentation).

Preliminary evaluation using coherence measures from Mimno et al. [10] indicate that our graph-based models outperform LDA-type models when run on many small "documents" or when fewer data is available, but inference is about 10 times more memory and computation intensive. We are working towards a parallel, optimized implementation of graph-based concept models that can run in real-time.

Modularity The distribution of concept labels in each module can be used to assess the module's coherence as well as the locality of the respective concept. Related work dealing with an LDA-based modularity assessment proposes to use entropy measures [8]. We can adapt this idea to the labeled names and define per-method entropy as

$$H(m) = -\sum_c p(c|m) \log_2(p(c|m))$$

With $p(c|m)$ being the proportion of concept c with regard to all labels in module m. The larger this number, the less coherent a module is with regard to naming. As a refinement we could eliminate implementation-specific concept labels from the analysis and only focus on the labels given to the method signatures. This would give

us a measure for the entropy in a module's interface and not consider the fan-out caused by implementation details.

Using the same mechanism, we can quantify cross-cuttingness of a concept by measuring its entropy across all modules:

$$H(c) = -\sum_m p(m|c) \log_2(p(m|c)),$$

with $p(m|c)$ given by Bayes' Rule from previously computed $p(c|m)$.

In combination with version control, we could track the entropy measures across change sets and score each change according to whether it was an improvement (decreasing $H(m)$) or a sign of deterioration (increasing $H(m)$). This difference might become more apparent when versions with many changes in between are being compared.

Programmatic Interface From tooling perspective, we are working on a reflection interface that extends existing meta-programming facilities with mechanisms to retrieve and manipulate concepts of arbitrary program artifacts. This interface can be used by tools, e.g., debuggers, browsers, and editors, to display concepts, sort or group by concepts, and use visual hints such as ing to indicate concept correspondence. A key idea is to extend abstract syntax trees (ASTs) emitted by parsers and used by a multitude of tools to further split each name into individual lexical sub-tokens (e.g., splitting camel case and underline-separated names off an identifier) and provide an attribute that links these sub-tokens to their concept.

3 Related Work

DESIRE and DM-TAO An early instance of program-assisted concept assignment was the *DESIRE* system [4]. It was founded on a model that distinguishes programming-oriented concepts (formal, unambiguous, data-manipulating) from human-oriented concepts (informal, possibly ambiguous, domain-based). Biggerstaff et al. identified the mapping between them as a crucial part in program comprehension similar to Naur and highlighted that natural language tokens and proximity of statements (including grouping, e.g., by linebreaks) are important to automatically discover these links. Particularly interesting is their first approach to an *intelligent agent* called DM-TAO that can both locate concepts in code and explain code in terms of domain-model concepts, thereby serving two of our main use cases. It relied on a semantic graph that, similar to a neural network, computes the likelihood of a domain-model concept given observable syntactical and term-like features, can be trained by user feedback and bootstrapped from an existing domain model.

Topic models on code The discipline of topic modeling, originally from the domain of natural language processing, has been applied to software in several circumstances. Linstead et al. [8] successfully applied LDA to a large repository of Java source files to discover globally prevalent concepts. They also introduced entropy as

a method to measure tangling and scattering of topics in the sense of aspect-oriented programming. Saeidi et al. [12] refined LDA and included expert feedback in the form of manually defined must-link and cannot-link constraints between names. Asuncion et al. [3] extend topic modeling to related artifacts surrounding the programming environment. This allows information retrieval and software traceability across large projects. Thomas et al. [16] propose to track topic evolution in source code, especially the points in time where new concepts were introduced or removed, by a specifically designed topic model. Binkley et al. [5] investigated LDA on source code in detail and provided relevant insights that helped parametrize our own LDA implementation as a first step towards automated concept assignment.

Kernel-based methods More recent research focuses on kernel-based learning to capture code similarity, and improving the bag-of-words representation of software modules to include other information, such as types, with lexical features [13]. Significant progress on integrating different views (source code, history, and call-graph) using shared subspace learning to better capture semantic relatedness between modules is being made [yet unpublished].

Other statistical models Beyond topic modeling, the discipline of graph clustering and community detection researches similar structures, such as the mixed-membership stochastic block model [2], which can be understood as a topic model on graphs.

The problem of structuring concepts in a hierarchy has been addressed for natural language and images using a hierarchical Dirichlet process [15], but not yet applied to source code.

Computational biology is concerned with detecting subsystems (analogous to concepts) in reaction pathways to understand and classify lifeforms in terms of these subsystems. The models used to summarize reaction pathways into subsystems resemble modern topic and graph clustering models [14]. The fact that abstractions (implementation details, interface) in programming resemble reactions in chemical pathways (reagents, products) makes these models partially applicable to our domain.

4 Future Work

Given a way of distributing concept labels to each name, and also knowing abstractions and dependencies between concepts, we elaborate on our planned tool integration and quantitative evaluation.

Quantitative Evaluation We are facing the problem that we need to compare graph-based models, topic models, and ideally related work from kernel-based and other embedding methods. All three use different evaluation metrics throughout literature, e.g., graph-based models often use *modularity* while topic models use *perplexity*.

To construct better metrics that capture how well a model captures the underlying semantics of a system, we are currently building a data set from about 10 billion Git commits obtained from GitHub that includes not only source code, but also history and meta-data.

The data set is currently designed for the following evaluation scenarios:

Clustering Using a concept model for clustering can determine how well it captures semantic relatedness of existing modules. We cluster a range of well structured software systems using a top-down approach based on the most likely decomposition according to the probabilistic models. We then compare and measure their path distance (PD) to the real architecture. A variation of this experiment would use hierarchical clustering to better compare to non-probabilistic related work.

Change Prediction We use the concept model and related models to recommend likely code locations that are related to a code change. We compare the recommendations to actual modifications that have been made throughout the version history using standard measures, such as *area under ROC curve*.

Refactoring Recommendation When commits explicitly intend to improve a code base, we plan to measure if a concept model rates the result of a modification as more desirable than the original version and identify cases in which programmer and model opinion diverge.

Name Suggestion Our model is capable of estimating the probability of names within an arbitrary context and thereby can be used to recommend identifier names or re-rank code completion suggestions. We can construct incomplete test data from complete source code and measure how well our model could auto-complete identifiers or select the right name from a list of proposals.

Depending on the preliminary results of the above evaluation we plan to extend the framework to measure model stability, e.g., how fast inferred concepts shift with each program version, and how modularity assessed using the concept model compares to a wide range of other software metrics aimed at locating maintainability issues.

Concept-augmented Class Diagram We propose to introduce a new tool to explore the program at concept-level and address the information needs stated above.

The design studies in Figure 2 and 3 show a class diagram with color-coded concepts. Selecting a concept from the top highlights methods which have a majority of names labeled with this concept, making clear where the concept is prevalent. It should be possible to collapse or hide classes not concerned with the selected concepts, since class diagrams tend to use up a lot of space for larger systems. Even if not selected, the other concepts are indicated by color markers alongside modules, they might as well show proportions rather than just a binary indicator, but with many concepts such miniature charts might get cluttered. A major challenge will be automated layouting, such that classes related by concept are arranged closer to each other. For a user-layouted class diagram, we can provide *linting*, e.g., assessing how

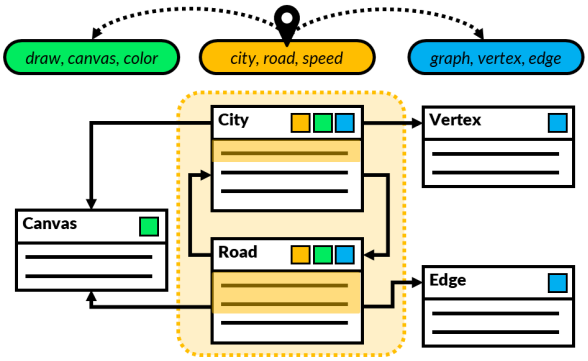


Figure 2: A class diagram with a selected concept. Concepts are represented as colored collection of relevant names, the corresponding parts of the program are highlighted using this color. Links to implementation-specific concepts are shown. Each module has colored indicators to show which additional concepts it is dealing with.

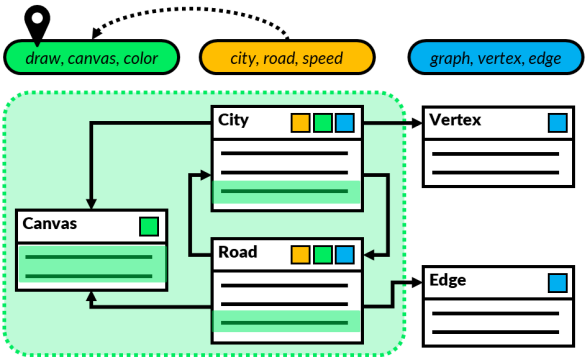


Figure 3: The same class diagram as in Figure 2 with a different selected concept

well the spatial grouping of classes reflects concepts and help re-arrange a diagram for better readability.

Concept-aware Browser and Editor For less explorative tasks or tasks requiring programming, the standard code browser and editor can be augmented to support first-class concepts in multiple ways. Similar to the class view, we can use color-coding to indicate relatedness of certain modules to a given concept. Concepts should be available anywhere, e.g., by selecting any identifier within the source code the editor should explain which concepts this identifier is labeled with (e.g., by showing a colored indicator near the identifier and some names associated with it). Interacting with any colored concept indicator, e.g., by hovering or clicking, should highlight every part of the program that belongs to the same concept.

Besides color-coding, the editor may warn programmers when they use names that are inconsistent with the context.

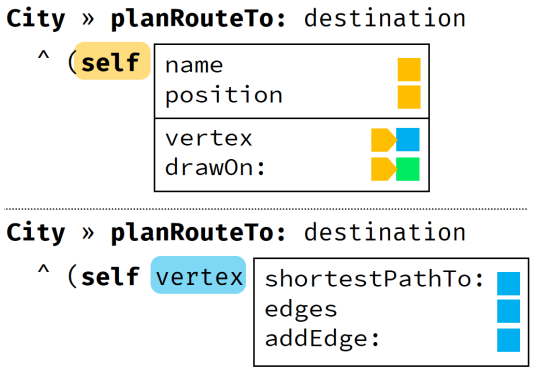


Figure 4: Code completion is scoped to a concept and additionally proposes completions from related concepts. When a concept transition is detected, code completion updates its concept context accordingly.

Concerning code completion, each typed name indicates what concept programmers are dealing with. Combined with the context (e.g., concepts in the currently active method and class), code completion can prefer those names that are either inside the concept, or in frequently used related concepts, like illustrated in Figure 4 (top). When a transition into another concept is detected, e.g., because a specific implementation detail uses a different concept, syntax completion can snap into that concept again, as in Figure 4 (bottom). This can be combined with traditional code completion ranking mechanisms, such as type inference and most-recently-used heuristics, but details on the exact mechanism need to be explored and tested yet.

Debugger A common challenge with debugging complex systems in graphical debuggers is that the call stack does not reflect abstractions in the original program design. Especially highly modular architecture expands to a call stack that slices

through a high number of abstractions which are likely unrelated to the problem at hand.

First-class concepts can help to structure a debugging session, since they can collapse call frames that belong to unrelated concepts, or at least mark the relevant ones.

Since modern debuggers have access to live objects in the running program, there is the opportunity to arrange or retrieve them by their concept.

Version Control With modern version control systems, changes can be tracked in small increments and attributed to their author. When reviewing individual change sets, the difference in terms of concepts (e.g., how many labels of each concept have been removed or added) may already give away what the code change was about. By aggregating the number of concept labels affected per author, we can compute what each author is an expert for.

Including Programmers' Feedback All of our designs rely on the fact that concepts detected using topic-modeling and similar techniques group names in a way that they are recognizable as coherent concept by humans. Since they only make use of artifacts created by programmers, they can only better arrange information that is already there, but not recover parts of the programmers' mental model that were never encoded in the program.

At any point where concept labels appear, experts should be able to correct the current label. This may trigger a cascade of automated updates trying to optimize the remaining, automatically chosen labels to reflect the change made by the expert. Other operations on concepts that we want to support include:

- Merge two concepts. This is simple, as each label of the second concept can just flip its color to the first concept.
- Automatically split a concept. This requires computing an optimal split, where each sub-concept retains the highest possible coherence while making sure that the names with the lowest semantic similarity are put into different sub-concepts.
- Manually split a concept. Experts drag and drop names into a new concept (similar to [12]). The environment may recommend additional names to move to the new concept, e.g., those that have high co-occurrence with the ones already moved.
- Shuffle a set of concepts. This might be the last resort when automated concept allocation fails. The good concepts remain fixed and the (randomized)

algorithm is re-run on the remaining concepts, probably finding a different solution.[1]

An interesting challenge manifests itself when concept-aware reflection facilities or client tools are used by multiple, collaborating developers. In a fully automated setting, the assignment algorithm may be tuned to compute the same results for each identical working copy of a shared repository. When decentralized expert feedback is taken into account, we need a way to synchronize concept assignments, for example, by serializing them into code comments or files managed by version control. Alternatively, a repository of concepts and their lexical features can be maintained independently from version control, feedback would be submitted to this repository, and automated analyses on different code bases can re-use this "crowd-sourced" knowledge.

References

[1] H. Abelson, G. J. Sussman, and J. Sussman. *Structure and Interpretation of Computer Programs – 2nd Edition*. MIT Press, 1996.

[2] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. "Mixed Membership Stochastic Blockmodels". In: *Advances in Neural Information Processing Systems 21*. 2009, pages 33–40.

[3] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor. "Software Traceability with Topic Modeling". In: *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering*. 2010, pages 95–104.

[4] T. J. Biggerstaff, B. G. Mitbander, and D. Webster. "The Concept Assignment Problem in Program Understanding". In: *Proceedings of 1993 15th International Conference on Software Engineering*. 1993, pages 482–498.

[5] D. Binkley, D. Heinz, D. Lawrie, and J. Overfelt. "Understanding LDA in Source Code Analysis". In: *Proceedings of the 22nd International Conference on Program Comprehension*. 2014, pages 26–36.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent Dirichlet Allocation". In: *J. Mach. Learn. Res.* 3 (2003), pages 993–1022.

[7] T. Griffiths. *Gibbs Sampling in the Generative Model of Latent Dirichlet Allocation*. 2011.

[8] E. Linstead, P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi. "Mining Concepts from Code with Probabilistic Topic Models". In: *Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering*. 2007, pages 461–464.

[1] Gibbs samplers are randomized algorithms. Especially when a large range of near-optimal, but completely different solutions exist, re-running a Gibbs sampler with a different starting configuration can give a better solution.

[9] T. Mattis, P. Rein, S. Ramson, J. Lincke, and R. Hirschfeld. "Towards Concept-aware Programming Environments for Guiding Software Modularity". In: *Proceedings of the 3rd Programming Experience (PX/17.2) Workshop*. PX/17.2. To appear. 2017.

[10] D. Mimno, H. M. Wallach, E. Talley, M. Leenders, and A. McCallum. "Optimizing Semantic Coherence in Topic Models". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2011, pages 262–272. ISBN: 978-1-937284-11-4.

[11] P. Naur. "Programming as Theory Building". In: *Microprocessing and Microprogramming* 15.5 (1985), pages 253–261.

[12] A. M. Saeidi, J. Hage, R. Khadka, and S. Jansen. "ITMViz: Interactive Topic Modeling for Source Code Analysis". In: *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension*. 2015, pages 295–298.

[13] A. Saeidi, J. Hage, R. Khadka, and S. Jansen. "On the Effect of Semantically Enriched Context Models on Software Modularization". In: *The Art, Science, and Engineering of Programming* 2.1 (2017), 2:1–2:39. DOI: 10.22152/programming-journal.org/2018/2/2.

[14] M. Shafiei, K. A. Dunn, H. Chipman, H. Gu, and J. P. Bielawski. "BiomeNet: A Bayesian Model for Inference of Metabolic Divergence among Microbial Communities". In: *PLOS Computational Biology* 10.11 (2014), pages 1–17. DOI: 10.1371/journal.pcbi.1003918.

[15] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. "Hierarchical Dirichlet Processes". In: *Journal of the American Statistical Association* 101 (2004).

[16] S. W. Thomas, B. Adams, A. E. Hassan, and D. Blostein. "Modeling the Evolution of Topics in Source Code Histories". In: *Proceedings of the 8th Working Conference on Mining Software Repositories*. 2011, pages 173–182. DOI: 10.1145/1985441.1985467.

GraalSqueak: A Fast Squeak/Smalltalk Implementation for the GraalVM

Fabio Niephaus

Software Architecture Group
Hasso Plattner Institute, University of Potsdam, Germany
Fabio.Niephaus@hpi.uni-potsdam.de

Language implementation frameworks aim to provide everything that is needed to build interpreters, simplify the process by making certain design decisions in advance, and suggest implementation strategies to virtual machine creators.

Truffle, for example, is a language implementation framework designed for building Abstract Syntax Tree interpreters that run on top of the GraalVM. Various programming languages have already been implemented in Truffle including SOMns, a Newspeak implementation in the tradition of Smalltalk and Self. However, an implementation of a traditional Smalltalk system is missing.

In this paper, we propose an approach for a fast Squeak/Smalltalk implementation in Truffle for the GraalVM. Our implementation GraalSqueak aims to provide full compatibility to existing Squeak/Smalltalk images including support for its programming environment and its unique language features. Although Truffle's warmup behavior has noticeable side-effects on the responsiveness of the programming environment, an early version of GraalSqueak already achieves competitive run-time performance in micro benchmarks.

1 Introduction

Implementing virtual execution environments for dynamic programming languages is a comprehensive activity when done from scratch. It is not only necessary to implement common components such as a parser, interpreter, or garbage collector, but also common optimizations for dynamic languages such as a just-in-time (jit) compiler. Language implementation frameworks such a RPython [1] and Truffle [24] have become more and more popular as they allow language implementors to use a high-level language and provide useful and reusable components and optimization mechanisms.

These frameworks, however, often enforce certain implementation styles because they are usually designed to support a specific kind of interpretation model. For example, RPython [1], a language implementation framework maintained as part of the PyPy project [21], is mainly used to implement bytecode interpreters, because its tracing jit compiler is most suited to operate on bytecode. Oracle's Truffle framework [24], on the other hand, is designed for implementing Abstract Syntax Tree (ast) interpreters and its jit compiler applies ast node rewriting and partial evaluation to significantly increase run-time performance of corresponding interpreters.

In this paper, we present GraalSqueak, a Squeak/Smalltalk [12] implementation for the GraalVM [24]. Squeak/Smalltalk is a Smalltalk dialect derived from the

Smalltalk-80 language specification [10] and GraalVM is the runtime environment for Truffle language implementations. With SOMns [16], a Smalltalk-like interpreter implemented in Truffle already exists. However, it operates entirely on asts as it is not image-based like traditional Smalltalk-80 systems and can therefore be well optimized by the GraalVM. GraalSqueak, on the other hand, aims to be fully compatible with Squeak/Smalltalk images including its programming environment and Smalltalk language features such as sender chain modifications and full reflective access. Some of these language features are unique to Smalltalk and for that reason harder to implement in Truffle than others.

In the following section 2, we introduce Squeak/Smalltalk as well as Truffle and the GraalVM ecosystem. Then, we present concepts and implementation strategies for Smalltalk-specific features that are especially hard to implement in Truffle in section 3. Afterwards, we give an overview over the implementation of GraalSqueak and highlight some design decisions in section 4. In section 5, we then evaluate the performance of GraalSqueak before discussing our approach in section 6 and related work in section 7. Finally, we conclude the paper in section 8 and give an outlook to future work.

2 Context

In this section, we introduce Squeak/Smalltalk as well as the GraalVM and the Truffle framework which we used to build GraalSqueak.

2.1 Squeak/Smalltalk

Squeak/Smalltalk [12] is a Smalltalk dialect derived from the Smalltalk-80 language specification [10]. It was originally developed by Alan Kay, Dan Ingalls, and others, most of which were also part of the Xerox PARC Learning Research Group which designed Smalltalk-80. The default framework for Squeak's graphical user interface (ui) is Morphic [14], which originally was developed for the Self programming language [22]. Over the years, many projects have evolved from Squeak/Smalltalk including Newspeak [6], Etoys [9, 13], Scratch [19], Pharo [4], and Babelsberg/S [11].

2.2 Truffle and GraalVM

GraalVM [24] is a multi-language runtime environment and based on the Java Virtual Machine (JVM). Its main component is the Graal compiler which performs ast node rewriting and partial evaluation to produce specialized machine code. The compiler can be used as a jit compiler at run-time or for ahead-of-time (aot) compilation as part of the SubstrateVM project [23].

Truffle [24] is a language implementation framework for building ast interpreters that run on top of the GraalVM. It not only allows language implementors to define ast nodes that can then be processed by the Graal compiler, but also to fine-tune

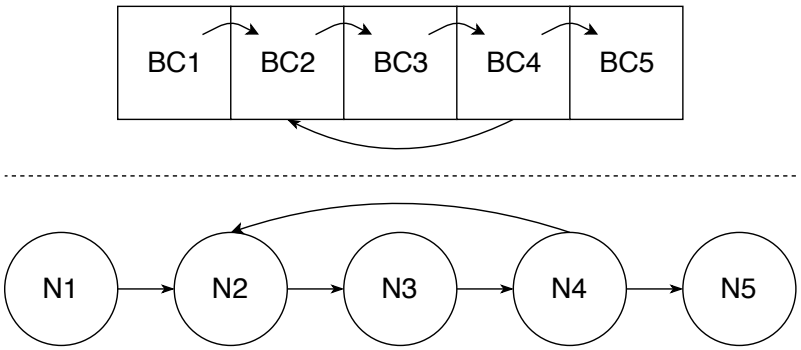


Figure 1: A bytecode stream and a sequence of ast nodes

the compiler with various hints. Moreover, it provides various profiling and caching mechanisms as well as optimized data structures that can be used to further improve run-time performance.

3 Approach

In this section, we propose an approach for implementing Squeak/Smalltalk in the Truffle framework. This includes implementation strategies to model Squeak/Smalltalk concepts in Truffle as well as language-specific performance optimizations.

3.1 Building a Bytecode Interpreter with Truffle

The Smalltalk-80 language specification includes a well-defined bytecode set that a Virtual Machine (vm) needs to implement to execute Smalltalk code. Since Squeak/Smalltalk is derived from this specification, a Truffle-based interpreter needs to support its bytecode set. However, Truffle only operates on ast nodes, so an ast needs to be produced from Squeak/Smalltalk bytecode. This can be done by decompiling its bytecode. Squeak/Smalltalk comes with a decompiler, but since it is implemented in Smalltalk and runs within the image, it is unavailable on interpreter-level. Although porting this decompiler to Truffle is relatively straightforward, it would add significant complexity to the interpreter. More importantly, compatibilty breaks in case the Squeak/Smalltalk compiler is changed since the vm now depends on in-image code because of a ported decompiler.

A better approach to implement a Smalltalk bytecode interpreter is to transform bytecode into a stream of ast nodes as depicted in Figure 1. For this, each bytecode is represented as a node and connected with one or, in the case of a jump, with two successor nodes which results in a linked list of ast nodes with jump pointers. Truffle and the Graal compiler, however, need additional hints to efficiently execute such asts. These hints will be discussed in more detail in section 4.2.

3.2 Providing Support for Squeak/Smalltalk Context Objects

Unlike many other programming languages, Smalltalk languages including Squeak/Smalltalk expose execution information as `Context` objects and it is possible to fully modify these objects. This, for example, allows an implementation of exception handling to be self-contained in Smalltalk because it is possible to jump from one context to another by modifying a context's sender.

Since this is not a common mechanism, sender modification is not supported by Truffle's `Frame` implementation. Because Truffle and Graal heavily optimize asts, they try to avoid materialization of these `Frame` objects if possible. Similar to Deutsch and Schiffman [7], we propose to use different representations (volatile, stable, hybrid) for Squeak/Smalltalk `Context` objects in Truffle. When such a `Context` object is not required, it is not allocated at all. We call this a fully virtualized context (*volatile*). When execution information is requested, it can be reconstructed from the corresponding Truffle frame. In case the execution context needs to be fully mutable, the vm allocates a fully materialized context (*stable*). Otherwise, a *hybrid* context is allocated if a method modifies only parts of a volatile context. In this case a context consists of a materialized Truffle `Frame` as well as a pointer array that overlays this frame. If, for example, the method is changed, it will be stored within this array and this array has precedence over values stored inside the frame object. This is also the case when the sender is replaced. Additionally, the Truffle frames need to be unwound as they no longer correspond to Squeak/Smalltalk's `Context` chain. This is an expensive operation, especially when always unwinding all frames. When this happens as part of the exception handling mechanism, for example, the new sender is likely to be part of the chain of Truffle frames. Therefore, unwinding can stop as soon as the new sender is found to minimize performance degradation.

3.3 Primitives and VM Plugins

Alongside a bytecode set definition, Smalltalk-80 also defines a set of language primitives. The most important primitives are numbered, but they can also be named and additional primitives can be provided through a plugin mechanism. To implement these primitives in Truffle, we suggest to use a dedicated node subclass as well as a Java annotation for storing primitive indexes and names. More importantly, the allocation of new frames should be avoided by eagerly detecting and evaluating these primitive nodes within the parent frame. Only if a primitive fails, a new call-target should be allocated and activated. This is a common performance optimization which has also improved performance in GraalSqueak significantly.

A crucial vm plugin is the BitBlt plugin which is responsible for drawing the ui on a canvas by providing bulk operations for modifying bitmaps. This plugin provides only few primitives, but their implementation is complex as it supports a great number of different so-called combination rules for mixing two bitmaps into one. The original sources of the BitBlt plugin are written in Slang, a Smalltalk subset that can be transpiled to C. These sources, however, can also be used inside an image for simulating loading other images for development and debugging purposes [18].

Similar to RSqueak/VM [8], we propose to use the simulation code from within the vm. This allows a very simple implementation of the BitBlt plugin. The BitBlt code, although not part of the vm code base, is still complex and can take relatively long to be partially evaluated by Graal. Therefore, we propose another hybrid implementation that we call *partial primitives*. First, we analyzed which operations within the BitBlt plugin are used most often and take the most time to run. Since primitives are implemented as nodes, we can provide optimized code for the most common operations using specializations and appropriate guards. An example will be discussed in more detail in section 4.3.

4 Implementation

In this section, we give an overview over the implementation of GraalSqueak 0.4.0 and highlight the most interesting design decisions. The source code of GraalSqueak is available on GitHub.[1]

4.1 High-level Overview

GraalSqueak's code base is organized similarly to other Truffle languages and can be built, tested, and used with the mx[2] command-line tool. The core project and Java package is called `de.hpi.swa.graal.squeak` and contains approximately 21k SLOC which implement the interpreter. Other projects contain tests and various boilerplate code required by Truffle. The image reader supports 32-bit and 64-bit Squeak/Smalltalk images using the Spur image format [8]. The VMMaker package[3] needs to be present in an image if opened in headful mode, as it contains the BitBlt simulation code. The test suite has only few test cases for testing specific parts of the vm. The most important test case, however, is called `SqueakSUnitTest` and runs Squeak/Smalltalk's tests within an image which covers most of GraalSqueak's code base and gives good feedback on its compatibility. GraalSqueak is automatically built and tested on different Java Development Kits (jdks) for each commit using Travis CI.

4.2 Implementing the Bytecode Interpreter

GraalSqueak's bytecode interpreter is implemented by the `ExecuteContextNode`. This node contains two interpreter loops: an optimized bytecode loop for executing a method from scratch (`startBytecode`) as well as a non-optimized version which is used when a method is resumed (`resumeBytecode`). This way, we can avoid that resumed methods are being optimized by Graal's partial evaluator.

[1] https://github.com/hpi-swa/graalsqueak/releases (last accessed 2018-10-18).
[2] https://github.com/graalvm/mx (last accessed 2018-10-18).
[3] http://source.squeak.org/VMMaker/ (last accessed 2018-10-18).

```
@ExplodeLoop(kind = ExplodeLoop.LoopExplosionKind.MERGE_EXPLODE)
void startBytecode(VirtualFrame frame) {
 CompilerAsserts.compilationConstant(bytecodeNodes.length);
 int pc = 0; int backJumpCounter = 0;
 AbstractBytecodeNode node = bytecodeNodes[pc];
 try {
  while (pc >= 0) {
   CompilerAsserts.partialEvaluationConstant(pc);
   if (node instanceof ConditionalJumpNode) {
    ConditionalJumpNode jumpNode = (ConditionalJumpNode) node;
    boolean condition = jumpNode.executeCondition(frame);
    if (jumpNode.getCountingConditionProfile().profile(condition)) {
     int successor = jumpNode.getJumpSuccessor();
     if (CompilerDirectives.inInterpreter() && successor <= pc) {
      backJumpCounter++;
     }
     pc = successor; node = bytecodeNodes[pc]; continue;
    } else {
     int successor = jumpNode.getNoJumpSuccessor();
     if (CompilerDirectives.inInterpreter() && successor <= pc) {
      backJumpCounter++;
     }
     pc = successor; node = bytecodeNodes[pc]; continue;
    }
   } else if (node instanceof UnconditionalJumpNode) {
    UnconditionalJumpNode jumpNode = (UnconditionalJumpNode) node;
    int successor = jumpNode.getJumpSuccessor();
    if (CompilerDirectives.inInterpreter() && successor <= pc) {
     backJumpCounter++;
    }
    pc = successor; node = bytecodeNodes[pc]; continue;
   } else {
    final int successor = getGetSuccessorNode().executeGeneric(frame,
     ↪ node);
    getUpdateInstructionPointerNode().executeUpdate(frame,
     ↪ successor);
    try { node.executeVoid(frame); } catch (NonLocalReturn nlr) { /*
     ↪ ... */ }
    pc = successor; node = bytecodeNodes[pc]; continue;
   }
  }
 } finally {
  LoopNode.reportLoopCount(this, backJumpCounter);
 }
}
```

Figure 2: Optimized bytecode loop with hints for the Graal compiler

Figure 2 shows a simplified and self-contained version of the optimized bytecode loop including appropriate annotations for the Graal compiler. These are now explained in more detail.

The `@ExplodeLoop` annotation in line 1 of Figure 2 instructs the compiler to unroll loops. In our case, it uses the `MERGE_EXPLODE` strategy which is designed especially for bytecode interpreters as it tries to explode all loops while merging copies of the loop body that have identical state. Next, we assert that the number of bytecodes is constant per method instance (line 3) and fetch the first bytecode node. Then, the actual interpreter loop begins with a check that the program counter is reduced to a constant during the partial evaluation phase (line 8). Afterwards, there are three options how control flow can continue.

First, if the node is a conditional jump node, the condition is being executed (line 11). Each conditional jump node maintains a probability value that represents how often the condition is `true` or `false` as part of a counting condition profile provided by Truffle. This profile is invoked on the condition result in line 12. Afterwards, the next program counter is determined depending on whether the condition was `true` or `false` (line 13 and 19). If executed in the interpreter and if the successor is smaller than the current program counter, a `backJumpCounter` is incremented. This `backJumpCounter` is reported to the compiler through the `LoopNode::reportLoopCount` API on method exit as part of the `finally` block in line 40. This information is used in Truffle's optimization heuristics to further improve the compilation process. Lastly, the successor becomes the current program counter and the interpreter loop continues with the next bytecode.

Second and in the case of an unconditional jump, the next program counter is fetched and analyzed for backward jumps if running is interpreted (line 26 to 31).

Otherwise, the bytecode is executed exactly as in the non-optimized version of the loop: the next program counter is determined (line 33) and stored in case it is requested through Squeak/Smalltalk's context object (line 34). Then, the node is fully executed in a try-catch block to handle `NonLocalReturn` correctly. Finally, the next bytecode node is fetched.

The Javadocs for the different Truffle hints provide further information on how they work or can be used, yet are unable to fully explain how to use them in combination with others or how exactly the Graal compiler benefits from them.

4.3 Partial Primitives

Many of Squeak/Smalltalk's primitive methods provide Smalltalk fallback code which is executed in case a primitive fails in the vm. This mechanism makes it possible to speed up computation-heavy operations if supported by the vm without breaking compatibility.

We believe the notion of specializations in Truffle provides an elegant way to implement only certain code paths which gives vm builder more flexibility.

As an example, we demonstrate how GraalSqueak implements the BitBlt plugin which is used for drawing the ui onto a canvas. A full reimplementation of the BitBlt plugin in Truffle would require a significant amount of work which we believe is

```
class BitBltPlugin extends AbstractPrimitiveFactoryHolder {
 @SqueakPrimitive(name = "primitiveCopyBits")
 abstract static class PrimCopyBitsNode extends AbstractPrimitiveNode {
  // Constructor omitted for brevity.
  Object executeCopy(VirtualFrame frame, PointersObject receiver,
    Object sourceForm, Object destForm) {
   return executeCopy(
    frame, receiver, receiver.at0(BIT_BLT.COMBINATION_RULE),
    receiver.at0(BIT_BLT.SOURCE_FORM),
     receiver.at0(BIT_BLT.DEST_FORM));
  }

  abstract Object executeCopy(VirtualFrame frame, PointersObject receiver,
    Object combinationRule, Object sourceForm, Object destForm);

  @Specialization(guards={"combinationRule == 24", "is32BitForm(sourceForm)",
                          "is32BitForm(destForm)"})
  Object doCombinationRule24(VirtualFrame frame, PointersObject receiver,
    long combinationRule, Object sourceForm, Object destForm) {
   // Apply combinationRule 24 to destForm using sourceForm...
  }
  // More specializations for other combinationRules...
  @Fallback
  Object fallbackToSimulation(VirtualFrame frame, PointersObject receiver,
   Object combinationRule, Object sourceForm, Object destForm) {
   // Fall back to in-image simulation code for BitBlt...
  }
  boolean is32BitForm(PointersObject target) {
   return (long) target.at0(FORM.DEPTH) == 32;
  }
 }
}
```

Figure 3: Simplified implementation of a partial primitive for BitBlt

unnecessary. When profiling calls into the plugin, we notice that only few code paths are activated often and only some of them are slow in performance as they are slow to execute or slow to compile. Truffle specializations allow us to implement these kind of primitives and entire plugins partially.

Figure 3 shows a simplified implementation of a partial `primitiveCopyBits` primitive in GraalSqueak. The `@SqueakPrimitive` is used to defined the name of the primitive while the module name is derived from the class name. Therefore, this primitive is triggered by a Smalltalk method starting with:

```
copyBits
  <primitive: 'primitiveCopyBits' module: 'BitBltPlugin' error:
  ↪ ec>
  "Fallback Smalltalk code...".
```

The node's entry method is defined in line 5 to 10 which simply extracts combination rule, source form, and destination form from the receiver and then calls the abstract method in lines 12 and 13. This method is automatically generated by Truffle's annotation processor. Its implementation first tries to call the specialization `doCombinationRule24`, otherwise it falls back to `fallbackToSimulation`. The specialization has three guards (line 15 to 16) which ensure that the combination rule of the receiver is 24 and that both `Form` objects have a depth of 32-bit. Only in this case the partial Java port of the primitive is activated (line 19) while in all other cases, the simulation code is called (line 25).

5 Evaluation

In this section, we compare GraalSqueak 0.4.0 with the OpenSmalltalkVM (tag `201810071412`), the standard vm for Squeak/Smalltalk, and RSqueak/VM (commit `533c836e`) in terms of run-time performance. For this, we run the Are-We-Fast-Yet micro benchmark suite [15] with ReBench[4] in the same 64-bit[5] Squeak/Smalltalk trunk image (update level: 18163) on the same hardware, a Dell PowerEdge 2950 (CPU: 2.33GHz Intel Xeon CPU E5410; Memory: 8x4GB DDR2-667MHz SDRAM ECC). The benchmark results, however, should still be taken with a grain of salt as neither memory consumption nor CPU time were measured. Compared to OpenSmalltalkVM and RSqueak/VM which both run on a single CPU core, GraalSqueak on GraalVM uses multiple Truffle compiler threads by default and can easily consume two gigabytes of memory in case of a 50 megabyte image.

The benchmark results are shown in Figure 4. In six of 12 micro benchmarks, GraalSqueak performs worse than OpenSmalltalkVM. The worst case is the DeltaBlue

[4] https://github.com/smarr/ReBench (last accessed 2018-10-18).

[5] RSqueak/VM does not support 64-bit Spur images yet, so we had to use an equivalent 32-bit version. Nonetheless, the corresponding results should be comparable as RSqueak/VM always operates in 64-bit mode independently of the image.

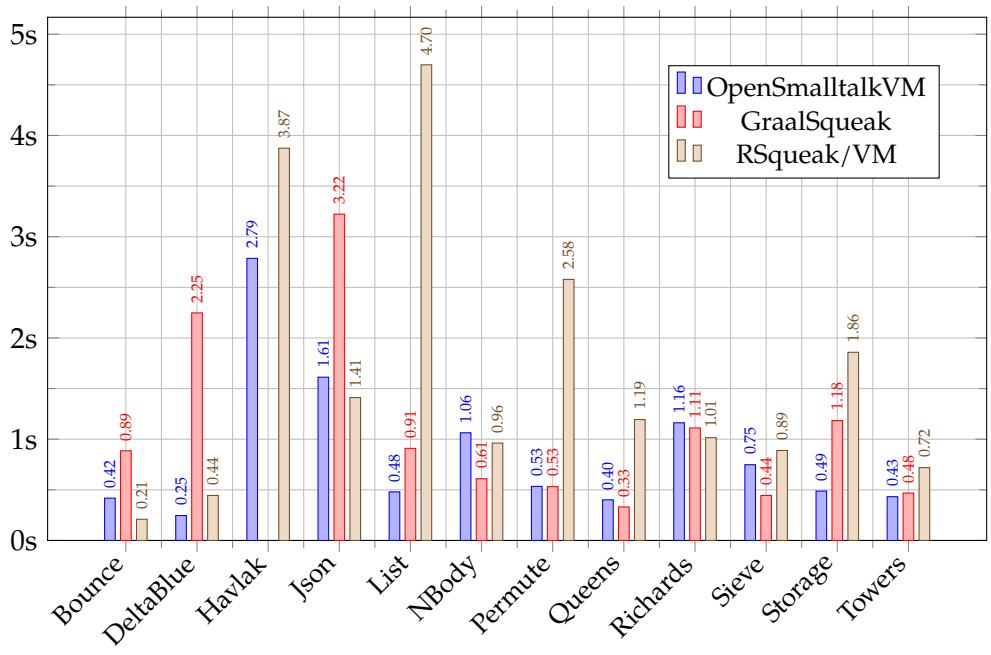


Figure 4: Are-We-Fast-Yet benchmark results in seconds (lower is better)

benchmark which runs approximately 9.14x slower on GraalSqueak. Its performance is comparable to the standard Squeak/Smalltalk vm in another four benchmarks. However, it is faster when running the NBody (+1.74x) and Sieve benchmarks (+1.68x). Compared to RSqueak/VM, it is faster, often significantly, in half of the benchmark suite, and slower or comparable in the other half. Although this benchmark was performed on an early version of GraalSqueak, its overall performance is already competitive with the two other Smalltalk vms.

6 Discussion

In this section, we discuss our approach and GraalSqueak. The main goal was to get the programming system up and running which GraalSqueak is capable of. This, however, requires a fully functional interpreter with support for many primitives including the ones provided by the BitBlt plugin. In a Squeak/Smalltalk trunk image (update level: 18163), about 274 of 374 SUnit test cases (73.26%) pass on GraalSqueak 0.4.0. Most of the failing tests target primitives that are not yet implemented (e.g. SqueakSSL primitives). GraalSqueak's code base consists of roughly 22,5k SLOC which is comparable to RSqueak/VM (approximately 21,3k SLOC) which in turn is written in PyPy's language implementation framework RPython.

With regard to run-time performance, we believe GraalSqueak still has potential as performance has been a secondary goal so far. One performance bottleneck, for example, is its internal representation for `BoxedFloat64` objects which are used in 32-bit

Squeak/Smalltalk images. Furthermore, Truffle is known for slow warmup times [2] which are especially noticeable when opening the programming environment. The reason for this is that the Graal compiler needs seconds, sometimes minutes to compile important code paths that are responsible for drawing the ui which results in poor response times in the first seconds. Responsiveness significantly improves over time while the user interacts with the environment.

GraalSqueak can also be compiled into a standalone binary with SubstrateVM. Although SubstrateVM improves warmup times as promised, an aot-compiled GraalSqueak binary is not yet able to provide snappy response times. More importantly, peak-performance is negatively effected. One reason for this could be that its garbage collector is currently unable to handle the allocation of many small objects well, which happens continuously in Squeak/Smalltalk.

7 Related Work

In this section, we present related work and compare it to our work on GraalSqueak.

OpenSmalltalkVM and Sista The OpenSmalltalkVM [17] is the default vm for Squeak/Smalltalk and variations of it include Sista [3] which stands for "Speculative Inlining SmallTalk Architecture". Instead of optimizing code purely on vm-level, the vm provides an API which can be used from inside a Smalltalk environment to retrieve profiling information. This information can then be used to apply optimizations on image-level which can also be persisted when saving the image. Compared to GraalSqueak, the OpenSmalltalkVM comes with its own jit compiler, garbage collector, and other components which need to be maintained as part of the project.

RSqueak/VM RSqueak/VM [5] is an alternative interpreter for Squeak/Smalltalk and written in the language implementation framework RPython [1]. Therefore, it leverages the same meta-tracing jit compiler that is also used in PyPy to optimize its bytecode loop. GraalSqueak, on the other hand, uses the Graal compiler which performs node rewriting and partial evaluation.

SOMns SOMns [16] is an implementation of the Newspeak language [6] in Truffle. Since it is completely file-based, it does not provide compatibility with image-based Newspeak and traditional Smalltalk systems. In contrast, GraalSqueak is designed to be compatible with existing Squeak/Smalltalk images and is able to display the programming environment. It also supports language features such as sender modifications and provides various vm plugins.

Sulong Sulong [20] is a bytecode-based interpreter for LLVM bitcode, written in Truffle, and maintained by Oracle Labs as part of the GraalVM project. Its bytecode loop employs similar Truffle hints as GraalSqueak to support the compiler in optimizing run-time performance.

8 Conclusion and Future Work

In this paper, we presented an approach as well as and an implementation of a fast Squeak/Smalltalk vm in Truffle for the GraalVM. The vm uses a bytecode interpreter to provide full compatibility to the language specification. Although Truffle is designed for implementing ast interpreters, GraalSqueak achieves good run-time performance because it provides appropriate hints for the Graal compiler. Moreover, the vm uses different internal representations for providing support of Smalltalk `Context` objects to further improve performance. Instead of implementing all primitives including the ones provided by the BitBlt plugin for drawing purposes, we demonstrated how only certain code paths can be accelerated with partial primitives implemented as node specializations in Truffle. A benchmark demonstrated that GraalSqueak's run-time performance is competitive compared to OpenSmalltalkVM, the standard vm for Squeak/Smalltalk, as well as RSqueak/VM, an RPython-based vm employing PyPy's jit.

In the future, we want to further increase performance as there is still potential for improvements. A closer look at micro benchmarks and at the benchmark results are good starting points. Additionally, it would be interesting to develop a benchmark to measure the responsiveness of Squeak/Smalltalk's ui as partial evaluation has noticeable side-effects as discussed in section 6. Since GraalSqueak combines a flexible Smalltalk-based programming system and the polyglot runtime system GraalVM, we think it also opens up opportunities for novel experiments in the area of polyglot programming.

Acknowledgments

We gratefully acknowledge the financial support of Oracle Labs,[6] HPI's Research School,[7] and the Hasso Plattner Design Thinking Research Program.[8]

References

[1] D. Ancona, M. Ancona, A. Cuni, and N. D. Matsakis. "RPython: A Step Towards Reconciling Dynamically and Statically Typed OO Languages". In: *Proceedings of the 2007 Symposium on Dynamic Languages*. 2007, pages 53–64. doi: 10.1145/1297081.1297091.

[6] https://labs.oracle.com/ (last accessed 2018-10-18).
[7] https://hpi.de/en/research/research-school.html (last accessed 2018-10-18).
[8] https://hpi.de/en/dtrp/ (last accessed 2018-10-18).

[2] E. Barrett, C. F. Bolz-Tereick, R. Killick, S. Mount, and L. Tratt. "Virtual Machine Warmup Blows Hot and Cold". In: *Proc. ACM Program. Lang.* 1 (2017), 52:1–52:27. doi: 10.1145/3133876.

[3] C. Béra, E. Miranda, T. Felgentreff, M. Denker, and S. Ducasse. "Sista: Saving Optimized Code in Snapshots for Fast Start-Up". In: *Proceedings of the 14th International Conference on Managed Languages and Runtimes*. 2017, pages 1–11. doi: 10.1145/3132190.3132201.

[4] A. P. Black, O. Nierstrasz, S. Ducasse, and D. Pollet. *Pharo by Example*. Lulu.com, 2010.

[5] C. F. Bolz, A. Kuhn, A. Lienhard, N. D. Matsakis, O. Nierstrasz, L. Renggli, A. Rigo, and T. Verwaest. "Back to the Future in One Week – Implementing a Smalltalk VM in PyPy". In: *Self-Sustaining Systems: First Workshop*. 2008, pages 123–139. doi: 10.1007/978-3-540-89275-5_7.

[6] G. Bracha, P. Ahe, V. Bykov, Y. Kashai, and E. Miranda. *The Newspeak Programming Platform*. 2008. url: http://bracha.org/newspeak.pdf (last accessed 2017-07-01).

[7] L. P. Deutsch and A. M. Schiffman. "Efficient Implementation of the Smalltalk-80 System". In: *Proceedings of the 11th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. 1984, pages 297–302. doi: 10.1145/800017.800542.

[8] T. Felgentreff, T. Pape, P. Rein, and R. Hirschfeld. "How to Build a High-Performance VM for Squeak/Smalltalk in Your Spare Time: An Experience Report of Using the RPython Toolchain". In: *Proceedings of the 11th Edition of the International Workshop on Smalltalk Technologies*. 2016, 21:1–21:10. doi: 10.1145/2991041.2991062.

[9] B. Freudenberg, Y. Ohshima, and S. Wallace. "Etoys for One Laptop Per Child". In: *Proceedings of the 7th International Conference on Creating, Connecting and Collaborating through Computing*. 2009, pages 57–64. doi: 10.1109/C5.2009.9.

[10] A. Goldberg and D. Robson. *Smalltalk-80: The Language and Its Implementation*. The Blue Book. Addison-Wesley Longman, 1983. isbn: 978-0-201-11371-6.

[11] M. Graber, T. Felgentreff, R. Hirschfeld, and A. Borning. "Solving Interactive Logic Puzzles With Object-Constraints: An Experience Report Using Babelsberg/S for Squeak/Smalltalk". In: *Proceedings of the Workshop on Reactive and Event-based Languages and Systems (REBLS), co-located with the Conference on Object-oriented Programming, Systems, Languages, and Applications (OOPSLA)*. 2014.

[12] D. Ingalls, T. Kaehler, J. Maloney, S. Wallace, and A. Kay. "Back to the Future: The Story of Squeak, a Practical Smalltalk Written in Itself". In: *SIGPLAN Not.* 32.10 (1997), pages 318–326. doi: 10.1145/263700.263754.

[13] A. Kay. *Squeak Etoys, Children & Learning*. 2005. url: http://vpri.org/pdf/rn2005001_learning.pdf (last accessed 2017-07-01).

[14] J. Maloney. *Morphic: The Self User Interface Framework*. 1995. url: http://ftp.squeak.org/docs/Self-4.0-UI-Framework.pdf (last accessed 2017-07-01).

[15] S. Marr, B. Daloze, and H. Mössenböck. "Cross-language Compiler Benchmarking: Are We Fast Yet?" In: *SIGPLAN Not.* 52.2 (2016), pages 120–131. doi: 10.1145/3093334.2989232.

[16] S. Marr, C. Torres Lopez, D. Aumayr, E. Gonzalez Boix, and H. Mössenböck. "A Concurrency-Agnostic Protocol for Multi-Paradigm Concurrent Debugging Tools". In: *Proceedings of the 13th ACM SIGPLAN International Symposium on Dynamic Languages*. 2017. doi: 10.1145/3133841.3133842.

[17] E. Miranda and contributors. *OpenSmalltalkVM*. 2017. url: https://github.com/OpenSmalltalk/opensmalltalk-vm (last accessed 2018-05-08).

[18] E. Miranda, E. Gonzalez Boix, C. Béra, and D. Ingalls. "Two Decades of Smalltalk VM Development". In: *Proceedings of the 10th ACM SIGPLAN International Workshop on Virtual Machines and Intermediate Languages*. 2018, pages 1–10. doi: 10.1145/3281287.3281295.

[19] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai. "Scratch: Programming for All". In: *Communications of the ACM* 52.11 (2009), pages 60–67. doi: 10.1145/1592761.1592779.

[20] M. Rigger, M. Grimmer, C. Wimmer, T. Würthinger, and H. Mössenböck. "Bringing Low-level Languages to the JVM: Efficient Execution of LLVM IR on Truffle". In: *Proceedings of the 8th International Workshop on Virtual Machines and Intermediate Languages*. 2016, pages 6–15. doi: 10.1145/2998415.2998416.

[21] A. Rigo and S. Pedroni. "PyPy's Approach to Virtual Machine Construction". In: *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*. 2006, pages 944–953. doi: 10.1145/1176617.1176753.

[22] D. Ungar and R. B. Smith. "Self: The Power of Simplicity". In: *Conference Proceedings on Object-oriented Programming Systems, Languages and Applications*. 1987, pages 227–242. doi: 10.1145/38765.38828.

[23] C. Wimmer, V. Jovanovic, E. Eckstein, and T. Würthinger. "One Compiler: Deoptimization to Optimized Code". In: *Proceedings of the 26th International Conference on Compiler Construction*. 2017, pages 55–64. doi: 10.1145/3033019.3033025.

[24] T. Würthinger, C. Wimmer, A. Wöß, L. Stadler, G. Duboscq, C. Humer, G. Richards, D. Simon, and M. Wolczko. "One VM to Rule Them All". In: *Proceedings of the 2013 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software*. 2013, pages 187–204. doi: 10.1145/2509578.2509581.

Examining Dependability in the Internet of Things

Lukas Pirl

Operating Systems and Middleware Group (OSM)
Hasso Plattner Institute for Digital Engineering (HPI)
lukas.pirl@hpi.de

Due to an increasing number of Internet of Things (IoT) technologies which are being deployed for critical applications, their dependability properties need to be increasingly considered likewise. We evaluate the concept of Software Fault Injection (SFI) for dependability assessments of IoT systems. Investigations are conducted in three settings: virtually in simulations, physically in the HPI IoT Lab and in the field. Special attention is payed to the representativeness of the results from the three aforementioned settings. On an abstract level, this research explores a systematic approach to dependability assessments of IoT systems. On a concrete level, we will contribute results from dependability assessments of IoT technologies.

In the context of a joint project (with, e.g., *DB Systel GmbH*), the wireless network standard *IEEE 802.11p* for Vehicle-to-Vehicle communication is being investigated for applicability in railway use cases. First experiments in the IoT Lab revealed that the latency of initiating connections and packet Round Trip Times (RTTs) are indeed lower compared to conventional wireless networks (i.e., in ad hoc or access point mode). This holds even in the presence of a third, maliciously disturbing (i.e., *jamming*) station. In current efforts, we analyze the implementation of the *IEEE 802.11p* network stack in the network simulator *ns-3* and how it can be leveraged to gather gainful insights from simulations, including SFI experiments.

1 Overview

The growing importance of distributed software systems for individuals, enterprises, and governments can hardly be disputed. As a consequence, the dependability of the critical systems has to be guaranteed with great confidence. At the same time, the satisfaction of this need is complicated by the increasing quantity, rate of change, and complexity of contemporary distributed software systems.

In established areas, such as Cloud computing, manifold best practices to ensure systems' dependability have evolved. In the Internet of Things (IoT) however, unprecedented challenges narrow the applicability of established Fault Tolerance Mechanisms (FTMs). Foremost, typical IoT devices face notable resource constraints regarding computation, storage, and communication. Further, the devices tend to operate in uncontrolled – if not hostile – environments. Finally, IoT systems are usually widely distributed, where ensuring dependability is notoriously difficult.

Apart from the technical challenges which render the development of dependable systems non-trivial even for experts, economic circumstances come into addition. For instance, the pressure for short time-to-market cycles is likely to hamper the maturing of the systems. Recently, the digital transformation increases the pressure

for innovation and Information Technology (IT) starts to pervade all industries. This leads to the circumstance that formerly non-IT-centric industries operate and even develop IT systems. Together with the aforementioned challenges regarding distributed software systems in general and the IoT specifically, it does not seem to be far-fetched that the lack of experiences in "digitally transforming" industries worsens the situation regarding dependability.

Since dependability concerns are not a novel phenomenon, development standards from several areas intend to raise the awareness for dependability, for example by requiring hazard and risk assessments [5]. Some standards even highly recommend Software Fault Injection (SFI) to test systems for their resilience to failure [7]. SFI is the concept of using software to forcefully introduce suspected error causes (faults) or erroneous states (errors) into a Service Under Consideration (SUC) in order to assess its dependability. Despite its prospects, SFI does not seem to be widely integrated into software development practices, not to mention assessments of production systems. Nevertheless, the famous example of Netflix' Chaos Monkey [11] proves that SFI can be integrated into development and even operation practices. The throughout practice of SFI, can thus foster a detailed understanding of the systems' dependability properties.

Especially with the advent of IoT technologies in critical infrastructures, such as utility or rail services, the approaches to dependability assurance need to be advanced further as well. SFI, with its experimental nature, its easily employable high-level black-box perspective and its mental model rooted in the failure space, might be a valuable approach to counter the challenges described. Since the IoT, as a deployment model, rather equals an ecosystem than a single SUC, it is an open question how SFI can be applied to leverage its full potential in this domain.

The remainder of this document is structured as follows: In the subsequent paragraph, we will describe the general approach and overall goal of our research. Starting with Section 2, this document will be oriented towards our particular current efforts and therefore take a more technical perspective.

Approach Given that IoT systems are likely to grow in quantity, complexity, and criticality in evermore areas, we want to examine approaches to assess their dependability. Due to the reasons given above, SFI will be the encompassing direction of our research. Joint projects with the manufacturing industry (e.g., *Siemens*), the rail services industry (e.g., *Deutsche Bahn*), and research organizations (e.g., *German Aerospace Center* (DLR)) will ensure the practical relevance and the topicality of our work. In these projects, further practical case studies will be conducted which will consolidate the foundation we base our theoretical work upon.

The case studies in the context of the joint projects will include dependability assessments of IoT systems. As a result, we will not only contribute the results from the concrete assessments, but also re-evaluations of established FTMs under the yet little explored circumstances. Adding to that, on a more abstract level, this will trial the usefulness of existing approaches to SFI for the IoT. From the obtained insights, we want to develop a structured approach to SFI for assessing dependability in the IoT. If we can demonstrate the benefit of SFI for the IoT and how to maximize this

benefit systematically, we hope to further promote SFI as a valuable tool to attain dependability for IoT systems and a corresponding awareness.

The investigations will be conducted in three different settings, i.e., virtually in simulations, physically in the HPI IoT Lab, and finally in the field. Simulations play a key role in experimenting with technologies, architectures, and algorithms prior their real-life deployment. It is especially helpful to gain insights prior real-life deployments in a well-controlled setting if, e.g., the deployments are large, expensive and laborious to set up. While investigations in a physical lab serve similar purposes, they additionally reveal effects which remained undisclosed in simulations. Those discrepancies between simulations and the reality can occur, e.g., when the models of the simulations are over-simplified or immature. Similar effects are to be expected when investigating the SUCs in the field, where a vast number of environmental factors start to play in. In addition to the insights directly related to the projects, we want to explore if patterns regarding the discrepancies between settings emerge; under which circumstances the discrepancies harm the transfer of insights between settings; and how the discrepancies can be lessened.

To enable a continuous integration of the dependability assessments in development and operation practices, processes need to enable full automation. In past research, we found that exploiting machine- and human-readable fault and dependability models for the automated identification and exercise of SFI helps in systematically assessing complex distributed software systems [3]. A systematic assessment is, e.g., characterized by being reproducible, as well as by being able to achieve a predictable and high coverage. Aside from the advantages for automation, the creation of the aforesaid models alone supports the understanding of the systems' dependability properties.

Related Work Although the IoT is a significant field of research, it can be stated that there is little publicly available research on introducing IoT technologies in railway infrastructures. A similar situation can be observed regarding SFI in particular: although this topics is being examined since the early years of computer science, few current works seem to apply the concept to IoT systems. Regarding dependability of IoT systems in general, software security concerns appear to be a more active field of research which is, contrastingly, not one of our main considerations. Scientific contributions which assess the dependability of IoT hardware using fault injection (e.g., [2]) confirm the suspected usefulness of SFI for the IoT.

One of the few closely related works is a dependability evaluation tool for the IoT presented by Silva et al. [10]. With the aforementioned tool, the dependability of IoT applications in the presence of failing hardware and network links can be assessed at early design time. A notable functionality of the tool is its ability of deriving a fault tree from an arbitrarily laid out network topology of the IoT application under consideration, e.g., to identify dependability bottlenecks. Other works have proposed the use of Markov Chains to evaluate the reliability and availability of IoT applications [6]. In contrast to our work, a compound and systematic approach including, e.g., real-life assessments and SFI – which we think is a key to dependable distributed software systems – has not been considered yet.

In [4], Gluhak et al. survey available test beds (as of 2011) for experimental research on IoT systems and present a taxonomy to classify them. Simulations are valued as an important tool for the design of IoT systems, but, in contrast to our work, are not a major concern in their survey. We see that large parts of their insights (e.g., 2- and 3-tier architectures, considerations regarding composition and the devices' operation environments) are still applicable to today's diverse understanding of the IoT. Even thought the authors propose ideas to close the identified gaps with concrete projects, it is outside their scope to provide abstract and systematic approaches to experimental dependability assessments for the IoT, not speaking of SFI.

Alipour recently identified fault types for SFI in event-based IoT systems [1]. This work however is possibly in an early stage, since the so far identified faults (i.e., loss, duplication and delay of events) appear to be well-known from distributed software systems research. It is nevertheless to be considered, if incorporating the identified fault types into the investigations in the context of this research would be beneficial.

2 Research Activities

Together with *DB Systel GmbH*, *Siemens Mobility AG*, *Deutsches Zentrum für Luft- und Raumfahrt e. V. / German Aerospace Center* (DLR), and *DRALLE Systementwicklungen*, we work in the publicly funded project *Rail2X – Smart Services*. The overall goal of this project is to explore, design, and develop Vehicle-to-Everything (V2X) applications for railways (therefore *Rail2X*; also in analogy to, e.g., Car2X, Ship2X). Three novel use cases of V2X applications for railways guide the activities of all consortium partners within the project:

- *use case 1: predictive maintenance*: trains collect and transport maintenance data of field elements instead of installing wiring under a lot of efforts or using expensive mobile networks;
- *use case 2: on-demand level crossing*: Car2X-enabled cars automatically request barriers to open instead of requiring drivers to get out of the car for calling the train dispatcher manually;
- *use case 3: flag stop*: stop buttons send their state to trains wirelessly instead of relying on visual communication which can be problematic near curves or under poor visibility conditions.

The role of the HPI Operating Systems and Middleware Group; as part of the *Rail2X* consortium; is to contribute expertise and research in the fields of middleware technologies, middleware architectures, and software systems' dependability (including IT security and data privacy).

Our efforts are currently focused on use case 1 which is visualized in Figure 1 and specifically on the underlying wireless network technology *IEEE 802.11p*. This technology is the basis for standardized vehicular communication in the US (*IEEE 1609*

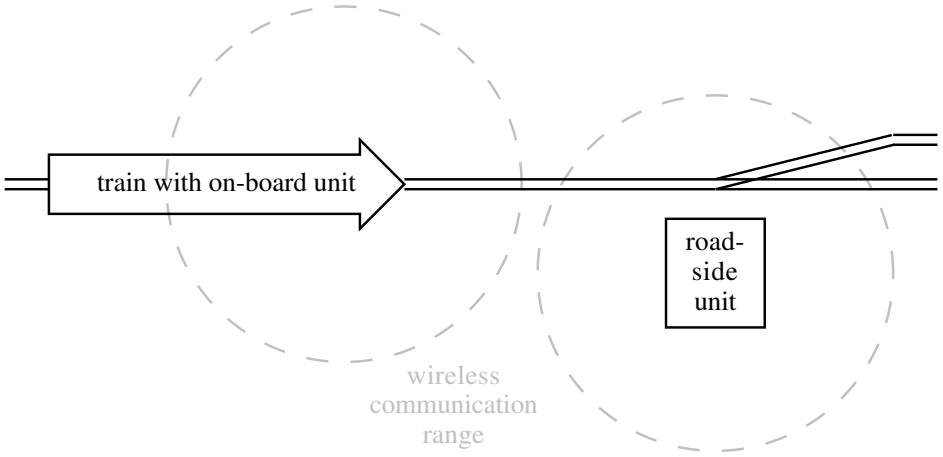


Figure 1: *Rail2X* use case 1: passing trains collect data from field elements (e.g., switches) using the short-range wireless network standard *IEEE 802.11p* to enable predictive maintenance

standards family) and Europe (*ETSI ITS-G5* by the European Telecommunications Standards Institute (ETSI) Intelligent Transport Systems (ITS) Technical Committee). *IEEE 802.11p* defines network protocols for the physical layer (OSI level 1) and data link layer (OSI level 2). On the physical layer, *IEEE 802.11p* operates in a 5.9 GHz band, close to the well-established *IEEE 802.11n* wireless network standard. While conventional *IEEE 802.11* wireless networks typically operate in Basic Service Set (BSS, i.e., an access point) mode, a notable innovation in *IEEE 802.11p* is the introduction of the Outside the Context of a BSS (OCB, comparable to broadcasts) mode. Since *IEEE 802.11p* is the basis for all *Rail2X* applications, it is especially relevant for the investigations in the context of the project. The fact that *IEEE 802.11p* is a relatively new standard, with a less extensive body of research and fewer practical experiences compared to well-established wireless network technologies, makes it particularly interesting for our research.

In a concise student's project, a simulation of *IEEE 802.11p* communication has been conducted in the event-based network simulator *ns-3*.[1] *Ns-3* is widely used and generally recognized as an advanced tool for simulating a wide variety of network technologies (e.g., the realistic simulation of contemporary cellular networks). However, the results from the aforementioned student's project suggest that the communication quality is largely unaffected by the distance between two stations when inside the communication range. Once the distance between two stations reaches the maximum communication range, the communication quality degrades abruptly. The results therefore suggested an unrealistic model of *IEEE 802.11p* in *ns-3* and a discrepancy between simulation and reality (lab measurements from our project partner DLR confirm this intuition). Consequently, these observations motivated a

[1] https://www.nsnam.org (last accessed 2018-10-18).

detailed assessment of the *IEEE 802.11p* models in *ns-3*, so that the models can be configured or enhanced to yield more realistic results.

Until now, we created a setup to analyze the *IEEE 802.11p* model in *ns-3* and conducted first experiments. The results clearly confirmed our assumption that the communication quality between two stations degrades abruptly when close to the communication range. Figure 2 shows that the packet loss ratio between two stations degrades approximately between 91 and 103 meters, while Figure 3 shows that the packet Round Trip Times (RTTs) between two stations degrade approximately between 99 and 103 meters. Considering that *IEEE 802.11p* has a communication range of up to 1000 meters by design and a less abrupt signal loss behavior [9], there is a lot of room for ongoing research.

Apart from assessments in the simulation setting, we also conducted physical *IEEE 802.11p* measurements in the IoT Lab [8]. Since the *IEEE 802.11p* intends to enable low-latency communication in network with highly dynamic topologies, we initially measured the latency of joining an Independent Basic Service Set (IBSS, i.e., an ad hoc network), a BSS, or no BSS in OCB mode. The results depicted in Figure 4 show that OCB mode in fact has the lowest latency with the lowest variability to join a network compared to the other two modes.

In further experiments, we conducted practical SFI experiments in the IoT Lab. Since vehicular networks are potentially exposed to malicious parties, their resistance to, e.g., so-called *jamming* attacks must be of major concern. Jamming describes the usually malicious practice of transmitting random or specifically crafted radio signals as fast and energetic as possible with the intention to disturb legitimate wireless communication. We investigated the degradation of RTTs between two stations in the presence of a third station which jammed the *IEEE 802.11p* frequency band (i.e., pseudo-random data on OSI level 3, therefore valid level 2 and 1 packets). As shown in Figure 5, *IEEE 802.11p* in OCB mode achieves the lowest RTTs with the lowest variability compared to the other two modes.

Future Work This research is work in progress and any future work will be aligned with the spanning research goals (see Section 1). And of course, the work in *Rail2X* will continue according to the project goal with its use cases and project plan (touched in the beginning of this Section).

Concretely, we will continue to assess *IEEE 802.11p* for the use in *Rail2X*, e.g., with the stations in movement (up to 300 km/h) or with hopping stations. As it is a central concern of this research, it is needless to say that we will continue to conduct SFI experiments; be it with the technique of denial-of-service attacks on OSI layer 2 and lower (e.g., jamming), layer 3–6 (e.g., packet flooding), or on the application layer. These next steps apply to all the three settings which we consider (i.e., in the simulation, lab, and field).

Specifically regarding the simulation in *ns-3*, we continue to identify configurations, and possibly patterns and processes to make the results from simulations more realistic. On the one hand, the maximum range has to be extended (i.e., a range of around 400–500 m [9]). On the other hand, the degradation of the signal quality has

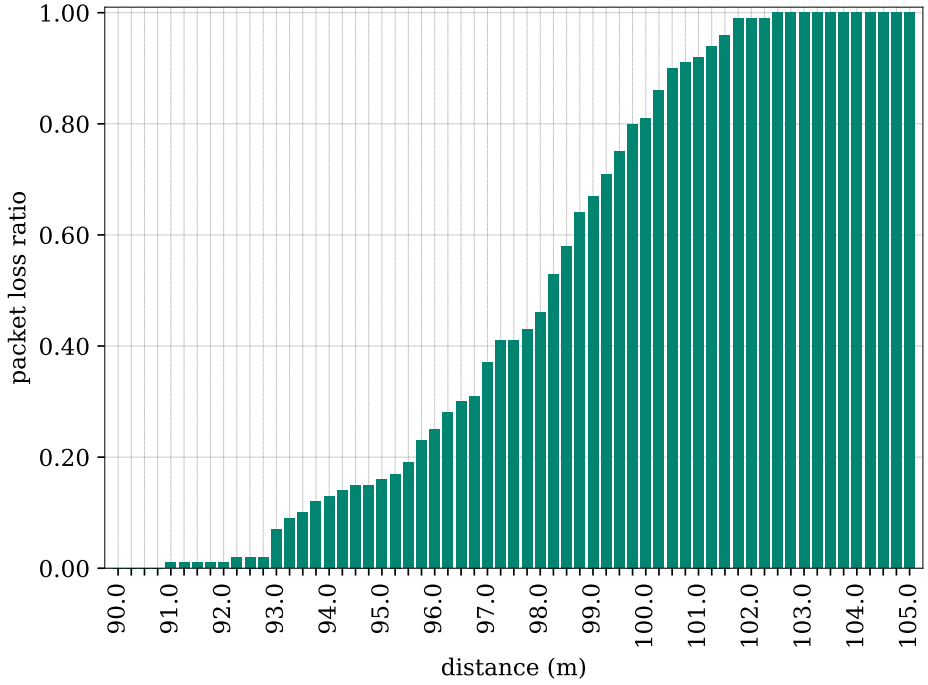


Figure 2: Packet loss ratios relative to the distance between two stations communicating over *IEEE 802.11p* (measured in *ns-3*, 1000 packets per distance)

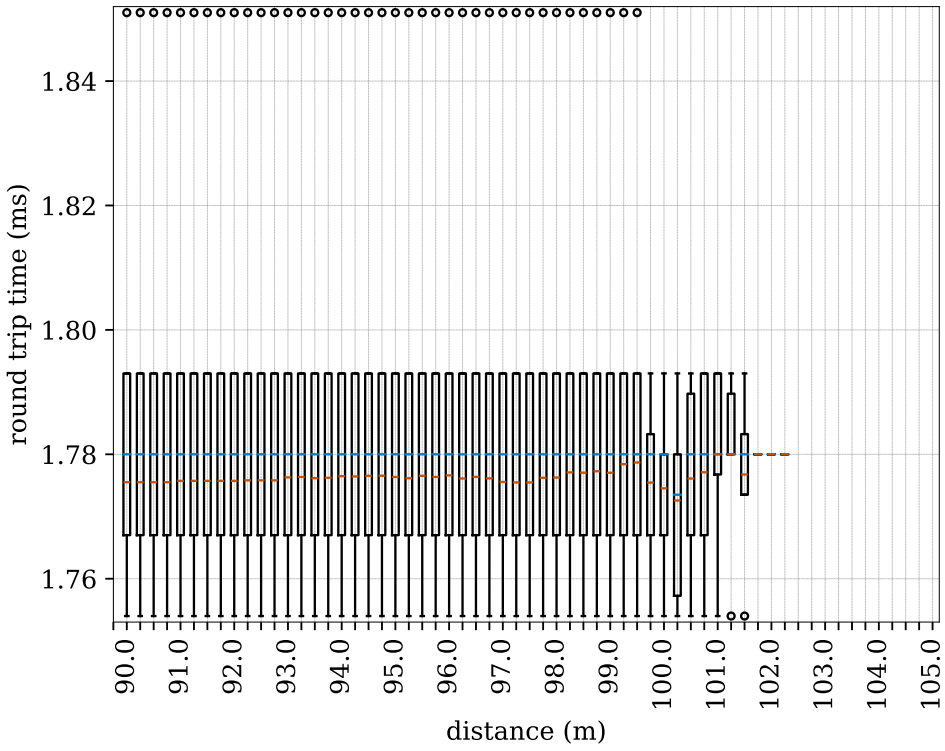


Figure 3: Packet round trip times relative to the distance between two stations communicating over *IEEE 802.11p* (measured in *ns-3*, 1000 packets per distance, blue: median, orange: mean)

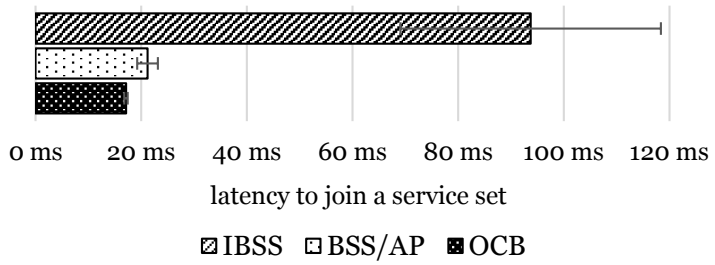


Figure 4: *IEEE 802.11p*'s OCB mode achieves the lowest latency with the lowest variability to join a network (measurements in the IoT Lab, 100 trials each)

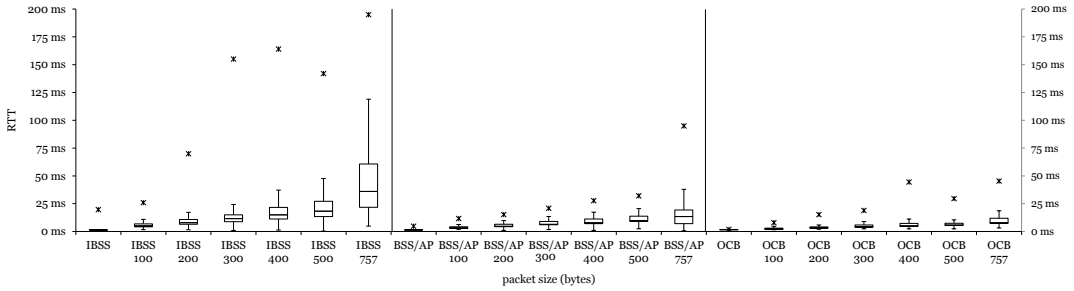


Figure 5: Especially with larger packet sizes, *IEEE 802.11p*'s OCB mode achieves the lowest RTTs with the lowest variability in the presence of a third jamming station (SFI experiments in the IoT Lab, 2000 trials each)

to degrade over nearly the whole communication range (i.e., approximately linear degradation, starting at around 50 m [9]).

Further investigations and SFI experiments in the IoT Lab and finally in the field will continue to contribute to *Rail2X* and the overall goal of this research.

References

[1] M. A. Alipour. "Fault injection in the internet of things applications". In: *Proceedings of the 1st ACM SIGSOFT International Workshop on Testing Embedded and Cyber-Physical Systems (TECPS'17)*. 2017. doi: 10.1145/3107091.3107095.

[2] C. A. Boano, K. Rmer, and T. Voigt. "RELYonIT: Dependability for the Internet of Things". In: *IEEE IoT Newsletter Januray* 13 (2015).

[3] L. Feinbube, L. Pirl, P. Tröger, and A. Polze. "Software fault injection campaign generation for cloud infrastructures". In: *IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C'17)*. 2017, pages 622–623.

[4] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo. "A survey on facilities for experimental internet of things research". In: *IEEE Communications Magazine* 49.11 (2011), pages 58–67.

[5] ISO. "26262: Road vehicles – Functional safety". In: *International Standard ISO/FDIS* 26262 (2011).

[6] D. Macedo, L. A. Guedes, and I. Silva. "A dependability evaluation for Internet of Things incorporating redundancy aspects". In: *Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control*. 2014, pages 417–422. doi: 10.1109/icnsc.2014.6819662.

[7] B. O'Connor. "NASA software safety guidebook". In: *NASA Technical Standard NASA-GB-8719.13* (2004).

[8] D. Richter, J. Beilharz, L. Pirl, C. Werling, and A. Polze. "Performance of real-time wireless communication for railway environments with IEEE 802.11p". In: *Proceedings of the 52th Hawaii International Conference on System Sciences (HICSS)*. To appear. 2018.

[9] V. Shivaldova and C. F. Mecklenbrauker. "Real-world measurements-based evaluation of IEEE 802.11 p system performance". In: *IEEE 5th International Symposium on Wireless Vehicular Communications (WiVeC'13)*. 2013, pages 1–5.

[10] I. Silva, R. Leandro, D. Macedo, and L. A. Guedes. "A dependability evaluation tool for the Internet of Things". In: *Computers & Electrical Engineering* 39.7 (2013), pages 2005–2018. doi: https://doi.org/10.1016/j.compeleceng.2013.04.021.

[11] A. Tseitlin. *Chaos Monkey released into the wild*. 2012. url: http://techblog.netflix.com/2012/07/chaos-monkey-released-into-wild.html (last accessed 2017-03-02).

Evolutionary Algorithms and Local Search in Combinatorial Optimization

Francesco Quinzan

Algorithm Engineering
Hasso Plattner Institute
francesco.quinzan@hpi.uni-potsdam.de

In the context of black box optimization, the most common way to handle deceptive attractors is to periodically restart the algorithm. To derive the *optimal restart time* of a black-box algorithm, perfect knowledge of the convergence probability function is necessary. This quantity is often unknown or difficult to compute. We analyze various pseudo-Boolean fitness landscapes on these problems, and show that the corresponding restart strategies are nontrivial. We show that Eas with restart strategy perform well on some combinatorial optimization problems.

We investigate the performance of a local search algorithm for the problem of maximizing functions under a partition matroid constraint. We consider non-monotone submodular functions and monotone subadditive functions. We discuss the applicability of our results to the maximum entropy sampling problem, and related determinant function problems. We conclude that local search heuristics are well-suited to approach these problems. This constitutes the basis of a more complex analysis with evolutionary algorithms.

1 Overview

The following is an exposition of my main current and past projects. The project on the optimization of noisy functions, now concluded, is briefly summarised and in line with what has already been presented on this topic. The project on restart strategy is also briefly presented. I also discuss new results, currently under review.

Typically, evolutionary algorithms require as input a *population* of strings of fixed length n. After an *offspring* is generated, a mutation factor is introduced, to ensure full objective space exploration. The fitness is then computed, and the less desirable result is discarded. The $(\mu + 1)$-EA and the $(\mu + 1)$-GA differ in how the offspring is generated. In the first case, the offspring is selected u.a.r. from the input population, while in the latter case a crossover operation is performed on two u.a.r. chosen strings. We use uniform crossover, which consists of assembling a new element by choosing coefficients of one parent or the other with probability $p = 0.5$.

A *restart strategy* for a black-box algorithm is an infinite sequence $(t_1, t_2, \ldots, t_t, \ldots)$ that specifies that the algorithm should be run from a uniformly random starting point for t_1 steps, then restarted uniformly at random and run again for t_2 steps and so forth. Thus, we consider only *a priori* restart strategies, although it has been observed that other kinds of restart techniques may fasten the expected run time of randomized algorithms (cf. de Perthuis de Laillevault et al. [18] and Lissovoi et al. [13]).

The *optimal restart strategy* is the sequence that minimizes the expected number of runs until a run returns the optimal solution. In the context of Las Vegas algorithms, when the density function of the convergence probability over time is known, then the optimal strategy can be computed exactly (cf. Luby et al. [15]). More formally, let $\mathcal{A}$ be any Las Vegas algorithm, and let τ denote its run time. In this context, τ is a r.v. that returns the number of calls to the fitness function, until a solution that satisfies a convergence criterion is reached. We consider convergence criteria that terminate the process when the global optimum is reached, or a solution that gives an approximation of it. Let $p(t)$ the probability of convergence at step t, and $q(t) = \sum_{t' \leq t} p(t')$ the probability of search termination before step t, i.e. the cumulative distribution function.

More specifically, let $\mathcal{A}$ be any Las Vegas algorithm, $p(t)$ the probability of convergence at step t, and $q(t) = \sum_{t' \leq t} p(t)$ the probability of converging before step t - i.e. the cumulative distribution function. The optimal restart strategy for $\mathcal{A}$ is the repeating sequence $\mathcal{S} = (t_*, \dots, t_*, \dots)$ with t_* defined as

$$t_* := \inf_t \left\{ \frac{1}{q(t)} \left(t - \sum_{t' < t} q(t') \right) \right\} \tag{1}$$

Thus, t_* is the first point in time that minimizes the probability of not converging over the probability of converging at previous steps. We refer to this result as Luby theorem.

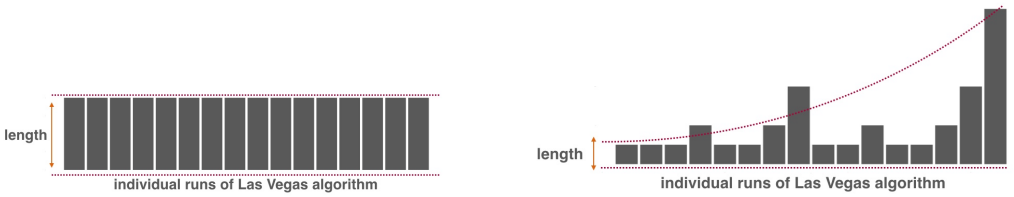
Luby et. al. prove $\mathcal{S}$ to be optimal, both for t_* a finite number, and $t_* = +\infty$ - in the latter case the optimal strategy being not restarting the process at all. In most real world settings, however, $p(t)$ is not known a priori, and a numerical approximation of it is not feasible. Because of this, Luby et. al. present a "universal" method to simulate $\mathcal{S}$ when $p(t)$ is unknown. This strategy is indicated by

$$\mathcal{S}^{\mathbf{univ}} = (1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 1, 1, 2, 4, 8, \dots)$$

Pseudo-code for an algorithm $\mathcal{A}$ running as indicated by $\mathcal{S}^{\mathbf{univ}}$ is given in Algorithm 1. The universal restart strategy and the optimal restart strategy are depicted in Figure 1a and Figure 1b. We adapt the results presented above to the case of evolutionary heuristics running with time budget constraints, or with solution quality constraints. We focus on finding an explicit relationship between the optimal restart point given in Equation 1, the total time budget, and the impact on overall performance. We experimentally compare the performance of the $(\mu + 1)$-EA and without restarts on a combinatorial optimization problem. The *Minimum Vertex Cover* problem consists of finding the smallest set such that each edge of the graph is incident to at least one vertex of the set. For a given graph $G = (V, E)$, with n nodes, a solution is stored in memory as a pseudo-boolean array, its length is the number of vertices of the graph. In all cases, the quality of the solution is evaluated against the following function

$$f(x) = n^2 u(x) + |x_i|_1 \tag{2}$$

We consider the Facebook (NIPS) available on the Stanford Network Analysis Project (SNAP), and search for the Minimum Vertex Cover (cf. Figure 2a). This dataset



(**a**) The best possible restart strategy consists of let the algorithm run for a fixed amount of time (length). The optimal length of each run depends on the algorithm and may be hard to compute.

(**b**) The universal strategy is defined as in Algorithm 1, and approximates the optimal strategy up to a logarithmic factor. This strategy has the advantage that no tuning is needed.

Figure 1: A comparison of two different strategies

Algorithm 1: The Universal Strategy $\mathcal{S}$ for $\mathcal{A}$.

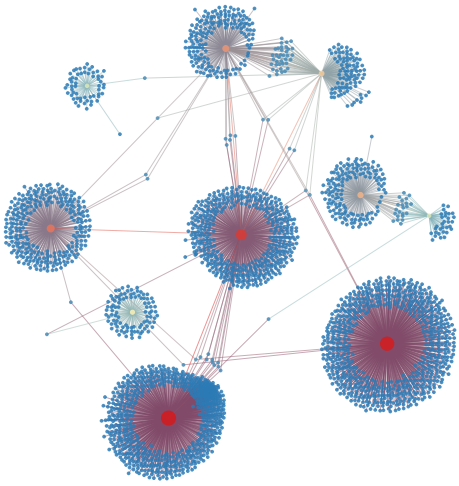
$i \leftarrow 0, t \leftarrow [\,]$
while *convergence criterion not met* **do**
  if $\exists k \in \mathbb{N} : i = 2^k - 1$ **then**
    run $\mathcal{A}$ for 2^{k-1} steps
    $t[i] \leftarrow 2^{k-1}$
  else if $\exists k \in \mathbb{N} : 2^{k-1} \leq i < 2^k - 1$ **then**
    run $\mathcal{A}$ for $t[i - 2^{k-1} + 1]$ steps
    $t[i] \leftarrow t[i - 2^{k-1} + 1]$
  end
  $i \leftarrow i + 1$
end

consists of 'friends lists' from Facebook. Facebook data was collected from survey participants using this Facebook app. The dataset includes node features (profiles), circles, and ego networks.

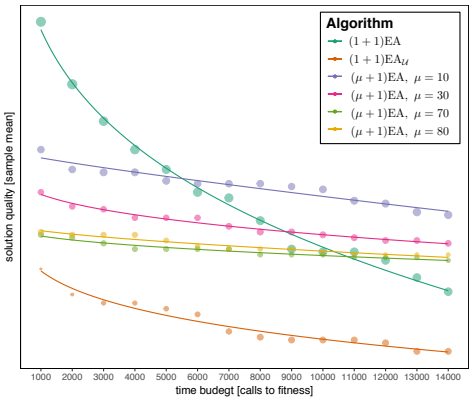
The results are presented in Figure 2b. We observe that for fixed time budget the $(\mu + 1)$-EA following the universal strategy clearly outperforms the $(\mu + 1)$-EA without restart strategies, for $\mu = 1$.

Together with evolutionary algorithms, we study a simple greedy algorithm (see Algorithm 6), for the problem of maximizing a non-negative submodular functions with bounded curvature, under a single matroid constraint. This type of analysis constitutes the bases for studying more complex algorithms on this problem. Greedy is the simple discrete greedy algorithm that appears in Algorithm 6. Starting with the empty set, Greedy iteratively adds points that maximize the marginal values with respect to the already found solution. This algorithm is a mild generalization of the simple deterministic greedy due to Nemhauser and Wolsey [17].

We empirically study Algorithm 6 on the following problem: Given a set of random variables (RVs), find the most informative subset of variables, subject to a



(**a**) The Facebook (NIPS) network, taken from the Stanford Network Analysis Project (SNAP). The nodes size is proportional to the degree. We clearly observe community structure.



(**b**) Number of fitness evaluations (run time) for the $(\mu+1)$-EA with and without restarts (with $\mu = 1$), to find the MVC on the network in Figure 2a. The size of each dot is proportional to the sample standard deviation.

Figure 2: Experiments on a large network

side constraint. This setting finds a broad spectrum of applications, from Bayesian experimental design [19], to monitoring spatio-temporal dynamics [20].

We consider the Berkley Earth climate dataset.[1] This dataset combines 1.6 billion temperature reports from 16 preexisting data archives, for over 39.000 unique stations. For each station, we consider a unique time series for the *average* monthly temperature. We always consider time series that span between years 2015–2017. This gives us a total of 2736 time series, for unique corresponding stations. The code is available at [removed for review].

We study the problem of searching for the most informative sets of time series under various constraints, based on these observations. Given a time series $\mathbf{X} = \{X_t\}_t$ we study the corresponding variation series $\overline{\mathbf{X}} = \{\overline{X}_t\}_t$ defined as $\overline{X}_t = X_t - X_{t-1}$. A visualization of time series $\overline{\mathbf{X}}$ is given in Figure 3a(a).

We compute the covariance matrix Σ between series $\overline{\mathbf{X}}$, $\overline{\mathbf{Y}}$, the entries of which are defined as

$$\mathrm{cov}(\overline{\mathbf{X}}, \overline{\mathbf{Y}}) = \frac{1}{m-1} \sum_{t=1}^{m} (\overline{X}_t - \mathbb{E}\left[\overline{\mathbf{X}}\right])(\overline{Y}_t - \mathbb{E}\left[\overline{\mathbf{Y}}\right]), \tag{3}$$

with $m = 35$ the length of each series. A visualization of the covariance the matrix Σ is given in Figure 3b.

[1] http://berkeleyearth.org/data/ (last accessed 2018-10-18).

Algorithm 2: The Greedy algorithm.

```
S ← ∅
while |S| ≤ Σ_{i=1}^{k} d_i do
    let ω ∈ V maximizing f(S ∪ {ω}) − f(S) and s.t.
    |(S ∪ {ω}) ∩ B_i| ≤ d_i, ∀i ∈ [k]
    S ← S ∪ {ω}
end
return S
```

Assuming that the joint probability distribution is Gaussian, we proceed by maximizing the *entropy*, defined as

$$f(S) = \frac{1 + \ln(2\pi)}{2} |S| + \frac{1}{2} \ln \det_{\Sigma}(S) \tag{4}$$

for any indexing set $S \subseteq \{0, 1\}^n$.

We consider two types of constraints. In a first set of experiments we consider the problem of maximizing the entropy as in (4), under a cardinal constraint only. Specifically, given a parameter d, the goal is to find a subsets of time series that maximizes the entropy, of size at most d of all available data. We also consider a more complex constraint: Find a subset of time series that maximizes the entropy, and s.t. it contains at most d of all available data of each country. The latter constraint is a partition matroid constraints, where each subset B_i consists of all data series measured by stations in a given country.

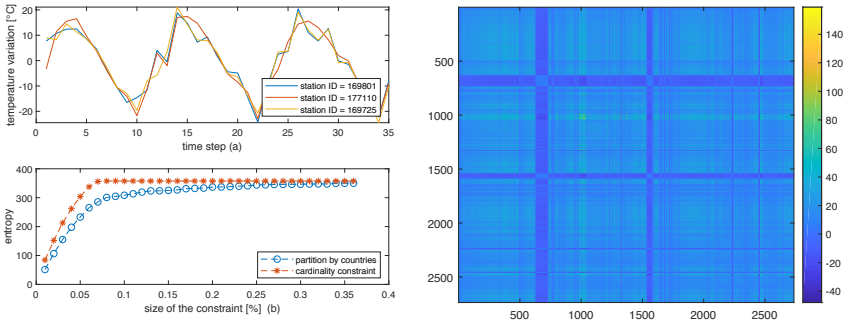
A summary of the results is displayed in Figure 3a(b). We observe that in both cases the entropy quickly evolves to a stationary local optimum, indicating that a relatively small subset of stations is sufficient to explain the random variations between monthly observations in the model. We observe that the Greedy reaches similar approximation guarantees in both cases. We remark that the Greedy finds a nearly optimal solution under a cardinality constraint, assuming that the entropy is (approximately) monotone [9].

In Figure 4 we display solutions found by Greedy for the cardinality and partition matroid constraint, with $d = 10\%$.

We observe that in the case of a cardinality constraint, the sensors spread across the map; in the case of a partition matroid constraint sensors tend to be placed unevenly. We remark that in the original data set, some countries have a much higher density of stations than others.

2 Approach

All tests are carried out by performing a global optimum search with objective space consisting of all pseudo-Boolean strings of fixed size n. In some cases, we add posterior Gaussian noise in order to simulate an environment where errors of controlled



(**a**) A visualization of the monthly temperature variations of three time series in (a), with particularly high variance. . Optimal solution found by Greedy for a uniform constraint and a partition matroid constraint by countries in (b). The f-value of each set of stations is the entropy (4), with Σ the covariance matrix of variation series as in (a) (see Figure 3b).

(**b**) A visualization of the covariance matrix Σ of time series available in the Berkley Earth climate dataset. We consider stations that have full available reports between years 2015-2017, for a total of 2736 stations. We consider the variation between average monthly temperatures of each time series. Each entry of this matrix is computed by taking the sample covariance as in (3).

Figure 3: Our results

standard deviation are produced, and we simulate a resampling operator: we compute the fitness function r times, and take the average.

For each set of experiments we look at the sample mean, sample standard deviation, and infer the trend toward asymptotic behaviour via model regression. All samples considered are of size $N \geq 10^2$; the exact size and relevant information is given in the description of each experiment. Statistical models with different properties are considered. In all cases, tests on the predictions made by the fitting models are performed. For a given experiment described by pairs $\{(x_i, y_i)\}_{i \in I}$, denote with $\bar{y}$ the sample mean, and let $\{f_i\}_{i \in I}$ be the predictions of a given model. We assume that the model is valid if $R^2 > 0.95$, with R^2 the coefficient of determination. This choice is intuitively motivated by the fact that R^2 is the "percent of variance explained" by the model.

3 Related Work

In the context of real-world optimization, there is usually very little analytical knowledge of the problem at hand, and classical numerical methods often fail. Problems of this kind can be approached with *black-box* optimizers – algorithms that access the function to be optimized only via the evaluation of possible solutions.

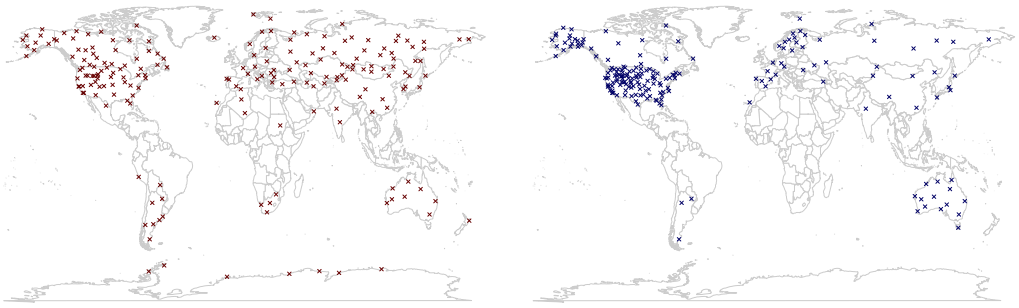


Figure 4: A visualization of the solution found by Greedy for $d = 10\%$ in the case of a uniform constraint (left), and a partition constraint by countries (right). In both case, a solution is obtained by maximizing the entropy as given in (4). The covariance matrix Σ for all possible locations is displayed in Figure 3b. We observe that in the case of a cardinality constraint, the informative stations tend to be spread out, whereas in the partition constraint by countries they tend to be grouped in a few areas. We remark that in the original dataset stations are not distributed uniformly among countries.

Optimizers of this kind often exhibit a large variance in run time, due to a strong dependence of the run time on initial conditions. Typically, a significant number of iterations are wasted trapped in deceptive basins of attraction: sets of states that lead the search away from the optimal solution. A simple and effective strategy for handling this problem is to use *restarts*. In this scenario, an algorithm is periodically re-initialized after some time (the *restart time*), with the hope that if it has already become trapped in a deceptive basin then it can be allotted another chance to discover the global optimum.

The utility of re-initializing with a uniformly chosen random point depends strongly on the function under consideration. Let $\mathcal{X}$ be a fixed domain, and consider any family of functions $\mathcal{F} \subseteq \{f \mid f : \mathcal{X} \to \mathbb{R}\}$ s.t. for all $x \in \mathcal{X}$ there is a $f \in \mathcal{F}$ that attains a unique global optimum at $f(x)$. Then the restart strategy that uses uniform measure on the samples is the optimal choice [7]. On the other hand, if there are more restrictions on the global structure of the space $\mathcal{F}$, different restart strategies can be employed. This is the idea behind *iterated local search* [14] in which restarts are nonuniform, and it is assumed that smaller perturbations can escape deceptive attractor basins. In this paper, we shall assume the former situation, in which no information is known and so the best policy is restarting uniformly at random. Such function families are a common object of study, for example in the field of black-box complexity.

It is possible to derive theoretically the *optimal restart time* – the value that yields the fastest possible convergence over all such restart times. This result is due to Luby, Sinclair, and Zuckermann [15], but was also discovered in the context of backpropagation training for neural networks [16].

The classical result of [5] shows that a greedy algorithm achieves a $1/2$ approximation ration when maximizing monotone submodular functions under partition matroid constraints. [17] showed that no-polynomial time algorithm can achieve a better approximation ratio than $(1 - 1/e)$. Many years later [3] where able to achieve this upper bound using a randomized algorithm. Recently [2] achieved a deterministic 0.5008-approximation ratio by derandomizing search heuristics.

Formally, a set function $f\colon 2^V \to \mathbb{R}$ is *submodular* if for all $U, W \subseteq V$, $f(U) + f(W) \geq f(U \cup W) + f(U \cap W)$. As these functions come from a variety of applications, in this work we will assume that, given a set $U \subseteq V$, the value $f(U)$ is returned from an oracle. This is a reasonable assumption as in most applications $f(U)$ can be computed efficiently. Often in these applications, a realistic solution is subject to some constraints. Among the most common constraints include matroid and Knapsack constraints – see [11]. From these families of constraints the most natural and common type of constraints are uniform matroid constraints also known as cardinality constraints. Optimizing a submodular function given k as a cardinality constraint is to finding a set U, with $|U| \leq k$, that maximizes $f(U)$. In this paper we consider submodular maximization under partition matroid constraints. These constraints are in the intersection of matroid and knapsack constaints and generalize uniform matroid constraints. In partition matroid constraints we are given a collection $B_1, \ldots, B_k$ of disjoint subsets of V, integers $d_1 \ldots d_k$. Every feasible solution to our problem must then include at most d_i elements from each set B_i. Submodular maximization under partition matroid constraints are considered in various applications, e.g. see [8, 12].

As the literature in submodular optimization is immense, we will only review the results on submodular maximization under matroid and knapsack constraints. The classical result of [5] shows that a greedy algorithm achieves a $1/2$ approximation ration when maximizing monotone submodular functions under partition matroid constraints. [17] showed that no-polynomial time algorithm can achieve a better approximation ratio than $(1 - 1/e)$. Many years later [3] where able to achieve this upper bound using a randomized algorithm. Recently [2] achieved a deterministic 0.5008-approximation ratio by derandomizing search heuristics.

The previous approximation ratios can be further improved when assuming that the rate of change of the marginal values of f is bounded. This is expressed by the curvature α of a function. The results of [4, 21] show that a continuous greedy algorithm gives a $\frac{1}{\alpha}(1 - e^{-\alpha})$ approximation when maximizing a monotone submodular function under any matroid constraint. Thus, when $\alpha \leq 1.58933$ the the continuous greedy outperforms the algorithm of [2] and when $\alpha \leq 1$ the continuous greedy outperforms the algorithm of [3]. Finally, [1] show that the deterministic greedy algorithm achieves a $\frac{1}{\alpha}(1 - e^{-\alpha})$ approximation when maximizing submodular monotone functions of curvature α, but only under cardinality constraints.

All of the aforementioned approximation results rely on the fact that f is monotone. In practice submodular functions such as maximum cut, combinatorial auctions, sensor placement and experimental design the functions need not be monotone. To solve such problems using simple greedy algorithms, often assumptions are made that the function f is monotone or that f is under some sense "close" to being monotone.

Practical problems that are solved using greedy algorithms under such assumptions can be found in many articles such as [1, 6, 10, 20].

4 Future Work

A core feature of evolutionary algorithms is their mutation operator. Recently, much attention has been devoted to the study of mutation operators with dynamic and non-uniform mutation rates. Following up on this line of work, we propose a new mutation operator and analyze its performance on the (1+1) Evolutionary Algorithm (EA).

This mutation operator competes with pre-existing ones, when used by the $(\mu + 1)$-EA on classes of problems for which results on the other mutation operators are available. We present a "jump" function for which the performance of the $(\mu + 1)$-EA using any static uniform mutation and any restart strategy can be worse than the performance of the $(\mu + 1)$-EA using our mutation operator with no restarts. We show that the $(\mu + 1)$-EA using our mutation operator finds a $(1/3)$-approximation ratio on any non-negative submodular function in polynomial time. This performance matches that of combinatorial local search algorithms specifically designed to solve this problem.

References

[1] A. A. Bian, J. M. Buhmann, A. Krause, and S. Tschiatschek. "Guarantees for Greedy Maximization of Non-submodular Functions with Applications". In: *Proc. of ICML*. 2017, pages 498–507.

[2] N. Buchbinder, M. Feldman, and M. Garg. "Deterministic $(1/2 + \epsilon)$-Approximation for Submodular Maximization over a Matroid". In: *CoRR* abs/1807.05532 (2018).

[3] G. Călinescu, C. Chekuri, M. Pál, and J. Vondrák. "Maximizing a Monotone Submodular Function Subject to a Matroid Constraint". In: *SIAM Journal of Computing* 40.6 (2011), pages 1740–1766.

[4] M. Conforti and G. Cornuéjols. "Submodular Set Functions, Matroids and the Greedy Algorithm: Tight Worst-case Bounds and Some Generalizations of the Rado-Edmonds Theorem". In: *Discrete Applied Mathematics* 7.3 (1984), pages 251–274.

[5] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser. "Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms". In: *Management Science* 23.8 (1977), pages 789–810.

[6] A. Das and D. Kempe. "Submodular meets Spectral: Greedy Algorithms for Subset Selection, Sparse Approximation and Dictionary Selection". In: *Proc. of ICML*. 2011, pages 1057–1064.

[7] X. Hu, R. Shonkwiler, and M. C. Spruill. *Random Restarts in Global Optimization*. Technical report 110592-015. School of Mathematics, Georgia Institute of Technology, 1994.

[8] S. Jegelka and J. Bilmes. "Submodularity Beyond Submodular Energies: Coupling Edges in Graph Cuts". In: *Proc. of CVPR*. 2011, pages 1897–1904.

[9] A. Krause, A. P. Singh, and C. Guestrin. "Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies". In: *Journal of Machine Learning Research* 9 (2008), pages 235–284.

[10] N. D. Lawrence, M. W. Seeger, and R. Herbrich. "Fast Sparse Gaussian Process Methods: The Informative Vector Machine". In: *Proc. of NIPS*. 2002, pages 609–616.

[11] J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko. "Non-monotone Submodular Maximization under Matroid and Knapsack Constraints". In: *Proc. of STOC*. 2009, pages 323–332.

[12] H. Lin and J. Bilmes. "Multi-document Summarization via Budgeted Maximization of Submodular Functions". In: *Proc. of HLT*. 2010, pages 912–920.

[13] A. Lissovoi, D. Sudholt, M. Wagner, and C. Zarges. "Theoretical Results on Bet-and-Run as an Initialisation Strategy". In: *Proc' of GECCO*. 2017, pages 857–864. doi: 10.1145/3071178.3071329.

[14] H. R. Lourenço, O. C. Martin, and T. Stützle. "Iterated Local Search". In: *Handbook of Metaheuristics*. 2003, pages 320–353.

[15] M. Luby, A. Sinclair, and D. Zuckerman. "Optimal Speedup of Las Vegas Algorithms". In: *Information Processing Letters* 47.4 (1993), pages 173–180.

[16] M. Magdon-Ismail and A. F. Atiya. "The Early Restart Algorithm". In: *Neural Computation* 12.6 (2000), pages 1303–1312.

[17] G. L. Nemhauser and L. A. Wolsey. "Best Algorithms for Approximating the Maximum of a Submodular Set Function". In: *Mathematics of Operations Research* 3.3 (1978), pages 177–188.

[18] A. de Perthuis de Laillevault, B. Doerr, and C. Doerr. "Money for Nothing: Speeding Up Evolutionary Algorithms Through Better Initialization". In: *Proc' of GECCO*. 2015, pages 815–822.

[19] P. Sebastiani and H. P. Wynn. "Maximum Entropy Sampling and Optimal Bayesian Experimental Design". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 62.1 (2002), pages 145–157.

[20] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser. "Efficient Informative Sensing using Multiple Robots". In: *Journal of Artificial Intelligence Research* 34 (2009), pages 707–755.

[21] J. Vondrák. "Submodularity and Curvature: The Optimal Algorithm". In: *RIMS Kokyuroku Bessatsu* B23 (2010), pages 253–266.

A Comparison of Implementation Techniques for Implicit Layer Activation

Stefan Ramson

Software Architecture Group
Hasso-Plattner-Institut
Stefan.Ramson@hpi.uni-potsdam.de

Context-oriented programming (cop) directly addresses context variability by providing dedicated language concepts: *layers*, units of modularity, store context-dependent behavior. During runtime, layers can be applied dynamically depending on the current context of the program.

Various activation means for layers have been proposed. Most of them require developers to model context switches explicitly. In contrast, implicit layer activation (ila) allows developers to bind the activation status of a layer to a boolean predicate. The associated layer stays automatically active as long as the given predicate evaluates to true.

Despite its declarative semantics, ila is usually implemented in an imperative manner. In context of our research, we investigated the applicability of reactive programming for an implementation of ila. Thus, we present a reactive implementation variant of ila in ContextJS alongside an imperative one. Furthermore, we discuss their trade-offs regarding code complexity as well as runtime overhead.

1 Introduction

Context variability is an inherent property to most modern software systems. However, wide-spread programming languages do not support concepts for context variability in a principled way. The context-oriented programming (cop) paradigm [3] directly addresses context variability by providing dedicated language concepts to describe contexts and context-dependent behavior. *Layers* allow to store all behavior related to a specific context in a single unit of modularity.

To apply context-dependent behavior, layers can be activated dynamically through various activation means. Most activation means require developers to model context switches in an explicit manner [5]. For example, using global layer activation, a layer becomes (de-)activated at a certain point in imperative control flow, usually guarded by a condition. Similarly, dynamic layer activation allows to explicitly activate a layer for the extent of a message send. Even integrations with event-based concepts, such as event transitions, require to emit the respective events explicitly.

In contrast to most activation means, implicit layer activation (ila) [8] provides a mechanism to declaratively define contexts. By describing the extent of a context rather than context switches, ila relieves the programmer from the task of manually describing the boundaries of contexts. While ila offers promising properties, only few cop implementations support ila [2, 4, 5, 8]. Despite its declarative semantics, ila is usually implemented in an imperative fashion, similar to other activation means.

In particular, the current layer composition stack is determined *imperatively* when calling a layered method. At this very point in time, the cop framework checks the conditions of all implicitly activated layers. However, this non-reactive approach is only possible, because most cop implementations are limited to adapting object and class methods [9]. These concepts are *passive* entities that only affect the program behavior when called explicitly. Therefore, the limitation to passive entities enables the cop framework to check the condition at a well-defined point in the program. In contrast, *active* entities, such as constraints, may initiate behavior by themselves without being called explicitly. Extending the concept of cop beyond method decoration requires to activate scoped entities at specific times.

Using a *reactive implementation*, a cop framework can *eagerly enable* implicitly activated layers and, thus, deal with active entities properly. Thus, a reactive implementation might pave the way to apply the concept of cop to other types of abstraction beyond partial methods. For example, a layer could be used to limit the scope of a constraint: when a condition becomes true, the corresponding layer becomes active and the constraint immediately takes effect [6], instead of waiting for an additional, explicit trigger.

Similar to active entities, *life-cycle callbacks*, such as `onActivate` and `onDeactivate`, should be executed immediately when a layer becomes active or inactive, respectively. As an example, an `onActivate` callback might set up some state required by the layer while the `onDeactivate` callback cleans up this additional state. An imperative implementation might lead to unintended behavior as it delays the execution of the callbacks unnecessarily. Instead, the life-cycle callbacks expect to be executed eagerly. Again, integrating ila properly with reactive concepts, such as life-cycle callbacks, requires a reactive implementation for ila itself.

Both examples above highlight advantages of a reactive implementation for ila. However, imperative implementations are more prevalent. Thus, we examine the different approaches to implement ila. In particular, we extend ContextJS [7] with implicit layer activation (ila) in two variants: an imperative implementation based on *an extended dispatch* and a reactive implementation based on *Active Expressions* [10]. Furthermore, we compare the two implementations regarding *code complexity* and *runtime overhead*.

2 Background

This section presents relevant prior work. In particular, we discuss the concepts of cop and ila as well as Active Expressions as a means to implement ila using reactive programming concepts.

2.1 Context-oriented Programming

Context-oriented programming (cop) is a programming paradigm dedicated to directly express context variability. To do so, cop allows developers to extend system behavior through heterogeneous adaptations given as partial method definitions:

```
1 fetch(url) {
2   console.log('fetch ' + url);
3   return proceed(url);
4 }
```

Partial method definitions may extent or base override the behavior of class or object methods. In the example above, the partial method `fetch` prints out the given url before continuing with the base behavior in line 3. One may combine multiple partial method definitions into a unit of modularity, called a *layer*:

```
1 const networkTracer = new Layer().refineObject(Networking, {
2   fetch(url) {
3     console.log('fetch ' + url);
4     return proceed(url);
5   }
6 };
```

A layer represents a set of behavior adaptations to be active while the system is in a specific context. To signalize that a certain context is present, the developer can activate a layer using one of various activation means:

```
1 Networking.fetch('example.com'); // prints nothing
2 networkTracer.beGlobal();
3 Networking.fetch('example.com'); // prints 'fetch example.com'
```

In the example above, line 2 activates the layer using global layer activation. Once a layer is active, its partial methods are applied to the program behavior. Thus, the resource request in line 3 is traced. In case of global layer activation, this behavior adaptations last indefinitely.

2.2 Implicit Layer Activation

Ila [8] is another activation means for cop layers. In contrast to global layer activation, ila has declarative semantics. To be precise, developers can associate a layer to an arbitrary object-oriented (oo) expression:

```
layer.activeWhile(expression)
```

By associating a layer with an expression, the layer is not activated or deactivated at a fixed time. Instead, the layer is active as long as the given condition holds, as depicted in the following example:

```
1 var shouldTrace = false;
2 networkTracer.activeWhile(() => shouldTrace);
3
4 Networking.fetch('example.com'); // prints nothing
5 shouldTrace = true;
6 Networking.fetch('example.com'); // prints 'fetch example.com'
```

Despite these declarative semantics, ila is typically implemented in an imperative manner. An underlying system keeps track of all layered methods, such as the `fetch` method. Then, when calling a layered method, the current layer composition stack is determined [5]. At this very point in time, the cop framework checks the conditions of all implicitly activated layers [1, 9]. If the condition evaluates to true, the method adaptation is taken into account for this method call, as the modified behavior in line 6 illustrates.

Considering the declarative semantics of ila, a reactive implementation does not seem a stretch [6]: an underlying reactive framework may monitor variables referenced by the given condition. When such a variable changes, the condition is re-evaluated and the corresponding layer is activated or deactivated accordingly.

2.3 Active Expressions

Active Expressions [10] is a basic reactive programming concept designed to aid language designers when implementing reactive programming concepts in oo environments. In particular, Active Expressions relieve developers from the tedious task of change detection by hiding implementation details behind a unified abstraction: oo *expressions*. Developer provide the expression to be monitored to the reactive framework by calling the `aepxr` function:

```
aexpr(expression).onChange(callback)
```

The specified callback gets executed whenever the evaluation result of the expression changes.

Using Active Expressions, one can significantly reduce the implementation effort when creating new reactive programming concepts. As ila fits well into the working principle of Active Expressions, we use Active Expressions in order to simplify the reactive implementation of ila. The provided expression may contain any oo mechanism, such as information hiding and polymorphism. As a consequence, the resulting ila implementation integrates well with oo environments.

3 Implementing Implicit Layer Activation in ContextJS

Currently, ContextJS [7] does not support implicit layer activation (ila) as an activation means.[1] However, due to the reactive and declarative nature of the web, ContextJS might benefit from this activation means. Thus, we show how to extend ContextJS with ila, first using an imperative implementation, then using a reactive implementation with Active Expressions.

3.1 Imperative Implementation

ContextJS supports multiple activation means, including global activation and dynamic activation for the extent of a function call. When calling a layered method, the `currentLayers` function is responsible for computing an appropriate layer composition, either by using a cached result or, if necessary, by determining a new one using global and dynamic layers:

```
1 export function currentLayers() {
2   // parts omitted for readability
```

[1] https://github.com/LivelyKernel/ContextJS (last accessed 2018-09-03, at commit 938e117; npm package: contextjs in version 2.0.0).

```
3   if (!current.composition) {
4     current.composition = composeLayers(LayerStack);
5   }
6   return current.composition;
7 }
```

For our extension,[2] we add a separate list of layers to represent layers potentially activated through ila, called `implicitLayers`. To implicitly activate a layer, we add the method `activeWhile` to the class `Layer`:

```
1 activeWhile(condition) {
2   if (!implicitLayers.includes(this)) {
3     implicitLayers.push(this);
4   }
5   this.implicitlyActivated = condition;
6   return this;
7 }
```

This method has two responsibilities. First, in line 3, it adds the layer to the list of implicitly activated layers, if necessary. Second, it stores the provided argument `condition`, a boolean function to specify whether the layer should be active, as seen in line 5. Using the list of implicitly activated layers, we can get all layers that are actually activated by filtering this list for layers with their conditions evaluating to true, as done by the `getActiveImplicitLayers` function:

```
1 function getActiveImplicitLayers() {
2   return implicitLayers.filter(layer => layer.implicitlyActivated());
3 }
```

Using the `getActiveImplicitLayers` function, we can now adjust the computation of the current layer composition in `currentLayers`:

```
1 export function currentLayers() {
2   // part omitted for readability
3   var current = LayerStack[LayerStack.length - 1];
4   if (!current.composition) {
5     current.composition = composeLayers(LayerStack);
6   }
7   return current.composition.concat(getActiveImplicitLayers());
8 }
```

To include implicitly activated layers, we append all layers activated through ila to the already computed layer composition. As a result, the returned layer composition contains dynamically activated, globally activated, and implicitly activated layers. Note that we cannot cache implicitly activated layer, because we have no means to invalidate the cache on changes to the condition.

3.2 Reactive Implementation

In contrast to the imperative implementation, we do not introduce a separate data structure for implicitly activated layers. Instead, we treat implicitly activated layers as being globally active as long as their condition evaluates to true. As a result, we

[2] https://github.com/active-expressions/programming-contextjs-plain (last accessed 2018-09-03, at commit 22deb54).

can reuse the existing layer composition algorithm once a layer becomes active. To do so, the `activeWhile` method has to setup dependencies to detect changes to the given condition and update the layer accordingly. For this implementation,[3] we wrap the given `condition` in an Active Expression. Using this Active Expression, we can easily implement the appropriate reactive behavior:

```
1 activeWhile(condition) {
2   aexpr(condition)
3     .onBecomeTrue(() => this.beGlobal())
4     .onBecomeFalse(() => this.beNotGlobal());
5   return this;
6 }
```

Using the `onBecomeTrue` (line 3) and `onBecomeFalse` (line 4) methods, the layer is eagerly activated or deactivated whenever the expression result becomes true or false, respectively. Thus, the layer is automatically taken into account as a globally activated layer by the existing layer composition algorithm. Additionally, those methods automatically adjust the initial state of the layer depending on the current result of the given expression.

4 Implementation Complexity

We quantitatively compare both implementations of ila in terms of code complexity. As a measurement for code complexity, we count the total number of Abstract Syntax Tree (ast) nodes in each implementation. We summarize our measurements in Table 1. According to Table 1, the reactive implementation based on Active Expressions has a lower complexity compared to the imperative implementation.

However, more important than these quantitative results is the way both implementations introduce the concept of ila to the ContextJS library. The imperative implementation introduces an additional layer type: implicitly activated layers. Furthermore, the imperative implementation requires knowledge about the underlying dispatch mechanism in order to extend the layer composition algorithm to take implicitly activated layers in account. In contrast, the reactive variant reuses the existing semantics by only modifying layers through already exposed methods. Thus, the reactive variant only requires knowledge about the usage of ContextJS, not about its internal working principles.

5 Performance Evaluation

To identify the performance penalties implied by the different implementation variants described in section 3, we provide and discuss multiple micro benchmark scenarios in the following. For each scenario, we compare the imperative implementation,

[3] https://github.com/active-expressions/programming-contextjs-aexpr (last accessed 2018-09-03, at commit 07437e8).

Table 1: Code complexity of the presented implementations compared to the existing ContextJS version in terms of the number of ast nodes

ast nodes (sloc)	Complete	Difference to ContextJS
Unmodified ContextJS	2485 (568)	
Imperative Implementation	2557 (580)	72 (12)
Reactive Implementation	2525 (575)	40 (7)

described in section 3.1, with the Active Expression-based implementation, described in section 3.2. Active Expressions allow to choose between multiple underlying implementation strategies. Thus, we compare against two strategies. First, the *interpretation* strategy uses dynamic interpretation to punctually insert property accessors into the system space. Second, the *compilation* strategy performs a heavy-weight source code transformation to notify about changes in the system state.

In section 5.1, we discuss our benchmark setup, test suite, and statistical methods.

5.1 Performance Benchmark Setup and Statistical Methods

All benchmarks were executed on the following system:

- CPU and memory: Intel(R) Core(TM) i7-6650U CPU @ 2.20GHz 2.21 GHz, 4 Logical cores; 16.0 GB Main Memory,
- System software: Windows 10 Pro (OS Build 15063),
- Runtime: Google Chrome version 57.0.2987.133; benchmarks executed using Karma test runner version 1.2.0 and Mocha test framework version 3.0.2,
- Transpiler and bundler: babel-cli 6.11.4 (no es2015 preset) and rollup 0.34.8,
- Libraries under test:
  - *programming-contextjs-plain* at commit 1360e1a[4]
  - *programming-contextjs-aexpr* at commit 07437e8[5]
- Benchmark suite: *aexpr-ila-benchmark* at commit 6a6395b.[6]

We measured the execution time of a benchmark by wrapping the benchmark in a function and measuring the time between calling the function and it returning. Each benchmark configuration was iterated 100 times, with only the final 30 iterations

[4] https://github.com/active-expressions/programming-contextjs-plain (last accessed 2018-09-03).
[5] https://github.com/active-expressions/programming-contextjs-aexpr (last accessed 2018-09-03).
[6] https://github.com/active-expressions/aexpr-ila-benchmark (last accessed 2018-09-03).

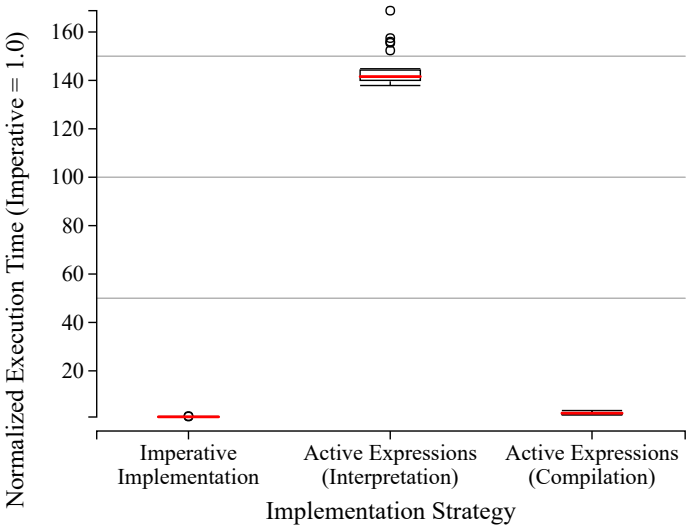


Figure 1: Execution times for declaratively associating 10,000 layers with a context using ila. The thick red line indicates the median, the upper and lower edges of the box are the second and third quartile and the end of the whiskers are the most outlying values in a 1.5 inter-quartile range distance from the second and third quartile.

taken into account for the overall performance measurement to mitigate the effects of the V8 just-in-time (jit).

Statistical Methods We make no assumptions on the underlying distribution and provide Tukey boxplots in Figure 1 to 3 to visualize the median and variation of the measured timings. Exact median timings are given in Table 2 to 4. Slowdowns are computed by dividing the median execution times of the measurements to compare. Confidence bounds of this statistic are given by the 2.5-th and 97.5-th percentile of the bootstrap distribution of the computed ratio.

5.2 Overhead for Initial Association

First off, we analyze the initial cost to create associations between a layer and a declarative context. For this purpose, we measure the time to associate 10,000 layers with the same context expression. As a running example, we use the following expression for all benchmarks:

```
() => context.enabled()
```

The variable `context` references an instance of the class `Context` with a single Boolean property representing whether the current is active. The `enabled` method provides access to this status. Thus, we execute `layer.activeWhile(() => context.enabled())` 10,000 times.

Table 2: Benchmark timings and relative slowdowns for declaratively associating 10,000 layers with a context using ila. Slowdowns given as ratio of medians with 95 % confidence intervals.

	timing [ms]	slowdown (vs Imperative)
Imperative	28.09	
Reactive (Interpretation)	3976.06	141.57 [138.74 - 144.77]
Reactive (Compilation)	70.29	2.50 [2.14 - 2.64]

Discussion As Figure 1 reveals, the imperative implementation has the lowest runtime for creating associations to declarative contexts. This result is to be expected, as the imperative implementation only adds the given layer to a global array when creating the association. In contrast, both reactive strategies have to set up their respective dependency mechanisms in order to monitor for changes. Accordingly, they impose high overhead as shown by the relative slowdowns in Table 2. While the compilation strategy runs the given expression in native JavaScript, the interpretation strategy uses a full-fledged JavaScript-in-JavaScript interpreter to determine relevant dependencies, which explains the very high impact of the interpretation strategy. The relative overhead compared to the imperative implementation is subject to the complexity of the given expression.

5.3 Frequent Context Switches

The above measurements show only the initial overhead of the respective implementation strategy. In the following, we identify the overhead that is imposed by implicit context switches. Continuing the previous scenario, we enable and disable a context object implicitly associated with a layer by ila. We disable and re-enable this context 500 times each. Then, we test the expected semantics by calling context-dependent behavior:

```
expect(adaptee.call()).to.equal(expected);
```

Thus, 1,000 context switches occur before using context-dependent behavior once. We measure the time it takes to repeat this process 100 times.

Discussion According to the results in Figure 2, both reactive implementations impose a very high overhead for frequent context switches. As highlighted in Table 3, the interpretation strategy is over two orders of magnitude slower and the compilation strategy is over three orders of magnitude slower. This high overhead is to be expected, because the imperative implementation does not invoke any additional behavior when switching contexts implicitly. In contrast, both reactive strategies activate the respective layer globally, thus, invalidating the current layer composition. The compilation strategy has a higher overhead than the interpretation strategy, because the applied source code transformation affects all computations. In con-

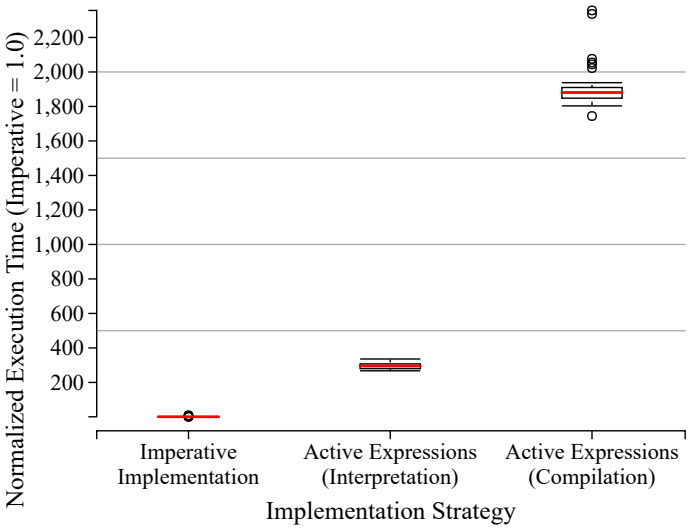


Figure 2: Performance benchmark results for a high ratio of context switches to invocations of context-dependent behavior. The exact ratio is 1000 to 1. All results are normalized by the imperative median. The normalization is the quotient of execution time of the respective implementation and the time of the imperative solution.

Table 3: Benchmark timings and relative slowdowns for frequently switching contexts

	timing [ms]	slowdown (vs Imperative)
Imperative	0.42	
Reactive (Interpretation)	125.63	295.61 [284.28 - 305.44]
Reactive (Compilation)	799.34	1880.81 [1851.14 - 1924.57]

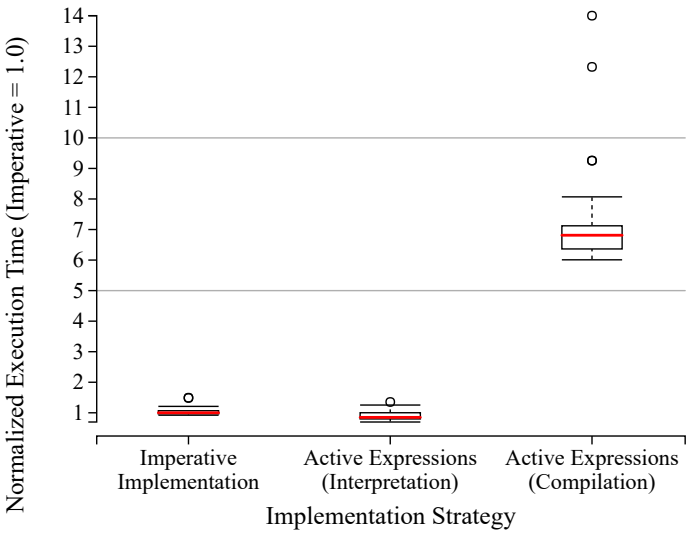


Figure 3: Performance benchmark results for calling context-dependent behavior 1,000 times before switching contexts. All results are normalized by the median of the imperative implementation.

Table 4: Benchmark timings and relative slowdowns for frequently invoking context-dependent behavior

	timing [ms]	slowdown (vs Imperative)
Imperative	17.63	
Reactive (Interpretation)	14.98	0.85 [0.80 - 0.96]
Reactive (Compilation)	120.12	6.81 [6.39 - 7.08]

trast, the interpretation strategy uses property accessors to punctually intercept the program execution.

5.4 Frequent Message Sends

The previous experiment hints a high overhead for reactive implementations when switching contexts frequently. In the following benchmark, we examine the overhead introduced by each implementation strategy when frequently calling context-dependent behavior. In particular, we switch context ten times, with invoking context-dependent behavior 1000 times after each context switch.

Discussion As Figure 3 reveals, both, the imperative and the interpretation-based implementation have similar performance for frequent invocation of context-dependent behavior. According to Table 4, the reactive variant using the interpretation strategy is slightly faster than the imperative implementation. The reason is that the

reactive implementation utilizes the caching mechanism of ContextJS. When calling context-dependent behavior subsequently, the dispatch mechanism checks whether the layer composition became invalid since the last dispatch. If not, ContextJS can reuse the existing layer composition. Because the reactive implementations update layers on change, no additional checks are required. In contrast, the imperative implementation has to check the current status of each implicitly activated layer on dispatch. Because the imperative implementation cannot anticipate context switches, the layer composition needs to be recomputed for each method invocation.

Even with this conceptual advantage, the compilation strategy is considerably slower than the imperative implementation. This result highlights the high performance overhead imposed by the source code transformation. The reason for this high overhead is that the invocation of detection hooks, such as access to object members, is *highly polymorphic*, and, therefore hard to optimize by jits. As every access to a property and every call of a member function is wrapped, this strategy can cause severe performance penalties.

⁂

The presented benchmark highlights the potential provided by reactive implementations of ila. In particular, systems with long living layers and frequent invocations of context-dependent behavior might benefit from such an implementation. However, the benchmarks also indicate potential performance problems. In particular, the production of jit-unfriendly code represents a major source of performance issues. Further studies on the effect of the different implementation variants are needed, especially with regard to real-world cop applications.

6 Conclusion and Future Work

We implemented implicit layer activation (ila) in two different variants: an imperative and a reactive one. Our comparison shows that the reactive implementation matches the declarative semantics of ila more closely. The runtime overhead of the two implementations highly depends on the specific usage scenario: the imperative implementation is suitable for a system with frequent context switches, while the reactive implementation is more suitable for systems with frequent invocations of context-dependent behavior. A reactive implementation seems viable and offers interesting possibilities: the eager (de-)activation of layers allows for the integration of ila with layer life-cycle callbacks and active entities, for example by scoping the effect of constraints [6].

The work reported here represents an application of our main research project, Active Expressions. We expect to continue to evolve both the concept of Active Expressions itself and its application on various problems. Thus, there are a number of directions for future work:

Bridging the Gap Between Object-oriented and Reactive Programming Besides ila, Active Expressions may act as a foundation of many other reactive programming mechanisms. A key design rationale to Active Expressions is to enable a seamless integration of those reactive mechanisms with the oo host language. Due to their basic nature, Active Expressions might represent the smallest possible increment on the object-oriented programming (oop) paradigm to integrate reactivity on a fundamental level. In this future work, we aim to identify whether Active Expressions are suitable for the role of integrating object-oriented and reactive programming, thereby, improving the day-to-day experience of software developers.

Transactional Integrity on Information Hiding Boundaries In contrast to many other reactive programming systems, Active Expression automatically infer dependencies from the given state description. Thereby, when encountering object methods, the underlying framework pierces these encapsulation boundaries to identify relevant state dependencies. When calling high-level state-altering methods, multiple state modifications might be required to carry out the intended behavior. In this case, Active Expressions may leak implementation details to users, thereby breaking encapsulation. Some reactive programming mechanisms do not desire this behavior. To improve the applicability of Active Expressions on such reactive programming mechanisms, we want to introduce optional transaction boundaries on units of encapsulation.

References

[1] M. Appeltauer, R. Hirschfeld, M. Haupt, J. Lincke, and M. Perscheid. "A Comparison of Context-oriented Programming Languages". In: *International Workshop on Context-Oriented Programming (COP)*. 2009, 6:1–6:6. doi: 10.1145/1562112.1562118.

[2] E. Bainomugisha, J. Vallejos, C. D. Roover, A. L. Carreton, and W. D. Meuter. "Interruptible Context-dependent Executions: A Fresh Look at Programming Context-aware Applications". In: *Symposium on New Ideas in Programming and Reflections on Software (Onward!), 2012*. 2012, pages 67–84. doi: 10.1145/2384592.2384600.

[3] R. Hirschfeld, P. Costanza, and O. Nierstrasz. "Context-oriented Programming". In: *Journal of Object Technology (JOT)* 7.3 (2008), pages 125–151. issn: 1660-1769.

[4] H. Inoue and A. Igarashi. "A Library-based Approach to Context-dependent Computation With Reactive Values: Suppressing Reactions of Context-dependent Functions Using Dynamic Binding". In: *15th International Conference on Modularity (MODULARITY)*. 2016, pages 50–54. doi: 10.1145/2892664.2892669.

[5] T. Kamina, T. Aotani, and H. Masuhara. "Generalized Layer Activation Mechanism Through Contexts and Subscribers". In: *14th International Conference on Modularity (MODULARITY), 2015*. 2015, pages 14–28. doi: 10.1145/2724525.2724570.

[6] S. Lehmann, T. Felgentreff, and R. Hirschfeld. "Connecting Object Constraints with Context-oriented Programming: Scoping Constraints with Layers and Activating Layers with Constraints". In: *7th International Workshop on Context-Oriented Programming (COP)*. 2015, 1:1–1:6. doi: 10.1145/2786545.2786549.

[7] J. Lincke, M. Appeltauer, B. Steinert, and R. Hirschfeld. "An Open Implementation for Context-oriented Layer Composition in ContextJS". In: *Science of Computer Programming (SCICO)* 76.12 (2011), pages 1194–1209. doi: 10.1016/j.scico.2010.11.013.

[8] M. von Löwis, M. Denker, and O. Nierstrasz. "Context-oriented Programming: Beyond Layers". In: *International Conference on Dynamic Languages (ICDL), 2007*. 2007, pages 143–156. doi: 10.1145/1352678.1352688.

[9] K. Mens, R. Capilla, N. Cardozo, and B. Dumas. "A Taxonomy of Context-aware Software Variability Approaches". In: *Workshop on Live Adaptation of Software SYstems (LASSY)*. 2016, pages 119–124. doi: 10.1145/2892664.2892684.

[10] S. Ramson and R. Hirschfeld. "Active Expressions: Basic Building Blocks for Reactive Programming". In: *Journal on The Art, Science, and Engineering of Programming* 1.2 (2017).

Deep Learning from Unbalanced Medical Imaging

Mina Rezaei

Internet Technologies and Systems
Hasso-Plattner-Institut
Mina.Rezaei@hpi.de

Unbalanced data is one of the major challenges in medical image segmentation, where the number of pixels belonging to a desired object, the organ or the tumors, are significantly lower than those belonging to the background. A model trained with imbalanced data tends to bias towards majority class distribution, which is not desired in clinical applications. We propose different learning algorithms that mitigates the challenge raised by unbalanced data using biased complementary labels, biased with synthetic minority classes, and biased with mini-batch normalization in data-level. We approach ensemble generative model, deep mutual learning, and adversarial weighted loss towards minority classes. We show evidence that the proposed frameworks is applicable to different types of medical images of varied sizes on different applications of clinical routine tasks such as diseases diagnosis, abnormal tissues localization, and semantic segmentation.

1 Overview

Medical imaging plays an important role in disease diagnosis, treatment planning, and clinical monitoring. However, one of the major challenges in medical image analysis is imbalanced data where there exists a majority class with normal or healthy data and a minority class with abnormal or non-healthy data. A model trained with imbalanced data tends to bias towards healthy data which is not desired in clinical applications and generally predicted outputs by these networks have high positive predictive value and low sensitivities.

There are methods in medical image analysis against class imbalance issue by cascade training [17], training with cost-sensitive function [11], equal selection of training samples [13], and samples re-weighting [5, 21] which directly adjusts the sample sizes of respective classes.

In this paper, we overview our recent approaches for handling imbalanced medical data in data-level and algorithmic-level using generative adversarial networks (GANs). At the data-level, the objective is to balance the class distribution through biased complementary labels [23], biased with synthetic minority classes [24],and biased with mini-batch normalization [27]. Algorithm-level based solutions address class imbalanced problems by modifying the learning algorithm to alleviate the bias towards majority class. Examples are the ensemble generative model [25, 26], deep mutual learning [27], and adversarial weighted loss towards minority classes [28].

2 Background

In a conventional generative adversarial network, generative model G tries to learn a mapping from random noise vector z to output image y; $G : z \rightarrow y$. Meanwhile, a discriminative model D estimates the probability of a sample coming from the training data x_{real} rather than the generator x_{fake}. The GAN objective function is a two-player mini-max game like Eq.(1).

$$\begin{aligned} \min_G \max_D V(D,G) = E_y[logD(y)]+ \\ E_{x,z}[log(1-D(G(x,z)))] \end{aligned} \tag{1}$$

In a conditional GAN, a generative model learns the mapping from the observed image x and a random vector z to the output image y; $G : x, z \rightarrow y$. Discriminative model on the other hand attempts to discriminate between generator output and ground truth of the training set Eq.(2).

$$\begin{aligned} \mathcal{L}_{adv} \leftarrow \min_G \max_D V(G,D) = E_{x,y}[logD(x,y)]+ \\ E_{x,z}[log(1-D(x,G(x,z)))] \end{aligned} \tag{2}$$

3 Methods for Handling Imbalanced Medical Imaging

We propose different conditional GAN frameworks to mitigate imbalanced imaging. Similar conditional GAN [19]; in our proposed methods, a generative model learns mapping from a given sequence of 2D multimodal MR images x_i to a sequence semantic segmentation y_{i_seg} and classification y_{cls}; $G : \{x_i, z\} \rightarrow \{y_{i_seg}, y_{cls}\}$ (e.g. i refers to 2D slice index between 1 and 155 from a total 155 slices acquired from each patient). The training procedure for the segmentation task is similar to two-player mini-max game as shown in Eq.(3).

While the generative model generates segmentation pixel labels, the discriminator classifies whether the predicted pixel output by generator is similar to the ground truth annotated by a medical expert or synthetic. The adversarial loss is mixed with two additional loss to attenuate the imbalanced data impact.

$$\begin{aligned} \mathcal{L}_{adv} \leftarrow \min_G \max_D V(G,D) = E_{x,y_{seg}}[logD(x,y_{seg})]+ \\ E_{x,z}[log(1-D(x,G(x,z)))] \end{aligned} \tag{3}$$

3.1 Cost-sensitive Learning

A class-imbalance in a medical dataset where non-healthy classes could not be trained as well as healthy classes, might dominate the gradient direction. Regarding to mitigate class-imbalanced impact, we mixed adversarial loss Eq. (3) with selective

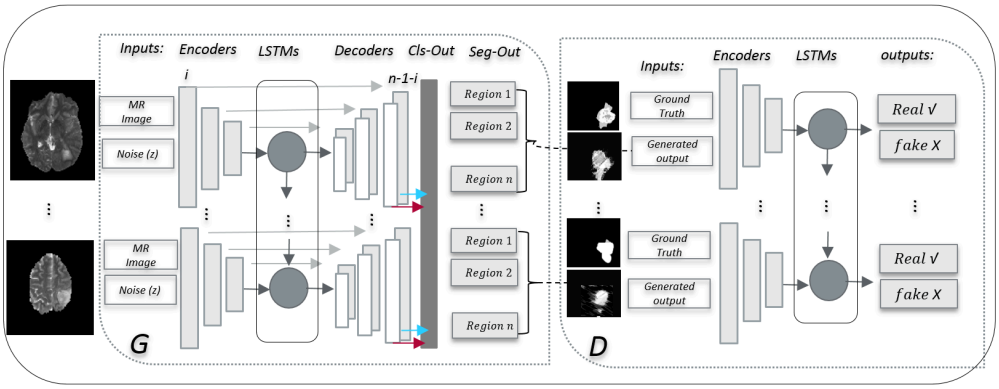


Figure 1: Our proposed architecture for learning semantic segmentation and diseases prediction. We design a set of auto-encoders combined with a LSTM unit in a circumvent bottleneck as the generator network with skip connections between each layer i and the corresponding layer n-1-i (mostly like UNet architecture). The discriminator is fully convolutional network substituted with LSTM unit. Both networks are trained together in an adversarial way with selective weighted categorical cross entropy loss for semantic segmentation and selective weighted L1 for diseases prediction.

weighted Eq.(4) categorical cross-entropy loss $\mathcal{L}_H(G)$ for semantic segmentation, and with selective weighted Eq.(4) $\ell 1$ loss Eq.(6) for classification of diseases.

$$w_c = \sqrt{\frac{\mid C_c \mid}{f_c + N}} \tag{4}$$

Where the weight for each class c is based on the ratio of the cordiality among N classes on entire training dataset (e.g. mostly healthy classes) by the frequency of samples with class c appears in the dataset. Since we have intense frequency differences, the square root is applied to prevent huge weights. This implies that larger classes in the training set have a weight smaller than 1 and the weights of the smallest classes are the highest defined by Eq.(4).

The final loss for semantic segmentation task is calculated through Eq.(5)

$$\mathcal{L}_{seg}(D, G) = \mathcal{L}_{adv}(D, G) + \mathcal{L}_H(G * w_c) \tag{5}$$

As shown in Fig.1, the concatenated depth features from last decoder layer with skip connection of encoder part passed into a couple of dense layers and map to the disease class. The objective function for class prediction is $\ell 1$ to minimize the absolute difference between the predicted value and the existing largest value Eq.(6)

$$\mathcal{L}_{cls}(G) = E_x \parallel y_{cls} - \sum_{i=1} G(x_i * w_c) \parallel \tag{6}$$

Where i indicates to 2D slice index from the same patient (e.g. in Brain dataset i is between 1 and 155 from a total of 155 slices acquired from each patient).

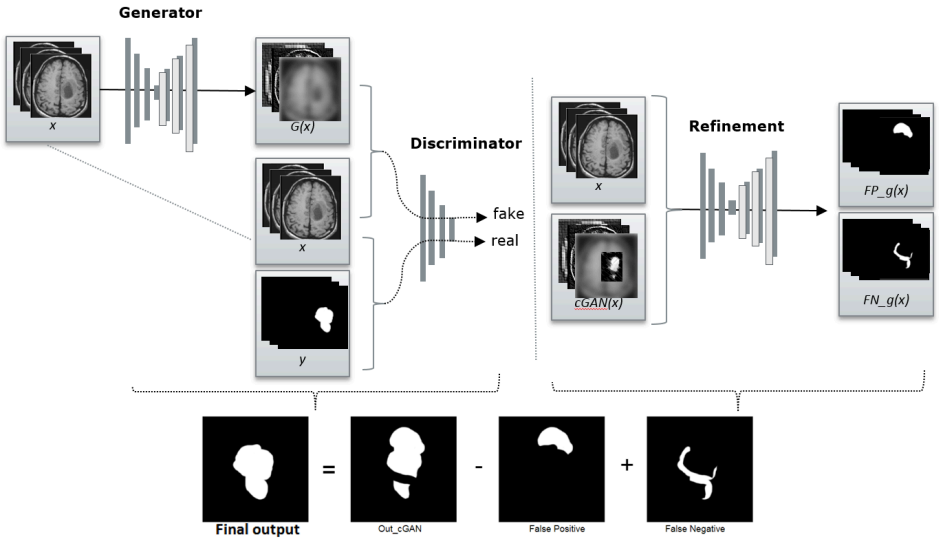


Figure 2: The proposed method for medical image semantic segmentation consists of a generator network, a discriminator network, and a refinement network. The generator tries to segment the image into pixel level, while the discriminator classifies the synthesized output as real or fake. The final semantic segmentation masks are computed through eliminating the false positives and adding the false negatives predicted masks by the refinement network.

In this work, similar to the work of Isola et al. [16], we used Gaussian noise z in the generator alongside the input data x. Without z, the network learns a mapping from x with a condition attached to the specific label y and would produce deterministic outputs, thus failing to match any distribution other than a delta function.

The final objective function for simultaneous semantic segmentation and classification is:

$$\mathcal{L} = \mathcal{L}_{seg}(D, G) + \mathcal{L}_{cls}(G) \tag{7}$$

3.2 Deep Ensemble Learning

To mitigate the problem of imbalanced data in medical image segmentation and achieve a much better trade-off between precision and recall, we proposed a conditional generative refinement network.Our proposed method consists of a generator and two discriminator where one discriminator feedback generator in true positive and true negative and another in false positive and false negative named refinement network.

The refinement network is trained to learn the false prediction of conditional GAN in details of false negatives Eq. (8) and false positives Eq. (9). The false negative error represents the number of pixels that were incorrectly labeled as background or wrong class (Fig. (2) third column). Similarly, the false positive indicates the number

of pixels that were incorrectly labeled as part of the region of interest (Fig. (2) last column).

$$\mathcal{L}_{fn} = clip((y - \mathcal{L}_{seg}), 0, 1) \tag{8}$$

$$\mathcal{L}_{fp} = clip((\mathcal{L}_{seg} - y), 0, 1) \tag{9}$$

where in both equations (8 and 9) y, $\mathcal{L}_{seg}$ respectively refers to the ground truth labels and predicted labels by adversarial loss.

Our final objective function $\mathcal{L}_{CR-GAN}$ for semantic segmentation relies on adding false negatives and subtracting false positives from outputs of adversarial network.

$$\mathcal{L}_{CR-GAN} = \mathcal{L}_{seg} - \mathcal{L}_{fp} + \mathcal{L}_{fn} \tag{10}$$

where $\mathcal{L}_{seg}$ indicates to predicted labels by adversarial loss.

3.3 Deep Mutual Learning

As shown in Figure 3, we propose generative multi-adversarial networks, forcing G to learn from ensemble of discriminators. This ultimately encourages G to produce samples from a variety of modes, since it now needs to fool the different possible discriminators that may remain in the ensemble. Variations in the ensemble are achieved by the feedback of each D with a certain probability p_d at the end of every batch. This means that G will only consider the loss of the remaining discriminators in the ensemble while updating its parameters at each iteration. Here, left side of Figure 3, the generator starts training from scratch by receiving feedback from different pre-trained and fine tune discriminators with different losses.

As depicted by Figure 3 in right side, we extend the generative networks equal to number of discriminative networks. Each generator communicate with correspond discriminator using adversarial loss. Moreover, the generator takes different mini-batches and discriminator are different in losses. The generator shared the mutual knowledge during training in way of using KL divergence, represented in Eq. (11)

$$L_{G_k} = L_{D_k} + \frac{1}{K-1} \sum_{l=1}^{k} D_{KL}(p_l \mid\mid p_k) \tag{11}$$

where K indicated the number of discriminators and generators, our mutual loss for each generator effectively takes the other $k-1$ networks in the cohort as $K-1$ discriminators to provide a learning experience. Note that we added the coefficient $1/k-1$ to make sure that the training is mainly directed by supervised learning of the true labels. The optimization with more than two networks is a straightforward extension. It can be distributed by learning each network on one device and passing the small probability vectors between devices.

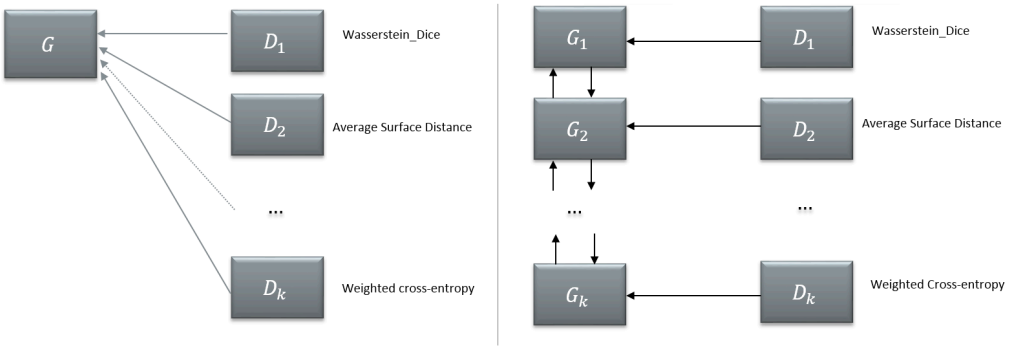


Figure 3: Our proposed architecture for learning semantic segmentation and diseases prediction. We design a set of auto-encoders combined with a LSTM unit in a circumvent bottleneck as the generator network with skip connections between each layer i and the corresponding layer n-1-i (mostly like UNet architecture). The discriminator is fully convolutional network substituted with LSTM unit. Both networks are trained together in an adversarial way with selective weighted categorical cross entropy loss for semantic segmentation and selective weighted L1 for diseases prediction.

3.4 Patient-wise mini-batch Normalization

Several popular techniques are developed for normalization, such as batch normalization [14], and max norm constraints [29], with the core idea of shifting the inputs to a zero mean and unit variance. The inputs are normalized before applying non-linearity to prevent the inputs from saturating extreme non-linearity. As described by Ioff *et al*. [14], batch normalization improve the overall optimization and gradient issues. In many cases, initial weights have a large deviance from true weights, delaying the convergence during training. Batch norm reduces the influence of weight deviance by normalizing the gradients this speeds up the training.

We initially normalized the inputs where the mean and variance are computed on a specific patient from the same acquisition plane (Saggital, Cronal, and Axial) and from all available image modalities (e.g., T1, T1-contrast, T2, Flair in the BraTS benchmark). In this regard, the deviances get increasingly large, and the back-propagation step needs to account for these large deviances which this restrict us for using a small learning rate to prevent gradient explosion.

For example, the mini-batch with 128 images includes the same patient images and four available modalities from the same acquisition plane. Algorithm. 1, shows how to compute normalization at each mini-batch by proposed patient-wise batch-norm technique.

3.5 Complementary Labels

Here, we mitigate the negative impact of the class imbalanced by inverse class frequency segmentation masks, that we name complementary segmentation labels.

Algorithm 1: Patient-wise mini-batch normalization. (i and n respectively refer to a number of 2D slices and number of patient e.g. $0 < i \leq 155$, n=230 in BraTS)

Input : Values of x over a mini-batch: $\beta = x_1, x_2, ..., x_{155}$
Parameters to be learned: γ, β
Output: $y_i = BN_{\gamma,\beta}(x_i)$

for $Patient : P_1, P_2, ..., P_n$ **do**
  for $AcquisitionPlane : x_i, y_i, z_i$ **do**
    for $Image \quad Modalities : T1, T2, T1c, Flair$ **do**
      $\mu_\beta \leftarrow \frac{1}{m}\sum_{i=1}^{n} x_i$
      $\sigma_\beta^2 \leftarrow \frac{1}{m}\sum_{i=1}^{n}(x_i - \mu_\beta)^2$
      $\hat{x_i} \leftarrow \frac{x_i - \mu_x}{\sqrt[2]{\sigma_x^2 + \varepsilon}}$
      $y_i \leftarrow \gamma\hat{x_i} + \beta = BN_{\gamma,\beta}(x_i)$
    end
  end
end

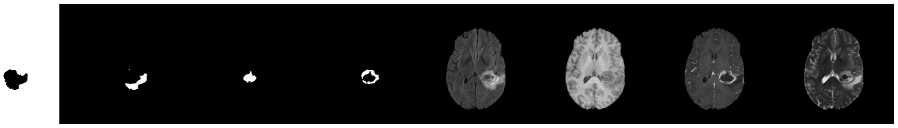


Figure 4: The brain MR image, from Brats 2018 after pre-processing. We extracted complementary mask from inverse of ground truth file annotated by medical expert, presented in the first column. Other binary masks extracted from ground truth file in columns 2–4 respectively are whole tumor, enhanced tumor, and core of tumor which they are used by the discriminator. The 5–8 columns are a slice of example 3D input of the segmentor.

Assume, Y is true segmentation label annotated by expert and $\bar{Y}$ is synthesize pair of corresponding images with complementary label where the $P(\bar{Y} = i \mid Y = j), i \neq j \in \{0, 1, ..., c-1\}$, and c is a number of semantic segmentation class labels. The complementary label $\bar{Y}$ is a negative label for the major class and a positive label for the $c-1$ class. Then, our network train with both true segmentation mask Y and complementary segmentation mask $\bar{Y}$ at the same time.

4 Experiments

To evaluate the performance of our network on imbalanced data segmentation and compared it with state-of-the-art methods, we trained recent popular annotated

medical imaging benchmarks for more detail please refer to [26]. Here we show an example on BraTS 2017 as described in Section (4.1).

4.1 Dataset and Pre-processing

The first experiment is carried out on real patient data obtained from BraTS2017 challenge [6, 7, 8, 18]. The BraTS2017 released data in three subsets train, validation, and test comprising 289, 47, and 147 MR images respectively in four multisite modalities of T1, T2, T1ce, and Flair which the annotated file provided only for the training set. The challenge is semantic segmentation of complex and heterogeneously located of tumour(s) on highly imbalanced data. Pre-processing is an important step to bring all subjects in similar distributions, we applied z-score normalization on four modalities with computing the mean and stdev of the brain intensities. We also applied bias field correction introduced by Nyúl et al. [20].

Additionally, we provided data augmentation such as randomly cropped, resizing, scaling, rotation between -10 and 10 degree, and Gaussian noise applied on training and testing time for three datasets.

4.2 Implementation

Configuration: Each proposed method is implemented separately based on a Keras library [10] with backend Tensorflow [1] and our code is publicly available.[1] We did different experiments on 2D, 2D sequences, and 3D images which all codes are publicly available in.[2]

4.3 Evaluation Results and Discussion

The segmentation of the brain tumour from medical images is highly interesting in surgical planning and treatment monitoring. The goal of segmentation as described by organizer [6, 7, 8, 18] is to delineate different tumour structures such as active tumorous core (TC), enhanced tumorous (ET), and edema or whole tumorous (WT) region.

Fig. (5) shows qualitative results of the cGAN network, and refinement network in detail. Based on Fig. (5), the result shows good relation to the ground truth for the segmentation after refinement network.

The Dice score, Hausdorff distance, sensitivity, and specificity are introduced by BraTS2017 as evaluation criteria for segmentation task. Tables (1, 2) present the brain segmentation results from proposed architecture and compare them with other related methods based on the pre-proceeding report [9].

From Table (1), the cGAN network (in second line) with one generator and discriminator achieved 12% less accuracy for whole tumour region segmentation com-

[1] https://github.com/HPI-DeepLearning/voxelGAN (last accessed 2018-10-18).
[2] https://github.com/HPI-DeepLearning/ (last accessed 2018-10-18).

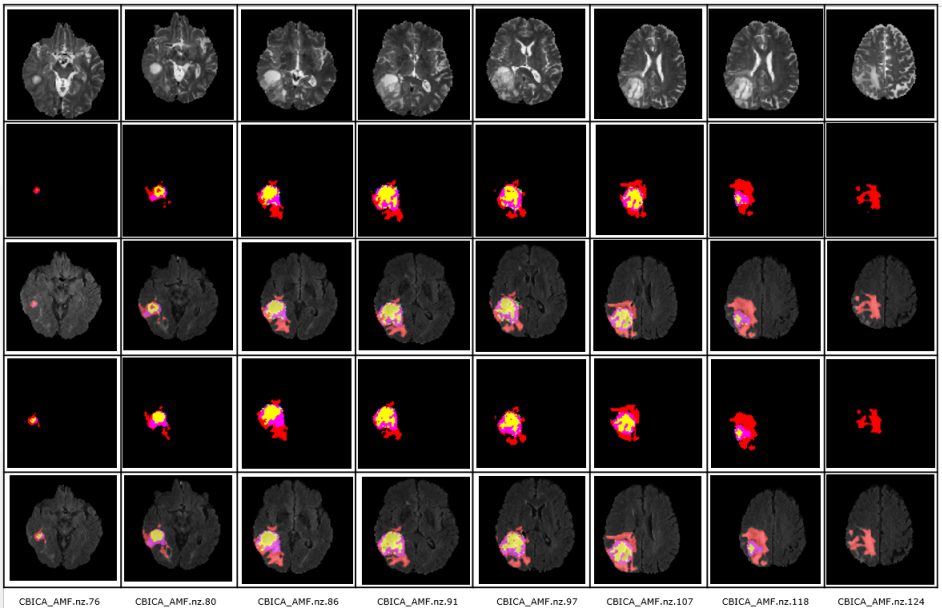


Figure 5: Visual results from our model on axial views of CBICA-AMF.nz.76-124 from the validation set. The first row shows Flair modality, while the second and fourth row show the output results respectively from cGAN and refinement architecture. The third row shows the semantic segmentation masks from cGAN overlaid Flair modalities where the fifth row shows outputs after refinement network. The red color codes the whole tumour (WT) region, while pink and yellow represent the enhanced tumour (ET) and the tumorous core (TC) respectively.

Table 1: Comparison of the achieved accuracy for semantic segmentation of different classes of tumour in terms of Dice and Hausdorff distance on validation data [6, 7, 8, 18] reported by the BraTS2017 organizer. The terms WT, ET, and TC are abbreviations of whole tumor region, enhanced tumor region, and core of tumor respectively.

Label	Dice-WT	Dice-ET	Dice-TC	Hdf-WT	Hdf-ET	Hdf-TC
Refinement GAN [26]	0.86	0.64	0.73	7.22	8.30	11.04
cGAN [23]	0.74	0.53	0.61	12.6	16.41	31.0
Recurrent-cGAN [24]	0.79	0.60	0.68	11.73	14.54	25.83
Residual-Encoder [3]	0.82	0.62	0.57	-	-	-
FCN [2]	0.83	0.69	0.69	11.06	11.49	12.53
3D-Unet [4]	0.81	0.76	0.72	13.65	22.36	13.88
Nifty-Net [12]	0.83	0.71	0.68	27.49	17.35	31.34
3D-CNN [22]	0.82	0.46	0.56	9.56	13.8	14.7
biomedia [21]	0.90	0.73	0.79	4.2	4.5	6.5
UCL-TIG [30]	0.90	0.78	0.83	3.8	3.2	6.4
MIC-DKFZ [15]	0.89	0.73	0.79	6.9	4.5	9.4

Table 2: Comparison and the achieved accuracy for semantic segmentation in terms of false negative rate or fnr= $1 - \frac{TruePositive}{TruePositive+FalseNegative}$ and false positive rate or fpr=$1 - \frac{TrueNegative}{TrueNegative+FalsePositive}$ on validation data. The terms of WT, ET, and TC are abbreviations of whole tumor region, enhanced tumor region, and core of tumor respectively.

Label	fnr-WT	fnr-ET	fnr-TC	fpr-WT	fpr-ET	fpr-TC
RefinementGAN [26]	0.11	0.16	0.29	0.02	0.02	0.02
cGAN [23]	0.22	0.34	0.32	0.02	0.04	0.03
Recurrent-cGAN [24]	0.19	0.32	0.30	0.02	0.03	0.02
biomedia [21]	0.11	0.22	0.24	-	-	-
UCL-TIG [30]	0.09	0.23	0.18	-	-	-
MIC-DKFZ [15]	0.11	0.21	0.22	-	-	-

pared to the segmentation results after the refinement network. In the first stage, the generator is trained by true positive and true negative masks. Meanwhile, the discriminator network tests how true is the predicted mask created by the generator. On the top of cGAN, the refinement learns the false negative and false positive masks. Table (2) presents discovery of false negative rate (1-recall) and false positive rate (1-specificity) in detail of network architecture. The final masks computed from the cGAN (or recurrent-cGAN) network with eliminating false negative and adding false positive predicted by refinement network.

Regarding results of false discovery rate presented in Table (2), we have achieved good results as second and third ranked teams in BraTS2017 competition when the segmented masks computed by recurrent conditional GAN and refinement network. Regarding quantitative results by Tables (1 and 2), the networks substituted by LSTM unit predicted more accurate results.

In test time, every group had 48 hours from receiving the test subjects to process them and submit their segmentation results to the online evaluation system. The average value of the Dice coefficient is 0.85 in test time, which the results from Table (3) obtained and evaluated by challenge organizer. Since the results of the challenge in testing are not publicly available, we are not able to compare the performance of the different approaches in the test time.

It is important to mention that our method takes only 58 seconds to segment one MR brain image consisting 155 slices at testing time.

5 Conclusion and Future Work

In this paper, we introduced some recent adversarial frameworks for handling imbalanced problems for the task of medical image segmentation. We developed and evaluated the four different algorithms in data-level and algorithm-level to mitigate

Table 3: The achieved accuracy for brain tumour semantic segmentation by proposed conditional refinement GAN in terms of Dice, sensitivity, specificity, and Hausdorff distance reported by the BraTS-2017 organizer

Evaluation	Validation			Test		
	WT	ET	TC	WT	ET	TC
Dice	0.86	0.64	0.73	0.85	0.61	0.72
Sens	0.89	0.84	0.71	-	-	-
Spec	0.98	0.98	0.97	-	-	-
Hdfd	7.22	8.30	11.04	8.73	59.2	25.9

the issue of unbalanced data for medical image analysis. We achieved promising results on two popular medical imaging benchmarks for the task of semantic segmentation of abnormal tissues as well as a body organ, together with a prediction of diseases. In the future, we plan to investigate the potential of one-class learning using unsupervised GANs for semantic segmentation task.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. "Tensorflow: A System for Large-scale Machine Learning." In: *OSDI*. Volume 16. 2016, pages 265–283.

[2] A. et al. "Brain Tumor Segmentation from Multi Modal MR images using Fully Convolutional Neural Network". In: *Proceedings of the 6th MICCAI BRATS Challenge*. 2017, pages 1–8.

[3] P. et al. "Residual Encoder and Convolutional Decoder Neural Network for Glioma Segmentation". In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. 2018, pages 263–273.

[4] P. Amorim, c. vinicius chagas, and e. guilherme escudero. "3D U-Nets For Brain Tumor Segmentation in MICCAI 2017 BraTS Challenge". In: *Proceedings of the 6th MICCAI BraTS Challenge*. 2017, pages 9–14.

[5] S. Ando and C. Y. Huang. "Deep Over-sampling Framework for Classifying Imbalanced Data". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2017, pages 770–785.

[6] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. Kirby, J. Freymann, K. Farahani, and C. Davatzikos. "Segmentation Labels and Radiomic Features for the Pre-operative Scans of the TCGA-GBM collection". In: *The Cancer Imaging Archive* (2017). doi: 10.7937/K9/TCIA.2017.KLXWJJ1Q.

[7] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. Kirby, J. Freymann, K. Farahani, and C. Davatzikos. "Segmentation Labels and Radiomic Features for the Pre-operative Scans of the TCGA-LGG collection". In: *The Cancer Imaging Archive* (2017). doi: 10.7937/K9/TCIA.2017.GJQ7R0EF.

[8] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. Kirby, J. Freymann, K. Farahani, and C. Davatzikos. "Advancing the Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features". In: *Nature Scientific Data* (2017).

[9] S. Bakas, editor. *2017 International MICCAI BraTS Challenge*. 2017, pages 1–352.

[10] F. Chollet et al. *Keras*. 2015.

[11] P. F. Christ, F. Ettlinger, F. Grun, M. E. A. Elshaer, J. Lipkova, S. Schlecht, F. Ahmaddy, S. Tatavarty, M. Bickel, P. Bilic, M. Rempfler, F. Hofmann, M. D'Anastasi, S. Ahmadi, G. Kaissis, J. Holch, W. H. Sommer, R. Braren, V. Heinemann, and B. H. Menze. "Automatic Liver and Tumor Segmentation of CT and MRI Volumes using Cascaded Fully Convolutional Neural Networks". In: *CoRR* abs/1702.05970 (2017). arXiv: 1702.05970.

[12] Z. Eaton-Rosen, W. Li, G. Wang, T. Vercauteren, B. Sotirios, S. Ourselin, and M. J. Cardoso. "Using niftynet to Ensemble Convolutional Neural Nets for the BRATS Challenge". In: *Proceedings of the 6th MICCAI BraTS Challenge*. 2017, pages 61–67.

[13] M. Havaei, A. Davy, D. Warde-Farley, A. Biard, A. Courville, Y. Bengio, C. Pal, P.-M. Jodoin, and H. Larochelle. "Brain Tumor Segmentation with Deep Neural Networks". In: *Medical image analysis* 35 (2017), pages 18–31.

[14] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015). arXiv: 1502.03167.

[15] F. Isensee, P. Kickingereder, W. Wick, M. Bendszus, and K. H. Maier-Hein. "Brain Tumor Segmentation and Radiomics Survival Prediction: Contribution to the BRATS 2017 Challenge". In: *2017 International MICCAI BraTS Challenge* (2017).

[16] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. "Image-to-Image Translation with Conditional Adversarial Networks". In: *CoRR* abs/1611.07004 (2016).

[17] K. Kamnitsas, W. Bai, E. Ferrante, S. McDonagh, M. Sinclair, N. Pawlowski, M. Rajchl, M. Lee, B. Kainz, D. Rueckert, et al. "Ensembles of Multiple Models and Architectures for Robust Brain Tumour Segmentation". In: *arXiv preprint arXiv:1711.01468* (2017).

[18] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, et al. "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)". In: *IEEE transactions on medical imaging* 34.10 (2015), pages 1993–2024.

[19] M. Mirza and S. Osindero. "Conditional Generative Adversarial Nets". In: *CoRR* abs/1411.1784 (2014).

[20] L. G. Nyúl, J. K. Udupa, and X. Zhang. "New Variants of a Method of MRI Scale Standardization". In: *IEEE transactions on medical imaging* 19.2 (2000), pages 143–150.

[21] N. Pawlowski, M. Rajchl, M. Lee, B. Kainz, D. Rueckert, and B. Glocker. "Ensembles of Multiple Models and Architectures for Robust Brain Tumour Segmentation". In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: Third International Workshop*. Volume 10670. 2018, page 450.

[22] G. R. C. Ramiro and A. Claudio. "Multimodal Brain Tumor Segmentation using 3D convolutional networks". In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. 2018, pages 226–240.

[23] M. Rezaei, K. Harmuth, W. Gierke, T. Kellermeier, M. Fischer, H. Yang, and C. Meinel. "A Conditional Adversarial Network for Semantic Segmentation of Brain Tumor". In: *International MICCAI Brainlesion Workshop*. 2017, pages 241–252.

[24] M. Rezaei, H. Yang, and C. Meinel. "Cardiac MRI Segmentation via Context Aware Recurrent Generative Adversarial Network". In: *Journal Computer Assisted Radiologist and Surgery* (2018). Under review.

[25] M. Rezaei, H. Yang, and C. Meinel. "Instance Tumor Segmentation using Multitask Convolutional Neural Network". In: *IEEE International Joint Conference on Neural Networks* (2018). Accepted.

[26] M. Rezaei, H. Yang, and C. Meinel. "MISS-GANs: Medical Image Semantic Segmentation via Generative Adversarial Networks". In: *Journal Medical Image Analysis* (2018). Under review.

[27] M. Rezaei, H. Yang, and C. Meinel. "Survival GAN: An adversarial Network for learning multiple clinical tasks". In: *Medical Image Computing and Computer Assisted Intervention* (2018). Under review.

[28] M. Rezaei, H. Yang, and C. Meinel. "Whole Heart and Great Vessel Segmentation with Context-aware of Generative Adversarial Networks". In: *Bildverarbeitung für die Medizin 2018 – Algorithmen – Systeme – Anwendungen. Proceedings vom 11. bis 13. März 2018 in Erlangen*. 2018, pages 353–358. doi: 10.1007/978-3-662-56537-7_89.

[29] N. Srebro and A. Shraibman. "Rank, Trace-Norm and Max-Norm". In: *Learning Theory*. 2005, pages 545–560.

[30] G. Wang, W. Li, S. Ourselin, and T. Vercauteren. "Automatic Brain Tumor Segmentation Using Cascaded Anisotropic Convolutional Neural Networks". In: *arXiv preprint arXiv:1709.00382* (2017).

Comparative Text Mining and News Comment Analysis

Julian Risch

Information Systems Group
Hasso-Plattner-Institut
julian.risch@hpi.de

This progress report describes my recent activities in the HPI Research School on Service-Oriented Systems Engineering. It presents the current state of my work and summarizes my research and teaching activities during the last two years.

1 Overview

My field of research is called comparative text mining, where I work on novel document representations. The challenge of document representations is to capture semantic similarities despite different language use. To this end, I develop new topic models, dense vector representations, and recurrent neural networks. Applications of my work are text classification systems and content-based recommender systems. For example, my research can be applied to reference recommendation for patents and scientific papers, content-based book recommendation, and semi-automated comment moderation at online discussion platforms.

1.1 Digital Libraries

I am part of the research community working on digital libraries, which deals with large collections of documents. These documents are typically text documents in our context but they are not limited to a specific type of text document. My datasets contain, for example, patents, scientific papers, book synopses, news articles, user comments, and medical documents. Together with my co-authors I published three different papers at the three major conferences on digital libraries, which are geographically distributed across different continents, but address a joint community. These publications have in common that they deal with large collections of documents and that they mine information from these documents in order to compare them. However, they deal with different datasets and aim at different goals. In our paper "What Should I Cite? Cross-Collection Reference Recommendation of Patents and Papers" published at the Conference on Theory and Practice of Digital Libraries (TPDL) we cluster patents and scientific papers by topic despite their different language use. We propose an extension to cross-collection topic models, which we also evaluated on other datasets and tasks in our paper "My Approach = Your Apparatus? Entropy-Based Topic Modeling on Multiple Domain-Specific Text Collections" at the Joint Conference on Digital Libraries (JCDL). Similar to these two publications, our most recent research is also a recommender system. In "Book Recommendation Beyond the Usual Suspects: Embedding Book Plots Together with Place and Time

Information" we model plot, place, and time of a story in a dense vector space. This space allows us to do arithmetics with books. For example, given a popular book, we can recommend another book that has a similar plot but is set at a different, user-defined place and time. This paper has been accepted at the International Conference on Asia-Pacific Digital Libraries (ICADL).

Recent advances of deep learning in natural language processing introduced the concept of word embeddings. Independently of any downstream applications, word embeddings provide a way to represent words as points in a high-dimensional vector space. Semantically similar words have also (numerically) similar representations in this space. However, to train word embeddings, huge corpora with billions of tokens are needed. The training process is also computationally expensive. Therefore the community mostly uses publicly available, pre-trained word embeddings. These word embeddings have been trained by large companies and research institutions, for example, on all English-language Wikipedia pages.

In our paper "Learning Patent Speak: Investigating Domain-Specific Word Embeddings", we compare generic word embeddings with domain-specific word embeddings. It has been published at the International Conference on Digital Information Management (ICDIM), which is topically related to previously mentioned conferences on digital libraries. Specifically, we find that domain-specific word embeddings can drastically improve precision and recall at an exemplary classification task. We trained these word embeddings on 38 billion tokens and publish them online so that other researchers can use them. Currently, I co-advise a master thesis that extends this work. Its goal is to model the hierarchical scheme for patent classification within a neural network. It also touches the field of few-shot learning, because the class distribution is highly imbalanced. For some classes only very few training samples exist, which makes the classification problem more challenging.

1.2 Comment Analysis

My second field of research is news comment analysis, which deals with user comments posted at online news platforms. Thanks to a collaboration with a large online news provider, I have access to a real-world dataset of millions of user comments with additional metadata about comments, users, and articles. Industry collaborations like this also give me valuable insights into the daily working routine of news editors. In our joint work we discuss our own views of what modern tech-driven journalism could be. Besides proprietary datasets, I also work on publicly available data from other online news platforms.

To some extent, collections of millions of comments can be seen as digital libraries, too. However, collections of comments are typically not used to retrieve archived information. Instead, I identified several other tasks, which I motivate and explain in the following. All these tasks share the research questions "How can we leverage machine learning to improve online discussions?" and "How can we mine information from online discussions?".

- Toxic Comment Classification: For editors at online news platforms it is costly in terms of working power and time to keep discussion sections clean from inappropriate content and to watch the compliance of users ("netiquette"). To support editors at the task of moderating and editing user-submitted content, we propose a semi-supervised machine learning approach. We model linguistic differences of toxic comments in comparison to non-toxic comments and classify whether a comment contains obscene language, insults, threats, hate speech, or any other toxic content that makes users leave a discussion. In a joint work together with PhD students from Beuth University of Applied Sciences, we participated in the Kaggle Toxic Comment Classification Challenge. Our approach is an ensemble of recurrent neural networks, convolutional neural networks, logistic regression, and boosted trees. We finished in the top 2% (place 54/4551) and continued our collaboration in the field of hate speech detection. This collaboration resulted in a paper publication on current research challenges, titled "Challenges for Toxic Comment Classification: An In-Depth Error Analysis" [1]. It will be presented at a workshop co-located with the Conference on Empirical Methods in Natural Language Processing (EMNLP). Further, we plan to do research on how to automatically explain why a particular comment is considered toxic or not.

  We apply our machine learning approach to English and Hindi comments in our paper "Aggression Identification Using Deep Learning and Data Augmentation" [5], but also to German tweets in our paper "Fine-Grained Classification of Offensive Language" [4]. In a collaboration with the German news provider ZEIT Online we work on a dataset of German news comments and published the results in our paper titled "Delete or not Delete? Semi-Automatic Comment Moderation for the Newsroom" [6]. This wide range of languages proofs that our approach is language-independent. These papers have been published at workshops co-located with the Conference on Computational Linguistics (COLING) and the Conference on Natural Language Processing (KONVENS).

  One challenge of toxic comment classification is the small amount of labeled training data. To this end, we propose a transfer learning approach. Instead of training our models with the limited training data of a specific task, we pretrain them on related tasks that provide larger amounts of data. For example we use English-language training data and machine-translate this data to German. After this process, we use the translated data to train models for German-language tasks.

- Highlighting Interesting Comments: A contrary approach to toxic comment classification is the automatic detection of so called "editor picks". Editor picks are comments that are highlighted because of their high writing quality and their relevance to other users. An example is a comment that provides background information that is not mentioned in the article itself. Another example is a comment that summarizes parts of a discussion and is helpful to other users by giving an overview. Currently, I co-advise a master thesis on how to improve automatic classification of comments by incorporating a comment's context,

such as previous comments and its corresponding news article. Further, we participate in the project „Yes comment!“ together with industry partners from online media. This project is framed by the „Masterclass Science Journalism“ funded by the Robert Bosch Foundation. It explores how to summarize and highlight interesting comments in order to enhance the communication in online comment sections. The outcome of the project is planned to be a case study.

- Comment Volume Prediction: News directors decide when to schedule which article for publication. With the help of a good prediction of expected interest and comments, they can balance the distribution of highly controversial topics across a day. Thereby, not only readers and commenters get a chance to engage in each single controversial discussion, but also the moderation workload for comment editors is distributed evenly. Further, knowing which articles will receive many comments helps in the moderation process: Guiding the main focus of attention of moderators towards controversial topics facilitates efficient moderation. We propose a logistic regression model based on article metadata, linguistic, and topical features to predict the number of comments that an article will receive. In this field, I co-advised a master's thesis with the title "Automatically Managing News Comments". Based on the results of this thesis, we published a paper with the title "Prediction for the Newsroom: Which Articles Will Get the Most Comments?" at the NAACL conference[2].

- Comment Ranking: Typically, online comments are presented in chronological order. As a consequence, the earliest comments attract the most attention, regardless of whether their content is interesting for other users. Especially early, provocative comments can stifle any chance of an open and meaningful debate. To encounter this problem, we propose new comment rankings that keep reply structures intact and address different user needs. For example, we consider rankings based on comment upvotes and downvotes or prestige scores for users. This is work in progress and overlaps with the next task: discussion summarization.

- Discussion Summarization: When online discussions grow, it becomes almost impossible to read through all comments. Popular news articles receive hundreds to thousands of comments. Large comment sections are split across multiple pages and most users only pay attention to the first page of comments. Because of the large number of comments, it is hard for users to get an overview of the discussion and extract relevant information. We propose to automatically summarize discussions, cluster subtopics, and extract principal arguments. Thereby, we aim to make longer discussions more engaging and more accessible to users.

- Modeling Linguistic Change: Language use underlies temporal changes. The detection, analysis, and visualization of vocabulary shifts, can give interesting insights. Further, whether a particular comment is appropriate for publication might also vary over time. As a consequence, machine learning models, such as

semi-automated comment classifiers need to adapt to vocabulary shifts. Our approach focuses on diachronic word embeddings, where a word is represented with a different vector depending on the temporal context of its use.

- Modeling Users' Commenting Behavior: While comment volume prediction and modeling linguistic change considers online communities as a whole, modeling the comment behavior of particular users is a more fine-grained task. Here, the goal is to predict which articles a particular user will most likely comment on or in what tone. Besides the accuracy of a model's predictions, its interpretability and explainability is an important aspect. The latter is an unsolved research problem especially for deep neural networks and motivates the use of other machine learning models, such as probabilistic graphical models.

1.3 Outlook

Two submissions are planned in November and January to the Web Conference 2019 (formerly WWW conference) and the International Conference on Research and Development in Information Retrieval 2019 (SIGIR). In November 2018, I will present our paper "Book Recommendation Beyond the Usual Suspects: Embedding Book Plots Together with Place and Time Information" at the 20th International Conference On Asia-Pacific Digital Libraries (ICADL) [3]. At the same time, our paper "Challenges for Toxic Comment Classification: An In-Depth Error Analysis" will be presented at the 2nd Workshop on Abusive Language Online (co-located with EMNLP) [1]. After my work on comment ranking and discussion summarization is finished, I plan to focus my research on how to model users' commenting behavior. Starting from March 2019, I will again teach an HPI Youth College course on *Information Retrieval and Web Search*.

References

[1] B. van Aken, J. Risch, R. Krestel, and A. Löser. "Challenges for Toxic Comment Classification: An In-Depth Error Analysis". In: *Proceedings of the 2nd Workshop on Abusive Language Online (co-located with EMNLP)*. 2018.

[2] C. Ambroselli, J. Risch, R. Krestel, and A. Loos. "Prediction for the Newsroom: Which Articles Will Get the Most Comments?" In: *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. 2018, pages 193–199.

[3] J. Risch, S. Garda, and R. Krestel. "Book Recommendation Beyond the Usual Suspects: Embedding Book Plots Together with Place and Time Information". In: *Proceedings of the 20th International Conference On Asia-Pacific Digital Libraries (ICADL)*. 2018.

[4] J. Risch, E. Krebs, A. Löser, A. Riese, and R. Krestel. "Fine-Grained Classification of Offensive Language". In: *Proceedings of the Germeval Task 2018 – Shared Task on the Identification of Offensive Language* (*co-located with KONVENS*). 2018.

[5] J. Risch and R. Krestel. "Aggression Identification Using Deep Learning and Data Augmentation". In: *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying* (*co-located with COLING*). 2018, pages 150–158.

[6] J. Risch and R. Krestel. "Delete or not Delete? Semi-Automatic Comment Moderation for the Newsroom". In: *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying* (*co-located with COLING*). 2018, pages 166–176.

Power-Law Distributions in Random Satisfiability

Ralf Rothenberger

Algorithm Engineering
Hasso Plattner Institute
ralf.rothenberger@hpi.de

One of the most fundamental problems in computer science is Propositional Satisfiability (SAT). Its computational hardness gave rise to many complexity-theoretical concepts, e.g. NP-hardness and lower bounds on algorithmic runtime via the (Strong) Exponential Time Hypothesis. In order to analyze the average-case complexity of SAT and to generate benchmarks for solvers, instances are created at random (random SAT). Random SAT initiated the development of sophisticated rigorous and non-rigorous techniques for analyzing random structures. Despite a long line of research and substantial progress, nearly all theoretical work on random SAT assumes a *uniform* distribution on the variables. In contrast, real-world instances often exhibit community structure and large fluctuations in variable occurrence, similar to big real-world networks.

This report documents my work analyzing a more realistic random SAT model inspired by one proposed by Ansótegui et al. [2]. I present my latest results, which show under which requirements the probability of generating satisfiable formulas abruptly changes from asymptotically almost surely (a. a. s.) one to a. a. s. zero.

1 Overview

Random k-SAT was introduced as a means to study the average-case complexity of Propositional Satisfiability (SAT). In it one creates random SAT instances with a given number of variables n, a given number of clauses m, and a given number of literals per clause k. These instances are generated in such a way, that each instance with these parameters has the same uniform probability to be created. One of the most prominent questions related to random k-SAT is trying to prove the satisfiability threshold conjecture. Intuitively, the satisfiability threshold is the clause-variable-ratio m/n at which the probability to generate a satisfiable formula goes from one to zero (see Figure 1).

The *satisfiability threshold conjecture* states that for a formula Φ generated with random k-SAT, there is a real number r_k such that

$$\lim_{n\to\infty} \Pr\{\Phi \text{ is satisfiable}\} = \begin{cases} 1 & m/n < r_k; \\ 0 & m/n > r_k. \end{cases}$$

Trying to prove this conjecture has been the subject of many works since the 1980's and there has been a lot of progress since then: For $k = 2$, Chvátal and Reed [4] and, independently, Goerdt [13] proved that $r_2 = 1$. For $k \geq 3$, explicit upper and lower bounds have been derived, e.g., $3.52 \leq r_3 \leq 4.4898$ [8, 14, 15]. Additionally, the cavity method from statistical mechanics [16] was used to suggest a numerical

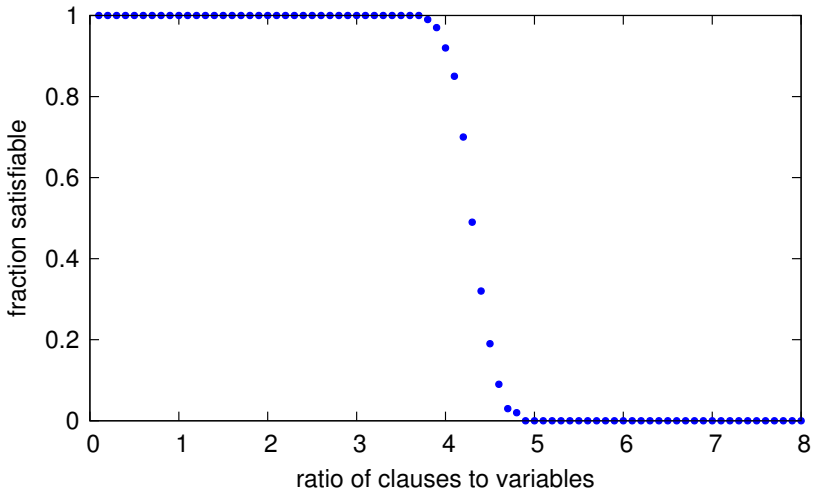


Figure 1: Fraction of satisfiable formulas for random SAT with clause length $k = 3$ and $n = 100$ variables. Each data point created with 500 random formulas. The threshold can be observed somewhere around a clause-variable-ration of $m/n \approx 4.26$.

estimate of $r_3 \approx 4.26$. Coja-Oghlan and Panagiotou [5, 6] derived a bound (up to lower order terms) for $k \geq 3$ with $r_k = 2^k \log 2 - \frac{1}{2}(1 + \log 2) \pm o_k(1)$. Recently, Ding, Sly and Sun [7] proved the exact position of the threshold for sufficiently large values of k.

Determining the location of the satisfiability threshold can give insights about structural properties of the random formulas. Furthermore, it can be observed that the runtime of SAT solvers is especially high around the satisfiability threshold [17] (see Figure 2). Therefore, knowing the location of the threshold can be helpful when trying to generate difficult benchmarks.

However, nearly all other theoretical work on random SAT assumes a *uniform* distribution on the variables. In contrast, real-world instances often exhibit community structure and large fluctuations in variable occurrence, similar to big real-world networks. For example, it has been found out that the degree distribution of many families of industrial instances follows a power-law [1, 3]. This means that the fraction of variables that appear i times is proportional to $i^{-\beta}$, where β is a constant intrinsic to the instance. To help close the gap between the structure of random and industrial instances, Ansótegui et al. [1] proposed a power-law random SAT model.

These insights inspired us to also look beyond random k-SAT with uniform distributions. Instead, we consider instances with a given ensemble of variable probability distributions $(\vec{p}_n)_{n\in\mathbb{N}}$. To create an instance with n variables and m clauses, the variables of each clause are drawn with a probability proportional to $\vec{p}_n$, then they are negated independently with a probability of $1/2$ each. This is repeated m times. We call this model *non-uniform random k-SAT*. The main questions we want to answer now are: Does an equivalent of the satisfiability threshold conjecture still hold, if

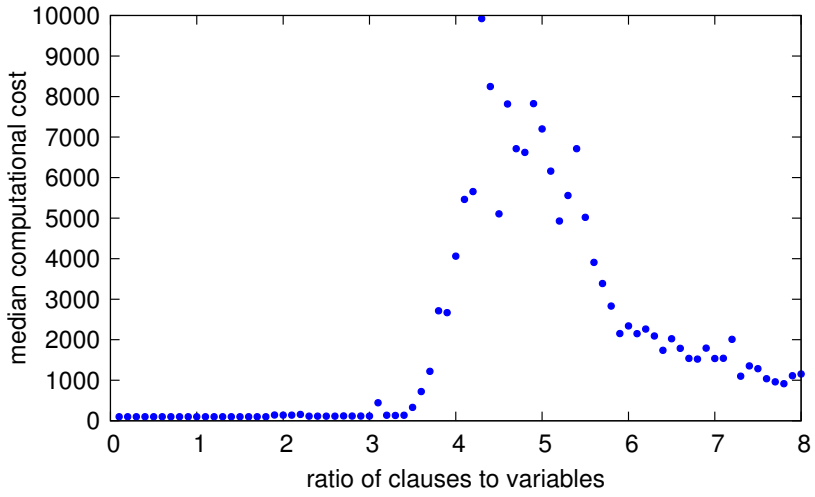


Figure 2: Median number of propagations when solving with MiniSAT (no preprocessing or randomization) for uniform random SAT with clause length $k = 3$ and $n = 100$ variables. Each point is the median of 500 random formulas. It can be seen that around the satisfiability threshold there is a peak in median runtime, which turns out to scale exponentially with the number of variables n.

we consider the non-uniform random k-SAT model? If so, for which ensembles of variable probability distributions does it hold and for which does it not?

In this report we summarize and discuss our latest results trying to answer these questions.

2 Preliminaries and Related Work

To give an answer to these questions, one first has to understand asymptotic and sharp thresholds for satisfiability. Let μ_m be the probability that a random formula is unsatisfiable when drawing $m(n)$ clauses over n variables. $m^* = m^*(n)$ is an *asymptotic threshold function* for unsatisfiability if for every $m = m(n)$

$$\lim_{n\to\infty} \mu_m = \begin{cases} 0, & \text{if } m = o(m^*) \\ 1, & \text{if } m = \omega(m^*). \end{cases}$$

We say that unsatisfiability has a *sharp threshold*, if there exists a function $m^* = m^*(n)$ such that for every constant $\varepsilon > 0$ and for every $m = m(n)$

$$\lim_{n\to\infty} \mu_m = \begin{cases} 0, & \text{if } m \leq (1-\varepsilon)m^* \\ 1, & \text{if } m \geq (1+\varepsilon)m^*. \end{cases}$$

Otherwise we call the threshold *coarse*.

In terms of threshold functions, the satisfiability threshold conjecture states that there is a sharp threshold for satisfiability at $m = r_k \cdot n$ and the constant r_k is the same for a fixed k and all sufficiently large n. For $k = 2$, Chvátal and Reed [4] and Goerdt [13] proved the conjecture and showed that $r_2 = 1$. However, random 2-SAT is easier to analyze than random k-SAT and their techniques do not work for bigger values of k. For uniform random k-SAT the "recipe" for proving the conjecture is as follows:

1. Show the existence of an asymptotic threshold function, i.e. show constant lower and upper bounds on r_k.
2. Prove that the threshold is sharp. In 1999 Friedgut [9] showed that the satisfiability threshold for uniform random k-SAT is sharp, although its location is not known exactly for all values of k. However, his result does not prove that r_k is the same for a fixed k and all sufficiently large values of n. Friedgut's proof relies on knowing the asymptotic threshold function.
3. Derive the actual constant r_k and that the threshold is sharp around it. Ding et al. [7] were the first to prove the exact value of r_k for values of k bigger than 2. Their proof relies on the result of Friedgut.

The approach we have is to show equivalents of these proofs for non-uniform probability distributions in order to show or refute the conjecture for certain probability ensembles.

3 Our Results

In our last reports we primarily reported results on the asymptotic threshold for *power-law random k-SAT*, i.e. non-uniform random k-SAT where the prescribed variable frequency is a power law. These results are summed up in the next section before we state our more general and more recent results in the section thereafter.

3.1 Power-Law Random k-SAT

We analyzed the power-law random k-SAT model and showed results regarding its asymptotic satisfiability threshold. First, we showed the following lower bound on the threshold position for power-law random 2-SAT [11]. An illustration of this bound compared to experimental results can be seen in Figure 3.

Theorem 1. *Scale-free random 2-SAT with power-law exponent $\beta > 3$ and clause-variable ratio $m/n < \frac{(\beta-1)(\beta-3)}{(\beta-2)^2}$ is almost surely satisfiable.*

We then showed that the satisfiability threshold for power-law random k-SAT with power-law exponents below $\frac{2k-1}{k-1}$ is sublinear in n [10].

Corollary 2. *Let Φ be a random k-SAT formula that follows an arbitrary power-law distribution and consists of $\Theta(n)$ clauses. If the power-law exponent is $\beta < \frac{2k-1}{k-1}$, Φ is unsatisfiable with high probability.*

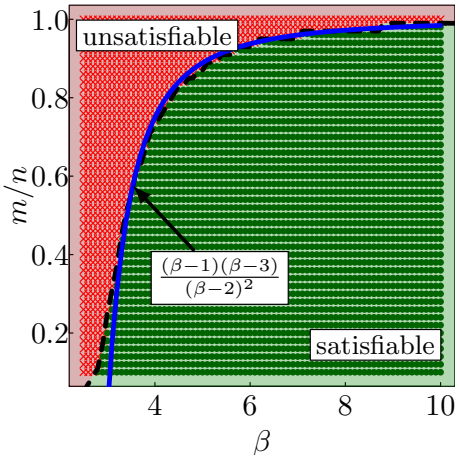


Figure 3: Phase diagram for scale-free 2-SAT formulas with $n = 10^7$ variables. We empirically observe a sharp phase transition (– –), which closely matches the theoretical bound of Theorem 1 (—).

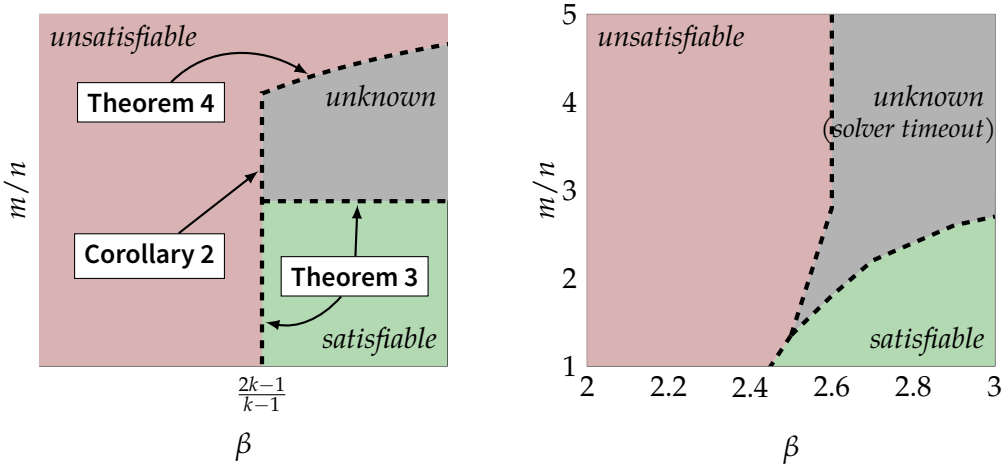


Figure 4: Illustration of our asymptotic results for the power-law satisfiability threshold location when $n \to \infty$ (left) compared with empirical results for randomly generated power-law 3-SAT formulas on $n = 10^6$ variables checked with the SAT solver MiniSAT (right). The timeout was set to one hour.

For power-law exponents above $\frac{2k-1}{k-1}$ we were able to show that the satisfiability threshold is at least linear in n by providing the following lower bound.

Theorem 3. *Let Φ be a random k-SAT formula that follows an arbitrary power-law distribution. If the power-law exponent is $\beta > \frac{2k-1}{k-1} + \varepsilon$ for an arbitrary $\varepsilon > 0$, Φ is satisfiable with high probability if $\frac{m}{n}$ is a small enough constant.*

Although the satisfiability threshold for every probability distribution can be at most, linear, we improved upon the trivial bound of $\ln(2)/\ln(\frac{2^k}{2^k-1})$ with the following theorem.

Theorem 4. *Let Φ be a random k-SAT formula with $k \geq 2$ and $r = \frac{m}{n}$ that follows a power-law distribution. Let further $N \in \mathbb{N}^+$ be any constant. If the power-law exponent is $\beta > 2$, then Φ is w. h. p. unsatisfiable if*

$$\left(\left(1-\tfrac{1}{2^k}\right)^r 2^{\frac{1}{N}} \prod_{l=1}^{N-1}\left[2-\exp\left(-(1+o(1))\, r \frac{k}{2^k-1}\frac{\beta-2}{\beta-1}\left(\frac{N}{l}\right)^{\frac{1}{\beta-1}}\right)\right]^{\frac{1}{N}}\right) < 1.$$

A summary of those results can be seen in Figure 4.

3.2 More recent results

Recently we considered non-uniform random k-SAT with other variable probability distributions besides a power law. Let $\mathcal{D}(n,k,\vec{p},m)$ be non-uniform random k-SAT, where m clauses are drawn with a variable probability distribution $\vec{p}$ over n variables. The first result we showed for $\mathcal{D}(n,k,\vec{p},m)$ asserts sharpness if the asymptotic threshold is known and if $\vec{p}$ has certain properties [12]. This result provides us with the second part of our "recipe" for proving the satisfiability threshold conjecture.

Theorem 5. *Let $\vec{p}$ be a variable probability distribution on n variables and let $m_c = t$ be an asymptotic satisfiability threshold for $\mathcal{D}(n,k,\vec{p},m)$ with respect to m. If*

$$\|\vec{p}\|_\infty = o\left(t^{-k/(2k-1)} \cdot \log^{-(k-1)/(2k-1)} t\right)$$

and

$$\|\vec{p}\|_2^2 = \mathcal{O}\left(t^{-2/k}\right),$$

then satisfiability has a sharp threshold on $\mathcal{D}(n,k,\vec{p},m)$ with respect to m.

The first step toward showing this result is considering a slightly different random k-SAT model. In $\mathcal{D}(n,k,\vec{p},m)$ a fixed number m of clauses is drawn at random. In the new model we consider, which we call $\mathcal{F}(n,k,\vec{p},s)$, we flip a coin for each of the $\binom{n}{k}\cdot 2^k$ possible k-clauses to see if we add it. The probability with which we add a certain clause is equivalent to the probability of drawing that clause in $\mathcal{D}(n,k,\vec{p},m)$ multiplied with a scaling factor s. We can now define an equivalent of asymptotic and sharp thresholds with respect to the scaling factor s instead of the number of clauses m. This model is a lot easier to analyze and we can show that the thresholds in both models behave similarly under the conditions of Theorem 5. The following theorem now asserts sharpness of the threshold for $\mathcal{F}(n,k,\vec{p},s)$.

Theorem 6. *Let $\vec{p}$ be a variable probability distribution on n variables and let $s_c = t$ be an asymptotic satisfiability threshold for $\mathcal{F}(n,k,\vec{p},s)$ with respect to s. If*

$$\|\vec{p}\|_\infty = o\left(t^{-k/(2k-1)} \cdot \log^{-(k-1)/(2k-1)} t\right)$$

and

$$\|\vec{p}\|_2^2 = \mathcal{O}\left(t^{-2/k}\right),$$

then satisfiability has a sharp threshold on $\mathcal{F}(n,k,\vec{p},s)$ with respect to s.

The proof of this theorem follows the lines of Friedgut's proof for the uniform case [9] and employs Fourier transformation techniques by O'Donnell [18]:

We assume toward a contradiction that the threshold is coarse. Then the Sharp Threshold Theorem [18] tells us that there have to be so-called "boosters" of constant size that appear with constant probability in the random formula. These boosters are subformulas which have the property that conditioning on their existence *boosts* the probability of the random formula to be unsatisfiable by at least an additive constant.

One kind of booster are unsatisfiable subformulas of constant size. Conditioning on these would boost the probability to be unsatisfiable to one. We rule these out by showing that they do not appear with constant probability.

Then, we consider subformulas which give the second highest boost: maximally quasi-unsatisfiable subformulas. These are subformulas which have only *one* satisfying assignment for the variables appearing in them and adding any new clause over those variables makes them unsatisfiable. We want to show that these cannot boost the probability of a formula to be unsatisfiable by a constant.

Again toward a contradiction, we assume, that conditioning on a maximally quasi-unsatisfiable subformula T is enough to boost the unsatisfiability probability by a constant. First, we prove that conditioning on T is equivalent to adding a number of clauses of size shorter than k to the random formula over variables not appearing in T. Then, we use a version of Friedgut's coverability lemma to show that, if adding these clauses of size smaller than k makes the random formula unsatisfiable with constant probability, then so does adding $o(t)$ clauses of size k. We prove that this probability is dominated by the probability to make the original random formula unsatisfiable for a slightly bigger scaling factor. However, due to the assumption of a coarse threshold, the slope of the probability function for unsatisfiability has to be small at one point in the threshold interval. If we consider exactly this point, the probability to make the original random formula unsatisfiable cannot be increased by a constant with our slightly increased scaling factor. This contradicts our assumption that the probability is boosted by a constant in the first place. Therefore, quasi-unsatisfiable subformulas cannot be boosters.

After showing this, every less restrictive subformula cannot be a booster either. That means, the only possible boosters are unsatisfiable subformulas, which we ruled out already. Therefore, the implication of the Sharp Threshold Theorem does not hold, which contradicts the assumption of a coarse threshold.

This theorem especially applies to both uniform random k-SAT and power-law random k-SAT with exponent $\beta > \frac{2k-1}{k-1} + 1$.

Corollary 7. *Let $\vec{p}$ be an arbitrary power-law distribution. If the power-law exponent is $\beta > \frac{2k-1}{k-1} + \varepsilon$ for an arbitrary $\varepsilon > 0$, then $\mathcal{D}(n, k, \vec{p}, m)$ has a sharp threshold with respect to m.*

4 Conclusion and Future Work

I analyzed the power-law random SAT model proposed by Ansótegui et al. [2]. For the case of power-law random 2-SAT I was able to prove a lower bound on the satisfiability threshold which depends on the power-law exponent of the variable distribution and seems to be tight in practice [11]. For k-SAT with $k \geq 3$ I showed that only the regime of distributions with power-law exponent bigger than $\frac{2k-1}{k-1}$ exhibits a satisfiability threshold which is constant in the clause-variable-ratio, while distributions with a lower exponent almost surely generate unsatisfiable formulas at constant clause-variable ratios. For power-law exponents above $\frac{2k-1}{k-1}$ I gave upper and lower bounds on the satisfiability threshold, which allowed me to show that an asymptotic threshold exists at $\Theta(n)$. These results can also be found in [10].

Furthermore, I studied a more general model for random k-SAT, which I called non-uniform random k-SAT. For this model I proved requirements on the sharpness of the satisfiability threshold depending on the variable probability distribution [12], c. f. Theorem 5. The requirements of the theorem are met, if the power-law exponent is above $\frac{2k-1}{k-1} + 1$, proving a sharp threshold in this case. This means, for power-law random k-SAT with power law exponents above $\frac{2k-1}{k-1} + 1$, I already provided the first two out of three ingredients towards proving an equivalent of the satisfiability threshold conjecture

At the moment I am further investigating the threshold behavior of non-uniform random k-SAT. Since random 2-SAT is easier to analyze I am considering non-uniform random 2-SAT. For this special case I want to show matching lower and upper bounds on the satisfiability threshold under certain conditions on the variable probability distribution. This could allow me to immediately prove the satisfiability threshold conjecture for certain probability ensembles. If these conditions are not met, I want to show that there is an asymptotic threshold, but this threshold is coarse. This will show a dichotomy for the sharpness of the threshold depending on the variable probability distribution.

I want to show similar results for non-uniform random k-SAT with $k > 2$. This means studying the asymptotic threshold and showing conditions under which the threshold is coarse. Furthermore, I want to try and improve my result on the sharpness of thresholds to demand only $\|\vec{p}\|_\infty = o\left(t^{-1/k}\right)$.

All results mentioned in this report and some of the results I am currently working on will be part of my dissertation.

References

[1] C. Ansótegui, M. L. Bonet, and J. Levy. “On the Structure of Industrial SAT Instances”. In: *15th Intl. Conf. Principles and Practice of Constraint Programming (CP)*. 2009, pages 127–141.

[2] C. Ansótegui, M. L. Bonet, and J. Levy. "Towards Industrial-Like Random SAT Instances". In: *21st Intl. Joint Conf. Artificial Intelligence (IJCAI)*. 2009, pages 387–392.

[3] Y. Boufkhad, O. Dubois, Y. Interian, and B. Selman. "Regular Random *k*-SAT: Properties of Balanced Formulas". In: *J. Automated Reasoning* 35.1-3 (2005), pages 181–200.

[4] V. Chvátal and B. A. Reed. "Mick Gets Some (the Odds Are on His Side)". In: *33rd Symp. Foundations of Computer Science (FOCS)*. 1992, pages 620–627.

[5] A. Coja-Oghlan. "The Asymptotic *k*-SAT Threshold". In: *46th Symp. Theory of Computing (STOC)*. 2014, pages 804–813.

[6] A. Coja-Oghlan and K. Panagiotou. "The Asymptotic *k*-SAT Threshold". In: *Advances in Mathematics* 288 (2016), pages 985–1068.

[7] J. Ding, A. Sly, and N. Sun. "Proof of the Satisfiability Conjecture for Large K". In: *47th Symp. Theory of Computing (STOC)*. 2015, pages 59–68.

[8] J. Díaz, L. M. Kirousis, D. Mitsche, and X. Pérez-Giménez. "On the Satisfiability Threshold of Formulas With Three Literals per Clause". In: *Theoretical Computer Science* 410.30-32 (2009), pages 2920–2934.

[9] E. Friedgut. "Sharp Thresholds of Graph Properties, and the *k*-SAT Problem". In: *J. Amer. Math. Soc.* 12.4 (1999). With an appendix by Jean Bourgain, pages 1017–1054. issn: 0894-0347.

[10] T. Friedrich, A. Krohmer, R. Rothenberger, T. Sauerwald, and A. M. Sutton. "Bounds on the Satisfiability Threshold for Power Law Distributed Random SAT". In: *25th European Symposium on Algorithms (ESA)*. 2017, 37:1–37:15.

[11] T. Friedrich, A. Krohmer, R. Rothenberger, and A. M. Sutton. "Phase Transitions for Scale-Free SAT Formulas". In: *31st Conf. Artificial Intelligence (AAAI)*. 2017, pages 3893–3899.

[12] T. Friedrich and R. Rothenberger. "21st Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT)". In: *International Conference on Theory and Applications of Satisfiability Testing (SAT)*. 2018, pages 273–291.

[13] A. Goerdt. "A Threshold for Unsatisfiability". In: *J. Comput. Syst. Sci.* 53.3 (1996), pages 469–486.

[14] M. T. Hajiaghayi and G. B. Sorkin. *The Satisfiability Threshold of Random 3-SAT is at Least 3.52*. Technical report RC22942. IBM, 2003.

[15] A. C. Kaporis, L. M. Kirousis, and E. G. Lalas. "The Probabilistic Analysis of a Greedy Satisfiability Algorithm". In: *Random Struct. Algorithms* 28.4 (2006), pages 444–480.

[16] M. Mézard, G. Parisi, and R. Zecchina. "Analytic and Algorithmic Solution of Random Satisfiability Problems". In: *Science* 297.5582 (2002), pages 812–815.

[17] D. G. Mitchell, B. Selman, and H. J. Levesque. "Hard and Easy Distributions of SAT Problems". In: *10th Conf. Artificial Intelligence (AAAI)*. 1992, pages 459–465.

[18] R. O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

Mobile Fabrication

Thijs Roumen

Human Computer Interaction
Hasso-Plattner-Institute
thijs.roumen@hpi.uni-potsdam.de

I explore the future of fabrication, in particular the vision of mobile fabrication, which I define as "personal fabrication on the go". We introduce this vision with two surveys, two simple hardware prototypes, matching custom apps that provide users with access to a solution database and custom fabrication processes we designed specifically for these devices. Our findings suggest that mobile fabrication is a compelling next direction for personal fabrication. From our experience with the prototypes we derive the hardware requirements to make mobile fabrication technically feasible.

Aside from presenting the vision of mobile fabrication, I will touch upon the different challenges for mobile fabrication to become a reality (how to model on the go, what the hardware for mobile fab can be, how to ensure portability of models) and present an overview of projects we are currently engaged in to push this agenda forwards.

Figure 1: According to our survey, one use case of mobile fabrication is to fix things that break while on the go. This user, for example, fabricates a hex key to fix his broken bike light using either (a) a custom "mobile" 3D printer or (b) a "human-assisted" 3D printer based on an extruder pen. We drive both using a mobile phone running our custom app.

1 Overview

Personal fabrication has emerged as a topic in human computer interaction [6]. 3D printers, initially considered tools for prototyping [9], are now explored as tools to

help users solve engineering problems, such as the design and assembly of furniture and vehicles [15], optimization of objects' aerodynamics [19], or repair of objects [17].

While fabrication currently takes place in offices, labs, and workshops, the current evolution of 3D printing hardware suggests that 3D printers are about to achieve a mobile form factor (e.g. iBoxNano (iboxprinters.com) or Olo (olo3d.com)). Future users may soon have access to such devices while on the go. In an analogy to mobile computing, such mobile fabrication could provide users with access to fabrication anywhere, anytime. This raises a number of questions: what will users do with such devices while on the go? How will the hardware develop? What issues and limitations will users encounter?

2 Approach

First we explore the vision of mobile fabrication, this is the main content of this document. The conclusion of this is a set of requirements to make this vision a reality. I am currently engaged in a series of follow-up projects that focus each on one of these requirements. One of them being grafter [12], which was presented in previous retreats, focusing on the challenge of modeling on the go (remixing and browsing is feasible on the go, 3D modeling from scratch is not).

To explore the vision of mobile fabrication, we conducted two surveys with the primary objective to create a basis for our subsequent work. The objective of the first survey was to create a list of objects potential future users might want to fabricate while on the go. The objective of the second survey was to prioritize this list. The resulting prioritized list of objects became the basis for our subsequent engineering.

2.1 Survey one: scenarios worth solving with fabrication

We recruited 40 volunteers from our institution (age 18-38, 12 female). Each participant filled in a questionnaire with the following wording "We are studying 3D printing, specifically a potential future where people might carry a tiny 3D printer with them at all times, as we do with mobile phone. We are wondering about potential use cases. Please list five on-the-go scenarios where being able to make a missing object quasi-instantaneously would really help you out."

Results: Each participant listed on average of 3.9 objects for a total of 75 distinct objects. The list contained 50 objects that could be produced by a mobile 3D printer.

Discussion The useful scenarios in which objects on the list would be required could be characterized by three main qualities: (1) Unexpectedness: users are unlikely to incidentally carry the required objects. (2) Importance: not having the object costs time, money, safety, or reputation. (3) Urgency: The problem requires a timely solution otherwise users would solve the problem later, by buying the object or by fabricating it at home or at work. To help us focus our subsequent engineering effort

on the most relevant objects, we conducted a second survey in which we prioritized objects on this list.

2.2 Survey two: prioritizing the scenarios

We recruited a separate set of 39 participants (age 20–45, 18 female). Each participant rated each of 12 use cases of mobile fabrication. We had created the list by sampling each of the 5 categories from the first survey and then making the list of objects more tractable by adding a brief description of a context of use. Participants rated objects in the resulting scenarios as "must have", "nice to have", and "uninteresting".

Results: The three scenarios that received the highest number of "must have" ratings were: (15) "make a key when you locked yourself out at home", (11) "create earplugs when there is somebody snoring next to you in a long distance bus", and (9) "make a carabiner to fix a bag strap on the way to plane."

Scenarios with the highest combined number of "must have" and "nice to have" ratings were: (38) "Make an Allen wrench to fix a bike lamp" (36) "Replacing shoelaces when they break during a longer hike", and (36) the carabiner mentioned before. The least popular scenarios (with largest number of "unimportant" ratings) were: (25) "replace a lost earring retainer", (11) "make a shopping cart clip", and (8) "make disposable cutlery".

Discussion The second survey provided us with a ranking of objects that helped us focus our subsequent engineering effort. Furthermore, the survey had raised the question about engineering issues, which we decided to address using a solutions database. Finally, the large number of "must have" ratings in the second survey suggests that mobile fabrication has the potential to matter.

2.3 Engineering and practicality check 1: modified 3D printer

Building on the results of our surveys, we set out to test the practicality of mobile fabrication. We started by creating the first iteration of what a mobile 3D printer might look like, wrote a matching app for it, and then tried to fabricate the list of desired objects from our survey. To satisfy the urgency objective, we re-engineered all objects so as to minimize build time. Our overall objective was to investigate what objects this type of device would be able to produce and where mobile 3D printers would fail in order to derive implications about mobile 3D printer hardware.

We started out with a 3D printer that extrudes plastic. We wanted to use the printer truly "on the go" and FDM worked well here because it continues to print when held sideways or even upside down and while being shaken. This criterion prevented us from using certain other printer designs, including those based on stereo-lithography as they use a container of liquid resin that needs to be upright and stationary while printing (e.g. Olo).

In order to optimize the form factor, we reduced the height of the device as shown in Figure 2. This reduced the printable height from 11.6cm to 2.2cm, and the print area to 9.1 $\times$ 8.4 cm. The resulting printer weighted 1190g and measured 9 x 18.5 x

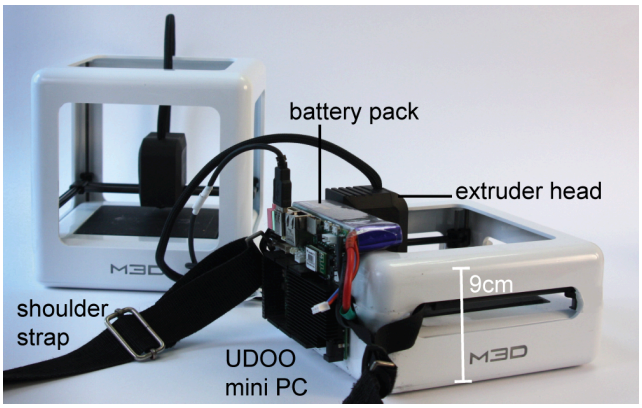


Figure 2: The modified printer, a Micro 3D printer reduced in height and extended with battery pack and UDOO miniPC

18.5cm, which makes its size comparable to a handbag or messenger bag. We added a shoulder strap, allowing users to wear the printer like a messenger bag, as shown in Figure 1 and Figure 2.

To allow the printer to work on the go, we retrofitted it with a battery pack that allowed for about one hour of printing time (1800mAh). We attached a single-board computer (UDOO Quad board) running a web-based host (OctoPrint [7]) on top of Linux (Udoobuntu 1.1). This allowed us to control the printer through a phone app connecting to the OctoPrint server as a web app.

To allow users to look up engineered solutions, we wrote the app shown in Figure 3. The app allowed users to search for solutions for the on-the-go scenarios from the survey and it returned hand-engineered annotated 3D models. The app ran in a web browser and was written in AngularJS. The app retrieved the 3D models from a cloud service (Firebase), which used a NoSQL database to store relevant data in JSON format. We stored the images accompanying the 3D models with an image service provider (Cloudinary). The app interfaced with Firebase and the server using HTTP requests.

Walkthrough Users operated the system as illustrated by Figure 3. Here a user uses the system to re-attach a bike lamp. (a) He wants to ride home in the dark when he discovers that the lamp sags – the mount of the bike lamp is broken. In order to make his ride home safe, he decides to re-attach the lamp. (b) Close inspection reveals a loose hex nut, but our user does not carry the matching hex key. (c) He starts our app and enters "hex key". (d) The system offers several models of hex keys – all custom designs reduced to the bare mini-mum to allow for fast fabrication. The first two are sized 5mm and 6mm, but the user is uncertain about the diameter of the nut. He therefore picks the third model, which offers two heads: one for 5mm and one for 6mm.

(e) The user produces the hex key, which takes 25 minutes. The printer works in any orientation and while moving. Since the app works while running in the background, the user is free to roam around and to use the phone in the meantime.

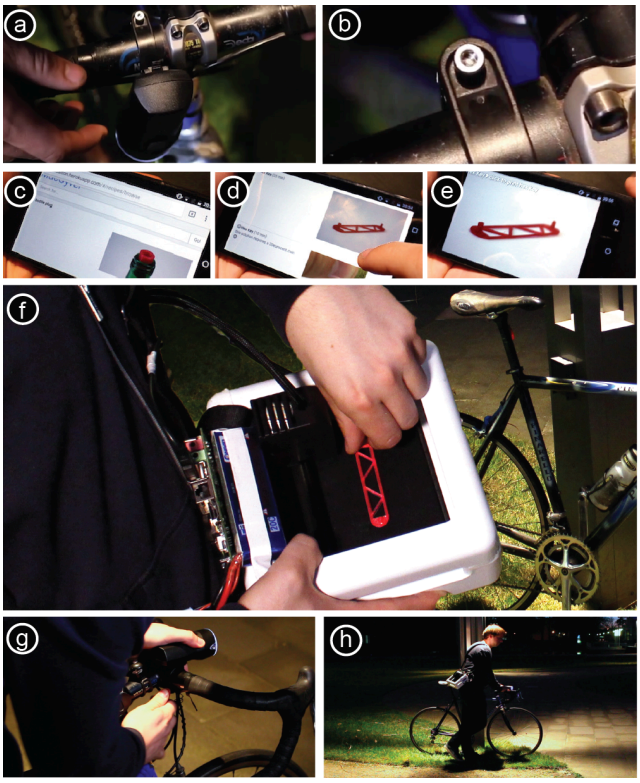


Figure 3: Walkthrough

(f) 25 minutes later the app plays a notification sound. The user removes the hex key from the print chamber, (g) re-attaches the lamp by tightening the hex nut, and (h) rides on safely.

Designing for mobile fabrication based on 3D printer In order to engineer the solutions in the database, we got together with a team of three lab members and recreated the 3D printable designs from the study list for our mobile 3D printer setup. We generally started with objects from an online repository (Thingiverse.com) and then optimized for use with our mobile printer. Most designs only required optimizing material use in order to maximize printing speed. We created five types of optimizations, which we discuss at the example of the aforementioned hex key (Figure 4 shows 5 close-ups).

(a) Use strong geometric structures. The handle of the hex key has to be strong enough to transmit large amounts of torque, yet we still want it to print quickly. As shown in Figure 4, we address this by creating a handle in the form of a flat structure of connected beams (aka truss) optimized to handle torque. (b) Avoid support material by designing in 2½D. To prevent buckling, we need to add several very narrow extra layers on top for a L-shaped cross section, but we add them on one side resulting in a 2½D design that prints is flat against the build platform and does not require support-material.

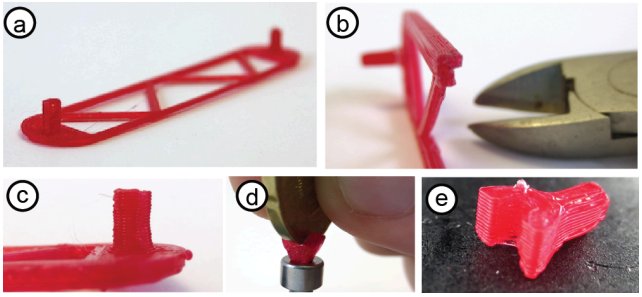


Figure 4: Our optimizations, at the example of the hex key included

(c) Reinforce weak points. The weakest point of the hex key design is the connection between handle and tip. We rein-force it by adding a smooth transition, aka a fillet. (d) Reinforce using metal parts. If we need to transmit even more torque, e.g., to unlock rusty nuts, make users embed metal objects they are likely to carry, here a coin.

(e) To maximize strength, we print this hex key sideways. This allows filament to weave back and forth between the top and the cradle for the coin, resulting in extra stability.

More challenging was the fabrication of objects that require printing around or through objects from the environment. Shoelaces are one example. While we can 3D print shoelaces, the nature of the PLA plastic makes it hard to tie a knot. We would therefore prefer to close up our shoelaces by 3D printing stoppers at each end, i.e., print, pause the printer to insert the lace into the shoe, and then finish the print (as demonstrated in [22] and [2]).

We were able to avoid this type of complication by adding 3D printed locking mechanism into our objects, i.e., print the entire object, insert it, and then close the mechanism. Figure 5 illustrates this, implementing (a) shoelaces as a zip tie and (b) a chain link as a carabiner.

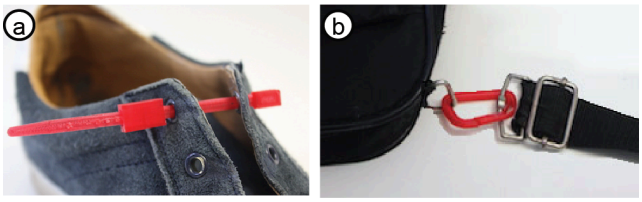


Figure 5: (a) Shoelace as zip tie, (b) chain link as carabiner

2.3.1 Limitation: Objects that have to fit the environment

While we managed to overcome the challenge of making objects that "loosely" connect with the environment, objects that "tightly" interact with the environment remain elusive.

The hex key from our walkthrough is a benevolent sub-category of this problem. While the user has to guess the size of the nut the hex nuts are standardized. This leaves a reasonably small number of choices, which we can address by simply implementing all the choices under consideration, i.e., a 5mm head and a 6mm head. In this particular example, we can do so with little overhead as most of the printing effort goes into the handle regardless. The overhead of producing multiple solutions, however, grows with the number of possible choices and the approach fails when trying to reproduce an infinite number of choices, such as ear buds that fit the 3D geometry of the user's ear.

Ultimately, mobile 3D printers will likely contain appropriate measurement tools in order to fabricate this class of objects. Measurement equipment could be as simple as calipers or as complex as a 3D scanner with sub-millimeter precision. We discuss this in the discussion section.

Since this type of scanning equipment is not quite ready for mobile use, we took a different approach and created a second mobile fabrication prototype that drops the 3D printer in favor of a more experimental, more flexible fabrication device.

2.4 Engineering and Practicality check 2: hand-held

Figure 6 shows our second prototype. Like our first prototype, it used a plastic extruder. However, instead of the 3D printer's x/y/z actuation mechanism, this prototype was built around a hand-held plastic extruder pen, i.e., the prototype actuates the extruder using the user's hand, resulting in a human-assisted [23] fabrication system.

Figure 6: Second prototype

At the expense of additional user effort, the ad-hoc benefits of the human-assisted approach include: (1) fabrication directly onto objects in their environments, (2) a device 10x smaller than our first prototype, and (3) substantially faster fabrication (by integrating external objects and a coarser extruder).

Again we wrote an app around the device that provides users with solutions for common problems, and then tried to fabricate the list of desired objects from our

survey. We re-engineered all objects so as to produce as fast as possible. Our overall objective was to explore what objects we could make and where our system would fail in order to derive implications for mobile 3D printer hardware, like in the first iteration.

2.4.1 The second prototype

For this prototype, we modified an off-the-shelf plastic extruder pen (a 3Doodler 2.0, the3doodler.com) to allow for mobile use (Figure 6). It was outfitted with two 120mAH rechargeable batteries to provide 20 minutes of printing time, enough to build one large object or three smaller ones. (Other extruder pens, such as the Creopop (creopop.com) or Bondic (notaglue.com) could have been used as well).

Like our first prototype, plastic extruder pens are based on FDM, which allows them to print in any orientation as well as while shaking. The 3Doodler 2 is 16cm long, allowing it to fit into a coat pocket and is weighs 60 grams including one strand of filament.

The app lets users prototype directly on the phone's screen, utilizing the screen as the build platform to trace blueprints at actual scale. To assure adhesion between filament and screen we covered the screen with adhesive transparent plastic film (0,3mm rigid-PVC, anti-reflex).

The app provided users with solutions for the scenarios from our survey. However, this time the app did not return 3D models, but fabrication instruction to be executed by a human (similar to, for example, Instructables.com). The app used the same back-end as our previous app.

2.4.2 Walkthrough

The walkthrough shown in Figure 7 demonstrates the same scenario as before, i.e., how to re-attach a bike headlight using a fabricated hex key. (a) As with our first prototype, the user pulls out the phone and queries the app for "hex key". (b) The app returns multiple results from its online database, each result containing a sequence of instructions to be executed by a human. The first result requires a coin, which our user does not have on him. He thus picks the second hit. (c) The app displays a brief synopsis. The user inspects it briefly to get an overview of what is ahead and flicks to get started.

(d) The app displays the first fabrication step; it shows how to manufacture a truss-shaped handle for the hex key. The system shows the instructions directly on the screen, and the user fabricates the part by tracing it using the extruder pen. The instructions detail how to interlock the individual layers of material to maximize stability. Following additional instructions, the user completes the hex key by (e) molding a tip directly on the nut and (f) fusing it to the handle (details on these techniques below), attaches the light, and rides on.

2.5 Editor

As we already saw, instructions in our app are essentially sequences of photos, making them easy to create. To simplify the creation of instructions even further we

implemented the basic editor shown in Figure 7. (a) It offers templates for each of the individual fabrication processes and (b) helps users create instructions to be traced by allowing them to scale photographs with rulers and grid tools as reference, before (c) drawing the lines to be traced on top of the photo.

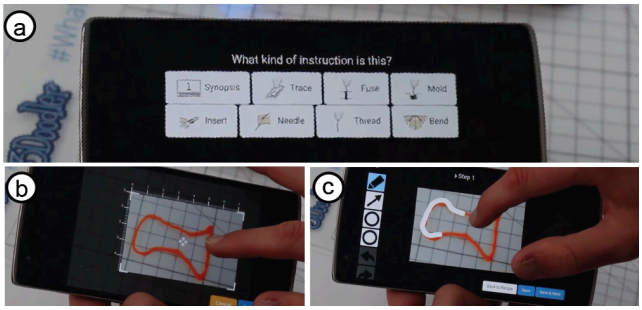


Figure 7: Editor to make new instructions

2.6 Contribution and Conclusions

Our main contribution is an exploration into the future of fabrication, in particular the vision of mobile fabrication. We explore this vision with two surveys, two simple hardware prototypes, matching custom apps that provide users with access to a solution database, engineering fabrication techniques specifically for these devices, including tracing and molding, and a user study conducted in situ on metro trains. All of this combined asks the question whether mobile fabrication will happen. We think it will.

3 Related Work

Our work builds on the work of HCI researchers who explored how to help non-engineers fabricate. We also build on humans assisted fabrication.

Fabrication systems that provide domain knowledge To help non-engineers engineer, researchers in HCI and graphics have developed tools that incorporate the required domain knowledge, such as interactive controls (Maker's Marks [14]), the dynamics of model airplanes (Pteromys [19]), the structural engineering of furniture (SketchChair [13]), or how to design enclosures (Enclosed [20]).

Design for fabrication on context With mobile fabrication, users create objects that are part of and connect to their context of use. Several research projects have investigated how to allow users to design objects in their context. Tactum, for example, lets users design bracelets directly on the wearer's arm [5]. CopyCAD [4]

brings physical objects virtually into the fabrication environment. MixFab [21] lets modeling and physical environment blend together. More recently Koyama et al. demonstrated how to generate geometry that connects 3D models to their context (AutoConnect [8]).

How-to instruction repositories We guide users in the fabrication process using how-to instructions. Several researchers studied the sharing [1] and evolution [10] of the communities surrounding such repositories. Recently, Torrey et al. [18] investigated searching for how-to instructions. The proposed instruction system builds on their findings by allowing users to filter by tool, providing a wide variety of keywords for every object, and by presenting instructions in a highly visual way. Two notable examples of commercial online repositories related to fabrication are Thingiverse.com and Instructables.com.

Human-assisted fabrication In "the wise chisel," Zoran et al. [25] provide a great over-view of handheld fabrication tools extended with guidance mechanisms. Free-D [24] elaborates on the concept of guidance by deactivating the tool when off-bounds. Similarly, Augmented Airbrush for Computer Aided Painting [16] applies this idea to airbrushes. D-coil [11] lets users fabricate 3D models out of wax.

While the systems above track users, Devendorf's being the machine [3] uses a form of human-assisted fabrication that does not require tracking. The system uses a laser pointer to point users to where to apply material, with the goal to make the fabrication process more expressive through human input and the resulting reduced precision.

The term human-assisted manufacturing was coined by Yoshida et al.; their system guides users through the fabrication of large objects by piling chopsticks using a chop-stick blower [23].

4 Future Work

My work continues to focus on filling in the open challenges of mobile fabrication. Currently we are working on developing small sized fabrication machines, sketching the trade-off between machine size and percentage of models that it can fabricate. In previous work, I looked into remixing as a strategy for 3D-modeling [12] which I consider the most advanced form of modeling on the go (seems unlikely that people will model from sratch on the go). There we took mechanisms and remixed then, in follow-up I want to study this more by making fully re-usable units from scratch. Grfater showed how to benefit from other people's models, but also identified the key issue of portability between fab machines. Which is another challenge I am attempting to tackle in the remainder of my PhD. All of these projects have relevance specifically to mobile fabrication but also beyond that in their own right.

References

[1] E. Buehler, S. Branham, A. Ali, J. J. Chang, M. K. Hofmann, A. Hurst, and S. K. Kane. "Sharing is caring: Assistive technology designs on thingiverse". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015, pages 525–534.

[2] X. Chen, S. Coros, J. Mankoff, and S. E. Hudson. "Encore: 3D printed augmentation of everyday objects with printed-over, affixed and interlocked attachments". In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 2015, pages 73–82.

[3] L. Devendorf and K. Ryokai. "Being the Machine: Reconfiguring Agency and Control in Hybrid Fabrication". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015, pages 2477–2486.

[4] S. Follmer, D. Carr, E. Lovell, and H. Ishii. "CopyCAD: remixing physical objects with copy and paste from the real world". In: *Adjunct proceedings of the 23nd annual ACM symposium on User interface software and technology*. 2010, pages 381–382.

[5] M. Gannon, T. Grossman, and G. Fitzmaurice. "Tactum: a skin-centric approach to digital design and fabrication". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015, pages 1779–1788.

[6] N. Gershenfeld. *Fab: the coming revolution on your desktop–from personal computers to personal fabrication*. Basic Books, 2008.

[7] J. Horvath. *Mastering 3D printing*. Apress, 2014.

[8] Y. Koyama, S. Sueda, E. Steinhardt, T. Igarashi, A. Shamir, and W. Matusik. "AutoConnect: computational design of 3D-printable connectors". In: *ACM Transactions on Graphics (TOG)* 34.6 (2015), page 231.

[9] S. Mueller, S. Im, S. Gurevich, A. Teibrich, L. Pfisterer, F. Guimbretière, and P. Baudisch. "WirePrint: 3D printed previews for fast prototyping". In: *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 2014, pages 273–280.

[10] L. Oehlberg, W. Willett, and W. E. Mackay. "Patterns of physical design remixing in online maker communities". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015, pages 639–648.

[11] H. Peng, A. Zoran, and F. V. Guimbretière. "D-coil: A hands-on approach to digital 3D models design". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015, pages 1807–1815.

[12] T. J. Roumen, W. Müller, and P. Baudisch. "Grafter: Remixing 3D-printed machines". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, page 63.

[13] G. Saul, M. Lau, J. Mitani, and T. Igarashi. "SketchChair: an all-in-one chair design system for end users". In: *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*. 2011, pages 73–80.

[14] V. Savage, S. Follmer, J. Li, and B. Hartmann. "Makers' Marks: Physical markup for designing and fabricating functional objects". In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 2015, pages 103–108.

[15] A. Schulz, A. Shamir, D. I. W. Levin, P. Sitthi-amorn, and W. Matusik. "Design and fabrication by example". In: *ACM Trans. Graph.* 33.4 (2014), 62:1–62:11. doi: 10.1145/2601097.2601127.

[16] R. Shilkrot, P. Maes, J. A. Paradiso, and A. Zoran. "Augmented airbrush for computer aided painting (CAP)". In: *ACM Transactions on Graphics (TOG)* 34.2 (2015), page 19.

[17] D. J. Solove. *The digital person: Technology and privacy in the information age*. NyU Press, 2004.

[18] C. Torrey, E. F. Churchill, and D. W. McDonald. "Learning how: the search for craft knowledge on the internet". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2009, pages 1371–1380.

[19] N. Umetani, Y. Koyama, R. Schmidt, and T. Igarashi. "Pteromys: interactive design and optimization of free-formed free-flight model airplanes". In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), page 65.

[20] C. Weichel, M. Lau, and H. Gellersen. "Enclosed: a component-centric interface for designing prototype enclosures". In: *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction*. 2013, pages 215–218.

[21] C. Weichel, M. Lau, D. Kim, N. Villar, and H. W. Gellersen. "MixFab: a mixed-reality environment for personal fabrication". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2014, pages 3855–3864.

[22] K. Willis, E. Brockmeyer, S. Hudson, and I. Poupyrev. "Printed optics: 3D printing of embedded optical elements for interactive devices". In: *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 2012, pages 589–598.

[23] H. Yoshida, T. Igarashi, Y. Obuchi, Y. Takami, J. Sato, M. Araki, M. Miki, K. Nagata, K. Sakai, and S. Igarashi. "Architecture-scale human-assisted additive manufacturing". In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), page 88.

[24] A. Zoran and J. A. Paradiso. "FreeD: a freehand digital sculpting tool". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2013, pages 2613–2616.

[25] A. Zoran, R. Shilkrot, P. Goyal, P. Maes, and J. A. Paradiso. "The wise chisel: The rise of the smart handheld tool". In: *IEEE Pervasive Computing* 13.3 (2014), pages 48–57.

Semantic Enrichment of Indoor 3D Point Cloud Models

Vladeta Stojanovic

Computer Graphics Systems
Hasso-Plattner-Institut
Vladeta.Stojanovic@hpi.de

The use of Building Information Modeling (BIM) for FM in the Operation and Maintenance (O&M) stages of the building life-cycle is intended to bridge the gap between operations and digital data, but lacks the functionality of assessing and forecasting the state of the built environment in real-time. To accommodate this, BIM data needs to be constantly updated with the current state of the built environment (*"as-is"* representation). Generation of *as-is* BIM data for a digital representation of a building is a labor intensive process. While some software applications offer a degree of automation for the generation of *as-is* BIM data, they can be impractical to use for routinely updating digital FM documentation. Current approaches for capturing the built environment, using consumer mobile devices methods, allow for routine capture of 3D point clouds that can be used as basis data for a Digital Twin (DT), along with existing BIM and FM documentation. 3D point clouds themselves do not contain any semantics or specific information about the building components they represent physically, but using machine learning methods they can be enhanced with semantics that would allow for reconstruction of *as-is* BIM and basis DT data. This research report present an overview of the current progress towards the development of a service-oriented platform for semantic enrichment of 3D point cloud representations and creation of basis data for *as-is* BIMs and DT's of indoor environments.

1 Introduction

With the current trend of adoption of ICT practices in the *Architecture, Engineering and Construction* (AEC) industry, the continuing integration of the BIM process for FM operations has created new opportunities and challenges [18]. One of the key challenges is updating and reflecting the changes in the BIM that represent the current state of the built environment, referred to *"as-is"* or *"as-built"* versus *"as-designed"* representation [10, 24]. Another key challenge is being able to monitor and forecast the current state of the built environment, especially if digital documentation is integrated with real-time or historic data that is used to provide feedback and enhance decision making [22]. Apart from BIM adoption for FM, concepts and practices from *Industry 4.0* are currently being evaluated and adopted for FM use [19]. The most important relative concept of this paradigm are *Digital Twin* (DT) representations. A DT is a digital duplicate of the physical environment, states and processes. While a BIM model contains *as-is* and *as-designed* data and representations, a DT can be used to assess the current state, and to potentially forecast the future state, using a digital duplicate of the built environment [7, 15]. The data used for a DT therefore needs to representative of both the static physical attributes, as well as the dynamic processes

and states of the built environment. In order to capture the current physical state of the built environment, 3D point clouds can be used as the basis for as-is BIM and DT representations [4]. This report provides a high-level overview concerning the design, development and testing of the main components of a service-oriented platform, used to generate semantically rich data from 3D point clouds. These semantically enriched 3D point cloud representations can be used as basis data for DTs. This allows for routine generation of semantically rich models for enhancing collaboration, decision making, and forecasting among FM stakeholders.

2 Overview of Methodology

The main focus of the presented research methodology is on semantic enrichment and integration of 3D point cloud representations of indoor environments for enhancing decision making concerning FM operations, particularly O&M operations. A 3D point cloud consists of what can be defined as *non-interpreted data* - data that is open to visual interpretation but does not have any semantics associated with it. 3D point clouds can be used by themselves to represent the current state of the physical environment for practical needs (e.g., assessment of space usage in a room), but for any further representations and assessment the 3D point cloud needs to be processed in order generate useful semantics. These can than be used as the basis for *as-is* BIMs or DTs. Although there is overlapping between a DT and BIM representations, for this research DTs are treated as a higher-level representation that includes both *as-designed* and *as-is* BIM data, and any other associated semantics (see Figure 1). A DT representation fuses these *as-designed* and *as-is* representations with additional information layers pertaining to the current state of the built environment (see Figure 2).

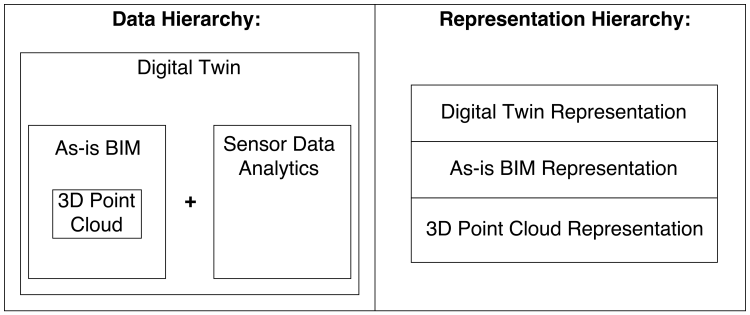


Figure 1: Data and representational hierarchy diagram for *as-is* BIMs and DTs

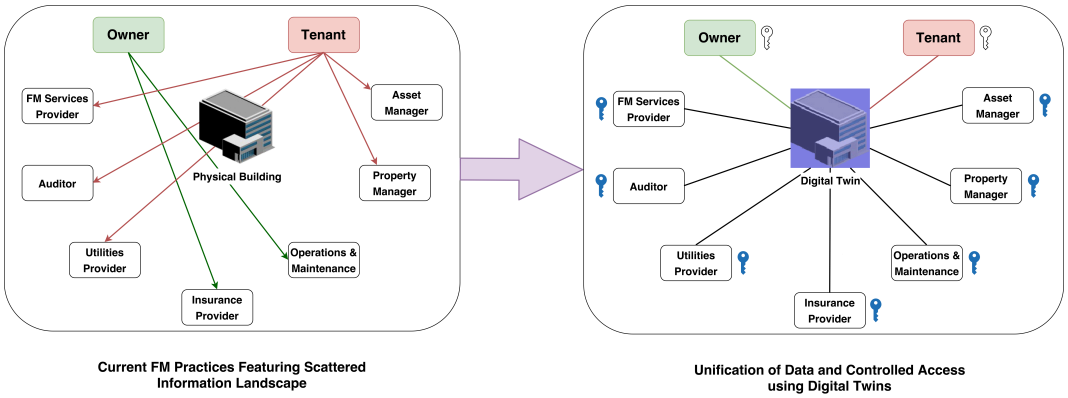


Figure 2: Illustrated example showing the current scattered state of FM operations in a typical building (left), in comparison to the unified and centralized access to operational status and digital data using a DT (right)

2.1 Point Cloud Acquisition

The main advantage of using consumer mobile devices (such as Google Tango specification compatible devices), is that they provide an affordable, flexible, and simple solution for capturing 3D point clouds of interiors in comparison to more expensive laser scanning devices [6]. Photometry-based processing methods can be implemented on mobile devices and allow the processing of a sequence of captured images to be converted into a point cloud representation. These 3D point cloud representations captured using mobile phones may feature lower fidelity in terms of visual details, but very detailed 3D representations of indoor environments are not generally required for routine FM applications. The visual representation described by a point cloud can be used to enhance FM practices and fulfills the need to be able to have up-to-date representations of interiors of a facility for enhancing and making informed decisions. The process for capturing point clouds is fairly straightforward, as the user has to simply walk around an interior space and capture the point cloud with the designated mobile tablet or phone device (see Figure 3). Once the scan has been generated, the resulting 3D point cloud needs to be filtered for overlapping duplicate points, and preferably sub-sampled to a reasonable degree in order to decrease processing time while preserving the visual fidelity of the 3D point cloud representation.

2.2 Reconstruction

As-is BIM reconstruction can generally be thought of as representing global characteristics of a built environment (e.g., comparison of as-built vs as-designed room layouts). Current general research in the AEC industry is focused on efficient geometry reconstruction at appropriate levels of detail using segmentation and geometry reconstruction methods. Before semantics can be added to the point cloud, the point cloud dataset has to be segmented in order to divide and mark the homogeneous

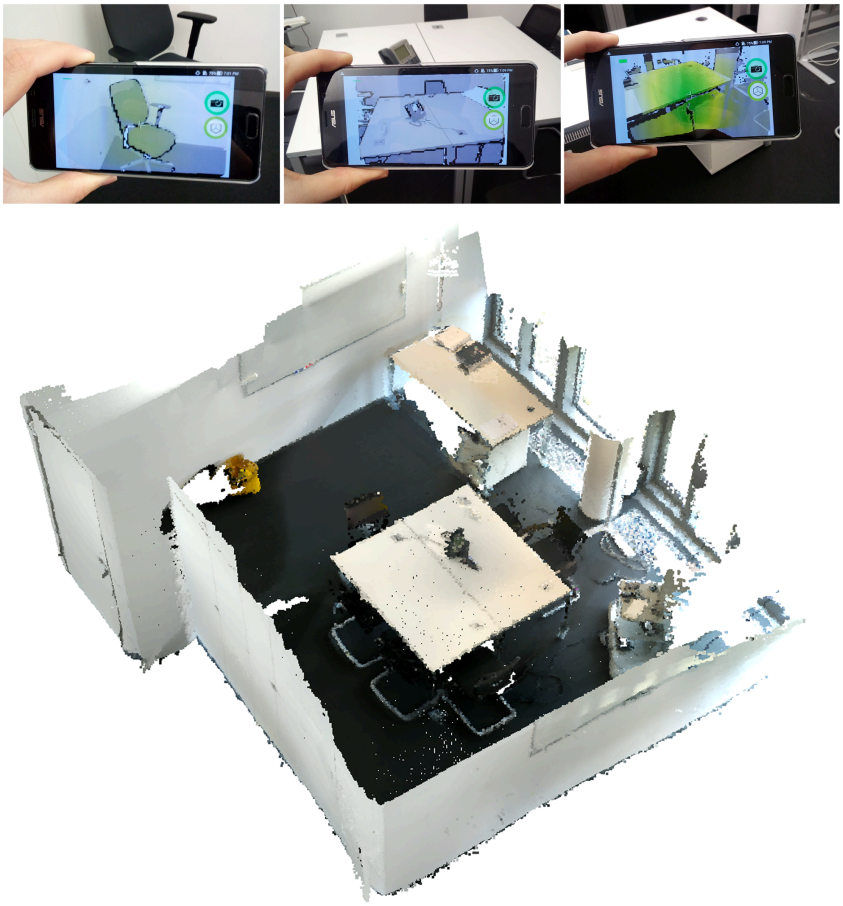
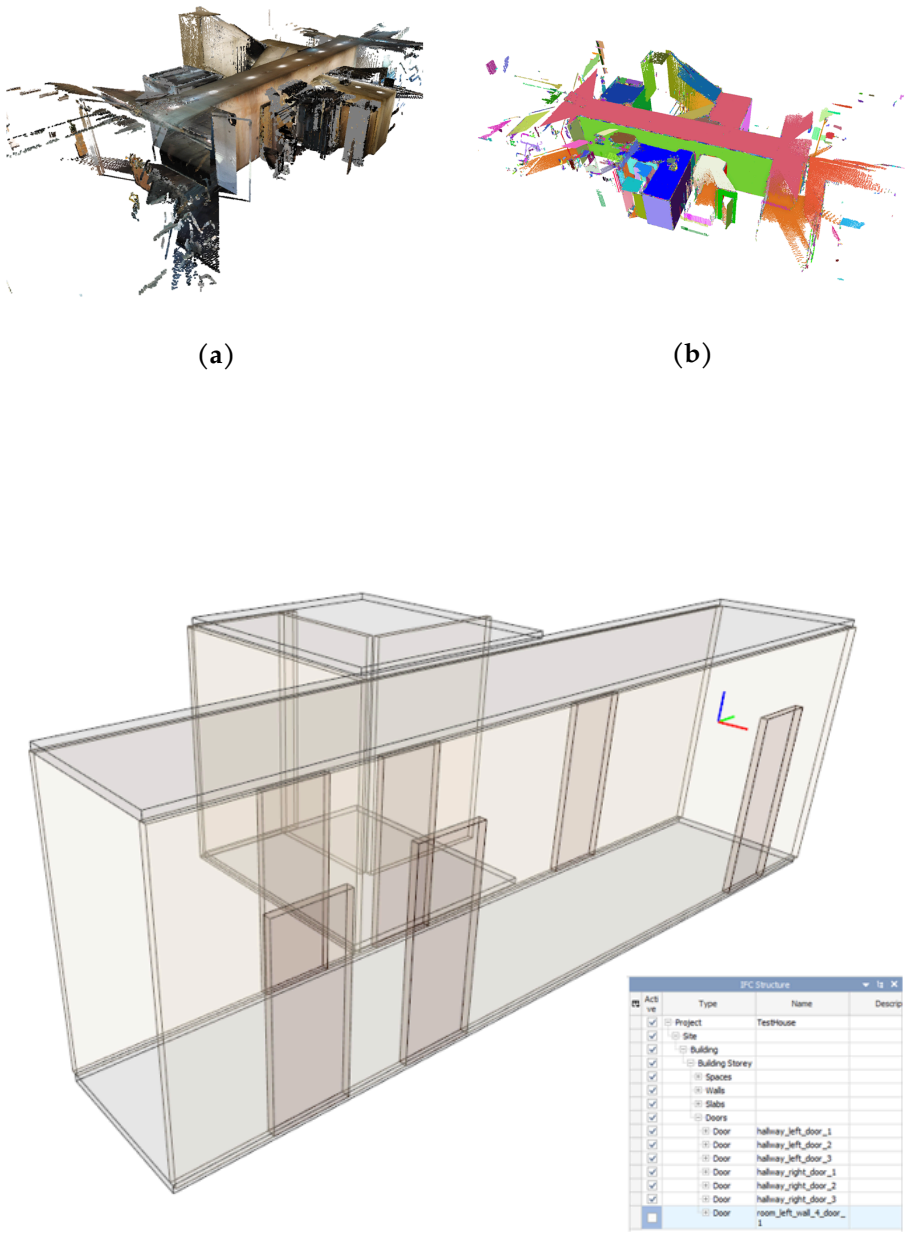


Figure 3: Acquisition of a 3D point cloud of office furniture using a Google Tango compatible mobile phone, and an example 3D point cloud of an office room generated using such a device

regions of point clusters, which allows for quicker identification of building features [14]. Segmentation can be either be done manually or can be automated by taking into account the spatial features of the point dataset (e.g., all point below a certain vertical axis coordinate can be considered as belonging to the floor cluster). Notable approaches for BIM reconstruction have been investigated by [1, 13, 21]. Segmented regions of the point cloud can then be used for reconstruction to a specified *Level of Detail* (LOD) BIM representation, usually stored in the *Industry Foundation Classes* (IFC) file format. This is illustrated in Figure 4.

Apart from reconstruction to BIM data, horizontally segmented point cloud representations of interiors can be used to generate 2D and 3D floor plan representations. Such representations can be very useful for analyzing space utilization, room layouts and emergency route planning. Research by [23] has presented a notable approach for generating room layouts for floor plan representations using segmented 3D point cloud representations of indoor buildings. Evaluation of boundary features derived from the spatial arrangement of point clouds can also be used to generate 2D and 3D



(a) (b)

(c)

Figure 4: a The original point cloud of an office hallway. b Shows the segmented point cloud that includes different colored planes to denote different point cluster orientations in comparison to other cluster regions in the point cloud data set, and c Shows the initial results of segmentation and reconstruction of an office hallway from a 3D point cloud at LOD-300.

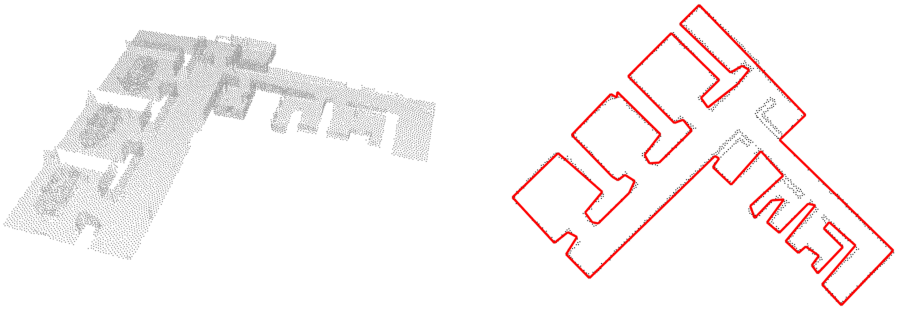
boundary approximation representations. Such representations can be useful when combined with existing 2D or 3D point cloud and CAD data in order to enhance 3D visualizations for initial assessment of the state of the building (e.g., energy usage, navigation planning and capacity simulation). Figure 5 illustrates an example of 2D and 3D floor plan boundary generation.

2.3 Classification

Segmented regions of the point cloud can be used to classify point cloud clusters as specific features of the built environment, such as furniture classification. In turn, this is the primary mechanism for semantic enrichment of segmented and/or reconstructed 3D point clouds. The process of classification is based on machine-learning and allows a *Convolutional Neural Network* (CNN) to recognize 2D or 3D spatial and visual features of a given point cluster [12, 16]. Training data used to train a CNN can either be segmented 3D point cloud clusters, or images of specific point cloud objects (e.g., images of point cloud representations of office furniture for example). Using 2D image classification methods, known as *"multi view classification"*, this process can be automated by having a trained CNN model classify point cloud clusters [2, 9]. a 3D pint cloud cloud can be partitioned using a 3D data structure such as an *Octree* or *kD-tree*, where each partitioned region can be used to generate classification images. These generated classification images can be classified in a service-oriented manner (see Section 2.5), and the classification results can be streamed back and associated with each partitioned region of the 3D point cloud. This allows for automated semantic-enrichment of the point cloud data, which in turn enables the point clouds to be used as basis data for *as-is* BIM or DT representations. The implementation of such a classification system is illustrated in Figure 6, and and example of such a classification result is shown in Figure 7.

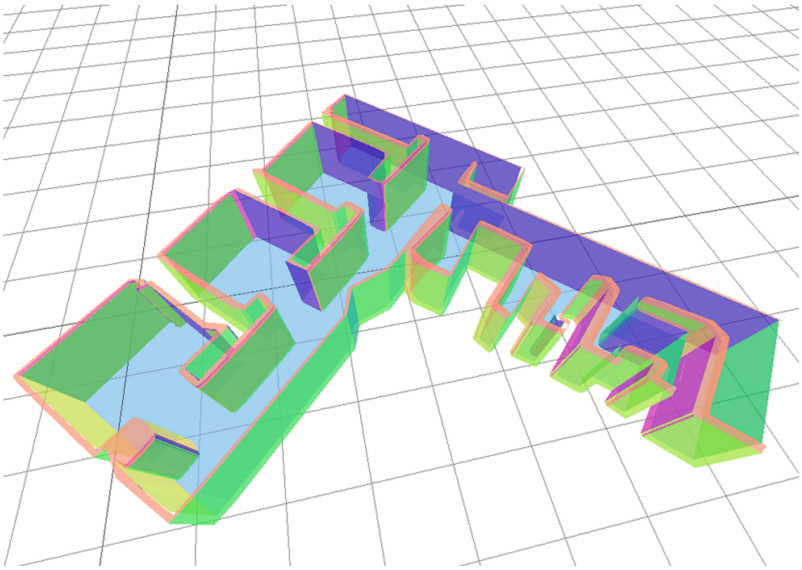
2.4 Visualization

The use of interactive 3D visualization can benefit FM stakeholder engagement by allowing real-time display and analysis of 2D and 3D visual outputs generated from the acquired data sources. Using modern computer graphics rendering approaches, complex visualizations can be presented to users in real-time on various configurations including commodity and older hardware. Service-based interactive visualization can enable streaming of complex visualization results to thin clients, such as smartphones and tablets [8]. With standardization of the WebGL 3D graphics API for most modern HTML5 compliant web browsers, it is possible to visualize in in real-time 3D various models and generated outputs for AEC applications. The use of high-level JavaScript frameworks for 3D web-based visualization, such as Three.js [3], allows for advanced real-time 3D features such as model loading (including PLY file format support for 3D point clouds), scene navigation, 3D data structures (octrees), and GPU-based rendering. Further, it features a flexible scene-management system where each component of the scene is added to a scene graph and accessed using a hierarchical function call system. One limit of Three.js for visualizing point



(**a**)

(**b**)



(**c**)

Figure 5: a Examples of floor plan generation outputs, using a 3D point cloud scan of an building interior. b A 2D vector contour around a regularized horizontal point cloud slice, and c an extruded 3D mesh representation of the primary 2D floor plan boundaries based on the generated vector contours.

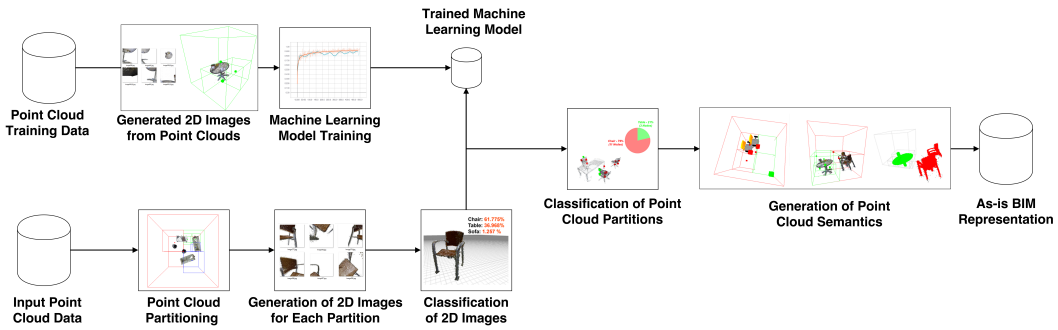


Figure 6: High-level overview of the presented point cloud classification approach using image-based classification

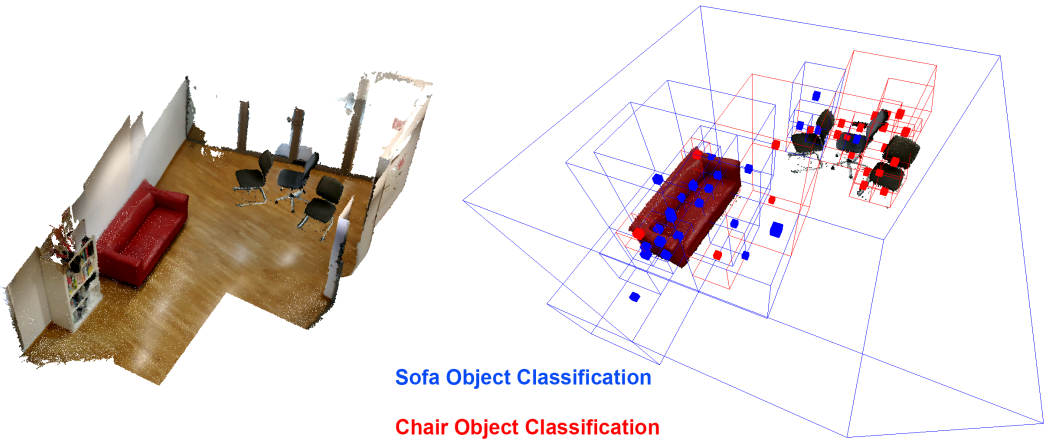


Figure 7: Classification results of an indoor office area containing a sofa and chairs. The left image shows the input point cloud and the segmented region that is to be classified. The right image shows the classification output. In the classification output, the blue cubes indicate possible spatial location of a sofa and the red cubes indicate possible spatial positions for chairs.

clouds is the lack of support for out-of-core rendering of massive amounts of point-cloud data, without resorting to the use of more sophisticated scene and memory management methods [17]. An example of the implemented web-based 3D visualization system used for display of semantically-enriched 3D point clouds of building interiors is illustrated in Figure 8.

2.5 Service Oriented Approach

Service-based interactive visualization can enable streaming of complex visualization results to thin clients, such as smartphones and tablets. Some mobile devices may be older generation and may not have the ability to process visualization data in real-time using their native hardware, thus data can be pre-processed and streamed from a server in real-time to the mobile device using an implemented service-oriented

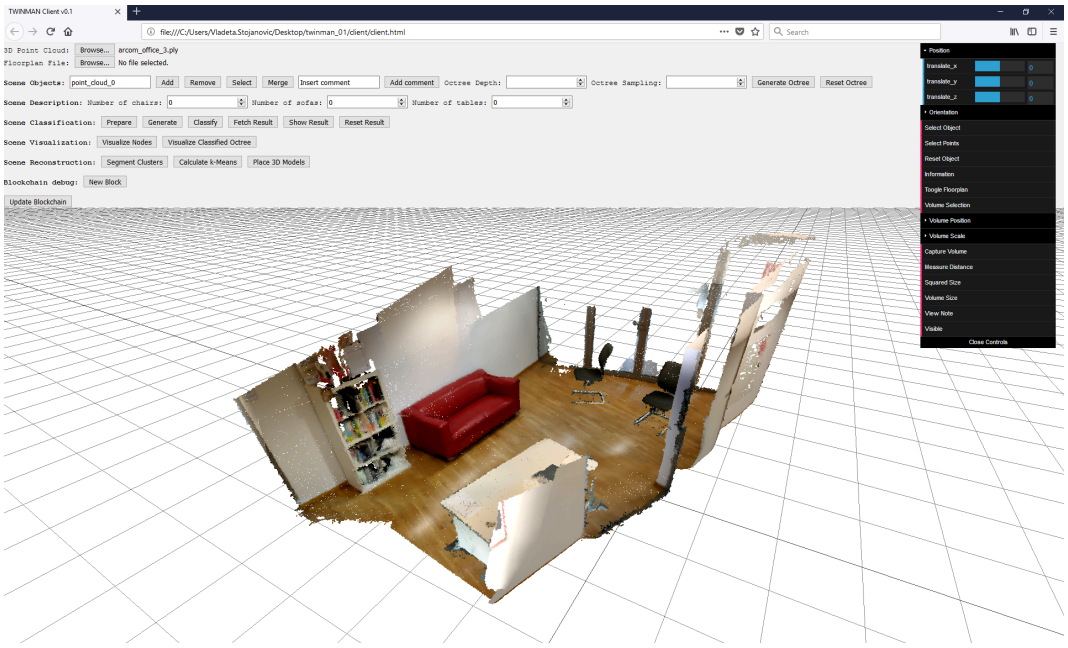


Figure 8: Example of a 3D scene being visualized using a prototype version of a web-based semantic enrichment tool for indoor point clouds

architecture solution [5]. Using a service oriented approach also allows for increased stakeholder collaboration, especially using 3D visualization of indoor environments [11]. The description of a BIM-based multidisciplinary collaboration platform, along with specific technical requirements, has been described in detail by [20]. A high-level overview of the implemented service oriented architecture for semantic enrichment of 3D point clouds using image-based classification is illustrated in Figure 9.

3 Conclusion and Outlook

Using a service-oriented paradigm, the scanned indoor environments obtained using consumer mobile devices can be reconstructed as semantically rich *as-is* 3D point cloud data, and as basis data for *as-is* BIMs and DTs. The described methodology provides a detailed overview of the processes required for acquiring, generating, and presenting this semantically rich 3D point cloud representations of indoor building environments. Current research efforts are focused on utilizing the presented methodology to generate results using real-world data, in order to effectively demonstrate the potential to increase engagement and enhance decision making for FM practitioners who are adopting BIM and Industry 4.0 practices.

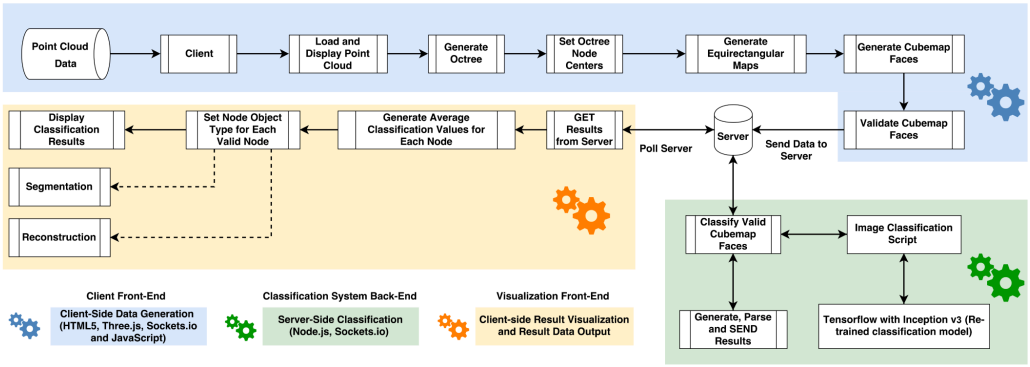


Figure 9: The presented client-server model showing the data flow, processes, and systems used for a service oriented implementation of a web-based semantic enrichment tool for indoor point clouds

References

[1] I. Anagnostopoulos, M. Belsky, and I. Brilakis. "Object boundaries and room detection in as-is BIM models from point cloud data". In: *Proceedings of the 16th International Conference on Computing in Civil and Building Engineering*. 2016, pages 6–8.

[2] K. Babacan, L. Chen, and G. Sohn. "Semantic segmentation of indoor point clouds using convolutional neural network". In: *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* 4 (2017).

[3] R. Cabello et al. *Three. js*. 2010. url: URL:%20https://github.%20com/mrdoob/three.js%20(last%20accessed%202018-10-18).

[4] S. Discher, R. Richter, M. Trapp, and J. Döllner. "Service-oriented processing and analysis of massive point clouds in geoinformation management". In: *Service oriented mapping: Changing paradigm in map production and geoinformation management*. 2018.

[5] J. Doellner, B. Hagedorn, and J. Klimke. "Server-based rendering of large 3D scenes for mobile devices using G-buffer cube maps". In: *Proceedings of the 17th International Conference on 3D Web Technology*. 2012, pages 97–100.

[6] M. Froehlich, S. Azhar, and M. Vanture. "An investigation of Google Tango® tablet for low cost 3D scanning". In: *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*. Volume 34. 2017.

[7] M. Grieves. "Digital twin: manufacturing excellence through virtual factory replication". In: *White paper* (2014).

[8] B. Hagedorn and J. Döllner. "High-level web service for 3D building information visualization and analysis". In: *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*. 2007, page 8.

[9] A. Ioannidou, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris. "Deep learning advances in computer vision with 3d data: A survey". In: *ACM Computing Surveys (CSUR)* 50.2 (2017), page 20.

[10] G. Kelly, M. Serginson, S. Lockley, N. Dawood, and M. Kassem. "BIM for facility management: a review and a case study investigating the value and challenges". In: *Proceedings of the 13th International Conference on Construction Applications of Virtual Reality*. 2013, pages 30–31.

[11] W.-L. Lee, M.-H. Tsai, C.-H. Yang, J.-R. Juang, and J.-Y. Su. "V3DM+: BIM interactive collaboration system for facility management". In: *Visualization in Engineering* 4.1 (2016), page 5.

[12] Y. Ma, B. Zheng, Y. Guo, Y. Lei, and J. Zhang. "Boosting multi-view convolutional neural networks for 3D object recognition via view saliency". In: *Chinese Conference on Image and Graphics Technologies*. 2017, pages 199–209.

[13] H. Macher, T. Landes, and P. Grussenmeyer. "From point clouds to building information models: 3D semi-automatic reconstruction of indoors of existing buildings". In: *Applied Sciences* 7.10 (2017), page 1030.

[14] A. Nguyen and B. Le. "3D point cloud segmentation: A survey". In: *RAM*. 2013, pages 225–230.

[15] J. Posada, C. Toro, I. Barandiaran, D. Oyarzun, D. Stricker, R. De Amicis, E. B. Pinto, P. Eisert, J. Döllner, and I. Vallarino. "Visual computing as a key enabling technology for industrie 4.0 and industrial internet". In: *IEEE computer graphics and applications* 35.2 (2015), pages 26–40.

[16] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *IEEE Proc. Computer Vision and Pattern Recognition (CVPR)* 1.2 (2017), page 4.

[17] R. Richter, S. Discher, and J. Döllner. "Out-of-core visualization of classified 3d point clouds". In: *3D Geoinformation Science*. 2015, pages 227–242.

[18] K. Roper and R. Payant. *The facility management handbook*. Amacom, 2014.

[19] SAP. *Leonardo Innovation Services Showroom*. 2018. url: https://discover.sap.com/innovation-showroom-demo/en-us/digital-twin.html%20(last%20accessed%202018-10-18).

[20] V. Singh, N. Gu, and X. Wang. "A theoretical framework of a BIM-based multidisciplinary collaboration platform". In: *Automation in construction* 20.2 (2011), pages 134–144.

[21] P. Tang, D. Huber, B. Akinci, R. Lipman, and A. Lytle. "Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques". In: *Automation in construction* 19.7 (2010), pages 829–843.

[22] P. Teicholz et al. *BIM for facility managers*. John Wiley & Sons, 2013.

[23] E. Turner and A. Zakhor. "Floor plan generation and room labeling of indoor environments from laser range data". In: *2014 International Conference on Computer Graphics Theory and Applications (GRAPP)*. 2014, pages 1–12.

[24] R. Volk, J. Stengel, and F. Schultmann. "Building Information Modeling (BIM) for existing buildings – Literature review and future needs". In: *Automation in construction* 38 (2014), pages 109–127.

Multi-Source 3D Geodata Analysis

Johannes Wolf

Computer Graphics Systems Group
Hasso Plattner Institute, Faculty of Digital Engineering, University of Potsdam, Germany
johannes.wolf@hpi.de

Different sensory equipment can measure geospatial information of our environment. Ground-penetrating 2D radar scans are captured for examination of surface conditions and below-ground variations such as lowerings and developing potholes in road areas. 3D point clouds captured above ground provide a precise digital representation of the ground's surface and the surrounding environment. Panoramic photos add color information and an easy to interpret overview of a certain area. When combining different data sources for the same area, a combined visualization is a valuable tool for infrastructure maintenance tasks. This report presents currently ongoing research in visualization techniques for combined visual exploration of multiple data sources and analyses that benefit from the different data characteristics.

1 Introduction

Our physical environment can be measured in many different ways. Varying sensors capture specific features of the surrounding world and together give us a complete overview of scanned areas. The number of applications which have a need for such geographical base data is continuously increasing. Use cases such as urban planning and development as well as environmental monitoring are dependant on a broad data basis.

While focusing on the analysis of 3D point clouds as central element for many applications, additional data sources such as panoramic photos and ground penetrating radar can improve the insights gained from the data.

3D point clouds can be seen as highly detailed representations of the scanned environment and are used as "digital twins". They are a collection of unstructured, unsorted, independent points in three-dimensional space with an arbitrary number of attributes attached to each point [16]. In contrast to 3D city models [4], 3D point clouds provide a "raw" view of the captured data and can be used for further processing.

Depending on the capturing technique, 3D point clouds often do not contain color information (see 2.1). For improved visualization in a more familiar way, color information can be gained from panoramic photos captured in parallel to the 3D point cloud creation. They can also be used for detailed analyses of, e.g., individual road signs or similar assets. Combining this information with the 3D point cloud data is a current area of research.

Both, 3D point clouds and panoramic photos, only represent the above-ground world. For applications such as road construction and archeological surveys, infor-

mation about the conditions below ground are of great interest. Ground penetrating radar can be used to gain that information and is the third data source that is to be integrated into a common data analysis and exploration framework.

We are working on different visualization and analysis techniques which combine the aforementioned data sources for a better understanding of the environment and faster data aggregation to create insights [19].

2 Data Sources

2.1 3D Point Clouds

3D point clouds can be captured by photogrammetric approaches, which derive the points from a series of pictures taken from different angles. A more precise approach uses laser scanners, resulting in a lot more detailled 3D point clouds. Many of our use cases are using these laser scanned data sets, where each point after capturing has only a three-dimensional position and an intensity value, representing to what extend the laser beam was reflected from the object it hit.

Point clouds can also be distinguished by the perspective from which they have been captured. Data from airplanes and drones results in aerial data that characteristically has a more or less homogeneous point density and provides data from the top-down view onto roofs, streets, and vegetation. A different approach to scan the environment is via mobile mapping [11, 15]. Mobile carrier platforms like cars or trains are used to capture entire infrastructure networks [13]. The resulting datasets contain information from ground perspective including vehicles, building façades, trees, road signs, train signals, and many more [1, 8].

The point clouds we are processing can contain billions of points, resulting in several terabytes of data, raising the need for efficient spatial data structures and processing algorithms.

2.2 Panoramic Photos

Due to the physical process of capturing a 3D point cloud with a laser scanner, captured points do have no color information. It is common to take panoramic photos in parallel, to colorize the point cloud with data from these images in postprocessing. While this doesn't result in perfectly colored point clouds due to perspective differences (see Figure 1), it can improve the visual data inspection for some use cases.

Panoramic photos are usually created from a number of individual photos taken in different directions which are then merged into a combined photo containing a full 360°view. Figure 2 shows an examplary panoramic photo projected onto a plane.

In situations where users want to get a detailled view of a certain area, panoramic photos can be used for a fast insight. Applications need to integrate panoramic photos

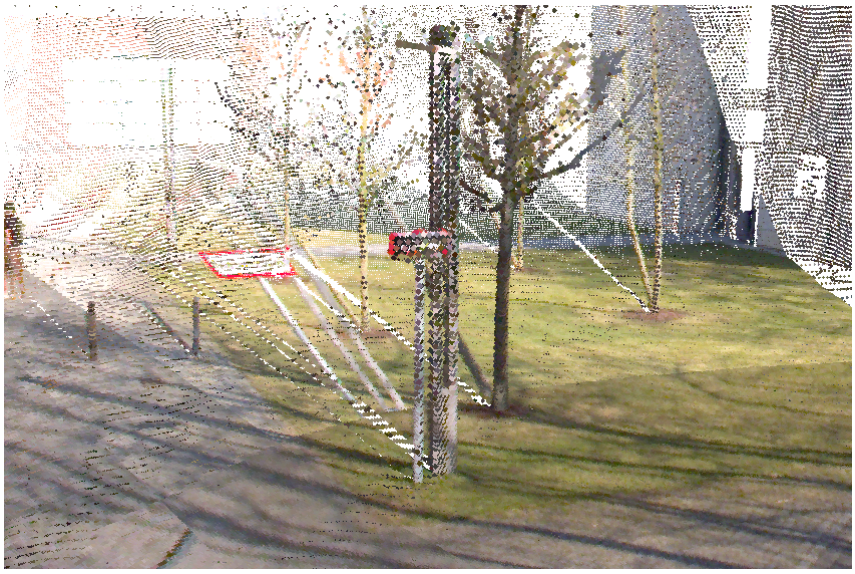


Figure 1: Visualization of a colored 3D point cloud with visible color mapping issues on the ground behind the red-bordered sign

Figure 2: Panoramic photo projected onto plane

in a supporting way, allowing for easy access to the region of the closest image at a certain point of interest.

2.3 Ground Penetrating Radar (GPR)

Ground penetrating radar (GPR) has been used for below-ground analyses for several decades. Ground penetrating radar scanners can measure material properties several meters below ground, creating insights about the non-visible foundation of roads and pathways [3]. A typical visualization of a ground penetrating radar data B-scan is shown in Figure 3. Ground penetrating radar scanners can also easily be mounted onto scanning vehicles, adding an additional data source for the region not accessible by LiDAR scanning [12].

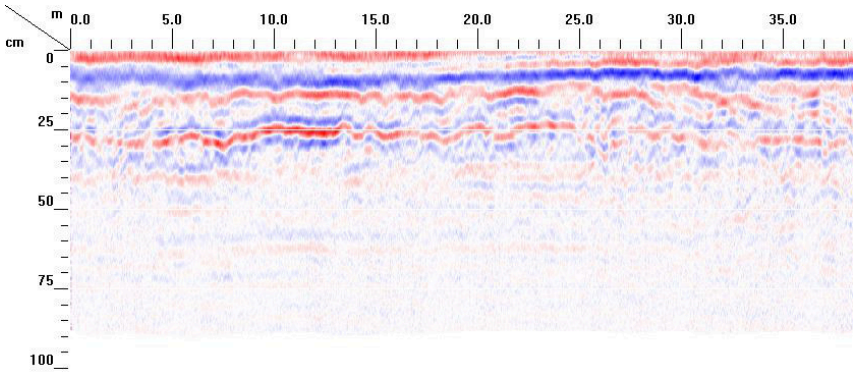


Figure 3: Visualization of a ground penetrating radar data B-scan

3 Data combination

3.1 3D Point Cloud and Panoramic Photos

3.1.1 Visualization

Panoramic photos are often placed into a spatial context by visualizing their locations on a map, like in Figure 4.

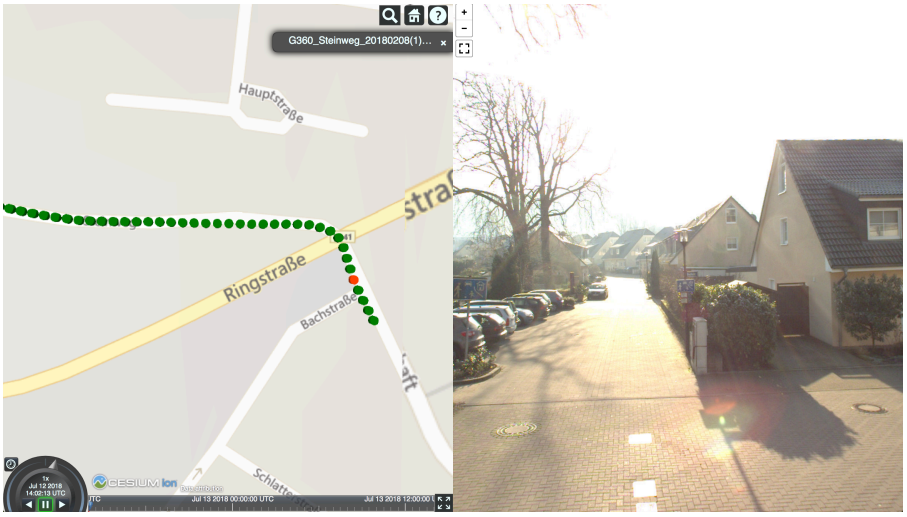


Figure 4: Rectangular section from a spheric projection of a panoramic photo (right) placed next to a map showing its location (left)

In a similar way, we are currently integrating panoramic photos directly into 3D point clouds. Areas of interest are blending between these two data sources, transitioning from one photo to the next in a capturing series and the respective movement within the point cloud.

3.1.2 Analysis

The panoramic images can be used for a more detailed analysis of objects if the geometric information is not sufficient, e.g., to identify road signs. The position of a sign can be analyzed in the 3D point cloud data and its type can be found by applying image recognition on the respective area in a suitable panoramic photo.

Additionally, 3D point clouds colored by the use of panoramic photos can be better analyzed in some situations, because the color for each point can be used as an additional input attribute for different metrics and analysis steps.

3.2 3D Point Cloud and GPR

3.2.1 Visualization

Giannopoulos describes how ground penetrating radar data can be visualized [7]. In addition to two-dimensional profiles, he shows an example for a three-dimensional visualization by rendering the three principal planes of a cuboid which is holding the data. Usually, two-dimensional ground penetrating radar profiles are shown individually, but multiple scans can be placed next to each other to create spatial feeling for the data [6].

We developed a GPR data visualization inside the 3D point cloud environment by projecting GPR data B-scans onto the captured GPS trajectory of the tracking vehicle. Because the visibility of inner B-scans is limited, the user can hide individual B-scans for a clear view onto the others.

For a better spatial impression, a textured cuboid-like structure is rendered onto the GPS trajectory, as shown in Figure 5. This structure covers the amount of space scanned by the GPR sensors. To fill the area in between the B-scans, their values are interpolated. A 3D-texturing approach guarantees the possibility of slicing the cuboid – both vertically and horizontally – as well as moving it along the trajectory.

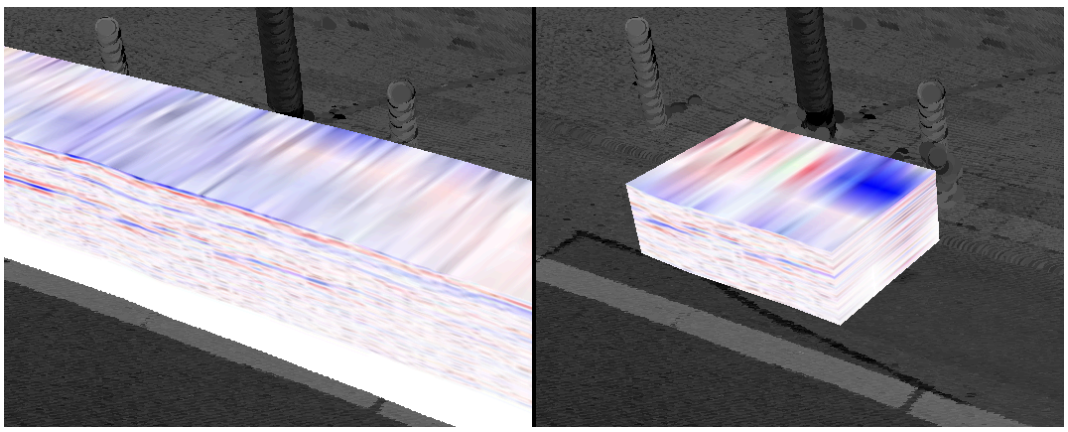


Figure 5: Cuboid rendered onto the GPS trajectory (left). Vertical and horizontal slicing can be used to explore data inside the cuboid (right).

We decided to raise the GPR data cuboid above the surface instead of rendering it in its accurate position below ground for better visibility and usability. To keep the spatial context, the corresponding points from the 3D point cloud located above the cuboid are translated alongside and highlighted for a better contrast to the remaining points (Figure 6).

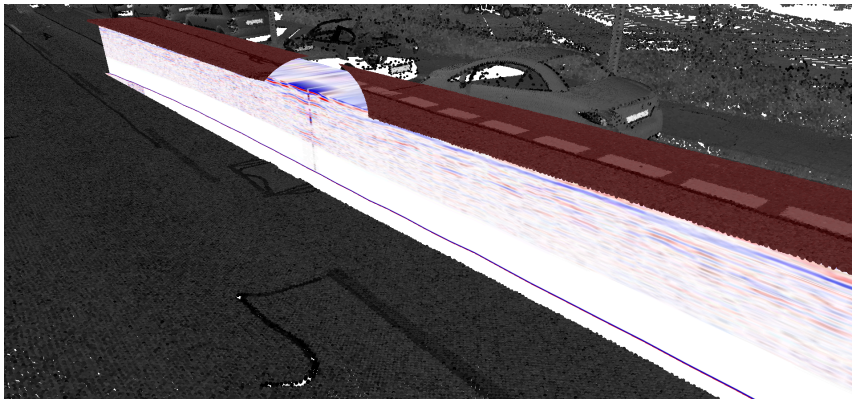


Figure 6: Points under the cuboid are elevated above it and highlighted. A magic lens shows the cuboid's surface below the points.

Furthermore, a magic lens approach has been implemented, which hides the points around the cursor and enables a direct view onto the cuboid's surface. The visualization can be switched to instead only show the points above the cuboid around the cursor, for focusing on the GPR data, while keeping once more the spatial context.

When presenting the prototypical implementation to experts in the field of GPR evaluation and city infrastructure planning, they approved of the visualization and stated it would greatly improve current workflows. We re currently working on additional analyses to highlight, e.g., anomalies in the data (see 3.2.2).

3.2.2 Analysis

The combination of above-ground 3D point clouds and below-ground 2D radar scans enables a more extensive analysis of road environments by using two combined sources instead of evaluating each on their own. A common use case for ground penetrating radar data inspection is to find certain areas with increased chance of developing pot-holes [9]. By adding road surface information from the 3D point cloud, false positives like manholes can easily be distinguished from other anomalies in the road's subsoil. Use cases are widely spread: The data can, e.g., be used in post-earthquake situations for ground analysis [14] as well as for general road surface modelling [10].

Benedetto et al. give an overview of how ground penetrating radar can be used for road inspections [2]. They discuss in detail which processing techniques can be used to analyze the pavement condition. Evans et al. summarize the abilities of ground penetrating radar for general pavement investigations [5]. Saarenketo and

Scullion explicitly list the localization of sinkholes and crack growth monitoring in their report about ground penetrating radar applications on roads and highways [17]. They further describe possible soil and road structure evaluations and required data interpretation techniques [18].

The described threshold and amplification manipulation assist during visually identifying anomalies in the GPR data. An extended automated analysis could help to find anomalous regions in the data by highlighting them for the user during inspection, which might further ease to find locations of interest.

Another possible improvement is an automated object detection within the 3D point cloud data that would enable the detection of manholes, so anomalies located closely to them could already be respectively tagged in the visualization. Other typical objects which can be filtered and tagged in a similar way are streetcar tracks and curbstones.

4 Conclusions and Future Work

GPR scans can be visualized together with 3D point cloud data for road inspection and the combination offers added value to typical users. A combined visualization in the same viewing space enables comparison between both data sources for easier evaluation. Anomalies in the GPR data can be compared with the 3D point cloud to detect irregularities visible from above ground. Manhole covers and gullies can easily be identified within the 3D point cloud and their location can be taken into account when evaluating anomalies from the GPR data.

Cropping the ground penetrating radar B-scans to a certain area of interest in length and height allows users to focus on details in a small area and avoids occluding too much context information. Individual ground penetrating radar B-scans can be enabled or disabled for the visualization to further decrease visual clutter of currently unneeded data, especially with regard to ground penetrating radar B-scans scanned with different frequencies.

An implementation of level of detail approaches for the GPR data might improve handling even larger data sets.

Visualizing the ground penetrating radar B-scans slightly raised above ground, also raising the ground points from the 3D point cloud in close proximity with them, enables a less occluded view onto the ground penetrating radar B-scans. By hiding those elevated points in a small region around the cursor, the top plane of the ground penetrating radar cuboid can still be inspected.

Also, the visualization of ground penetrating radar B-scans from different scans in areas of intersections and for roads with multiple scanning runs holds potential for further development.

References

[1] S. Becker and N. Haala. "Combined feature extraction for façade reconstruction". In: *Proceedings of the ISPRS Workshop Laser Scanning*. 2007, pages 241–247.

[2] A. Benedetto, F. Tosti, L. B. Ciampoli, and F. D'Amico. "An overview of ground-penetrating radar signal processing techniques for road inspections". In: *Signal Processing* 132 (2017), pages 201–209.

[3] J. L. Davis and A. P. Annan. "Ground-penetrating radar for high-resolution mapping of soil and rock stratigraphy". In: *Geophysical prospecting* 37.5 (1989), pages 531–551.

[4] J. Döllner, H. Buchholz, M. Nienhaus, and F. Kirsch. "Illustrative visualization of 3D city models". In: *Electronic Imaging 2005*. International Society for Optics and Photonics. 2005, pages 42–51.

[5] R. D. Evans, M. W. Frost, M. Stonecliffe-Jones, and N. Dixon. *A review of pavement assessment using ground penetrating radar (GPR)*. Technical report. Universty of Birmingham and EuroGPR, 2008.

[6] *Geo Radar: 3D and GPR*. url: http://www.geo-radar.pl/en/methods/georadar/3d/ (last accessed 2018-06-25).

[7] A. Giannopoulos. "Modelling ground penetrating radar by GprMax". In: *Construction and building materials* 19.10 (2005), pages 755–762.

[8] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. "Surface reconstruction from unorganized points". In: *Computer Graphics (SIGGRAPH '92 Proceedings)* 26.2 (1992), pages 71–78.

[9] D. R. Huston, N. V. Pelczarski, B. Esser, and K. R. Maser. "Damage detection in roadways with ground penetrating radar". In: *Eighth International Conference on Ground Penetrating Radar*. Volume 4084. 2000, pages 91–95.

[10] A. Jaakkola, J. Hyyppä, H. Hyyppä, and A. Kukko. "Retrieval algorithms for road surface modelling using laser-based mobile mapping". In: *Sensors* 8.9 (2008), pages 5238–5249.

[11] R. Li. "Mobile mapping: An emerging technology for spatial data acquisition". In: *Photogrammetric Engineering and Remote Sensing* 63.9 (1997), pages 1085–1092.

[12] *Mobile GPR*. url: https://www.catsurveys.com/Services/MGPR (last accessed 2018-06-26).

[13] Nusantara Secom InfoTech. *Road condition laser surveying (MMS: mobile mapping system)*. url: http://www.nsi.co.id/road-condition-laser-surveying-mms-mobile-mapping-system (last accessed 2017-09-21).

[14] M. J. Olsen and R. Kayen. "Post-earthquake and tsunami 3D laser scanning forensic investigations". In: *Forensic Engineering 2012: Gateway to a Safer Tomorrow*. 2013, pages 477–486.

[15] I. Puente, H. González-Jorge, J. Martínez-Sánchez, and P. Arias. "Review of mobile mapping and surveying technologies". In: *Measurement* 46.7 (2013), pages 2127–2145.

[16] R. Richter, M. Behrens, and J. Döllner. "Object Class Segmentation of Massive 3D Point Clouds of Urban Areas Using Point Cloud Topology". In: *International Journal of Remote Sensing* 34.23 (2013), pages 8408–8424.

[17] T. Saarenketo and T. Scullion. *Ground penetrating radar applications on roads and highways*. Technical report. Texas A&M University System; Arlington, TX, United States, 1994.

[18] T. Saarenketo and T. Scullion. "Road evaluation with ground penetrating radar". In: *Journal of applied geophysics* 43.2-4 (2000), pages 119–138.

[19] J. Wolf, S. Discher, L. Masopust, S. Schulz, R. Richter, and J. Döllner. "Combined visual exploration of 2D ground radar and 3D point cloud data for road environments". In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-4/W10 (2018), pages 231–236.

Aktuelle Technische Berichte des Hasso-Plattner-Instituts

Band	ISBN	Titel	Autoren / Redaktion
128	978-3-86956-464-7	**The font engineering platform : collaborative font creation in a self-supporting programming environment**	Tom Beckmann, Justus Hildebrand, Corinna Jaschek, Eva Krebs, Alexander Löser, Marcel Taeumel, Tobias Pape, Lasse Fister, Robert Hirschfeld
127	978-3-86956-463-0	**Metric temporal graph logic over typed attributed graphs : extended version**	Holger Giese, Maria Maximova, Lucas Sakizloglou, Sven Schneider
126	978-3-86956-462-3	**A logic-based incremental approach to graph repair**	Sven Schneider, Leen Lambers, Fernando Orejas
125	978-3-86956-453-1	**Die HPI Schul-Cloud : Roll-Out einer Cloud-Architektur für Schulen in Deutschland**	Christoph Meinel, Jan Renz, Matthias Luderich, Vivien Malyska, Konstantin Kaiser, Arne Oberländer
124	978-3-86956-441-8	**Blockchain : hype or innovation**	Christoph Meinel, Tatiana Gayvoronskaya, Maxim Schnjakin
123	978-3-86956-433-3	**Metric Temporal Graph Logic over Typed Attributed Graphs**	Holger Giese, Maria Maximova, Lucas Sakizloglou, Sven Schneider
122	978-3-86956-432-6	**Proceedings of the Fifth HPI Cloud Symposium "Operating the Cloud“ 2017**	Estee van der Walt, Isaac Odun-Ayo, Matthias Bastian, Mohamed Esam Eldin Elsaid
121	978-3-86956-430-2	**Towards version control in object-based systems**	Jakob Reschke, Marcel Taeumel, Tobias Pape, Fabio Niephaus, Robert Hirschfeld
120	978-3-86956-422-7	**Squimera : a live, Smalltalk-based IDE for dynamic programming languages**	Fabio Niephaus, Tim Felgentreff, Robert Hirschfeld
119	978-3-86956-406-7	**k-Inductive invariant Checking for Graph Transformation Systems**	Johannes Dyck, Holger Giese
118	978-3-86956-405-0	**Probabilistic timed graph transformation systems**	Maria Maximova, Holger Giese, Christian Krause
117	978-3-86956-401-2	**Proceedings of the Fourth HPI Cloud Symposium "Operating the Cloud” 2016**	Stefan Klauck, Fabian Maschler, Karsten Tausche
116	978-3-86956-397-8	**Die Cloud für Schulen in Deutschland : Konzept und Pilotierung der Schul-Cloud**	Jan Renz, Catrina Grella, Nils Karn, Christiane Hagedorn, Christoph Meinel

ISBN 978-3-86956-465-4
ISSN 1613-5652