Tiziana Margaria | Bernhard Steffen | Christian Kubczak

Evolution support in heterogeneous service-oriented landscapes

Suggested citation referring to the original publication:
Journal of the Brazilian Computer Society 16 (2010) pp. 35–47
DOI https://doi.org/10.1007/s13173-010-0004-4
ISSN (print) 0104-6500
ISSN (online) 1678-4804

J Braz Comput Soc (2010) 16: 35–47
DOI 10.1007/s13173-010-0004-4

INVITED PAPER

Evolution support in heterogeneous service-oriented landscapes

What if your mediation scenario for the semantic web service challenge suddenly runs SAP?

Tiziana Margaria · Bernhard Steffen · Christian Kubczak

Received: 5 March 2010 / Published online: 29 April 2010

Abstract We present an approach that provides automatic or semi-automatic support for evolution and change management in heterogeneous legacy landscapes where (1) legacy heterogeneous, possibly distributed platforms are integrated in a service oriented fashion, (2) the coordination of functionality is provided at the service level, through orchestration, (3) compliance and correctness are provided through policies and business rules, (4) evolution and correctness-by-design are supported by the eXtreme Model Driven Development paradigm (XMDD) offered by the jABC (Margaria and Steffen in Annu. Rev. Commun. 57, 2004)—the model-driven service oriented development platform we use here for integration, design, evolution, and governance. The artifacts are here semantically enriched, so that automatic synthesis plugins can field the vision of Enterprise Physics: knowledge driven business process development for the end user.

We demonstrate this vision along a concrete case study that became over the past three years a benchmark for Semantic Web Service discovery and mediation. We enhance the Mediation Scenario of the Semantic Web Service Challenge along the 2 central evolution paradigms that occur in practice: (a) Platform migration: platform substitution of a legacy system by an ERP system and (b) Backend extension: extension of the legacy Customer Relationship Management (CRM) and Order Management System (OMS) backends via an additional ERP layer.

Keywords Evolving systems · Semantic web services · Service mediation · Web services · SOA

T. Margaria (✉)
Chair of Service and Software Engineering, Universität Potsdam, 14482 Potsdam, Germany
e-mail: margaria@cs.uni-potsdam.de

B. Steffen · C. Kubczak
Chair of Programming System, TU Dortmund, 44227 Dortmund, Germany

B. Steffen
e-mail: steffen@cs.uni-dortmund.de

C. Kubczak
e-mail: christian.kubczak@cs.uni-dortmund.de

1 Introduction

Especially in the area of enterprise application integration the new possibilities opened by the Service Oriented Computing paradigm, applied to web services, are appealing: the idea of service-oriented computing and the envisioned availability of thousands of services to be leveraged to provide agile business processes is a vision shared by all major players in the enterprise software market and by their customers.

A concrete realization will therefore likely base on such an SOA/SOC paradigm, as outlined for example in the Policy Oriented Enterprise Management approach of [5], and substantiated in the *one thing approach* proposed by [19] conceptually and described in [20] from the methodological and realizational point of view.

However, service oriented applications are still described merely in terms of the service's signatures, as in current WSDL. The lack of rich information on the kind of service offered, and under which circumstances it is applicable and/or adequate is largely responsible for the fact that composite services and applications that use services are typically created and maintained manually. Semantic technology, of the kind that enhances the service descriptions by annotations that precise its behavior, for a certain application domain, beyond the signature information, is a minimum requirement. Semantics is understood in this context

as extended descriptions by means, e.g. of ontologies in the maximal case, or at least of some pre- and post-conditions. If this kind of semantics were carefully designed and automatically supported by tools, the tasks of service discovery, selection, negotiation, and binding could be automated, lifting service-oriented applications to a new level of adaptability and robustness.

There, semantic issues play a central role in two directions:

- expressing the *purpose* of the compound service, i.e.

  What should the compound service achieve?

- expressing the *suitability* of the single services that appear in the service composition, i.e.

  Once it is clear *how* to orchestrate single service functionalities to achieve the described purpose, *what* (available) services are adequate to provide the functionalities required in the orchestration?

This reminds strongly of the question *From the How to the What* we addressed at VSTTE 2005 in Zürich [15], where we considered the VSTTE Grand Challenge under a very specific (and Service-Orientation friendly) perspective: the enabling of application experts without programming knowledge to reliably model their processes/applications in a fashion that allows for a subsequent automatic realization on a given platform. This goal, which aims at simplifying the tasks of the many at the cost of ambitious and laborious tasks for the few, adds a new dimension to the techniques and concepts aimed at by the Grand Challenge: the application-specific design of platforms tailored for the intended goal. We were convinced already then that the outlined perspective provides a realistic and economically important milestone for the Grand Challenge.

The SWS Challenge [1] addresses exactly this goal, albeit in a web service context instead of general programming:

- how can one specify, as clearly and declaratively as possible, the *What's* in the previous two questions, and
- how can one achieve as automatically, adaptably, and robustly as possible the implied *How's*, concerning *composition* (orchestration) and *matchmaking* for the single services (or components).

In fact, many different approaches to semantic Web service descriptions are already available, and many frameworks are built around them, yet a common understanding, evaluation scheme, and testbed to compare and classify these frameworks in terms of their abilities and shortcomings is still missing.

The purpose of the *Semantic Web Service Challenge* [1] was precisely to develop the lacking common understanding of the various technologies intended to facilitate the automation of mediation, choreography and discovery for web services using semantic annotations. This was meant to explore trade-offs among existing approaches, reveal their strengths and weaknesses, and aspects of the problem space that are not yet covered.

In [22], we examined the concrete settings, the dimensions of complexity that appear in the Challenge, and reflected on the essence of the observations so far. More detailed information on the activity and results so far can be found in [26], the book collecting the revised results from the first period of the Challenge.

1.1 The problem scenarios

The challenge problems are realistic e-business scenarios in purchase order management. They are organized into major problem levels with sub-problem variations, with changes in the web services, the protocol of interaction, and the purchase order in consecutive variations.

Two scenarios address different aspects of the Semantic web techniques:

- The *mediation* scenario concerns making a legacy order management system inter-operable with external systems that use a simplified version of the RosettaNet PIP3A4 specifications.[1] It concerns therefore finding an adequate *orchestration* that adapts two conversation partners that mismatch both in the interaction protocol and in the granularity and format of data.
- The *discovery* scenario concerns the dynamical discovery, selection, binding, and invocation of the most appropriate shipment service for a set of given shipment requests. This scenario addresses *matchmaking* for the single services.

We concentrate in this paper on the Mediation Scenario.

In the following, Sect. 2 gives an overview about the chosen case study: the Mediation Scenario of the Semantic Web Service Challenge. After a brief overview of the jABC/jETI Technology (Sect. 3), this basis scenario is then enhanced in Sect. 4 by bringing in SAP's ERP Enterprise Services to address two different business purposes. A solution to these enhanced scenarios that uses jABC and its semantically enhanced facilities is provided in Sects. 5 and 6. Afterward, Sect. 7 shows on an example how to prove conformance to business policies, and Sect. 8 draws our conclusion.

[1] http://www.rosettanet.org/PIP3A4.

2 The original SWSC mediation scenario: process and data mediation

The very basic scenario concerns here purchasing goods using a simplified version of the RosettaNet PIP3A4 specifications. Figure 1 shows its three main components:

- the *Company Blue*, a customer (service requester) ordering products,
- the *Mediator*, the sought-for piece of technology providing automatic or semi-automatic mediation for the Moon Company, and
- the *Legacy System* of the Moon Company

While the external interfaces must follow the RosettaNet specification, internally Moon uses a propriety legacy system whose data model and message exchange patterns differ from those of RosettaNet. Participants shall basically enable Moon to "talk RosettaNet" and implement the *Purchase Order receiving role* part of the interaction described in the RosettaNet PIP 3A4.

Both the Moon legacy systems and the customer Web services (Blue) are provided by the challenge organizers as technical infrastructure accessible online, and cannot be altered (although their description may be semantically enriched). The sketch of the mediator of Fig. 1 requires two services (one from the RosettaNet request to the CloseOrder, called Part 1 and one for the order confirmation, called Part 2) and shall be implemented by the participants.

To manage its order processing, Moon uses two back-end systems: a Customer Relationship Management system (CRM) and an Order Management System (OMS), both accessible on the SWSC testbed through public web services described using WSDL. The scenario describes how Moon has signed agreements to exchange purchase order messages with its client (Blue) using the RosettaNet PIP 3A4 specification.

In order to address integration of the Blue and Moon companies, the participating groups are encouraged to use SemanticWeb service technology to facilitate conversation between all systems, to mediate between the PIP 3A4 and the XML schema used by Moon, as well as to ensure that the message exchange between all parties is correctly choreographed. In particular,

- *Data mediation* is involved in mapping the Blue RosettaNet PIP 3A4 message to the messages of the Moon back-end systems.

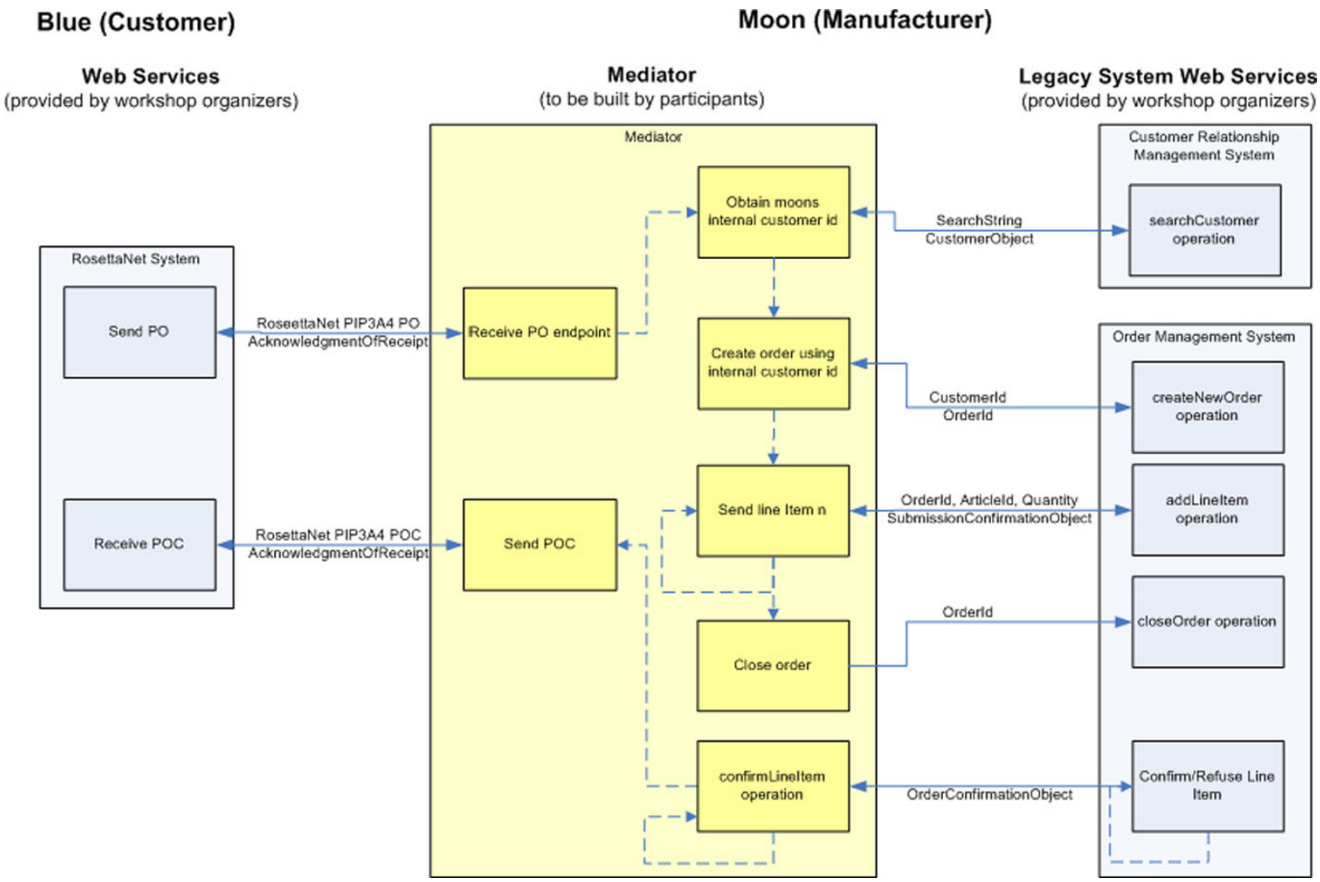

Fig. 1 Abstract view of the SWS mediation scenario

- *Process mediation* is involved in mapping of message exchanges defined by the RosettaNet PIP 3A4 process to those defined in the WSDL of the Moon back-end systems.
- *Conversations* between the systems including data and process mediation operate on semantic descriptions of messages, thus requiring the transformation from messages used by existing systems to the ontological level.

The SWSC organizers provide a set of challenge problems that build upon this initial mediation problem. Correct solution of the basis Mediation Scenario is determined automatically by the SWSC testbed: it tests and certifies that the solution is able to carry out the basic conversation. Subsequent levels foresee changes in some aspects of the problem. The evaluation criteria concern here the degree of declarativity of the solution: ideally, using semantics the middle layer should be able to autonomously react to changes made inside the process specification.

2.1 Outline of the solutions

The generic structure of the solution is shared by all the approaches:

- extract the *relevant* information from the *PurchaseOrderRequest*
- call Moon's Customer Relation Management (CRM) to find the customer data inside the database, if she already has an account.
- Use the *CustomerID* to create an order using Moon's Order Management System (OMS).
- add *LineItems* as needed and then
- close the order.
- Finally the middle layer receives an *OrderConfirmationObject* and
- sends a *PurchaseOrderConfirmation* back to Blue.

2.2 Profile of the jABC-based solutions

So far, we have provided a large set of solutions to the Mediation scenario problem, with increasing level of automation and dynamism. All of them are model-driven and service oriented, and use the jABC framework [4, 20] for design, composition, and execution of the mediator and the jETI [8, 13] technology for the integration and enactment of (a) remote planners/model generation tools and (b) remote heterogeneous services.

We briefly sketch here the main characteristics of the two families of approaches.

2.2.1 Model-driven design

In [7, 8], we showed how to use the business and technical knowledge about the scenario setting in order to semi-automatically extract service and business object representations from the scenario description and the provided WSDLs. The representations are automatically sorted in a simple taxonomy (as in Fig. 5) that can be used as a guidance in composing the solution workflow at the business logic level. This is done graphically. The service representation in the jABC are automatically Java classes, thus they are themselves executable and internally grounded to real services. Using these services within the jABC becomes very easy, so that in principle engaged (non-IT) professionals and even end users are able to compose new applications (that are service orchestrations) and to modify existing applications.

2.2.2 Automatic synthesis

In [6, 10, 11] and more recently in [14], we showed how to organize the business and technical knowledge already present in the just described MDD approach into taxonomies and semantically enhanced service descriptions, and how to use this knowledge to generate automatically the solution's workflow. To this aim,

- the taxonomies are internally represented as relations, in a notation close to Prolog facts or RDF triples, visualized in the jABC as shown in Fig. 5;
- the service descriptions corresponding to the information in Table 3 are formulated analogously.
- Constraints are formulated in SLTL [21], a semantically enriched linear time logic that allows expressing constraints over processes, using constructs like before/after.
- Business objectives are also formulated in SLTL.

Business designers can incrementally express their business knowledge as additional information/constraints, and use an automatic synthesis algorithm to obtain the collection of feasible service compositions that satisfy that objective and are consistent with the given constraints.

This way, business process composition elegantly arises without any need of programming, as a consequence of the enterprise knowledge and of the expressed business goals. Changes in the goal, in the situation, or in the objectives are automatically reflected in the new computed solution space. Thanks to the jABC framework, these solutions are immediately executable.

In particular, we showed in [11] how a naive formulation of the goals leads to solutions that processes empty orders, and how easy it is to add business-level knowledge to automatically obtain solution that are not only technically correct, but also useful and sensible from the business point of view.

Having the entire solution space available is a great help for managers: Ensuring that further global properties, typically linked to business conduct rules, or to regulations, or to security/authorization, are always satisfied is then possible (by automatic proof on the graph of the solution space). Examples of such business-level properties are:

- P1: *No item can be billed multiple times, even in split billing scenarios*, or
- P2: *Operations in the Order Management System and in the corresponding ERP system should always be kept consistent*.

The enhancement we propose here are profits of the jABC/jETI technology that we here briefly introduce. For an introduction from the point of view of the users we address here, we refer to [19], and for a comprehensive discussion of the underlying techniques we refer to the website and to the summary contained in [4]. We focus on the viewpoint of the Change Management Manager and Engineer: we assume that the previous Mediator solutions are already available to them, and they now wish to enhance them in the described way.

3 Overview of the jABC/jETI technology

For the process definition and management, we use a multi-purpose domain-independent modeling framework, the *Java Application Building Center* (*jABC*) [4], complemented by the *Java Electronic Tool Integration* framework (*jETI*) [8, 13] for dealing with integration and execution of remote services. Both are based on well-established software technology and have been used successfully in different application domains [16], most recently targeting bio-informatics processes [9, 13].

jABC is a control-flow oriented environment for service design and analysis. It enforces a *one-thing-approach* [19, 20] to eXtreme Model Driven Design [15, 17], thus avoiding the typical discontinuity in modeling style and technologies between the business and the IT aspects of a running process [12, 16].

Business processes become increasingly networked, parallel, conditional, event driven, recursive, and asynchronous: this is the kind of complexity sources whose control is at the core of jABC's strengths. Additionally, clear formal semantics is the precondition for a formal analysis and verification of properties of the designed workflows based on automatic mathematical proofs. jABC is built with this formal verification capability in the focus, and to scale for large models.

As detailed in [4], jABC is a sophisticated environment for integrating, orchestrating, and providing end-to-end services. Business processes are built by constructing service compositions (called Service Logic Graphs, or SLGs) that orchestrate basic services (in the form of *SIBs—Service-Independent Building Blocks*) along the flow of control.

All the user interaction happens within an intuitive graphical environment, hardly requiring any classical programming skills. Figure 3 shows the GUI: the available projects are listed in a browser (top left). There, one can switch to the SIB palette view (Fig. 4) from where they can be dragged onto the drawing area (right), where the SLG construction takes place. Different inspectors (bottom left) can be used for the detailed configuration and analysis of components and models.

The advantages of the framework are manifold: being a control-flow-oriented service definition environment, it is adequate to support complex control structures as primitives. For example, iterations over lists or matrices are provided as SIBs in the environment. At the same time, the data dependencies (which are secondary to the control flow) do not clog the representation: even large processes with complex dataflows are still easily readable.

Moreover, SLGs are at the same time mathematically analyzable objects: they are directed graphs, whose nodes (the SIBs) represent basic services and whose edges (the branches) define the flow of control. They are thus amenable to the sophisticated analyses provided by modern model checking techniques, as shown in Fig. 3 and explained on an example in Sect. 3.

The jETI platform is used jABC to accomplish the communication with remote tools. This includes acting as a client for (SOAP and REST) web services, CORBA IDL, or other RPC standards. This way the provision of appropriate SIBs happens as far as possible by generation based on a standardized service description, for instance WSDL (see Fig. 4). We have already successfully imported in the past entire SIB palettes from, e.g. FASTA or Muscle services in bioinformatics. The complete deployment of applications into new web services can be realized by jETI as well.

jETI provides also a specific technology for making file-based Java or command line applications accessible via the internet. In jETI, the application provider maintains a *server* that accesses (a collection of) applications on the one side, and on the other it provides an interface to the internet. At runtime, the server receives service requests from a client (the business process' SLG) and forwards them to the actual applications, then collects the results and builds adequate response messages. Similar to the web services's WSDL descriptions, relevant request parameters as well as the actual calls used by the jETI server to execute the applications are defined in an XML file. This information is used by jETI

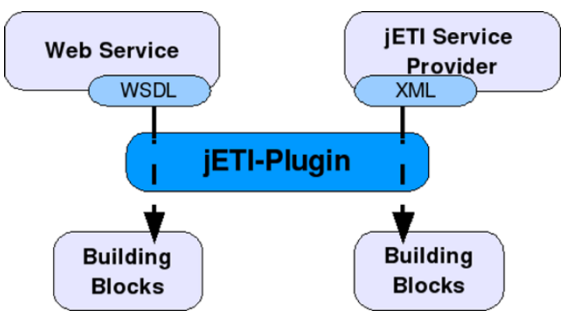

Fig. 2 Service integration via the jETI technology

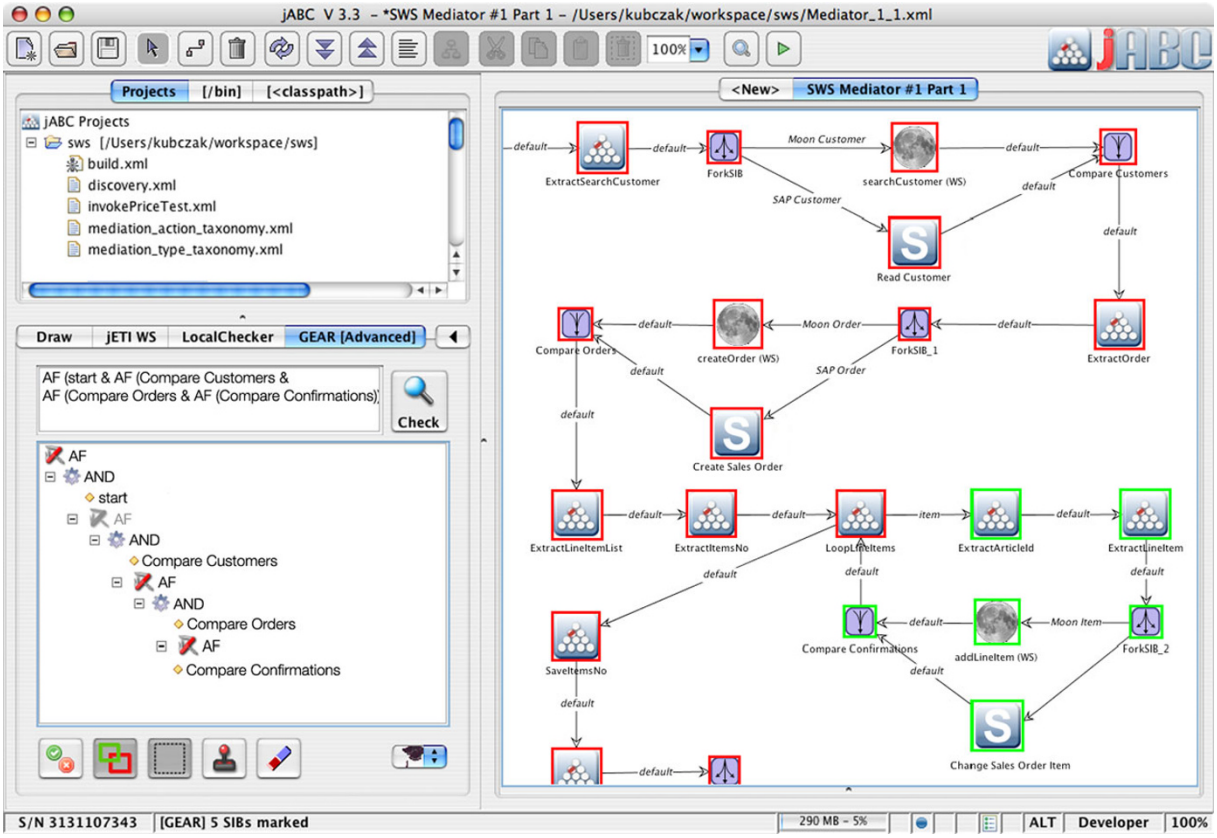

Fig. 3 jABC's advanced features: model checked solution of Fig. 10

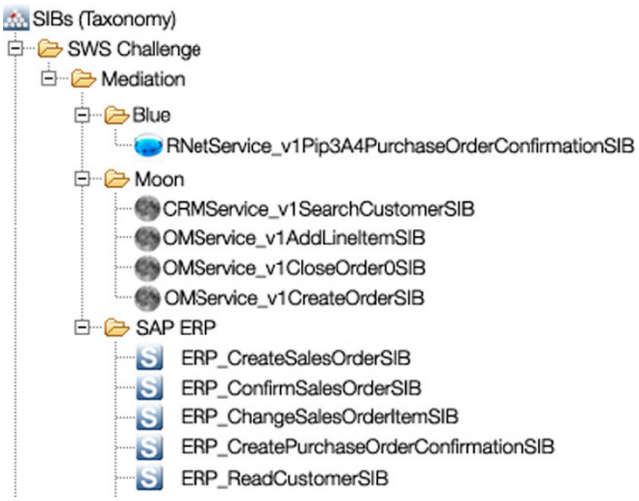

Fig. 4 jABC SIB palette with added SAP ERP services

to automatically generate appropriate SIBs (Fig. 2). Integration of services by means of jETI is convenient especially in the case of legacy application, REST services, and whenever else the setup of a classical web service is not adequate or feasible.

4 ERP-enhancement with the enterprise service bundle

Starting from the original SWS-C Mediator scenario, we consider two scenarios that involve its enhancement by means of a widespread, but also notoriously difficult to integrate, commercial product: SAP's ERP software.

In particular, we show how to exchange Moon's proprietary legacy system for services from SAP's Enterprise Service bundle [2, 3], showing that this solution is still compliant to additional requested properties.

Afterwards we also enhance Moon's proprietary legacy system with SAP'S ES bundle as additional ERP backend and show how to use the same technology presented in [11] to automatically obtain the augmented business project based only on our semantic technologies.

Enterprise services bundles are collections of enterprise (web) services that can be used to extend the functionality of SAP ERP 6.0 or other solutions of the SAP Business Suite. SAP ES provide service packages for all the central ERP applications: *Corporate Services Financials, Human Capital Management, Procurement, Product Development and Manufacturing, Sales and Services, Transportation, Warehousing*. These service bundles can be browsed and reviewed using SAP's SDN website [3].

Bundles consist of enterprise services that will be made available as part of the enhancement packages for *SAP ERP 6.0* or as a specific software add-on for other SAP Business Suite solutions. They provide a new set of services along with documentation of how the services can extend and reconfigure processes in a specific business scenario. Each ES bundle includes explanations of the relevant processes, busi-

Table 1 The service replacement map for Moon's new ERP backend

MOON SERVICE	SAP ENTERPRISE SERVICE
Search Customer	Read Customer
Create Order	Create Sales Order
Add LineItem	Change Sales Order Item
Close Order	Confirm Sales Order
Confirm/Refuse LineItem	Create Purchase Order Conf.

Table 2 The business objects replacement map for Moon's new ERP backend

SAP ENTERPRISE SERVICE	BUSINESS OBJECT
Read Customer	Customer
Create Sales Order	Sales Order
Change Sales Order Item	Sales Order
Confirm Sales Order	Sales Order
Create Purchase Order Conf.	Purchase Order Conf.

Table 3 Service collection for the platform migration scenario: new ERP activities

Activity name	Input type	Output type	Description
`Moon`			The Moon backend system
`searchCustomer`	*SearchString*	*CustomerObject*	Gets a customer object from the backend database
`createOrder`	*CustomerID*	*OrderID*	Creates an order
`addLineItem`	*LineItem*	*SubmConfObj*	Submits a line item to the backend database
`closeOrderMoon`	*OrderID*	*TimeoutOut*	Closes an order on the backend side
`confRefLineItem`	*Timeout*	*orderConfObj*	Sends a conf. or ref. of a prev. subm. LineItem
`SAP`			The SAP backend system
`Read Customer`	*SearchString*	*CustomerObject*	Gets a customer object from the backend database
`Create Sales Order`	*CustomerID*	*OrderID*	Creates an order
`Change Sales Order Item`	*LineItem*	*SubmConfObj*	Submits a line item to the backend database
`Confirm Sales Order`	*OrderID*	*TimeoutOut*	Closes an order on the backend side
`Create Purchase Order Conf.`	*Timeout*	*orderConfObj*	Sends a conf. or ref. of a prev. subm. LineItem

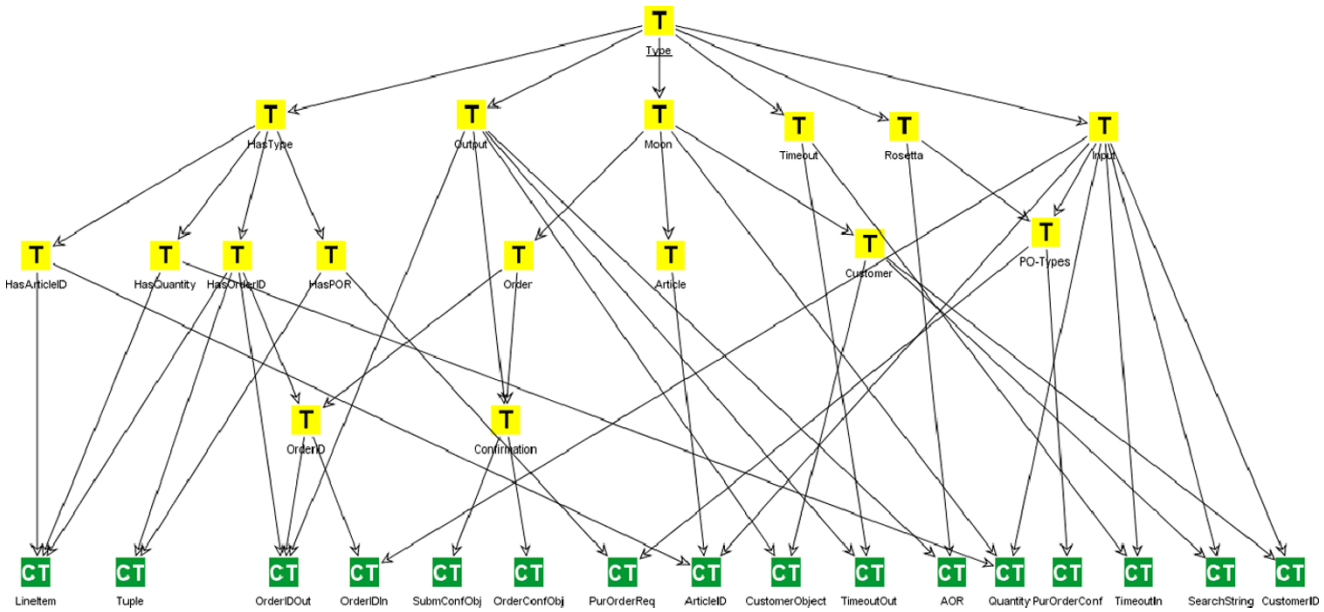

Fig. 5 The type taxonomy for the platform migration scenario: nothing needs to be modified

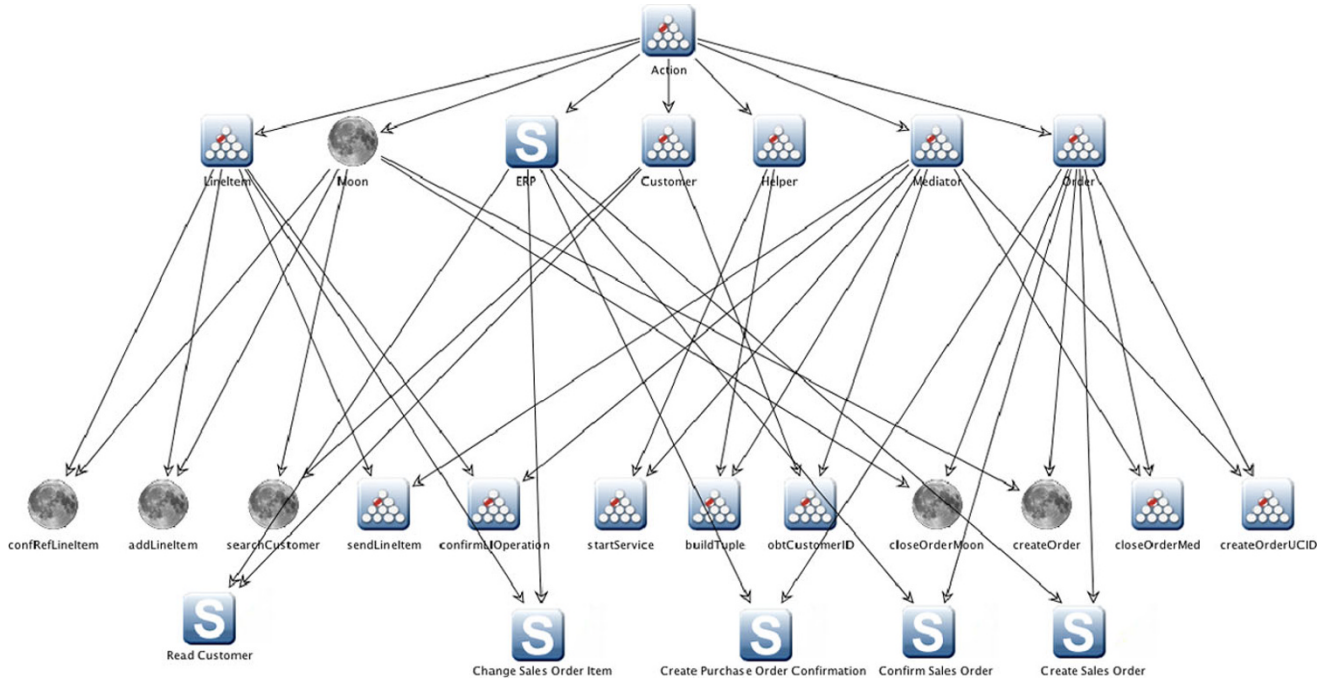

Fig. 6 Platform migration: the action taxonomy with added SAP ERP services

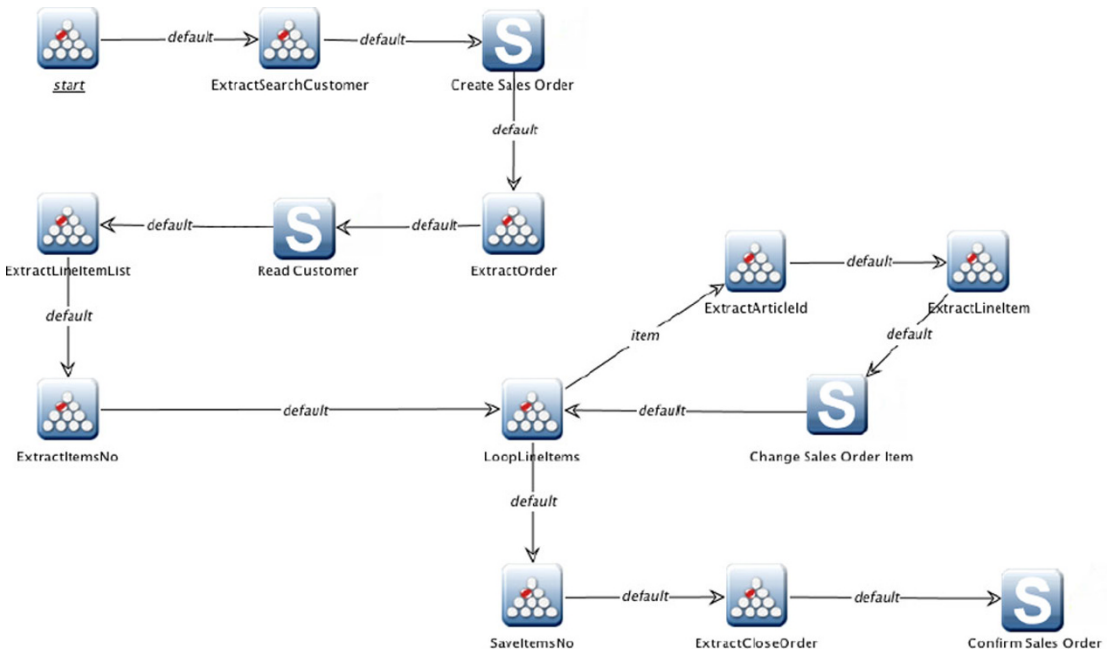

Fig. 7 Platform exchange: the solution to the mediation scenario part 1

ness scenario, and roles involved along with descriptions of business objects and tips about how to put the services to work.

The services delivered through ES bundles are typically used to create composite applications using *SAP NetWeaver's* enterprise services development and modeling tools within SAP's tool world. We show here instead how to use them inside the jABC/jETI framework, to enhance preexisting end-to-end composite services that are totally independent of the SAP ecosystem.

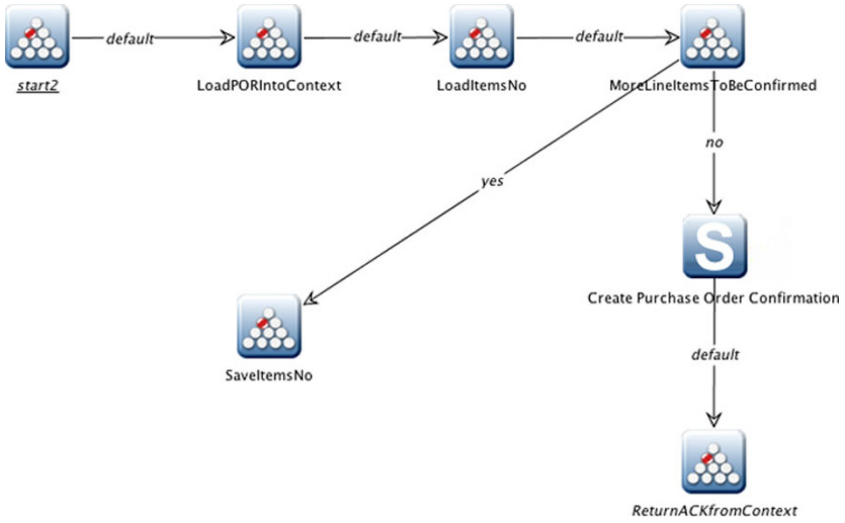

Fig. 8 Platform exchange: mediation scenario part 2 (order confirmation)

Table 4 Service collection for the backend enhancement scenario: new ERP activities and types

Activity name	Input type	Output type	Description
`Moon`			The Moon legacy system
`searchCustomer`	*MoonSearchString*	*MoonCustomerObject*	Gets a cust. obj. out of the back. database
`createOrder`	*MoonCustomerID*	*MoonOrderID*	Creates an order
`addLineItem`	*MoonLineItem*	*MoonSubmConfObj*	Submits a line item to the back. database
`closeOrder`	*MoonOrderID*	*MoonTimeoutOut*	Closes an order on the backend side
`confRefLineItem`	*MoonTimeout*	*MoonOrderConfObj*	Sends a conf. or ref. of a prev. subm. Item
`SAP`			The SAP backend system
`Read Customer`	*SAPSearchString*	*SAPCustomerObject*	Gets a cust. obj. out of the back. database
`Create Sales Order`	*SAPCustomerID*	*SAPOrderID*	Creates an order
`Change Sales Order Item`	*SAPLineItem*	*SAPSubmConfObj*	Submits a line item to the back. database
`Confirm Sales Order`	*SAPOrderID*	*SAPTimeoutOut*	Closes an order on the backend side
`Create Purchase Order Conf.`	*SAPTimeout*	*SAPOrderConfObj*	Sends a conf. or ref. of a prev. subm. Item

5 Platform migration: replacing moon with SAP'S ERP ES bundle

In a platform migration scenario, Moon becomes SAP customer and introduces its ERP component. Here, we face the task of completely replacing Moon's current SWS-C backend with an SAP system. This requires

- representing the internal process flow of Moon's legacy system with *SAP ERP 6.0* and
- provisioning the corresponding web services using components of appropriate Enterprise Service bundles.

In the Mediation scenario, we intend to use bundles from the *Sales and Services* package, that includes as thematic bundles *Customer Fact Sheet*, *Customer Service Execution*, *Order to Cash* and *Quote to Order for Configurable Products*.

The *Order to Cash* bundle contains services that are directly adequate to replace Moon's OMS. It also provides basic Customer Relations functionality, so we decide to use appropriate services for customer identification from this bundle until SAP's CRM will be fully supported by SAP ES (forthcoming).

5.1 Semantic modeling

Regarding this scenario update, the new realization with Enterprise Services leads to a replacement of Moon's service components as depicted in Table 1. The corresponding mapping of the Business Objects is shown in Table 2.

We previously generated the current Moon mediator process with the semantic-supported technique summarized in Sect. 2.2.2, which in particular comprises taxonomies for types and activities and concrete service descriptions.

This leads to the following semantic descriptions:

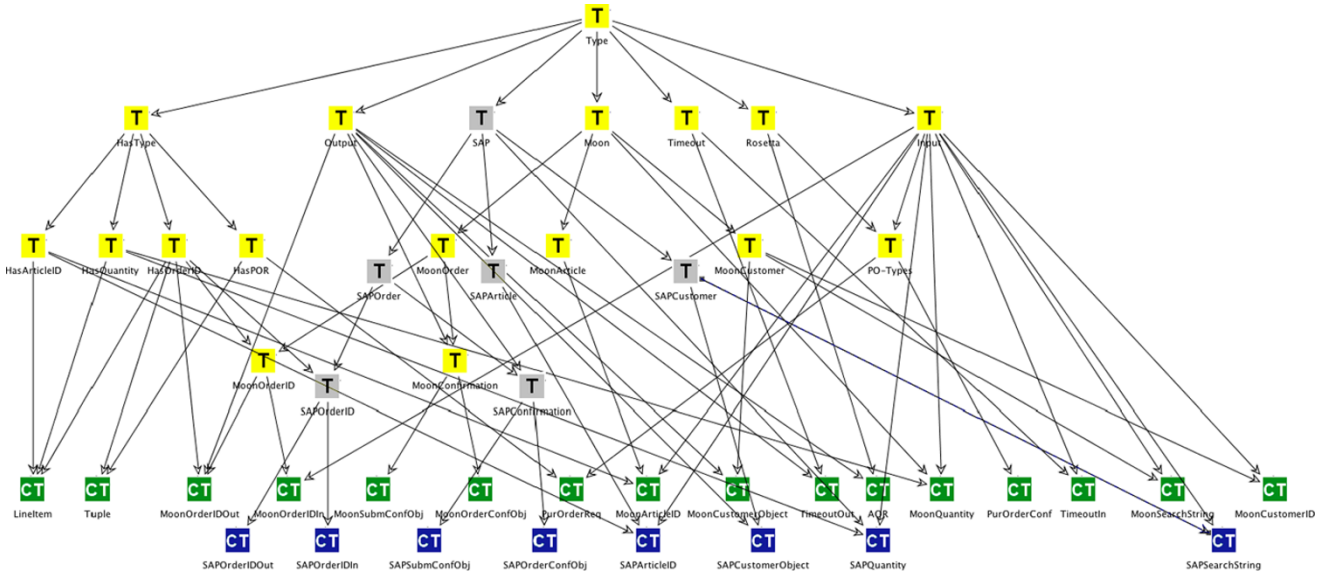

Fig. 9 The type taxonomy for the backend enhancement scenario, with ERP-related instance and concept types

- We extend the collection of available services with the new ERP ES palette, as shown in Table 3.[2] As we can see, the supported semantic types are the same as for Moon's services.
- The type taxonomy is therefore unchanged: Fig. 5 is identical to what shown in [11]. Here, the is—a relation between type concepts (in yellow, or light) and type instances (in green, or dark) is expressed graphically. Internally, we use a Prolog-like triple-based formalism embeddable in OWL(-S), we have a corresponding transformation to the Protege'tool [23].
- The activity taxonomy Fig. 6 now includes the additional ES services: a new facet SAP is added as concept and the individual services are added as its instances as well as instances of preexisting concepts, like e.g. the ReadCustomer service concerns the Customer concept.

5.2 The new mediator

As shown in [8], we can easily import the WSDLs of the ES bundle services into jABC SIBs. As a consequence, we immediately have a collection of (Java-based) components, that over jETI technology can execute the real ES bundle services installed in Walldorf.

The added ES bundle services appear as a new SIB palette in the jABC 4, and are ready to be used in a manual modeling scenario. Here, we prefer to follow the same generation based approach of Sect. 2.2.2, so we reuse the final business goal specification that we had found in [11].

[2]We could also completely eliminate Moon's service collection, but in migration cases it is customary to first extend the platform, and purge obsolete functionalities only once the new version operates correctly.

Use the mediator service to produce a Purchase Order Confirmation.

The corresponding formal specification is simple: we need to initiate the service (module startService) and consider on the way the single items listed in the order (`LineItem`) which need corresponding confirmations (`confRefLineItem`). We may simply write:

```
(startService < LineItem <
                confRefLineItem )
```

where the symbol < means *before* or *preceeds*.

The jABC process model shown in Fig. 7 yields the expected required solution for the first service required by the mediator (Part 1).

Analogously, we can address the Mediator Part 2: From the `confRefLineItem` we need to reach a purchase order confirmation (`PurOrderCon`, a type), expressed in SLTL as

```
(confRefLineItem <PurOrderCon)
```

and whose resulting jABC process model, shown in Fig. 8, yields the expected required solution for the first service required by the mediator (Part 2).

As we see, the business-level descriptions are easily updated, the goals are reused, and the incremental effort for this evolution is minimal.

6 Backend extension: providing an ERP backend to moon

In a Backend Extension scenario, Moon's old legacy systems remain in place, but they are backed by ERP func-

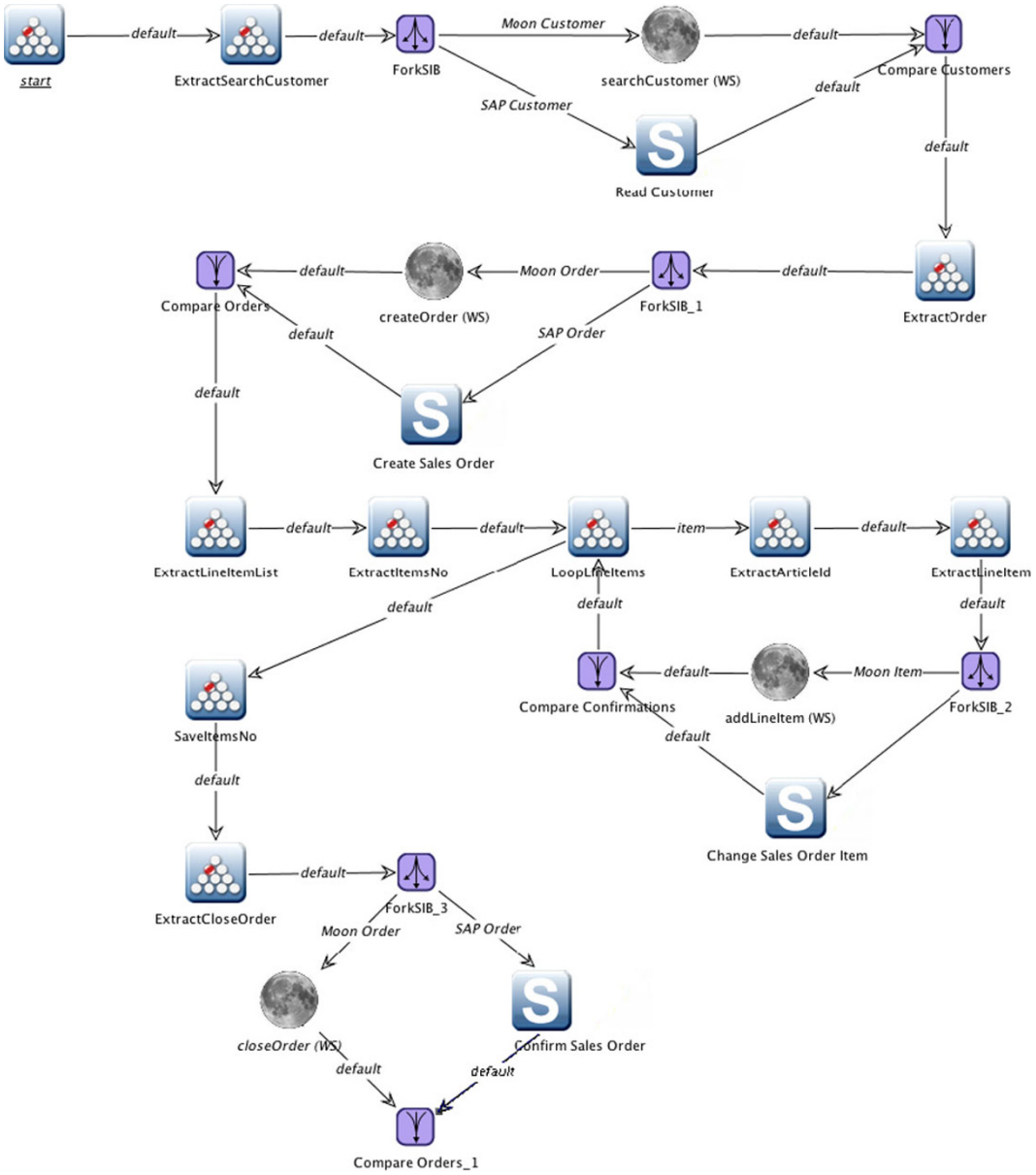

Fig. 10 Platform migration: new mediator solution part 1

tionality offered by the ES bundle. Service identification within the ES bundle leads to the same results as in the previous case (see Table 1), but a new intermediate layer is needed to map customer operations through Moon's services to the SAP backend. This touches the semantic types.

6.1 Semantic modeling

This leads to the following semantic descriptions:

- The collection of available services requires new types, as shown in Table 4.[3]
- The activity taxonomy is unchanged wrt. the platform migration scenario: we still refer to the activity taxonomy of Fig. 6.
- The type taxonomy (Fig. 9) now includes the additional ES types: next to Moon, a new facet SAP-type (grey icon)

[3]Technically, we could also do with the same types, using additional constraints, but this is the most intuitive solution and we intend here to stress the ease of use of the approach.

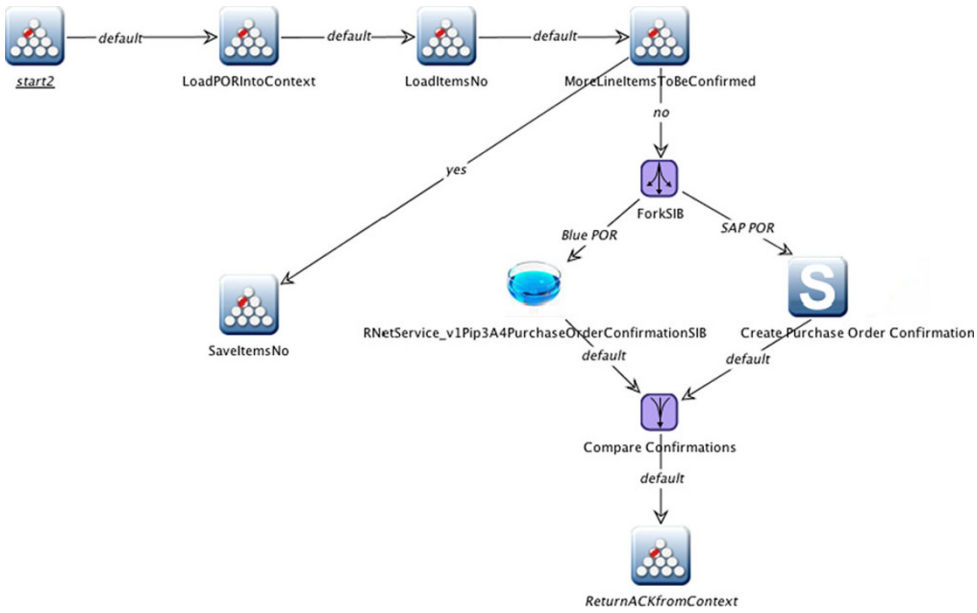

Fig. 11 Platform migration: new mediator solution part 2

is added as concept with new sub-concepts as appropriate, and the individual types are added as their instances (here with a dark blue icon), as well as instances of preexisting concepts like, e.g. the SAPArticleID-type is a SAPArticle-type and hasArticleID.

6.2 The new mediator

Binding the SIBs to the new semantic types concerns only the taxonomies (the enterprise physics-like pieces of business knowledge) but does not involve any change at the level of the SIB realizations. Therefore, we can proceed with the same SIB palette we had before, shown in Fig. 4.

We can this time manually extend the solution of [11] by forcing a lockstep behavior of the workflow at Moon's side: every time the legacy systems (the CRM and the OMS) perform a step, a corresponding recording in the new ERP system must also occur. To this aim, every CRM and OMS service invocation has a parallel invocation of the corresponding ERP service. In jABC, we use for this fok/join constructs. The resulting solutions for the mediator Part 1 and Part 2 are shown in Figs. 10 and 11, respectively.

7 Proving conformance to business policies

The lockstep behavior of the workflow, which is a requirement close to the property P2 of Sect. 2.2.2 might not hold for all the operations, maybe due to errors in the manual model enhancement. It is therefore desirable to be able to express business level policies that are then checked on the enhanced mediator business logic.

We enhance the join SIBs of the workflow with semantics, adding properties that express the business-level meaning of a flow join at that point. In Fig. 3, we chose to do it directly on the SIB: each join-SIB is then annotated with the business-level entity that is held consistent between the two backend systems at that point in the business logic. For example, the join-SIB following the first fork is only reached if the same *Customer* is successfully found in Moon's CRM and ERP subsystems. Accordingly, it is called CompareCustomer.

The business policy the can be expressed as following: "*It is always true that customers, orders, and confirmations must successfully match*".

This is expressed internally as a property in CTL that the jABC can automatically verify on the SLG by means of model checking [24, 25]. In this case, we see that the match is not always given for all the three observed business objects: a large part of the business process is marked in red, signalling that the property is not verified in that portion of the flow. In fact, as shown in Fig. 3, if an empty order is processed, the loop that handles the order's line items is not entered, thus the CompareConfirmation synchronization checkpoint within the loop cannot be reached.

It is a matter of business policies to decide whether this is a serious problem or not. If auditing and governance mandate a record for this operation for every processed order, then the workflow must be modified. If the rules require this only for non-empty orders, or the system ensures elsewhere that empty orders are not accepted, then the (technical) violation is not a problem and the workflow still conforms to the business policies. Then, however, the formulation of the business policy should be precised, expressing that this is valid only for non-empty orders. In both cases, however, the model checker helps to exhibit inconsistencies between required business policies and the actual implementation that would very likely otherwise remain undetected.

8 Conclusion

We have shown how to enhance the Semantic Web Service Challenge Mediation Scenario toward real life business applications using SAP Enterprise Service bundles, once as a substitute and once as an enhancement to the proprietary services of the Moon business partner. We used for this several capabilities offered by the jABC, a framework for model-driven, service-oriented, and platform-independent development of complex business processes. Thanks to the semantic-enhanced technologies it provides and to the integration with web services, the realization was straightforward. We were able to show how declarative, simple descriptions of the business knowledge and business goals can be used to describe services and business objects. We also showed how these can be used to automatically generate adaptations of end-2-end workflows including functionalities of commercial, state-of-the-art ERP systems, or to prove the compliance of those adaptations/modifications wrt. business policies.

In particular, we have shown on adaptations of a case study from the Semantic Web Service Challenge that much of the technology needed to make automatic support of service evolution viable is already available today. Today's most advanced platforms might start introducing these concepts and technologies in real businesses, and this was realizing the promise of a *Continuous Model-Driven Engineering* approach [18].

Acknowledgements We thank Charles Petrie, Matthias Kaiser, and Peter Emmel for many discussions on the Challenge examples and its relation to Enterprise SOA.

References

1. Semantic web service challenge website. http://www.sws-challenge.org
2. Efeoglu A. SAP enterprise service workplace handbook
3. SAP enterprise service website. http://www.sdn.sap.com
4. Jörges S, Kubczak C, Nagel R, Margaria T, Steffen B (2006) Model-driven development with the jABC. In: HVC—IBM Haifa verification conference, Haifa, Israel, 23–26 October 2006. LNCS. Springer, Berlin
5. Kaiser M (2007) Towards the realization of policy-oriented enterprise management. IEEE Comput 11:65–71
6. Kubczak C, Margaria T, Kaiser M, Lemcke J, Knuth B (2008) On-the-fly synthesis of the mediator scenario with jABC and POEM. In: Proceedings of the 6th international workshop on evaluation of ontology-based tools and the semantic web service challenge, with ESWC 2008, Tenerifa, Spain, June 2008
7. Kubczak C, Margaria T, Steffen B, Nagel R (2009) Service-oriented mediation with jABC/jETI. In: Petrie C, Margaria T, Zaremba M, Lausen H (eds) Semantic web services challenge—results from the first year. Springer, Berlin, pp 71–99
8. Kubczak C, Margaria T, Steffen B, Naujokat S (2007) Service-oriented mediation with jeti/jabc: verification and export. In: Workshop on service composition & SWS challenge, part of WI-IAT'07, the IEEE/WIC/ACM international conference on web intelligence, Stanford (CA), November 2007. IEEE, New York
9. Lamprecht A-L, Margaria T, Steffen B (2008) Seven variations of an alignment workflow—an illustration of agile process design/management in bio-jeti. In: ISBRA 2008, 4th international symposium on bioinformatics research and applications, Atlanta, GA, May 2008. LNBioinformatics, LNCS, vol 4983. Springer, Berlin, pp 445–456
10. Special session on SerComp & SWS challenge 2007 workshop, IEEE/WIC/ACM international conference on web intelligence (WI 2007), November 2007
11. Margaria T, Bakera M, Kubczak C, Naujokat S, Steffen B (2009) Automatic generation of the SWS-challenge mediator with jABC/ABC. In: Petrie C, Margaria T, Zaremba M, Lausen H (eds) Semantic web services challenge—results from the first year. Springer, Berlin, pp 119–138
12. Margaria T (2007) Service is in the eyes of the beholder. IEEE Comput 40(11):33–37
13. Margaria T, Kubczak C, Steffen B (2007) Bio-jETI: a service integration, design, and provisioning platform for orchestrated bioinformatics processes. In: BioMed central (BMC) bioinformatics supplement dedicated to network tools and applications in biology 2007 workshop (NETTAB 2007), V.9/4 2007. http://www.biomedcentral.com/1471-2105/9?issue=S4
14. Margaria T, Meyer D, Kubczak C, Isberner M, Steffen B (2009) Synthesizing semantic web service compositions with jMosel and Golog. In: Proceedings ISWS 2009, international semantic web conference, October 2009. LNCS, vol 5823. Springer, Berlin, pp 392–407
15. Margaria T, Steffen B (2005) From the how to the what. In: VSTTE: verified software—theories, tools, and experiments, proceedings of the IFIP working conference, Zurich, October 2005
16. Margaria T, Steffen B (2006) Service engineering: linking business and IT. IEEE Comput, 53–63
17. Margaria T, Steffen B (2004) Aggressive model driven development for the management of service evolution. Annu Rev Commun, 57
18. Margaria T, Steffen B (2009) Continuous model-driven engineering. IEEE Comput 42(10):106–109. doi.ieeecomputersociety.org/10.1109/MC.2009.315
19. Margaria T, Steffen B (2009) Business process modeling in the jABC: the one-thing-approach. In: Cardoso J, van der Aalst W (eds) Handbook of research on business process modeling. IGI Global, pp 1–26
20. Steffen B, Narayan P (2007) Full lifecycle support for end-to-end processes. IEEE Comput 40(11):64–73
21. Margaria T, Steffen B (2007) LTL guided planning: revisiting automatic tool composition. In: ETI proceedings SEW2007, 31st IEEE annual software engineering workshop, Loyola College, Baltimore, MD, USA, March 2007. IEEE, New York
22. Margaria T (2008) The semantic web services challenge: tackling complexity at the orchestration level. In: Proceedings ICECCS'08, 13th IEEE international conference on engineering of complex computer systems, Belfast, UK, April 2008
23. Protege' webpage. http://protege.stanford.edu/
24. GEAR: game-based easy and reverse model-checking. http://jabc.cs.uni-dortmund.de/gear/
25. Emerson EA, Jutla CS, Prasad Sistla A (1993) On model-checking for fragments of μ-calculus. In: Proceedings CAV
26. Petrie C, Margaria T, Lausen H, Zaremba M (eds) (2008) Semantic web services challenge—results from the first year. Springer, Berlin