Competences of Undergraduate Computer Science Students

Kathrin Bröker, Uwe Kastens, Johannes Magenheim
Institute of Computer Science
University of Paderborn
Fürstenallee 11
33102 Paderborn Germany
{kathyb, uwe, jsm}@upb.de

Abstract: The paper presents two approaches to the development of a Computer Science Competence Model for the needs of curriculum development and evaluation in Higher Education. A normative-theoretical approach is based on the AKT and ACM/IEEE curriculum and will be used within the recommendations of the German Informatics Society (GI) for the design of CS curricula. An empirically oriented approach refines the categories of the first one with regard to specific subject areas by conducting content analysis on CS curricula of important universities from several countries. The refined model will be used for the needs of students' e-assessment and subsequent affirmative action of the CS departments.

Keywords: Competences, Competence Measurement, Curriculum Development, Computer Science Education, Recommendations for CS-Curricula in Higher Education

1 Introduction

The central demand of the Bologna reform is the outcome-oriented implementation of study courses, which include competence-oriented descriptions of the curricula and respective competence-oriented assessment procedures. These demands are often implemented only in a superficial way. To accomplish the demands of Bologna it is not enough to adapt the curricula. Developing a competence-oriented course of studies is a process, which consists of several activities. According to Schaper (2012) these are e.g.: developing a competence oriented curriculum, approaches for a competence oriented organization of teaching and learning, methods for a competence oriented assessment and also approaches for a course related advancement of competences.

The base for most of these activities is a competence model. It is needed for the definition of learning outcomes on different levels of granularity of a study course like modules, lectures or seminars. Additionally for the needs of assessment and in order to identify affirmative action a concept of competence measurement is necessary.

This paper explains two different methodologies for developing a competence model for computer science undergraduates. The results of these approaches and the additional value of relating them to each other are described.

The first methodology represents a normative approach. The competence model is the agreed and confirmed result of a broad discussion between CS domain experts on the basis of a generic psychological competence model. In this case, a CS expert group of the German National Computer Society (GI) developed a CS competence model in order to issue design recommendations for CS curricula in Higher Education; see *Methodology A* in Fig. 1.

The second methodology represents an empirically oriented approach and derives the competence model by means of the content analysis of international CS curricula; see *Methodology B* in Fig. 1. The applied content analysis reveals common CS-competences addressed in different CS study courses of universities. The students at the end of their CS-bachelor studies should have achieved them. This research is part of a project conducted by the Computer Science Learning Center (LZI)[1] of our university. The main objective of the project is to support CS-students' learning processes by identifying learning barriers and contribute to resolve them. An empirically grounded competence model in the area of software engineering, software development and programming forms the basis of competence measurement instruments, which can be used for purposes of diagnosis, students' self-assessment and for affirmative action of the LZI.

1 Funded by the Ministry of Education and Research (BMBF).

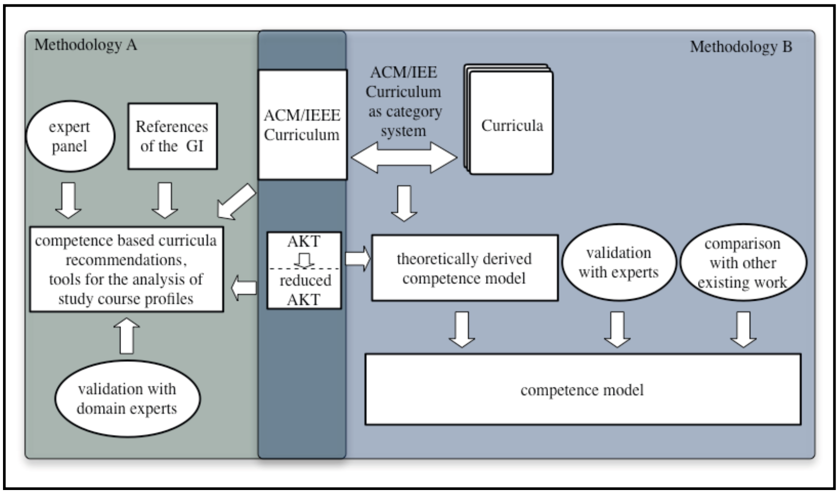

Methodology A
Methodology B
expert panel
References of the GI
ACM/IEEE Curriculum
ACM/IEE Curriculum as category system
Curricula
competence based curricula recommendations, tools for the analysis of study course profiles
AKT
reduced AKT
theoretically derived competence model
validation with experts
comparison with other existing work
validation with domain experts
competence model

Figure 1: Methodologies for competence model development

2 Concepts of Competence

During recent years the research landscape in the field of competence models and competence measurement diversified. Origin of these research efforts was the Bologna process and STEM-related research, especially the PISA project. Nevertheless, only little research for higher education and academic competences exists. Before we introduce both approaches to a competence model we give a short definition of the term competence followed by a short introduction to AKT and related psychological models of cognition and a short overview on theoretical aspects of competence models and measurement.

2.1 Competences and Models of Cognition

The term competence has many definitions in different subjects. In the area of curriculum development e.g. the competence structure model provided by the EU-Tuning project is often applied (Tuning, 2013). This is not the place to discuss all the different definitions. Consequently we only introduce the definition used in our projects. As applied in most of the recent projects we use the definition of Weinert (2001) (original in German, translation by the author). Competences are "the existence of learnable cognitive abilities and skills which are needed for problem solving as well as the associated motivational, volitional and social capabilities and skills which are essential for successful and responsible problem solving in variable situations." (Weinert, 2001, p. 27) This definition implies, that competences are learnable by interventions. Accordingly we can use a competence assessment to measure if any intervention during the educational process helps to develop competences or to achieve a higher competence level. To take account of this aspect during curricula development we need a competence model that allows grading. Therefore in both methodologies presented here, we refer to the "Taxonomy for learning, teaching, and assessing" by Anderson and Krathwohl (Krathwohl, 2002) also named as AKT, which is a revision of the well known taxonomy of Bloom. It can be applied for grading of the cognitive dimensions of competence. The AKT model regards learning objectives as a combination of a certain type of knowledge and a certain cognitive process, forming the two dimensions of the original Bloom's taxonomy in the following way (see Krathwohl, 2002, p. 214): A: *Knowledge Dimension* (Factual, Conceptual, Procedural and Metacognitive knowledge); B: *Cognitive Process Dimension* (Remember, Understand, Apply, Analyze, Evaluate, Create). While the knowledge dimension offers classification of knowledge types, which are relevant in the context of learning, the second dimension of cognitive processes can be used for a hierarchical classification of competences. The AKT model is also applied in both methodologies for CS curriculum development and competence assessment presented in this paper.

2.2 Competence Modeling and Measurement in CS

In Computer Science many projects exist in the context of key competences, especially ICT competences. In contrast there are only a few projects where subject specific competence models are developed. Two of these projects are the German KUI (Berges et al., 2013) and MoKoM (Linck et al., 2013) project. The MoKoM project has its focus on developing a competence model for students at schools. They developed a competence model for informatics modeling and system comprehension. In contrast to that of the project KUI which focuses on competences for future teachers. The competence model here is divided into 3 parts: competences on subject matter knowledge (CK), competences on pedagogical content knowledge (PCK) and non-cognitive competences (NCC) (Hubwieser, Magenheim, Mühling, and Ruf, 2013), (Schaper et al., 2013). In addition there are some projects with research in students' competences at universities. Nevertheless until now, no national or international project has developed a concrete competence model for subject specific competences in computer science for academics. The only existing models are for example the IEEE Software Engineering Body of Knowledge (SWEBOK) (Society, Bourque, and Dupuis, 2004) or the ACM/IEEE Curriculum (The Joint Task Force on Computing Curricula Association for Computing Machinery IEEE-Computer Society 2012, 2013). Nevertheless these documents rather describe knowledge, which should be part of a curriculum, than real competences. Furthermore, these documents are not empirically verified. Consequently we don't know if universities really teach the topics mentioned in these curricula. Only accreditation rules give us a first hint. Moreover they are not as specific as the ACM/IEEE Curriculum. However these documents form a good basis for the development of a specific CS competence model. Therefore, we refer to these approaches when we contextualize and specify our cognitive competence according to the AKT-model. In addition to missing CS competence models there are hardly assessment results for CS competences, which are based on a sound methodology derived from a CS competence model by now.

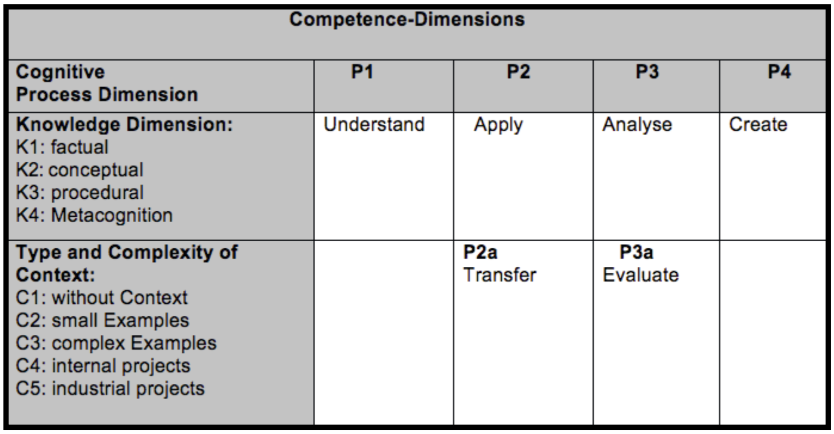
2.3 Competence Models and Measurement

Competence Models are the basis for developing measuring instruments and the description of their results. According to Koeppen, Hartig, Klieme, and Leutner (2013) there are several requirements for competence models. First such a model should represent the internal competence structure. Second the levels of competences should be described and at last these models should consider changes arising in educational processes. The competence structure models should reveal existing relations between the accomplishments of different requirements. Competence level models in particular can describe which requirements different people can accomplish. These models give us the opportunity to identify persons with distinguished competences. Competence level models are used for measuring the outcomes of educational processes, for example, if we want to measure the effectiveness of interventions at the Informatics Learning Center. For the measurement of competences psychometric models are used. For both forms of models the *Item-Response-Theory (IRT)* (Hambleton, Waminanthan, and Rogers, 1991) is applicable. For CS Fuller (Fuller et al., 2007) developed a competence model based on AKT. The differences and similarities between the CS-AKT presented here and the concept of Fuller is not subject of this article. In the authors' opinion the CS-AKT meets better the demands for developing CS program guidelines, while Fuller's concept fits very well for the analysis of specific competence-oriented CS learning processes.

3 Theoretical Approach to a CS Competence Model

In the following we describe two different strategies for developing a competence model. The first approach was applied by an expert group of the German Informatics Society (GI). The task of the group was to develop recommendations and guidelines for the educational design of Bachelor and Master CS study programs at German Universities and Universities of Applied Science. The group's strategy to resolve this problem comprises the following tasks (see Fig. 1):

1. Selection and rationale of a generic competence model that enables the grading of cognitive competence components: AKT
2. Adaption of the AKT to the needs of a CS competence model: adapted CS-AKT
3. Selection and rationale of core CS subject areas according to former and current national and international CS curricula and CS curricula guidelines: CS core subject areas (CS-CSA)
4. Definition and classification of CS competences according to CS-AKT and CS-CSA
5. Describing the competence profile of CS study courses according to the expected competences students' should achieve when attending mandatory modules, lectures and seminars during the study program.



Competence-Dimensions				
Cognitive Process Dimension	**P1**	**P2**	**P3**	**P4**
Knowledge Dimension: K1: factual K2: conceptual K3: procedural K4: Metacognition	Understand	Apply	Analyse	Create
Type and Complexity of Context: C1: without Context C2: small Examples C3: complex Examples C4: internal projects C5: industrial projects		**P2a** Transfer	**P3a** Evaluate	

Figure 2: Adapted CS-AKT for cognitive CS competences

(1): In order to describe CS competence structures and levels we use the AKT-matrix (Krathwohl, 2002) as a generic cognitive competence model. The AKT provides us with a differentiated concept of competence components regarding action-oriented process dimensions of knowledge achievement as well as dimensions of knowledge that take internal aspects of knowledge processing in the human brain into account. We are aware of the fact, that the AKT originally addresses learning objectives. Therefore, we relate the activities, which were described in the cells of the matrix, to specific CS topics, combined CS subject-related activities to a specific context of action and also considered the complexity of the tasks. Thus, the descriptions in the cells meet the requirements of the competence definition according to Weinert. The motivational and volitional aspects of competence remain largely excluded from this classification.

(2): In order to reduce the complexity of the matrix we deleted those cells, which are not or less relevant for CS competences according to the following principles:

a. Competences that are characterized only by remembering without understanding are not sufficient in higher education CS study programs. Therefore, the column 'remembering' of the original AKT-matrix was deleted.
b. At the other end of the process dimension scale the generation of new knowledge is mostly out of scope in CS Bachelor study programs, but it is not deleted in the adapted CS-AKT in anticipation of future descriptions of master's degrees.
c. In order to enable a compact and coherent description of competences of the process dimension with regard to different types of knowledge, we merged the original four rows into one. The addressed types of knowledge in the competence description are indicated by respective annotations W1 to W4.
d. For the description of CS study programs, it is important that competences are characterized with regard to their requested application context. Since the original AKT-matrix doesn't offer a specific structure for this demand, we added a second row that addresses the types and the complexity of the application context. By means of annotations we can classify competences with K1-K5 when describing them (see Fig. 2). For the level 'understand' we didn't introduce this concept, because small examples are always used in this area of knowledge achievement and the application of more complex examples already can be assigned to the process dimension 'apply'.

e. The competence component *P2a 'transfer'* considers the ability to perform competence related action in different application contexts, while *P3a 'evaluate'* indicate a student's ability to evaluate an informatics system on the basis of a previous analysis. In CS study programs the competence components 'transfer' and 'evaluate' do make sense in our opinion only with regard to the context. Therefore, those components are assigned to the dimension (row) 'Types and complexity of context'. The accuracy of the competence descriptions is not affected by this assignment.

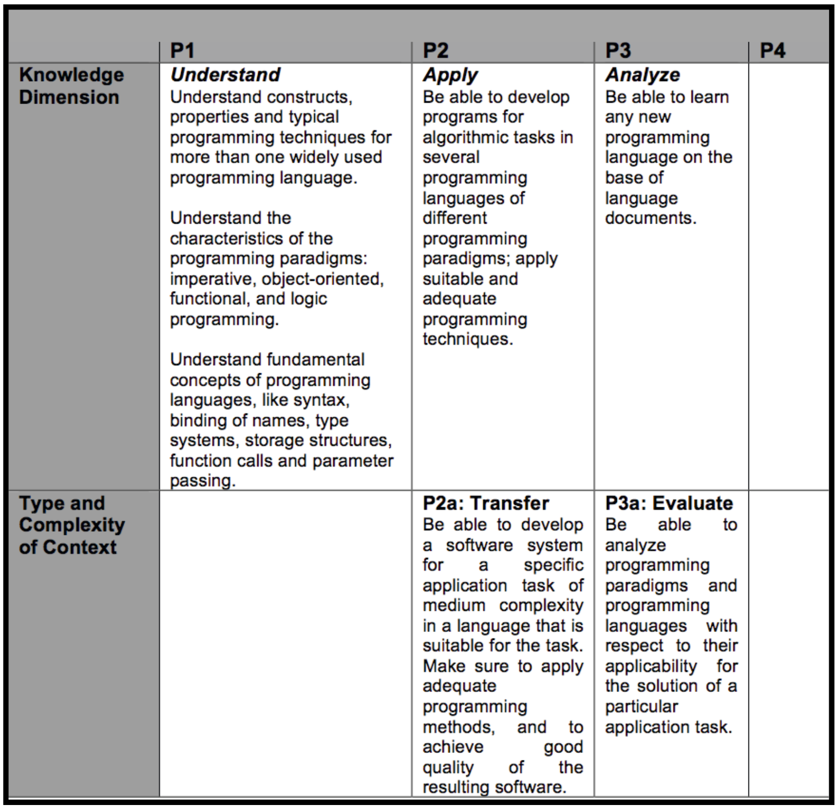
The adapted CS-AKT has proven to be effective for describing competences of CS Bachelor programs. The reduction and adaption should have preserved the underlying AKT methodology.

(3): After an intensive discourse considering current CS curricula recommendations, (e.g. ACM/IEEE, 2013) and former GI CS recommendations the GI expert group selected core CS subject areas (CS-CSA), where students' should have achieved competences according to the CS-AKT model at the end of their Bachelor programs.

The subject areas were derived from the knowledge areas of the ACM/IEEE curriculum without considering the different aspiration levels.

(4): For each of those CS-CSA, competences will be described according to the CS-AKT. In order to illustrate this procedure, we show an example for using the adapted CS-AKT:

Fig. 3 instantiates the CS-AKT scheme of Fig. 2 for the subject area of *Programming Languages and Programming Methods*, which is one of about 20 subject areas to cover Bachelor programs in Computer Science. The stated competences have been cross-checked against the ACM Computer Science Curriculum (ACM/IEEE, 2013), the (FTI, 2004), and a particular Bachelors Program at the University of Paderborn (UPB, 2009).



	P1	P2	P3	P4
Knowledge Dimension	***Understand*** Understand constructs, properties and typical programming techniques for more than one widely used programming language. Understand the characteristics of the programming paradigms: imperative, object-oriented, functional, and logic programming. Understand fundamental concepts of programming languages, like syntax, binding of names, type systems, storage structures, function calls and parameter passing.	***Apply*** Be able to develop programs for algorithmic tasks in several programming languages of different programming paradigms; apply suitable and adequate programming techniques.	***Analyze*** Be able to learn any new programming language on the base of language documents.	
Type and Complexity of Context		**P2a: Transfer** Be able to develop a software system for a specific application task of medium complexity in a language that is suitable for the task. Make sure to apply adequate programming methods, and to achieve good quality of the resulting software.	**P3a: Evaluate** Be able to analyze programming paradigms and programming languages with respect to their applicability for the solution of a particular application task.	

Figure 3: CS-AKT applied for Programming Languages and Programming Methods

In the *Knowledge* dimension the overall objective is formulated in field *P3*: At the end of a Bachelors program students shall be able to learn any new language autonomously on the base of published language documents. That competence is essential, because several new languages will be developed and some paradigm shifts will happen during their professional life. The competences formulated in *P1* and *P2* are preconditions for that of *P3*. The example Bachelor program (UPB, 2009) addresses these competences by a sequence of lectures: *Foundations of Programming* in the first and second semester, where students are introduced into programming in one particular language (Java, in this case), *Foundations of Programming Languages* in the second semester, where they learn to understand constructs and properties of different languages and programming paradigms, and finally in the fifth semester an advanced course on *Programming Languages and Compilers*, where they achieve a deeper insight in language development and implementation.

In the *Complexity and Context* dimension the competences stated in field *P3a* also build on those stated in *P2a*. Both address the ability of the students to use their understanding of programming languages and methods to develop software of good quality in projects of increasing complexity and with decreasing guidance of supervisors. In our example Bachelor program (UPB, 2009) the competences of *P2a* are to be achieved in a practical course in the third and fourth semester; *P3a* is usually a result of the implementation developed in context of the Bachelor thesis.

In the cognitive process dimension *P5: Create* usually does not apply for Bachelor programs. It is kept in the adapted CS-AKT to be used for the description of competences achieved in Masters programs. For this subject area one would for example describe in the knowledge dimension the competence to develop and implement new domain-specific languages. Of course, it requires that more advanced competences are achieved by Masters students in the lower levels of the process dimension.

(5): Finally the competence profiles of CS study programs can be described according to the expected competences students' should achieve during their studies. A classification of types of scientific activities in the modules, seminars and lectures of the study program will be provided, that applies the CS-AKT. The classification of scientific working will be conducted on the basis of those competences, which are expected that they are mediated primarily in the respective course or module. The profile of a program is then obtained from the frequency of occurrence of types of scientific work and its distribution in the average view on a program. By providing this differentiated profile of a study program the CS-AKT also contributes to resolve problems of the bipolar categorization of 'scientific-oriented' and 'application-oriented' study programs.

4 Empirically Oriented Approach to a CS Competence Model

As a second strategy for developing a CS competence model we now describe the approach that is part of an initiative of our learning center (LZI). This is an empirically oriented approach using content analysis. The result should be a more detailed competence model in a specific topic area in contrast to the model developed by the GI experts. Therefore, we can use this strategy to specify the competences of the experts mentioned before. Since the LZI initiative aims ultimately on affirmative action for students to successfully graduate in CS, the results of the research must be very specific and target group oriented.

This is the reason why we focus in this approach on the areas of Software Development, Software Engineering and Programming. The applied research methodology B (see Fig. 1) consists of the following steps:

1. Content analysis of existing CS curricula of selected Universities on the basis of the topic categories provided ACM/IEEE recommendations (ACM /IEEE, 2013).
2. Conducting expert ratings in order to validate the derived competences in the identified topic areas.
3. Aggregate the results of the expert interviews with existing empirical derived competence models in order to gain a consolidated competence model in the topic area.
4. Development of test instruments for students' self-assessment and the evaluation of seminars and lectures in order to derive affirmative action at the LZI.

In the following sections we focus on the empirical research indicated in (1).

4.1 A Competence Model derived by Content Analysis

To develop a Competence Model by means of content analysis we adapted the strategy used in previous projects like MoKoM and KUI. Fig. 1 provides an overview of this process (Methodology B). Firstly we analyzed several Computer Science Bachelor Curricula of universities from all over the world applying the method of deductive content analysis according to Mayring (2010). Here we start with the ACM/IEEE Curriculum (ACM/IEEE, 2013) in order to firstly derive classification categories. For practical coding we used the software *MaxQDA* (www.maxqda.com).

In the ACM/IEEE Curriculum, the basis of our analysis experts defined a catalog of 18 knowledge areas to cover computer science. Each knowledge area consists of several knowledge units. For each of these knowledge units there is a description of the content and the learning outcomes. In addition, it is defined how many hours should be taken for each knowledge unit during a computer science program. In short the importance of each knowledge unit is shown.

We used these knowledge areas and knowledge units as our categories and subcategories during the content analysis. As the text corpus served different computer science bachelor curricula: The top ten of the QA Ranking[2] and in

2 http://www.topuniversities.com/university-rankings/university-subject-rankings/2013/computer-scienceand-information-systems

addition the top Universities of Switzerland, Singapore, India and eight of the Top Ten of German Universities in Computer Science[3] We decided to use these additional Universities, because we want to gain an overview of different countries. The Top Ten of the QA Ranking contains mostly Universities from the US. Because curricula often have obligatory and non-obligatory courses we decided to analyze only the obligatory courses. Otherwise there are too many variants. Additionally, our observations at the Learning Center and at our institute indicate that students' problems often begin early during the basic courses. Thus, if we want to measure competences according these courses, a look at obligatory subjects is sufficient.

After building the category system and the text corpus the minimal (single word/subject term) and maximal (paragraph) coding units were defined. During the following analysis, the coder assigned the identified coding units to the appropriate categories of the ACM/IEEE curriculum. It was possible to map all code units to our category system derived from the ACM/IEEE curriculum, so there was no need to add other categories to the system during our analysis.

It has to be mentioned that the richness in details of the curricula varies considerably. Some only contain information about the course like title and some organizational information. Other curricula provide detailed descriptions of the course contents and learning outcomes or competences. In addition the representation of the learning contents differs in its form. On the one hand curricula provide only pure lists of keywords, on the other hand some list concrete competences. Fig. 4 shows two examples for the difference in course descriptions.

3 http://www.wiwo.de/ranking-die-besten-unis-und-fachhochschulen/8046582.html

Ludwig-Maximilian University Munich[4] Course: Introduction to Programming	**Indian Institute of Technology Bombay**[5] Course: Computer Programming and Utilization
Qualification Aims: The students will be able to implement solutions for small and manageable problems algorithmically and to realize them with a high level programming language as executable programs. Furthermore, students develop an understanding of the general principles of programming and programming languages. This lays the foundation to ensure that the students (after further experiences in the course of study) may become familiar quickly and accurately with any programming language. Additional there exists a course description with the covered topics.	Description: This course provides an introduction to problem solving with computers using a modern language such as Java or C/C^{++}. Topics covered will include: * Utilization: Developer fundamentals such as editor, integrated programming environment, Unix shell, modules, libraries. * Programming features: Machine representation, primitive types, arrays and records, objects, expressions, control statements, iteration, procedures, functions and basic i/o. * Applications: Sample problems in engineering, science, text processing and numerical methods.

Figure 4: Cuttings from two different curricula

On the basis of these curricula analysis we have got an overview about CS teaching content of several universities and additionally if there exists a kind of core knowledge to be achieved by the students. Because we focus on the areas of Software Development, Software Engineering and Programming we only covered the ACM/IEEE knowledge areas: Programming Languages, Software development Fundamentals and Software Engineering during our curricula analysis.

4.2 Results of the Curricula Analysis

We now present our first results of the curricula analysis after the coding process. As mentioned before the richness in details of the curricula descriptions varies considerably. As a result we do not consider the number of occurrences

4 http://www.uni-muenchen.de/studium/studienangebot/studiengaenge/studienfaecher/informatik/bachelor1/modulhandbuch/16_mhdb_nf_informatik_60_ects_psto_07_10_2010_en.pdf [l.v. 14.02.2014]
5 http://www.cse.iitb.ac.in/page95 [l.v. 14.02.2014]

of a single coding, but only the existence of a coding for a category in a curriculum. The frequencies of curricula that address a category can be seen in Fig. 5. Fig. 5 shows that about half of these subcategories occur in 50 % or more of the curricula. For building our competence model we decide to concentrate on the coding of these subcategories, which are shown in Fig. 6.

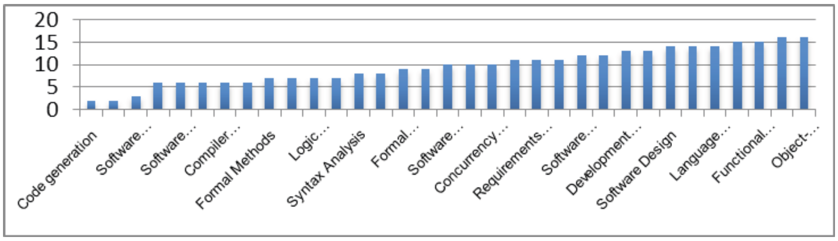


Figure 5: Results of the Curricula Analysis

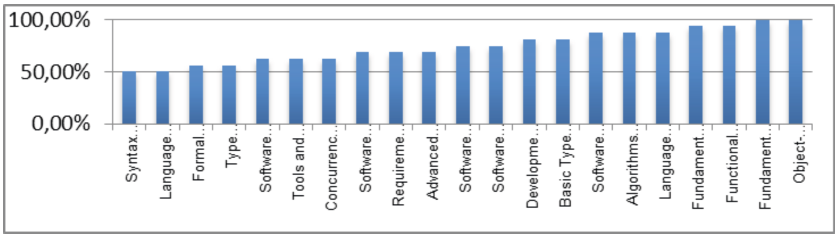


Figure 6: Subcategories occur in 50 % or more of the curricula

In a second step we compared these categories with the ACM/IEEE Curriculum again. In the ACM/IEEE Curriculum all knowledge units are classified by the categories "Core" or "Elective" where core is subdivided into "Tier-1" and "Tier-2". With these categories the ACM/IEEE Curriculum defines the relevance of the included subject areas and knowledge units for CS-curricula. Additionally a curriculum should include significant elective material (ACM/IEEE, 2013).

During the comparison of our results with such categories belonging to Tier-1 or Tier-2 we only found four subcategories, which belong to Tier-2 and which occur in not more than 50 % our curricula. These categories are: Software Reliability (3), Software Evolution (6), Program Representation (2) and Event Driven (6). During our next step, to develop the competence model, we will not consider the codings of these categories. Consequently we have to check, if our later validation of the competence model shows a lack regarding these categories. According to this validation we will add or not add the competences for these categories to our competence model. Nevertheless our results show that many curricula contain the topics the ACM/IEEE Curriculum suggests and we can say that these categories are part of a common computer science knowledge every computer scientist should have. On the basis of to such an empirically derived subject areas we will develop our competence model considering the CS-AKT described above.

4.3 Next steps – Follow up research

According to (2) and (3) a validation of the derived competence model by experts, a comparison with the IEEE Software Engineering Body of Knowledge (SWEBOK) (Society, Bourque and Dupuis, 2004) and other existing research like e.g. the model of Fuller (Fuller et. al., 2007) will be conducted. This aggregated and consolidated competence model will structure the competences in competence components and will graduate levels for each of these components. Finally, according to (4) we want to develop an e-Assessment tool for competence measurement in this specific topic area. E-Assessment will give reason for specific affirmative action at the LZI at our university and analyses e.g.: equality of learning conditions for each tested person, direct interpretation of the data, instant feedback and greater flexibility with respect to location and time. Additional reasons for an e-Assessment regarding to the aims of our Learning Center are that its use provides the opportunity to utilize such an assessment as a ubiquitous self-Assessment tool. That means students can use our assessment tool for their own to check their personal skills and identify their deficits. This is for example useful before they start a master study. An additional advantage is the opportunity to build an adaptive test or an adaptive e-Learning course later, based on the outcomes of the test. With such an instrument we can directly support our students to overcome their deficits and learning barriers with the greatest flexibility regarding location and time. For an assessment we have to create different exercises and items. At the end we need at least one item for each competence component mentioned in our competence model. In addition we have to add the aspiration level for each item.

To measure the competences, a competence model and test items are not enough, additionally a psychometric model is needed. With such a model we can map the answers of a person to the corresponding test criterion (here a competence). Here the *Item-Response-Theory (IRT)* will be applied. The reason for using IRT is, that it gives us the opportunity to measure different levels of competences. Based on our competence model, derived items and an assigned psychometric model e-assessment and measurement of competences at our LZI will be conducted.

4.4 Comparison

In contrast to the GI-experts who developed a competence model for curricula development, the results of the empirically oriented approach should be a more detailed competence model for a specific topic area. Furthermore, the competence model of the experts can be used as a framework for the concrete competences of the empirically oriented approach. For example, the competence: "Understand the characteristics of the programming paradigms: imperative, object-oriented, functional and logic programming." of the experts doesn't name the concrete characteristics of the programming paradigms. Certainly they are not important for curricula development. In contrast this information is important for developing lectures and competence assessments. Here the empirically oriented approach provides a more detailed competence model. Our first results show topics like object-oriented, functional and logic programming. In a next step we will derive competences for these topics. After this our competence model should describe the concrete competences for programming paradigms in object-oriented, functional and logic programming like the characteristics for each. All in all it refines the components of the CS-AKT-model and describes its competences in a more specific way and focused on a specific subject area.

5 Conclusion and Further Work

In this paper we present two methodologies for developing a computer science competence model. The first approach is a normative one where an expert consortium develops the competence model on the basis of existing curricula and references. This approach has the objective to develop national guidelines and recommendations for CS curriculum design in Higher Education. The results of this approach can be seen as a framework for the second methodology we present.

This second approach is an empirically oriented one and starts with a content analysis on different CS-curricula. Afterwards a competence model is build on the basis of the codings of the content analysis. This competence model should be more concrete than the expert one and can be used for the development of an instrument for competence measurement in the subject area of software development and programming. Next steps in this second approach are to derive the competences from the coding's and to validate and revise the resulting competence model. After this step we can start with the development of the assessment tools. This procedure of refinement may be applied also for other subject areas of the CS-AKT model. With our competence models we firstly provide the ability to develop guidelines and recommendations for CS curricula in Higher Education and secondly, in addition with our assessment, the ability to find out the deficits of our students in a more concrete way than by means of observation only. These concrete results will give computer science departments the opportunity to develop competence oriented curricula and specific interventions to help students in overcoming their deficits and in addition to measure the effect of these interventions accordingly.

References

Berges, M., Hubwieser, P., Magenheim, J., Bender, E., Bröker, K., Margaritis-Kopecki, M., Neugebauer, J., Schaper, N., Schubert, S., Ohrndorf, L. (2013). Developing a competency model for teaching computer science in schools. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education (ITiCSE '13) (p. 327)*. ACM, New York, NY, USA.

Denning, P. J. (November 2003). Great principles of computing. In *Communications of the ACM*, 34(11), pp. 15–20.

Fakultätentag Informatik (2004). *Recommendations for Setting Up Consecutive Bachelor's and Master's Programs in Informatics at Universities*. Resolution passed by the Plenary Session of the German Council of Informatics Departments. Retrieved from: http://www.ft-informatik.de/uploads/tx_sbdownloader/BaMaRecommendations.pdf. last access: 17.02.2014.

Fuller, U. et. al (2007). Developing a computer science-specific learning taxonomy. SIGCSE Bull.39, 4, pp. 152–170.

Hambleton, R. K., Swaminathan, H., Rogers, H. J. (1991). *Fundamentals of Item Response Theory*. Newbury Park, CA. Sage Press.

Hubwieser, P., Magenheim, J., Mühling, A., Ruf, A. (2013). Towards a conceptualization of pedagogical content knowledge of computer science. In *Proceedings of the ninth annual international ACM conference on International computing education research*, pp. 1–8.

Koeppen, K., Hartig, J., Klieme, E., Leutner, D. (2013). Competence Models for Assessing Individual Learning Outcomes and Evaluating Educational Processes – A Priority Program of the German Research Foundation (DFG) In Blömeke, S., Zlatkin-Troitschanskaia, O., Kuhn, C., Leutne, D. (Eds.), *Modeling and Measuring Competencies in Higher Education* (pp. 171–192). Volume 1 of Professional and Vet Learning. AW Rotterdam: SensePublishers.

Krathwohl, D. R. (2002). A Revision of Bloom's Taxonomy: An Overview, *Theory Into Practice*, 41(4), pp. 212–218.

Linck, B., Ohrndorf, L., Schubert, S., Stechert, P., Magenheim, J., Nelles, W., Neugebauer, J., Schaper, N. (March 2013). Competence model for informatics modelling and system comprehension. In *Global Engineering Education Conference (EDUCON'13)*, pp. 85–93.

Mayring, P. (2010). *Qualitative Inhaltsanalyse: Grundlagen und Techniken*. Weinheim: Beltz Pädagogik, Beltz.

Schaper, N. Kompetenzorientierung in Studium und Lehre (2012). *Fachgutachten für die Hochschulrektorenkonferenz*, Bonn: HRK.

Schaper, N., Magenheim, J., Schubert, S., Hubwieser, P., Bender, E., Margaritis, M., Ohrndorf, L., Berges, M. (2013). Competencies of Teaching Computer Science. In

Blömeke, S., Zlatkin-Troitschanskaia, O. (Eds) *KoKoHs Working Papers no.3,* pp. 32–34.

Society, I. C., Bourque, P., Dupuis, R. (eds.) (2004). *Software Engineering Body of Knowledge (SWEBOK)*. EUA: Angela Burgess.

The Joint Task Force on Computing Curricula Association for Computing Machinery IEEE-Computer Society 2012 (2013). *Computer Science Curricula 2013*

Tuning, Educational Structures in Europe, Competenceshttp://www.unideusto.org/tuningeu/competences.html last access: 14.02.2014.

UNESCO (2012). *UNESCO ICT Competency Framework for Teachers*. UNESCO. Retrieved from: http://unesdoc.unesco.org/images/0021/002134/213475E.pdf, last access: 14.10.2013.

Universität Paderborn (2009). *Module Handbook 2009 Version 2-Bachelor and Master Program*. Retrieved from: http://www.cs.uni-paderborn.de/fileadmin/Informatik/Institut/englishPage/ModulHandBook/Module_Handbook_GB_11_02_2013-ModuleMaster.pdf, last access: 17.02.1014.

Weinert, F. E. (2001). Vergleichende Leistungsmessung in Schulen – eine umstrittende Selbstverständlichkeit. In Weinert, F. E. (Ed.). *Leistungsmessungen in Schulen* (pp. 17–31). Weinheim: Beltz

Biographies

Kathrin Bröker is research associate and PhD-Student at the Computer Science Education Group at the University of Paderborn, Germany. There she is responsible for the Computer Science Learning Center. Her research areas are CSE and E-Learning.

Uwe Kastens is head of the Research Group Programming Languages and Compilers at the University of Paderborn. The research interest of this group centers around the design, the compilation, and the application of programming languages and domain-specific languages. He is member of the board of the German Council of Informatics Departments and Chairman of the Study Commission. He is also member of IFIP WG 2.4.

Johannes Magenheim is a Professor for Computer Science Education at the Institute of Computer Science at the University of Paderborn. His primary areas of research and teaching are CSE and E-Learning. He is member of different working groups of the German Informatics Society (GI) and member of IFIP WGs 3.1 & 3.3 and national German representative to TC3.