

A strategy to structure the learning process towards understanding of informatics systems

Peer Stechert, *stechert@die.informatik.uni-siegen.de*

University of Siegen, Didactics of Informatics and E-Learning, Germany

Sigrid E. Schubert, *schubert@die.informatik.uni-siegen.de*

University of Siegen, Didactics of Informatics and E-Learning, Germany

Abstract

Based on theoretical results towards understanding of informatics systems in secondary education, the step into practice has been done. A case study has been conducted to obtain empirical feedback, i.e., qualitative results. To permit generalising the results and a refinement of theory, a typical class has been chosen, which relates to object-oriented modelling. The exercises for understanding of informatics systems performed by students with respect to basic competencies have been evaluated. The hypothesis is that the structure of the exercises during the case study permits identifying focal points of an education model towards understanding of informatics systems, whereas misconceptions of students offer hints to refine existing theory. Therefore, we enhance theory through identifying domains of understanding of informatics systems by analysis of international curricula and recommendations to structure the learning process. These domains have to be connected by learners to see the whole picture of an informatics system. Bloom's Revised Taxonomy has been applied for classification of learning objectives.

Keywords

Exercise classes, classification of informatics systems, knowledge representation, understanding of informatics systems

MOTIVATION

Understanding of informatics systems is necessary for mature citizens. We define an informatics system as "the specific combination of hardware, software and networking facilities needed to solve some application problem" (Claus & Schwill, 2006). In such informatics systems, there are lots of networked informatics concepts and their interactions to be understood. The didactic challenge is to present a successful educational model for understanding of informatics systems that will foster networked thinking for new cognitive relations. Our approach to understanding of informatics systems combines educational research on the procedure to identify fundamental ideas of informatics (Schwill, 1997) and selected object-oriented design patterns (Gamma et al., 1995). For this purpose Stechert has developed a learner-centred classification of design patterns (Stechert, 2006ab) and a learner-centred framework implying learning objectives for understanding of informatics systems (Stechert, 2006c).

RESEARCH METHODOLOGY

Our research methodology (Figure 1) is inspired by successful researchers who investigated their theories in practice, i.e., intervention by performing field studies. Such research methodology permits critically reflecting upon research results, because it adds knowledge and insights regarding a phenomenon or problem identified by the researcher (Bassegy, 1999). In secondary education, there is the following situation specific to informatics education: one school seldom has enough courses in parallel to enable researchers to design studies involving experimental and control groups. Furthermore, informatics education depends on the school

where it takes place, e.g., because many existing curricula cover a wide spectrum of informatics but without defining mandatory parts.

We also include the Didactic System (Brinda and Schubert, 2002) in the research methodology. It consists of a) knowledge structures, b) exercise classes, c) learning software. Prerequisites of learners, methods, and concepts are represented as nodes in a graph, i.e., a knowledge structure. Exercise classes help defining levels of competencies. Construction, description and application of learning software has to be integrated into our research methodology because learning software is attractive to students and provides the opportunity of new types of exercises being both, subject matter and medium. The strength of the Didactic System is its application to different domains of informatics since 2001, e.g. object-oriented modelling, “Internetworking”, and understanding of informatics systems. It comes along with continued refinement of theory (Schubert, 2005; Freischlad & Schubert, 2006; Stechert, 2006b).

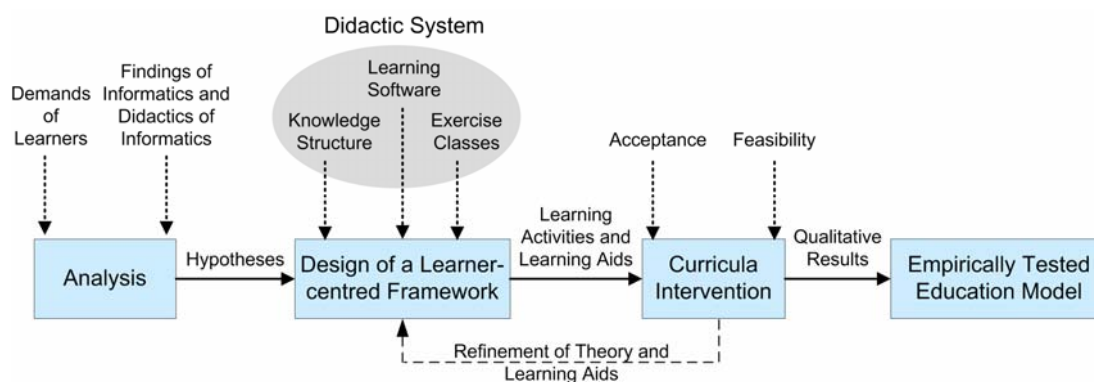


Figure 1: Design, intervention, evaluation cycle of curricula development.

During the analysis of the research field, demands of learners have been investigated. Findings of Informatics and Didactics of Informatics permit formulating hypotheses about the learning process towards the understanding of informatics systems. In combination with the Didactic System, we have designed a learner-centred framework, which consists of three dimensions. First, it describes principles of informatics systems, which have to be considered. Second, the need of a knowledge representation to ease understanding and learning is explained. Object-oriented design patterns are discussed and found to be potential in this field. Third, learning objectives according to Bloom’s Revised Taxonomy (Anderson & Krathwohl, 2001) are presented. It distinguishes between 1) factual, conceptual, procedural, metacognitive knowledge, and 2) six succeeding cognitive processes, i.e., remember, understand, apply, analyze, evaluate, create. In the next step of the research methodology, we have described learning activities and learning aids. Therefore, we are developing the learning software “Pattern Park” (Stechert, 2006a).

The effect of being researcher and teacher in one person has to be discussed: quantitative studies can suffer from such a constellation, but we aim at qualitative results. The case study has offered results about general feasibility and acceptance by the learners. The evaluation leads to a refinement of theory and learning aids including exercise classes: “Previously developed theory is used as a template with which to compare the empirical results of the case study” (Bassey, 1999, p. 31). The aim of the research is an empirically tested education model.

ADAPTION OF THE EDUCATION MODEL TO THE FIRST CURRICULUM INTERVENTION

The education model was designed to promote conceptual and procedural methods for understanding of informatics systems. It was based on object-oriented modelling, which is a dominant concept in upper secondary education. In contrast to the implementation of several programs, the emphasis was on a holistic view on informatics systems, which integrated evidence with theory. The major goal of the unit was for the students to avoid trial-and-error approaches to informatics systems, and to investigate how behaviour of informatics systems depends on networked informatics concepts. For understanding of informatics systems, we have divided the learning process into three main domains (Si) that are alternating:

- S_A: Understanding of essential aspects of observable behaviour of the informatics system,
- S_B: Understanding of essential aspects of the internal structure of the informatics system that are based on fundamental ideas,
- S_C: Understanding of construction details during the development of a concrete realization (implementation details).

During the unit on strategies for understanding of informatics systems, we focussed on S_A and S_B, because we are aiming at basic competencies necessary for mature citizen but not at skills for software developers. We exemplarily describe learners' activities towards access control according to the levels of Bloom's Revised Taxonomy and apply Iterator design pattern and Proxy design pattern, which are based on object aggregation. Iterator provides a way to access an aggregate like a queue sequentially. Proxy describes a placeholder to control access to another object (Gamma et al., 1996). This short description also shows that access control is inherent in both design patterns, even if it is not the central idea in the Iterator. Examples for sub-objectives of S_A are

- S_{A,1}: Understanding of the fundamental idea iteration for a queue data structure via accessing the Iterator instead of the queue.
- S_{A,2}: Understanding of the fundamental idea access control for different access rights with Proxy design pattern.

S_A has to be realized by investigating input and output, which is done in informatics experiments to disclose unexpected behaviour (e.g., faults). Examples for sub-objectives of S_B are:

- S_{B,1}: Understanding of iteration by designing an object-oriented model of a queue data structure and an Iterator as validated in S_{A,1},
- S_{B,2}: Understanding of interfaces and inheritance by applying the Proxy design pattern.
- S_{B,3}: Understanding of access control by designing an object-oriented model of a Proxy as validated in S_{A,2},

For S_B, a documentation of the system, different kinds of modelling (data, functional) and diagrams (class, object, sequence, state diagram) should be used. An example for a sub-objective of S_C is:

- S_{C,1}: Understanding of programming parts of the designed object-oriented model including Proxy design pattern.

THE FIRST CASE STUDY

We have conducted a unit on strategies for understanding of informatics systems based on the education model in autumn, 2006. In the informatics course at upper secondary level, 23 students at age 17 have taken part. Altogether, there have been 13 lessons; three lessons per week. The students' previous informatics knowledge had comprised object-oriented modelling and programming, class-diagrams, and list data structure. The course was part of a regular upper secondary level without special emphasis on natural sciences or informatics education. From our point of view, the students were average achievers in informatics. The advantage of object-oriented modelling is that it is widely used and accepted in international informatics education. Therefore, the case study is potential with respect to results that can be generalised to improve theory. Our approach to understanding of informatics systems is not necessarily limited to upper secondary education. Assuming a spiral curriculum, many competencies can already be learned before, and further objectives based on object-oriented modelling can build on previous knowledge.

The structure of the course has been as follows: in one lesson, the behaviour of an informatics system is analysed with respect to fundamental ideas of informatics that cause the behaviour. In the succeeding lesson, the internal structure of the informatics system has been investigated, because it is a representation of fundamental ideas (Stechert, 2006c). The third lesson is reserved for combination of both previous aspects and repetition. This pattern repeats every week. Step by step, the network of fundamental ideas, that cause the behaviour of the informatics system, has been evaluated and learned. We investigated two systems: medical software, and a program example realising access control as it is known from operating system.

One typical exercise was to experiment with the behaviour of an informatics system. To describe the procedure of the experiment, we will also assign cognitive processes and the knowledge domains according to Bloom's Revised Taxonomy. The procedure model of an experiment had to be applied (apply; procedural knowledge). It consisted of six learning strategies, which are described as sub-objectives of $S_{A,2}$:

1. Answer questions about the intended behaviour of a given informatics system (remember; factual knowledge),
2. Outline the procedure for conducting the experiment (understanding; procedural),
3. Document the data observed (apply; factual knowledge),
4. Examine data and identify informatics concepts, e.g., access control (analyze; conceptual knowledge),
5. Establish a set of special cases to be tested and evaluate them (evaluate; conceptual knowledge),
6. Create a series of hypotheses for using the informatics system in a real-world or more complex scenario (create; factual knowledge).

Another typical exercise was to explore the internal structure of the informatics system given the source code to derive a class diagram. An extension was to derive sequence diagrams for dynamic aspects of special scenarios given the class diagram. The procedure model for the experiment had to be applied (apply; procedural knowledge). It consisted of six learning strategies, which are described as sub-objectives of $S_{B,3}$:

7. List involved classes, objects, and answer questions on access control (remember; factual knowledge),

8. Classify operations and classes, e.g., whether and how far they are necessary for access control (understand; conceptual knowledge),
9. Arrange the correct relations between the participating classes in a class diagram (apply; conceptual knowledge),
10. Examine the class diagram and identify how informatics concepts are implemented (analyze; conceptual),
11. Check special cases and involve testing for internal inconsistencies (evaluate; conceptual),
12. Plan to reorganise and reuse the internal structure for a different application (create; conceptual).

Work sheets and solutions created with a software tool for class diagrams were collected. To identify misconceptions, the collected sheets and files were analysed for typical errors and working strategies. Lessons and exercises were discussed with the teacher of the course. All in all, students had to perform 47 exercises during the learning process, i.e., 31 during the lessons and 16 in the final test. For students, there was also a questionnaire aiming at the question whether students have accepted the unit and demanding for knowledge about their own cognition (metacognitive knowledge domain). One result was that 82% of the students agree that the unit was valuable towards application of informatics systems. Finally, there was a guided interview with the informatics teacher who confirmed that the degree of difficulty was adequate.

RESULTS OF THE CASE STUDY

Enhancement of theory towards understanding of informatics systems

The aim of the article is a structure of exercise classes in the learning process that is potential for understanding of informatics systems and adequate with respect to general education. Observations of the learning process during the case study have disclosed the need for enhanced theory, because difficulties of learners are caused by a lack of knowledge about either informatics concepts, their realization in typical informatics systems or the possibility to describe the structure of the system. In particular, it is necessary to have exercises that succeed each other to avoid misconceptions of learners, i.e., in a sequence of exercises one task corresponds to a basic competence that is necessary to accomplish the following exercise. By exercise classes we do not want to define distinct demands for understanding of informatics systems but a system of classes potential to achieve basic competencies. Curricula and recommendations for informatics education demand an understanding of informatics systems, but they only specify learning objectives and not comprehensive competencies. The authors have analysed different national and international curricula including the UNESCO-IFIP curriculum for ICT in secondary education (2002). To order learning aspects towards understanding of informatics systems, we have structured the domain based on criteria indicated in the research results of the authors, i.e., models of layers, design patterns, and networked fundamental ideas. Therefore, we have identified the following categories: The first category comprises typical representatives of informatics systems, the second common models to structure informatics systems. Last, fundamental ideas have been identified that are suitable for an education model for understanding of informatics systems.

1. Typical representatives of informatics systems

Topics in the first module of the ICT curriculum comprise different software applications students need to cope with, e.g., databases, spreadsheets, and word processing systems. In addition to such software applications, students “should understand how computers and the basic operating system work and demonstrate that the computer is under their control. They should not be mystified by computers”

(UNESCO, 2002, p. 68). To distinguish between those different aspects, we subdivide typical representatives of informatics systems into “system software” and “application software”.

II. Structuring informatics systems

Potential models of informatics systems should be learned. One common model is the “von Neumann architecture”. Such architecture is a concept to describe structure and functioning of an informatics system, e.g., separation of storage from the processing unit. A single storage structure holds both instructions and data. An informatics system following the von Neumann architecture is a universal computing machine that can process any arbitrary, well-formed sequence of instructions.

To describe computer-computer interaction as well as the gap between human-computer interaction and binary machine code, i.e., between virtual machine and real machine, “models of layers” are essential (Schubert, 2005). Therefore, they are a further subcategory. In particular, students should be able to assign their activities to different levels like hardware, systems software, and application software.

Smaller units to structure informatics systems are “design patterns”: A “design pattern identifies the participating classes and instances, their roles and collaborations, and the distribution of responsibilities“ (Gamma et al., 1995, pp.3-4). Once identified in the field of object-oriented modelling as solutions to recurring design problems, they have been defined for many areas of informatics, even to describe structure and functioning of hardware, e.g., for register-transfer-level and system-level design. These software and hardware patterns, both, have in common that their description is based on object-oriented modelling techniques. They are used during the development to ensure qualities of informatics systems. For the learning process, networked fundamental ideas in object-oriented design patterns have been identified in a learner-centred classification (Stechert, 2006b). Thus, to understand the behaviour of an informatics system design patterns offer a potential knowledge representation. For example, access control is essential to almost all informatics systems, where different access rights have to be defined. Students know it by their own real-life experience. The Proxy design pattern realizes the fundamental idea access control and is a domain specific external representation offering a description of its structure and processes. The Iterator pattern represents iteration, access control (see SA,1), and can be applied for recursion. And in addition, design patterns are designed to be networked, which is of high importance for the development of an education model. The scenarios and informatics systems can be extended subsequently.

III. Networked fundamental ideas of informatics

According to their definition, the fundamental ideas of informatics are observable in different areas of informatics and, thus, a network of fundamental ideas can be identified in informatics systems. To make learners aware of these relations permits overcoming cognitive barriers build by complex informatics systems: the concepts in the ACM K-12 model curriculum are described to achieve “competency in [...] networking fundamentals” (ACM, 2006, p. 17). The hypothesis is that there are some fundamental ideas, which are necessary for the learning process towards understanding of informatics systems. In particular, the curriculum intervention discloses the need to identify informatics concepts supporting understanding of informatics systems and, if necessary, to prove that they fulfil the criteria for being a fundamental idea. States and state machines are fundamental ideas, which are important to understand systems. Further potential candidates are authentication, integrity, availability and reliability in the context of dependability and security (cp.

ICT curriculum) as well as interfaces, information hiding, emulation, fairness, and human-computer-interaction.

Refinement of the structure of the learning process

In an education model for understanding of informatics systems, systematic exploration and evaluation of an informatics system is central. The starting point of the case study was the distinction between three domains S_A , S_B , and S_C . Application of selected examples of the learning software “Pattern Park” focussed on combining real-world example and internal structure. However, the analysis of the case study including the final test has shown that these domains have to be refined. In particular, the experiment to investigate the behaviour of an informatics system was very difficult for students (cp. previous chapter). Evaluation of work sheets with topics queue and iteration ($S_{A,1}$) has shown that they were hardly able to formulate adequate hypotheses, which is a prerequisite to conduct an informatics experiment.

To describe exercises, we therefore propose the following more fine grained distinction between the domains:

S_A . Observable behaviour

Behaviour of an informatics system can be investigated by observation and in informatics experiments. Therefore, hypotheses have to be proved in experiments. Students apply a concrete informatics system, e.g., running a software application that implements and illustrates fundamental ideas of informatics by its behaviour. Experiments are repeatable. Animations of behaviour can be provided by learning software. Exercises can also be applied to sensitize students to unexpected behaviour. Subcategories comprise requirements analysis, and description of use cases.

S_{AB} . Combination of behaviour and internal structure

Learners describe the observed behaviour during an experiment by functional models. During the learning process, one task was: *“Given definition and structure of queue data structure on the one hand, and the structure of the combination of Iterator and queue on the other hand, which behaviour would arise during the use of software based on both, respectively?”* Fundamental ideas, necessary concepts, and design problems are identified and represented externally to visualize their structure. The learners are able to explain, why a certain internal structure has been chosen with respect to the intended behaviour of the informatics system.

S_{AC} . Combination of behaviour and construction of a concrete realization

By systematic testing, the behaviour of an informatics system can be investigated. Test data corresponds to the behaviour, whereas checkpoints have to be set in the source code. During the learning process, one task was: *“Change the algorithm of the Iterator pattern. Classify the behaviour of the system by the order of the resulting sequence of patients in the queue, i.e., the same order means that the results belong to the same equivalence class.”*

S_B . Internal structure

In general, the internal structure is only known by developers but not by users. It can be investigated rather through analysis of the components than through experiments. Learners apply different diagrams, e.g., class, object, state, and sequence diagrams to visualize the internal structure. It is necessary to consider dynamic and static representations. Design patterns as solutions to previously identified problems can be applied and connected.

S_{BC}. Combination of internal structure and construction of a concrete realization

Structuring of implementation details is necessary in order to make a specification. Such structuring elements are below the abstraction of design patterns, e.g., idioms depend on a concrete programming language but have a predefined structure.

S_C. Construction of a concrete realization

Construction of a concrete realization needs programming skills and specific knowledge depending on a programming language. Studies show that learners often fail to construct and design programs even if they know essential programming concepts, because they do not see the whole picture (McCracken, 2004). They fail to network the concepts. Implementation details support advanced understanding of informatics systems, but for general education we will not focus on this aspect.

These exercise classes bring together different views on informatics systems to form the whole picture. Students construct their knowledge and need to prove their cognitive model by matching these views. In particular, misconceptions can be reduced in the learning process because students learn to transfer their knowledge. To achieve basic competencies, it is not sensible to regard every combination of views. In particular, we do not need the categories S_C and S_{BC} , because the focus is on general education, not programming. Instead, we want to demystify behaviour of informatics systems, thus, the categories S_A , S_{AB} , and S_{AC} are mandatory. We will include S_B , because describing static and dynamic processes is needed in general education.

Implications for exercises of the first case study

The exercises of the curriculum intervention have been created to set emphasis on the bridge between behaviour and internal structure of an informatics system and fundamental ideas. We have created the exercises to sensitise learners to the new challenges coming along with informatics systems. We have put emphasis on real-life experiences of learners to make the unit more exciting. For the evaluation, we have to match the exercises of the curriculum intervention to both enhanced theory with respect to international curricula and recommendations, as well as the more fine-grained subdivision of views on informatics systems. Then, it becomes obvious, which parts are underrepresented in the current education model.

Given the dimension “typical representatives of informatics systems (I.)”, “structuring informatics systems (II.)”, and “(networked) fundamental ideas (III.)”, as well as the categories S_A , S_{AB} , S_{AC} , and S_B , we will order the exercises performed in the curriculum intervention (cp. previous chapter) in Table 1. Some assignments of exercises are not unique, since some exercises combine different aspects, e.g., to arrange a class diagram (S_B) for access control (III.) results in the construction of a structure similar to the Proxy design pattern (II.). Then we have assigned the exercise to S_{B-II} , since a design pattern out of the collection of design patterns that is potential for the learning process represents a structuring element as well as a network of fundamental ideas. The assignments need interpretation. There have been exercises including all categories from typical representatives, via design patterns to networked fundamental ideas. However, analysis of curricula and recommendations has shown that investigation of behaviour of typical representatives has to be strengthened.

S _A -I	S _{AB} -II	S _{AB} -III	S _B -II	S _{AC} -III
1, 2, 3, 6	8	4, 5, 10	7, 9, 12	11

Table 1: Assignment of exercises of the curricula intervention (cp. chapter on the case study) to domains of the enhanced theory

The assignments show that the education model has put emphasis on the category structuring of informatics systems (exercises 7, 8, 9, 12), but we only focused on design patterns and have to include models of layers and von Neumann architecture. To model hardware components in an object-oriented way is relatively simple, because they are pre-structured objects and no further previous knowledge is needed. Furthermore, we see that S_{AC} (exercise 11) is underrepresented, although it is a very potential category because changes in the system can be observed by students. Many exercises are based directly on fundamental ideas (exercises 4, 5, 10, 11) and indirectly through design patterns (exercises 7, 8, 9, 12).

We conclude that a clear distinction between the learning domains S_i and emphasis on further aspects of understanding of informatics systems that have been identified in the international curricula lead to a refinement of the described procedure of informatics experiments with informatics systems: The case study has shown that students cannot experiment without experience, because creating hypotheses about a concept is difficult, but necessary: “When the task is an unfamiliar problem...then, to *understand conceptual knowledge* is a prerequisite to being able to *apply procedural knowledge*” (Anderson & Krathwohl, 2001, p. 77). Therefore, the exercise has to be split: The first part should emphasise on observation that results in hypotheses about networked fundamental ideas. The observation has to be documented. It is a precondition of gaining hypotheses instead of conjectures. In particular, the input and starting situation can be analyzed in combination with the output of the documentation with respect to qualities and concepts like fundamental ideas. The second part emphasises creating hypotheses about behaviour, planning the experiment, and conducting it in a repeatable way according to known concepts that results in hypotheses about the internal structure. Similar refinements according to the fine-grained distinction of views can be done with the second procedure, which has been discussed in chapter of the case study, and the remaining exercises. The refined structure of the learning process permits creating sustainable exercises that combine different views on informatics systems and different principles of informatics systems at the same time: so, exercises will be very tightly focused.

SUMMARY AND CONCLUSIONS

In this article, we have presented a strategy to structure the learning process towards understanding of informatics systems. Thus, we have refined and validated a theoretical education model after conduction of a case study. In particular, the empirically tested exercises of the curriculum intervention have disclosed the need to apply exercises of different domains with respect to systems and to identify informatics concepts supporting understanding of informatics systems. To achieve basic competencies, analysis of curricula and recommendations has shown that emphasis has to be put on typical representatives during the next curriculum intervention.

Such refined structure of the learning process implies setting the focus of the next curriculum intervention to the cognitive approach of students creating hypotheses about principles of informatics systems. Tasks will include a clear description for students of what to observe as an approach to informatics experiments. In particular, the new cognitive relations are independent of knowledge on object-

oriented modelling. Hence, further research has to include concepts dominant at lower secondary education, e.g., models of layers and the von Neumann architecture, which should be discussed according to the analysed curricula. Furthermore, the approach based on object-oriented modelling comes along with dominant errors of this domain, so analysis of typical errors does not disclose knowledge about misconceptions with respect to understanding of informatics systems, e.g., whether there is a typical observation error in experiments. We need more adequate means to get insights of students' conception. So, we are currently performing preliminary studies involving the think aloud method. In particular, we aim at finding cognitive barriers related to abstraction, formalization, and generalization.

REFERENCES

- ACM (2006) Association for Computing Machinery. A Model Curriculum for K–12 Computer Science. Final Report of the ACM K–12 Task Force Curriculum Committee. Second Edition. New York.
- Anderson, L. W. and Krathwohl, D. R. (eds.) (2001). *A taxonomy for learning, teaching and assessing: A revision of Bloom's Taxonomy of educational objectives*. Addison Wesley Longman, New York.
- Bassey, M. (1999) *Case study research in educational settings*. UK: Open University Press.
- Brinda, T. and Schubert, S. (2002) Didactic System for Object-oriented Modelling. In Proceedings of Networking the Learner. *Computers in Education*. Edited by D. Watson and J. Andersen. Kluwer, Boston, pp. 473 – 482.
- Claus, V. and Schwill, A. (2003) *Duden Informatik*. Duden Verlag, Mannheim.
- Freischlad, S.; Schubert, S. (2006) Media Upheaval and Standards of Informatics. In: IFIP TC3 / WG 3.1, WG 3.3 & WG 3.5 Joint Conference "Imagining the future for ICT and Education". Alesund, Norway.
- Gamma, E.; Helm, R.; Johnson, R. and Vlissides, J. (1995) *Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA.
- McCracken, W. M. (2004) *Research on Learning to Design Software. Computer Science Education Research*, Edited by S. Fincher, and M. Petre. Taylor & Francis, London, p. 155-174.
- Schubert, S. (2005) From Didactic Systems to Educational Standards. In *8th IFIP World Conference on Computers in Education (WCCE 2005)*. Edited by B. Samways, Documents/397.pdf.
- Stechert, P. (2006a) Informatics System Comprehension - A learner-centred cognitive approach to networked thinking. In: *IFIP TC3 / WG 3.1, WG 3.3 & WG 3.5 Joint Conference "Imagining the future for ICT and Education"*. Alesund, Norway.
- Stechert, P. (2006b) Informatics System Comprehension - A learner-centred cognitive approach to networked thinking. In: *Education and Information Technologies*, ISSN: 1573-7608, Springer Netherlands.
- Stechert, P. (2006c) Understanding of Informatics Systems – A theoretical framework implying levels of competence. In: *6th Baltic Sea Conference on Computing Education Research – Koli Calling*. Koli, Finland.
- Schwill, A. (1997) Computer science education based on fundamental ideas. In Proceedings of Information Technology - Supporting change through teacher education. Edited by D. Passey and B. Samways. Chapman Hall, pp. 285 – 291
- UNESCO/IFIP (2002) *Information and Communication Technology in Secondary Education – A Curriculum for Schools*. Edited by T. van Weert. Paris: UNESCO, [WWW document, 01.15.2007] URL: <http://unesdoc.unesco.org/images/0012/001295/129538e.pdf>

Biographies



Peer Stechert was born in 1979. He received his diploma in informatics (2005) from the University of Lübeck. Currently, he is teaching informatics at the University of Siegen and working towards his Ph.D. His research interests are concepts for understanding of informatics systems in secondary and higher education.



Since 1979 **Sigrid Schubert** has taught informatics in secondary, vocational and higher education. She received her diploma in physics, her doctoral degree in informatics and has been professor of "Didactics of Informatics and E-Learning" (Universities Siegen and Dortmund, Germany) since 1998. She is a member of IFIP TC3 and the Working Groups on Secondary and Higher Education.

This paper was presented at IMICT 2007: IFIP WG3.1 & WG3.5 Joint Conference on Informatics, Mathematics, and ICT: a 'golden triangle'. College of Computer and Information Science, Northeastern University, Boston, Massachusetts, USA 27th – 29th June 2007

Copyright Statement

Copies of this document, electronic or otherwise, may NOT be made without the express permission of the first-named author.