

Informatics System Comprehension

A Learner-Centred Cognitive Approach to Networked Thinking

Peer Stechert

Didactics of Informatics and E-Learning
University of Siegen, D-57068 Siegen, Germany
stechert@die.informatik.uni-siegen.de

Abstract

The author presents results of research on informatics education with emphasis on system comprehension for 21st century local, national and global needs. Learners have to create a sustainable cognitive model of a computer to demystify such an informatics system, which can only be reached by fostering system comprehension. The underlying hypothesis of this article is that knowing fundamental ideas of informatics and their combination in a system help learners to develop a cognitive approach to informatics systems. Particularly, the development of networked thinking as a cognitive precondition for mental models of systems is focused by this article. We will contribute to the question, what kind of comprehension and how far system comprehension can be supported. Assuming two pillars of the subject informatics, i.e. informatics modelling and comprehension of informatics systems, object-oriented design patterns join both. Knowing about multifaceted interdependencies between components of a system and the cognitive analysis of such a system is of great value to overcome the tendency of searching for a single cause of an effect. With this in mind, the author offers a theoretical basis why design patterns are an essential component of the subject informatics at secondary level. However, new media need a new cognitive approach. Regarding the Didactic System introduced by Brinda and Schubert, exploration modules are an appropriate way to support teaching and learning design patterns in practice. In this article, a current project developing an exploration module to introduce design patterns with emphasis on system comprehension to learners at upper secondary level is described.

Keywords: Exploration Modules, Learning Model, Knowledge Representation, System Comprehension, Networked Thinking

Motivation

This article is structured as follows:

First, we give a short motivation for the assumption that understanding background processes will help learners for 21st century local, national and global needs. Therefore, we will present the fundamental ideas of computer science, first introduced by Schwill as an educational principle (Schwill, 1997). Note that for consistency reasons we will refer to these ideas as fundamental ideas of informatics or fundamental ideas, for short.

Afterwards, we will bridge the gap between fundamental ideas and system comprehension by regarding design patterns. Classifying design patterns that contain fundamental ideas fulfilling a breadth issue of informatics as a sci-

ence, we lead over to system comprehension. Therefore, we show that design patterns have a strong systemic quality. Based on the educational value of fundamental ideas of informatics we present a classification of design patterns from learner's perspective.

From a historical perspective, informatics has been seen as a science considering a single computing machine as its theoretical basis. Since the 1990s, regarding computers as single machines is no longer appropriate. Generally, they are part of networks and taking part in the Internet.

Regarding the IFIP-publication "Information and Communication Technology for Secondary Education, A Curriculum for Schools", "students should be able to differentiate between the basic components of a computer system [...]. They should be aware of the connectivity of computers in a local and an external network and be familiar with the appropriate functions of the networks can give" (van Weert, 2000, p. 57).

Magenheim takes into consideration the process of construction, modelling, deconstruction, re-engineering, and anticipating socio-technical aspects of systems (Magenheim, 2005).

Aim of our research is a learner-centred cognitive approach to informatics system comprehension. In the following we will define what we mean by the term informatics system.

Definition: An **informatics system** is the composition of hardware, software, and network infrastructure to solve problems including all non-technical aspects (in the application domain) like systems design, user qualification, security issues, and consequences of use (Claus and Schwill, 2003, p. 301). The term informatics system is used to avoid overemphasis of technical aspects.

Consequently and consistent with the IFIP definition, informatics is "the science dealing with the design, realization, evaluation, use, and maintenance of information processing systems; including hardware, software, organizational and human aspects, and the industrial, commercial, governmental and political implications" (van Weert, 2000, p. 9).

To set the theoretical basis for further discussion, we will present the concept of fundamental ideas of informatics in the following (Schwill, 1997). Fundamental ideas of informatics orient the learning process towards the structure of informatics. Reflecting on fundamental ideas in the design process of informatics systems, Schwill identifies three master ideas for which the criteria of fundamental ideas hold: algorithmization, structured dissection, and language.

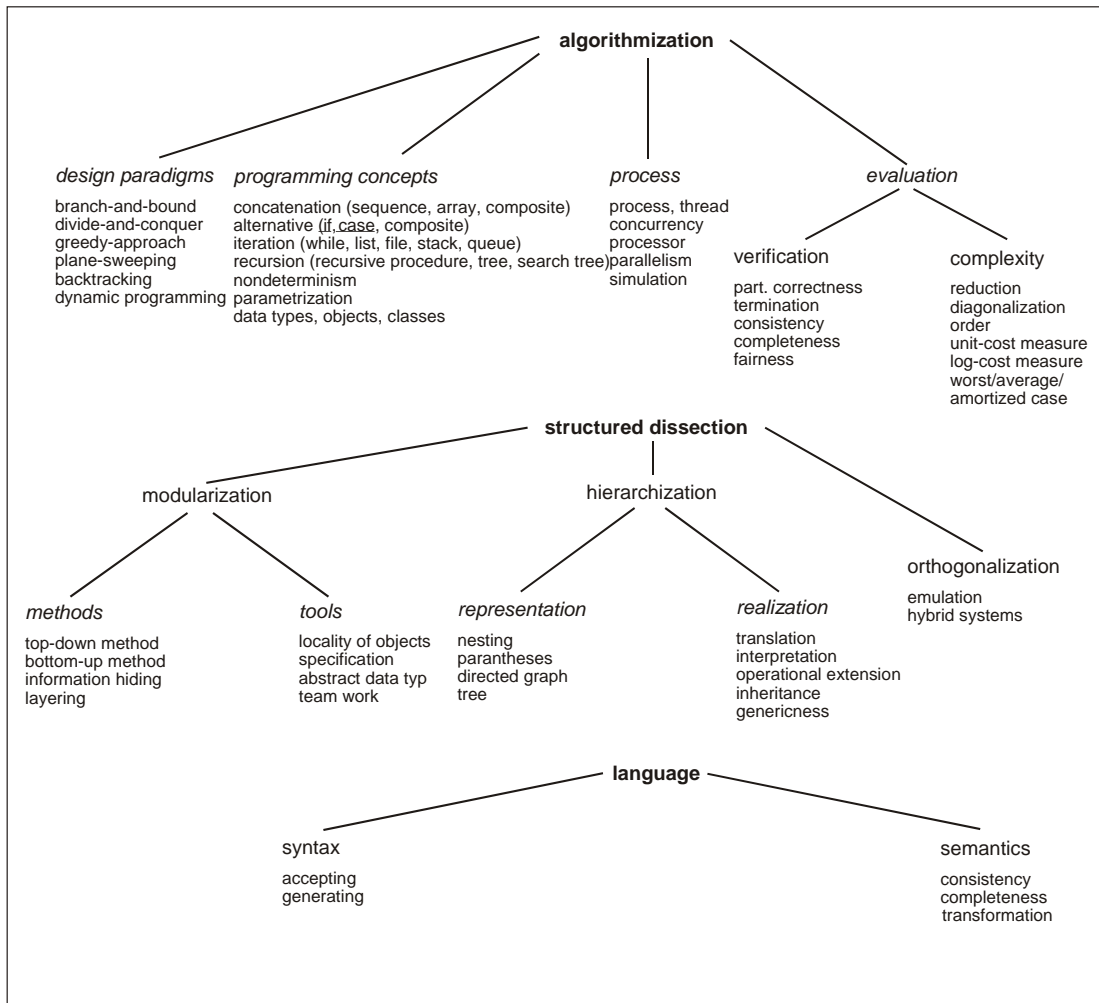


Figure 1: Fundamental Ideas of Informatics¹

The collection of fundamental ideas given by Schwill (see Figure 1, (Claus and Schwill, 2003, p. 299)) is far from complete and the placement of some fundamental ideas in the trees is debatable, because the assignment of ideas to master ideas is not unique. As proposed by Schwill, we can regard them as examples. However, the master ideas algorithmization, structured dissection, and language are consensus with regard to their educational value. Including fundamental ideas in the learning process means to teach and learn ways of thinking that permit non-specific transfer. Instead of manual skills that only permit specific transfer, learners are able to recreate new representations of their knowledge permanently. Thus, topics concerning informatics – or their underlying principles – can be evaluated based on these criteria with respect to their relevance to the subject informatics. This paper contributes to the question how to modify curricula in order to assign a central role to fundamental ideas.

¹ Names written in italics have been added for systematization only and denote groups of ideas but are not ideas themselves

Concept to Foster System Comprehension

In this article we will present the result of research in Didactics of Informatics, i.e. a concept to foster system comprehension in the subject informatics at secondary level.

Therefore, we have to consider three parties:

- A) the learner,
- B) Didactics of Informatics as a discipline, and
- C) Informatics as a science.

The parties A and B are interested in strategies of learning and (new) cognitive approaches, whereas B and C cope with informatics systems at a scientific level. Regarding informatics systems, structure, functioning, and principles of systems are of interest.

The author is convinced that topics like architecture of computers and their building principles including representation of data, operations, and function units contribute to system comprehension. In particular, further considerations concerning structure and order of function units like layered architectures and object-oriented architectures are essential, whereas former approaches like block diagrams fail describing the network aspect of informatics systems.

Taking the object-oriented approach, inheritance hierarchies, the model-view-controller-architecture (MVC) as an object-oriented equivalent of a layered architecture, and the combination of object-oriented design patterns to build a system are of great importance with regard to systems. Object-oriented design patterns have been identified by software developers as general recurring solutions to common problems in object-oriented software design that offer a high level of abstraction and a new knowledge representation to develop networked thinking.

Building on the high level of abstraction, we will discuss a design pattern approach to system comprehension in this article. Considering design patterns, we will identify fundamental ideas of informatics inherent in them, which are important in the fields of activities of B and C and research the relationships and connections between them. We assume that learning single fundamental ideas of informatics does not contribute to system comprehension in an adequate manner. In the context of design patterns, we will use them to describe functioning of informatics systems and heuristic knowledge about the design of such systems. Thus, by fostering comprehension of informatics system we attain an additional aim: we orient the subject towards the science informatics by putting emphasis on networked fundamental ideas. Therefore, we assume that a broad spectrum of fundamental ideas, i.e. including all master ideas, is adequate for upper secondary education.

Thus, we have to build the cognitive bridge between fundamental ideas and system comprehension and there is potential for networked thinking on learner's side in a learner-centred cognitive approach. Hence, the didactic challenge is to present a successful learning model to system comprehension that will foster networked thinking.

Design Patterns from the Aspect of System Comprehension

From a didactic view, design patterns as best practice solutions of common problems offer educational advantages. They represent good and elegant solutions and they are representatives of heuristic knowledge. Hence, they are valuable learning materials. However, we will focus on design patterns in the context of systems. Thus, we notice that design patterns describe decentralized object-oriented models: Guzdial claims that learners often have a centralized mindset, i.e. they prefer centralized models that are easy to understand and efficient, but contrary to the rules of object-oriented modelling (Guzdial, 1995). Therefore, design patterns help to overcome this problem. Decentralized models lead to a better understanding of concurrent and distributed processes like collaboration of objects that are responsible for themselves. Thus, a decentralized mindset is important to analyze systems.

In conventional lessons based on design patterns, fundamental ideas inherent in design patterns were not used as an opportunity to learn a broad spectrum of fundamental ideas. Nevertheless, fundamental ideas are attractive to learners because of simplicity, efficiency, and elegance.

Consequently, we want to compensate known disadvantages of system comprehension by offering design patterns as knowledge representation

1. to learn essential principles and heuristics, and
2. to learn networked fundamental ideas.

Hence, the cognitive approach based on design patterns reduces complexity by classification of knowledge networks.

Schubert presents further aspects of systems and links design patterns to the Didactic System consisting of exercise classes, exploration modules, and knowledge networks (Schubert, 2005). It is also part of our research to connect data structures, algorithms, and basic principles of object-oriented modelling with system comprehension. By microstructure of a system the fact is meant that design patterns and the combination of design patterns form subsystems, whose combinations form the whole system. The macrostructure of a system denotes the architecture of the system, which is based on layers.

Concluding that design patterns and models of layers contribute to comprehension of informatics systems, four levels should be integrated into the knowledge network (S_i) of lower secondary education:

- S_1 : “the fundamental concept of recursion with the pattern Composite without models of layers,
- S_2 : the two-layer-model of architecture, which separates the GUI (View) from other components,
- S_3 : the pattern Facade,
- S_4 : the combination of Facade and Composite.” (Schubert, 2005, p. 4)

At upper secondary, five additional levels have to be integrated:

- S_5 : “the three-layer-model, which separates the GUI (View), the application of inference machine (Control) and the database (Model) as macrostructure of the architecture,
- S_6 : the pattern Observer,
- S_7 : the network of patterns as microstructure of the architecture,
- S_8 : the pattern Proxy,

S₉: the combination of Proxy, Observer, Facade and Composite.”
(Schubert, 2005, p. 4)

With this in mind, we will analyze and structure the learning process of system comprehension. By system comprehension, we mean that learners should have knowledge and hypotheses about causes and effects of errors, involved components and objects. Furthermore, learners have to know initial points and states of components to analyze system behaviour from predefined states.

Considering an informatics system, there are three points of interest: external behaviour, internal structure, and further internal properties that specify a system (Claus and Schwill, 2003). The advantage of a design pattern approach is the existing specification of those patterns. Hence, learners can investigate each point in a learner-centred cognitive approach. Later, learners investigate variations of single design patterns and their combinations to permit transfer of previous knowledge.

With regard to system comprehension, there are two aspects of design patterns to be considered. First, they demand for external context, i.e. they are part of an object-oriented model. Second, they are complex enough to be seen as a subsystem forming larger subsystems by combination with other design patterns. Hence, the Composite pattern is an informatics way to teach recursion, whereas Fibonacci numbers are a mathematical representation. The demand for variable context allows learners to develop a cognitive model for fundamental ideas of informatics as a part of an informatics system.

Knowing a set of design patterns and designing a system, their context can be assumed to be known, which permits focusing on a subsystem and results in better solutions. Moreover, during analysis of a system knowing such subsystems and especially its external behaviour provides knowledge about structure and functioning of the surrounding system.

Besides, the classification of design patterns given by Buschmann et al. (Buschmann et al. 1996) based on levels of abstraction, i.e. distinguishing between idioms, design patterns and architectures, illustrates interdependence of different components. For example, the Observer design pattern can be identified as a part of the Model-View-Controller Architecture. This example shows the essential concept of layers and their connection with design patterns. Two layers that realize a separation of data and view respectively three layers that realize a separation of data (model), view, and aspects of interaction (controller) form the basis for modularization.

Regarding the fundamental ideas and linking them to design patterns, we have a means to introduce a system into the learning process understandable by learners: we can ensure educational coherence between the fundamental ideas inherent in the design patterns as subsystems and the principles of the whole system.

Learners' Classification of Design Patterns

The choice of adequate design patterns for a learner-centred cognitive approach is sophisticated. The design patterns described in literature – particu-

larly by Gamma et al. (Gamma et al., 1995) – form the pool from which we will make a classification. Note that we do not want to create new design patterns for the learning process only, according to a propaedeutic to Informatics as a science.

For our classification scheme we will investigate the following criteria:

1. Characteristics of design patterns as systems:
 - a. external behaviour,
 - b. internal structure,
 - c. properties,
2. The design pattern has to be based on fundamental ideas; particularly it should be based on (all) master ideas, respectively the selection has to cover all master ideas without overemphasis of a single master idea:
 - a. algorithmization,
 - b. structured dissection,
 - c. language,
3. Level of complexity.

Considering design patterns, we distinguish between external and internal behaviour with respect to systems. Interfaces and abstract classes describe external behaviour. To visualize external behaviour, the class diagram of the Unified Modeling Language (UML) is adequate (OMG, 2005). By internal behaviour, we mean objects and object interaction that can be visualized by object diagrams and sequence diagrams.

In the following, we will investigate two design patterns with respect to their suitability for the learning process. Therefore, we will compare Composite and Bridge based on the classification given above. That is, considering the master ideas algorithmization, structured dissection, and language connected with the description of a system, i.e. external behaviour, internal structure and internal properties. We will start with the fundamental ideas, because they have an impact on the analysis of the system. The level of complexity will be discussed during the final evaluation.

With regard to Gamma et al., Bridge is a structural design pattern based on object aggregation (Gamma et al., 1995). It separates an abstraction from its implementation. Thus, the fundamental idea decomposition is inherent in it. Hence, Bridge fulfils aspects of the master idea structured dissection. Further properties are design reuse and information hiding (encapsulation).

Furthermore, there are aspects of algorithmization: parameterization (which implementation to choose), alternative and polymorphism (dynamic binding). To develop a cognitive approach to system behaviour, polymorphism is an important concept. It denotes different behaviour of operations and variables depending on their context. Regarding an informatics system, learners have to cope with changing characteristics and behaviour of a component, respectively parameters of components. Polymorphism is essential for interfaces and inheritance hierarchies that, in turn, are essential for system comprehension.

Every design pattern can be represented by the UML. Thus, there are aspects of language as a master idea, but on a low level.

Regarding the external behaviour of Bridge as a system, separation of implementation and abstraction is essential. A client interacts with instances of the abstraction.

The abstraction passes requests from a client to an object of the implementation. Moreover, the fundamental ideas identified so far have to be investigated, because they disclose internal behaviour and structure.

Further properties of Bridge are depending on a concrete implementation or a concrete model specifying number and kinds of implementation or abstraction.

Now we turn our attention to Composite, which is a structural design pattern based on object aggregation, and composes objects into tree structures (Gamma et al., 1995). Individual objects and compositions of objects are treated uniformly.

Considering fundamental ideas, recursive decomposition is essential to Composite. Hence, there are strong relations to the master ideas structured dissection and algorithmization. Particularly, the divide-and-conquer algorithm inherent in it that is realized by recursion is valuable for the subject informatics.

Beyond its representation by the UML, Composite also contributes to the master idea language. As mentioned by Schneider, textual objects can be built applying this design pattern (Schneider, 2003). A text is divided into sections, which are subdivided into smaller sections. Single characters form the leaves of the resulting tree structure. To set emphasis on dynamic aspects of algorithmization we use further classes to represent nonterminal expressions of different kinds like repetition and selection and map the context of the design pattern to regular expressions and grammars. Thus, information about syntax is stored in the resulting inheritance hierarchy. Hence, we can use Composite to parse phrases and to cope with syntax. Note that in the context of formal language, the Composite design pattern is also known as Interpreter design pattern.

Considering the external behaviour of Composite as a system, the unique interface to its components is characteristic. The client executes an operation on Composite where every component executes it depending on the specification of the component.

The internal structure of Composite depends on recursion and polymorphism. Internal properties are depending on a concrete example, i.e. with regard to the Interpreter, a context object has to be used as a parameter.

Finally, we will evaluate the results. It is obvious that both design patterns are complex and considering their properties as a small system is valuable. However, there is one advantage regarding Composite: it contributes to the master idea language beyond a representation via UML. Textual objects as described by Schneider are considered in the sixth grade. Hence, there is much potential to introduce the Interpreter variation into the learning process. Based on experiences at the University of Siegen, considerations described above, and the work of Schneider, the author recommends Composite as a valuable content of a subject informatics. However, Bridge is not as recommendable as Composite. First, there is the fact that language as a master

idea is not supported beyond UML. Furthermore, Bridge has to be considered very complex (see Table 1), which is interesting with regard to design patterns as a system, but it may only be appropriate at the very end of secondary level. Finally, we admit that there are other design patterns not investigated in this article, which show separation of concerns in a much simpler way, e.g. the Observer design pattern. The Strategy design pattern is more adequate with regard to the fundamental idea “alternative” that belongs to algorithmization.

	Suitability for Learner-Centred Cognitive Approach to Informatics System Comprehension		
	High	Medium	Low
Design Patterns	Composite (**)		
		Bridge (***)	
	Singleton (*)		
	Observer (**)		
	Strategy (**)		
			Abstract Factory (**)
		Facade (*)	

Table 1: Classification of design patterns with respect to suitability for “Didactic Systems” based on investigations concerning fundamental ideas and systemic analysis

Applying this classification to a set of design patterns, we get the result denoted in Table 1. Notice that the levels of complexity (low: *; medium: **; high: ***) have been assigned by Harrer and Schneider (Harrer and Schneider, 2002).

The advantage of the classification described above is that also combinations of design patterns can be integrated easily by the learner. As an example, we will consider the combination of Composite and Proxy. Therefore, we assume the following scenario: a figure consisting of subfigures, which is realized by Composite, has to be put into a text document. To permit working on the text in a comfortable way, the figure should not be included itself but a lightweight placeholder object, i.e. a Proxy. Now the learner can apply the criteria given above and investigate the suitability for his system comprehension. Instead of using the Proxy, we could also combine Iterator and Composite or further design patterns. Regarding the literature, Composite is often combined with Builder, Prototype, Chain of Responsibility, Decorator, Flyweight, Iterator, Visitor and Proxy. Applying the given classification, learners can prove all combinations and decide whether to use their cognitive approach.

Usually, such subsystems are the sum of the potential of their components with regard to fundamental ideas, but with an additional cognitive value through networking these ideas. However, modifications have to be applied to combine design patterns. Moreover, the criteria concerning the system, particularly properties and external behaviour, change with regard to the importance of the subsystem.

An Exploration Module for Upper Secondary Education

To support the approach to system comprehension we want to foster new ways of learning and new ways of teaching – nevertheless, we can also integrate the concept into traditional learning situations. In the area of e-learning it is consensus that “Blended Learning”, i.e. use of learning environments guided by teachers, has the potential to add advantages of e-learning to the learning process without negative aspects of pure e-learning. Particularly, learning environments can help motivated learners and teachers who want to apply their knowledge within an adequate domain.

Therefore, learning environments adapt to learners’ needs, i.e. different levels of difficulty and different learning paths have to be constructed.

Moreover, modelling with abstract building blocks is of high importance with regard to working life where expert systems are often used.

Brinda and Schubert introduced exploration modules as new aids for learning object-oriented modelling in learner-centred informatics education (Brinda and Schubert, 2002a). Their description of how using exploration modules as a part of the Didactic System can help learners to discover a solution of exercise classes. Combining requirements of such an exploration module with the requirements for system comprehension in a more general way, the author presents a recent approach of constructing an exploration module dealing with design patterns. Since June 2005, students of informatics with specialization media informatics are developing an exploration module for design patterns at school (upper secondary level). They are modelling a scenario realizing different levels of abstraction and learner interaction. The project is Java-based and will last until Summer 2006.

Exploration modules support learners’ activity. Activating multidimensional aspects of learning, which are addressing different senses, demanding learners’ activity and giving feedback inside the learning environment, learners explore the environment, plan actions to solve problems posed by themselves or the teacher, and verify the result. Applying the didactic technique of exploration modules, we hope to support gaining knowledge about substantial concepts of informatics, general problem solving strategies, and creating mental models by networked thinking. Self-competence and team competence are strengthened in the lesson designed and guided by the teacher as a discovery process.

Van Weert distinguishes the following kinds of learning: assignment-, task-, problem- and situation-based learning (van Weert, 2001, p. 48f).

Assuming a tactical role of learners at problem based learning, learners explore simple model elements at first. Abstraction of these simple elements has to be done during a discussion guided by the teacher. Additionally, learners construct model views assisted by the exploration module, which provides heuristics to identify further (abstract) elements in the problem domain.

Assuming a strategic role of learners at situation based learning, learners have to identify problem and solution methods. Consistency between different views and completeness of the model are essential. The exploration module supports error awareness of learners by identifying inconsistencies

between different views, but logical errors have to be discussed in class. To deepen comprehension of an informatics system, the exploration module itself can be investigated at a meta-level by its design- and analysis documents together with its source code (Brinda and Schubert, 2002b).

In the following, the author presents the development of an exploration module for introducing design patterns at secondary level by examining the design decisions taken so far.

Therefore, we have to assume and generate knowledge structures for the structuring of the educational process. Further questions are to integrate non-object-oriented principles into the learning environment. We have to provide a view showing when and how to modify design patterns. In particular, composition of different design patterns at a high level of abstraction is relevant to cognitive development and therefore to improve informatics system comprehension. Such composition also demands for creativity.

For convenience, we will sum up the requirements given in the requirements specification:

- a short introduction to object-oriented modelling
- creation of an attractive, gender-sensitive learning scenario
- approach to design patterns represented by UML-diagrams
- explorative interaction concerning design patterns: UML-representation, creation of design patterns in the context of a system, source-code-representation in Java, animations, exercise classes, properties of design patterns
- extensibility of the exploration module
- metadata for the learning processes
- different levels of difficulty
- high degree of freedom with respect to learner's activity
- intuitive exploration and evaluation of properties of systems

Integration of the design patterns Proxy, Observer, and Composite is mandatory. Combination of design patterns is intended.

The overall scenario of the exploration module is a theme park. Within the scenario the learners are free to choose a level of difficulty meeting their needs and to select the design patterns arbitrarily, e.g. the one whose context appeals most to them. The scenario has been chosen, because it is considered to be gender-sensitive on the one hand, and it will provide various possibilities of extension with respect to deeper system comprehension on the other hand. Guided use of selected basis functions within a bounded subscenario is typical for exploration modules. Thus, modularity is the guiding principle during the development of the exploration module.

With respect to different views, class diagrams, object diagrams, real-world view, and source-code view will be implemented. Particularly, the class diagram view shows behaviour of a design pattern and can be interpreted as a system view. Every view can be edited by learners. Therefore, changes in one view will be immediately applied to any other view. An additional feature can also be an animation visualizing processes of fundamental ideas of informatics, e.g. recursion.

Tests and exercises will be realized by multiple-choice, puzzle, and analysis and construction of UML-based design descriptions, i.e. combination of design patterns. To encourage learners to be creative, we will not restrict their creativity by only posing multiple-choice tests. Hence, learners' design descriptions should be evaluated in team work.

Related Work

This paper is closely related to Schneider's article (Schneider, 2003) where Composite and Observer design patterns are presented for use at secondary level. Based on Schneider's article, the author links the design patterns to the fundamental ideas introduced by Schwill and emphasizes the contribution of design patterns to informatics system comprehension. Furthermore, the development of an exploration module visualizing the design patterns in a whole system is presented. Thus, the article contributes to the question how to integrate design patterns into the subject informatics posed by Schneider.

Summary and Conclusions

In this article, we have presented a system-oriented approach to the subject informatics at secondary level. Assuming that knowing fundamental ideas of informatics permits understanding informatics systems, design patterns as a means to introduce networked fundamental ideas are presented. Moreover, there is a strong cognitive quality of design patterns. They are complex enough to be seen as system representations themselves forming subsystems combined with other design patterns, and they are part of an external object-oriented model.

With regard to the classification scheme presented in this paper, knowledge representation will be investigated as a further criterion with respect to learners at secondary level.

The next step is to discuss the concept within the scientific community. The extended classification scheme will be used to introduce design patterns into the learning process. After the development of a concept for the learning process, application of this concept has to be done at school together with student teachers. The empirical data collected during the application phase will enforce adjustments of the approach to learners' needs. An objective is to create workshops addressing teachers to include an approach to system comprehension into the subject informatics.

References

Brinda, T. and Schubert, S. (2002a) Didactic System for Object-oriented Modelling. In: Watson, D.; Andersen, J. (eds.): Networking the Learner. Computers in Education. Kluwer Academic Publisher, Boston, 2002, pp. 473 – 482.

Brinda, T. and Schubert, S. (2002b) Learning aids and learners' activities in the field of object-oriented modelling. In *TeLE-Learning - The Challenge for the Third Millennium*, D. Passey and M. Kendall (eds), Kluwer Academic Publishers, Boston, pp. 37 – 44

Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P. (1996) Pattern-Oriented Software Architecture: A System of Patterns, John Wiley & Sons, New York.

Claus, V. and Schwill, A. (2003) Duden Informatik. Duden Verlag, Mannheim.

Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. (1995) Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA.

Guzdial, M. (1995) Centralized Mindset: A Student Problem with Object-Oriented Programming. In Proceedings of 26th SIGCSE, ACM Press, pp. 182 – 185.

Harrer, A. and Schneider, M. (2002) Didaktische Betrachtung zur Unterrichtung von Software-Mustern im Hochschulbereich. In Lecture Notes in Informatics, Volume 22, pp. 67 – 76

Magenheim, J. (2005) Towards a Competence Model for Educational Standards of Informatics. In (Samways, 2005), Documents/452.pdf.

OMG - Object Management Group (2005) UML Resource Page, [WWW document, proved 12.07.2005] URL: <http://www.uml.org>

Samways, Brian (ed.) (2005) 8th IFIP World Conference on Computers in Education (WCCE 2005); 4 – 7 July 2005 – University of Stellenbosch. Cape Town, South Africa: Document Transformation Technologies cc.

Schneider, M. (2003) Design Pattern, a topic of the new mandatory subject informatics? In Proceedings of Informatics and The Digital Society: Social, Ethical and Cognitive Issues, IFIP TC3/WG3.1&3.2 Open Conference on Social, Ethical and Cognitive Issues on Informatics and ICT, July 22-26, 2002, Dortmund, Germany. Edited by T. van Weert and R. Munro. IFIP Conference Proceedings 244 Kluwer 2003, pp. 157 – 171

Schubert, S. (2005) From Didactic Systems to Educational Standards. In (Samways, 2005), Documents/397.pdf.

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

van Weert, T. and Tinsley, D. (eds.) (2000) Information and Communication Technology in Secondary Education – A Curriculum for Schools. Paris: UNESCO, [WWW document, proved 11.24.2005] URL: <http://www.edu.ge.ch/cptic/prospective/projets/unesco/en/welcome.html>

van Weert, T. (2001) Co-operative ICT-supported learning. A practical approach to design. In Informatikunterricht und Medienbildung. Edited by R. Keil-Slawik and J. Magenheim, Köllen, Bonn, pp. 47 – 62.