GI, the Gesellschaft für Informatik, publishes this series in order

- to make available to a broad public recent findings in informatics (i.e. computer science and information systems)
- to document conferences that are organized in cooperation with GI and
- to publish the annual GI Award dissertation.

Broken down into the fields of "Seminars", "Proceedings", "Monographs" and "Dissertation Award", current topics are dealt with from the fields of research and development, teaching and further training in theory and practice. The Editorial Committee uses an intensive review process in order to ensure the high level of the contributions.

The volumes are published in German or English

Information: http://www.gi-ev.de/LNI

ISSN 1614-3213
ISBN 3-88579-435-7

Magenheim, Schubert (Eds.): Informatics and Student Assessment, 2004

S-1

GI-Edition

Lecture Notes in Informatics

Johannes Magenheim, Sigrid Schubert (Eds.)

INFORMATICS AND STUDENT ASSESSMENT

Concepts of Empirical Research and Standardisation of Measurement in the Area of Didactics of Informatics

Volume 1

Dagstuhl-Seminar of the German Informatics Society (GI) 19.–24. September 2004 on Schloss Dagstuhl

Seminars

Johannes Magenheim, Sigrid Schubert (Eds.)

INFORMATICS AND STUDENT ASSESSMENT

Concepts of Empirical Research and Standardisation of Measurement in the Area of Didactics of Informatics

GI-Dagstuhl-Seminar

September 19-24, 2004, Schloss Dagstuhl, Germany

German Informatics Society (GI) 2004

GI-Edition – Lecture Notes in Informatics (LNI) – Seminars
Series of the German Informatics Society (GI)

Volume S-1

ISSN 1614-3213
ISBN 3-88579-435-7

Volume Editors
Prof. Dr. Johannes Magenheim
University of Paderborn, Didactics of Informatics
Fürstenallee 11, D-33102 Paderborn, Germany
e-mail: jsm@uni-paderborn.de
Prof. Dr. Siegrid Schubert
University of Siegen, Didactics of Informatics and E-Learning
Hölderlinstrasse 3, D-57068 Siegen, Germany
e-mail: schubert@die.informatik.uni-siegen.de

printed by Köllen Druck+Verlag GmbH, Bonn

Preface

Mission Statement

The Dagstuhl-Seminar 'Concepts of Empirical Research and Standardisation of Measurement in the Area of Didactics of Informatics' is organised in order to make a contribution to the development of didactics of informatics in general and to foster empirical research in the area of informatics education particularly. It is also intended to link the discussion of national experts about standards of informatics education to the discussion of the international scientific community within this area. Connected with the development of a theory of didactics of informatics educational standards are regarded as standardized objectives of qualification in subject related learning processes. They contain educational objectives of informatics and thus also describe implicitly the contribution of informatics as a subject in schools to general education.

The history of informatics, of informatics education and of didactics of informatics is a very short one in comparison to other more traditional sciences and subjects. Due to this legal and educational framework it is necessary to establish a tradition of discussion of didactical concepts in the area of informatics education in order to develop a subject related didactical theory. Though there is a strong relation between empirical research and the development of didactical theory, we unfortunately have to register a lack of empirical studies in the area of didactics of informatics. To develop concepts of empirical analysis of learning processes in informatics education and regarding them as results of realisation of practical aspects of a didactical theory are main issues of the seminar.

Therefore, during the seminar different concepts of empirical research will be presented. Especially the process of operationalisation of test items related to educational standards will be discussed. In comparison with research concepts of class room work in other subject areas empirical research methods in informatics education must direct their attention additionally to the use of software-tools and integrated development environments. Empirical analysis of class-room work in informatics must include collaborative processes within learning groups and individual and collaborative aspects of human-computer interaction. The intention is to gain more sophisticated empirical instruments which fit in a special way with the specific demands of the subject area.

Educational Standards of Informatics

This Dagstuhl-Seminar will give reason for the use of educational standards within the area of informatics education and emphasise the importance of standards for empirical research. The intellectual techniques of informatics such as problem oriented modelling, formalisation and abstraction change research and lecture in other subject areas, including pedagogics, and support meta-knowledge in order to master complexity.

The educational value of informatics is determined by this method of cognition within other sciences even apart from informatics systems. To learn about design and construction of informatics systems as a process of balancing interests between stakeholders makes people realise that exerting influence on system design and the considered use of technical systems is an important issue of democratic societies.

Based on the fundamental educational importance of informatics there are recommendations, national and international curricula and demanding educational concepts concerning informatics education. They include mainly not approved and empirically verified educational standards, e.g. methodical skills and domain related knowledge. In a wide range of educational topics in which students' learning success is scored there is a tendency towards internationally harmonized test methods for the educational outcomes of institutional learning. At the moment such comparative data are missing for informatics, especially for the impact of informatics on general education issues. In order to formulate educational standards within informatics education comparable teaching-and-learning-materials must be developed.

The concept of „Didactic Systems" ensures such a collection of coordinated teaching-and-learning-materials, which, as part of a class scenario, may lead to different skills very flexibly according to the respective target group and enables the integration of secondary informatics education into international student assessment. Thus, educational learning processes in informatics will become more transparent and comparable. In the aftermath of that a certain level of standardisation will contribute to the quality assurance and sustainability of the general educational impacts of informatics education.

New Research Results

In the last years we observed the consolidation of a new part of informatics, the field of didactics of informatics through a row of powerful doctoral theses, e.g. from Torsten Brinda, Ira Diethelm, Berit Holl, Ludger Humbert, Eckhart Modrow, Carsten Schulte, Marco Thomas, and the postdoctoral thesis of Peter Hubwieser. Therefore, the time is ready to establish a new level of cooperation to solve open questions and pressing tasks based on such successful research designs and tools. The invited expert group of the Dagstuhl-Seminar 2004 was asked to give their experience to the task force "Educational Standards of Informatics".

The research by *Torsten Brinda* shows the way from objectives to educational standards for the field of object-oriented modelling (OOM). The key idea of this approach is the identification, structuring and testing of new exercise classes. He developed this exercise classes as part of his specific concept "Didactic System for OOM". This concept provides such exercise classes to enhance the quality of learning. The power of the research results lies in the connection of a competence level model with informatics cores, subjects and types of exercises. On this basis he deduced competence levels from cognitive and planning preparation of OOM (level 0) to the advanced OOM and assessment of models (level 4).

Volker Claus describes how to educate students to be future successful applicants of informatics with a learning and teaching method called "Basic Reciptique". This method is the core of a new kind of didactics of Informatics, the "Service Didactics" of informatics Application. As an expert of theoretical informatics he illustrates the connection between the skills and the essential knowledge for this specific target group. The new "Service Didactics" could guarantee an efficient and serious informatics application strategy for other sciences. He recommends experiments in the virtual laboratory as a technique of interdisciplinary learning of informatics and other sciences.

Ira Diethelm, Leif Geiger, Christian Schneider, Albert Zündorf present two papers concerning the problems of measuring modelling activities. The first paper 'Measurement of Modelling Abilities' discusses the difficulties of measuring modelling abilities within empirical examinations. Besides a description of diverse aspects of the subject area, especially the challenge to operationalize cognitive processes at different levels of abstraction of a model, the authors provide us with a specific solution for grading modelling abilities of 3rd term students. Their second paper 'Automatic Time Measurement for UML Modeling Activities' outlines the current state-of-the-art in automatic time measurement in CASE tools and what may be achieved in the near future. This is done with respect to empirical studies for learning and teaching processes.

Ludger Humbert, Hermann Puhlmann analysed kinds of phenomena of informatics, such with direct, such with indirect and such without connection to an informatics system but with informatical structure or informatical reasoning. They discuss the conclusions of these properties for a phenomena-driven approach in informatics education and the phenomenon-based test items. The relation between modelling skills and different techniques of formalization was described together with examples of appropriate test items. These last findings were summarized to design conclusions of test items to determine the degrees of literacy in informatics.

Dietmar Johlen's paper "Learning Process' Evaluation in Vocational Schools for the IT Sector's Training Occupations" presents the concept of learning areas for the IT sector's training occupations. The scenario-approach is introduced, which represents a methodical-didactic reference system for the development and execution of instruction. From this starting point the evaluation of learning process in vocational training, especially in regard to the advancement of competencies were discussed. The author stresses that the scenario-approach puts the concept of learning fields in precise terms and that this approach is also an appropriate research environment for the evaluation of learning processes.

The empirical studies by *Peter Micheuz* show the results of a project in informatics education of learners at the age between 10 and 12 years in comprehensive secondary schools in Carinthia/Austria. The learners are in the beginning highly motivated to master the fourth cultural technique, but the enthusiasm of all learners (girls as well as boys) decreases significantly after one year. In teamwork a minimal standard curriculum was established and a pool of exercises. The project confirms two well-known facts; first the preparations for informatics lessons are extraordinarily intensive and second the teachers prefer to work with materials they prepared themselves.

Eckhart Modrow's paper 'The Contribution of Computer Science to Technical Literacy' deals with the idea of general education and how informatics at school may foster students' appreciation of technical systems, especially informatics systems. The author stresses the importance of that issue in regard to students' occupational choice. For the discussion of educational standards and for the selection of content in the area of didactics of informatics it is also very important to analyse the contribution of informatics to technology related topics and its relation to general education. The paper also examines how the term "technical general education" may be substantiated and discusses on the basis of some examples the consequences for the class room work in informatics.

Olaf Scheel describes the use of learning objects in an interactive computer-based learning environment for Blended Learning called Informatics Learning Lab (ILL). Students should use learning objects in a self-organized learning process in this open collaborative learning environment. The paper focuses on the construction of the learning objects and examines the coding types and levels of abstraction of the learning objects' media. An empirical research design is presented that should give reason for the design of problem based learning scenarios and analyses the effects of interactive animations in order to achieve software engineering related objectives.

Markus Schneider presents a matrix of measurable quantities which connects fundamental concepts of informatics, complexity levels of the exercises (low, intermediate, high) and the test results of students (female and male separately) in higher informatics education (first academic year). He discovered important results. Various program styles should be learned in the order of increasing syntactic complexity. Lectures are not suitable for the support of the students' self-activity. Female students start their first academic year with the handicap of missing knowledge on program languages and application strategies. Adequate study scenarios are to be developed in future work.

Carsten Schulte describes how to measure the effectiveness of learning-processes in informatics that rely on the use of programming environments. The paper deals with empirical research concepts which examine the influence of media on learning processes, especially in the area of informatics. According to the thesis that media may not influence learning under any conditions, the emphasis shifts from searching the best media to the search of effective learning environments. The conclusion to be drawn from this paradigm shift is with regard to empirical studies to supplement empirical pre-post design by instruments which enable to analyse human computer interaction with the software tools.

The research by *Andreas Schwill* shows that educational standards of informatics need a clear definition of the expressions "idea" and "term". He analyses works of Plato, Descartes, Locke, Leibniz, Hume and Kant. He describes the impact of the properties of ideas for the process of education, e.g. the influence of basic ideas on more complex ideas. From this he deduces the specific role of "idea" and "term" in the process of cognition, e.g. terms are structuring the subject area of cognition and ideas are controlling the process of cognition. This article complements his publications on "Fundamental Ideas of Informatics" (e.g. algorithmizing, structured decomposition, language).

Through the publication of these new research results, we hope to intensify the dialog among the German researchers and the international community in didactics of informatics, to promote educational standards of informatics and their integration into the Programme for International Student Assessment (PISA).

We hope that many readers in the informatics community will benefit from these contributions.

Johannes Magenheim and Sigrid Schubert

Paderborn and Siegen, August 2004.

Content

Preparing Educational Standards in the Field of Object-Oriented Modelling

Torsten Brinda

Didactics of Informatics and E-Learning
University of Siegen
Hölderlinstr. 3
57068 Siegen, Germany
brinda@die.informatik.uni-siegen.de

Abstract: In Germany the results of the international PISA study disclosed a demand for an increase in the quality and an improvement in the comparability of educational results. In the subjects German, maths and first language (i.e. English) this demand already resulted in the development and publication of first educational standards. With the aim to prepare educational standards for the Informatics field of object-oriented-modelling (OOM) at first characteristics of such standards were analysed. It was justified that learning subjects from the OOM field fulfil current learning objectives of Informatics education. To prepare a competence level model for OOM, an important component of educational standards, a method was presented for selecting, abstracting, analysing and structuring exercises, which has effectively been applied to more than 320 exercises and also successfully been tested in Informatics education. Especially by the structuring step of exercises according to their dimensions Informatics core, subject and exercise type, a good basis for the justification of a competence level model is given. The results of this analysis were finally combined with the PISA competence level for maths to an outline of a corresponding model for OOM.

1 Motivation

The results of the international PISA- and the additional PISA-E-study showed that in Germany and its federal states the performance of the learners in secondary schools varies more than in any other participating country. While in the upper performance ranges Germany can keep up with most of the OECD countries, in the lower ranges the German learners considerably fall behind the participants of other countries [Ba02]. This was interpreted as a hint of a lack of minimum standards, which must be achieved in the education of e.g. reading and mathematical competence. While the German educational system so far was only controlled by the input, e.g. curricula and examination guidelines, nowadays a shift towards output orientation, e.g. towards the performance of schools and above all towards the performance of learners, can be observed.

As a consequence the German Ministry of Education and Research commissioned Ger-

man educational researchers to investigate the development, implementation and consequences of national educational standards to increase the quality of school education, the comparability of secondary school qualifications and the perviousness of the educational system. First results were published in 2003 [GMER03a] and influenced the work of the Standing Committee of the German Federal Ministers of Education and Cultural Affairs, which passed first educational standards for 10th grade in the subjects German, maths and first language in its resolution from December 4th, 2003 [CMEC03a, b]. In the future the fulfilment of such standards will regularly be checked. Since the further development of the German system for secondary education by the introduction of nationwide educational standards is a wide-ranging intervention in a well-established school system, all subjects and its didactics, teachers and teacher educators as well as the school administrations need to be involved in this process.

In the subject Informatics the development of nationwide educational standards is impeded by the fact that in contrast to almost all other subjects still no binding basic education exists for all learners. Existing educational recommendations, curricula, educational concepts and lesson examples moreover show that there still exists no generally accepted consensus about the competences learners should acquire and the exercise classes learners should be able to manage.

This paper concentrates upon the development of educational standards in one important field of Informatics education, namely object-oriented modelling (OOM). The important role of OOM within secondary Informatics education was shown in [Br04a]. It was shown and justified, how the components of a so called didactic system (for OOM in this case), a compound of traditional and new components of the learning process with so called exercise classes, exploration modules and knowledge structures as main constituents, can be applied to prepare educational standards in the OOM field.

2 Method of research

With the aim to prepare educational standards for the field of object-oriented modelling, general components and quality criteria of educational standards are analysed on the basis of publications. Essential components in this context are the educational objectives to be reached, a step-by-step model of the competences to be reached by the learners as well as exercises and testing methods to verify the reaching of certain competence levels. The connection with the aims of a general, secondary Informatics education is realized by the linking with publications from the field of didactics of Informatics and with general educational guidelines. For the preparation of a competence level model a method for the analysis of Informatics exercises and for the development of so-called exercise classes, which was developed and successfully tested by the author, is applied. As a part of this method exercises become classified due to the dimensions Informatics core, subject and exercise type. Identified values of these dimensions are used to identify different levels of demands. Afterwards the results from this analysis are combined with the PISA competence level for maths, which due to analogies seems very appropriate as a basis for the Informatics field of modelling.

3 Components and characteristics and of educational standards

Educational standards take up general educational objectives. They lay down, which competences children and young persons should (at least) have acquired until a certain grade. The competences are described so concretely that they can be illustrated and implemented in exercises and in principle be measured by the help of testing methods [GMER03a]. *Good* educational standards relate to a certain learning field and work out the discipline's basic principles, do not cover the whole width of the learning field but concentrate on a core field instead, focus on competences, which have cumulatively and altogether been built up until a certain point in the course of a learning history, express minimum requirements, which are expected by all learners, are formulated clearly, concisely and understandably and are challenging for learners and teachers but attainable with realistic effort [ibid.]. In contrast to other definitions competences are understood in this context as available or learnable cognitive abilities and skills to solve certain problems as well as the involved motivational and social readiness and ability to apply the problem solutions in variable situations successfully and responsibly.

To benefit of educational standards within the quality development of schools, educational objectives need to be considered and competence models and exercises as well as testing methods need to be developed. Educational standards orientate themselves by ***educational objectives***, which the learning in schools shall follow, and implement them into concrete demands. They put these objectives in concrete terms in the form of competence demands and lay down, which competences a learner should have available, if important school objectives can be considered to be reached. These demands are systematically ordered in ***competence models***, which present aspects, levels and courses of development of competences. The determination of competence levels to establish minimum educational standards is a main research goal. Educational standards as results of learning processes finally become illustrated in ***exercises and testing methods***, with which the competence levels learners have reached can reliably be captured with empirical research methods. Since this is a time consuming process, the first published standards referred to a middle level of demands [CMEC03a, b].

In the following the dimensions *educational objectives*, *competence model* and *exercises and testing methods* are analysed for the field of object-oriented modelling.

4 Development of educational standards for the OOM field

4.1 Educational objectives

At the sight of the rapid development of Informatics as a science, the objectives of Informatics education are a continual subject of the didactic discussion since the origin of the school subject. Important results for the aim of this paper were the "Fundamental ideas of Informatics" by Schwill [Sc97], with which selection criteria for concepts for school education were made available, the "Information centred approach" by Hubwieser [Hu97], in which the importance of modelling for Informatics education was justified

in detail and the overall conception of secondary Informatics education [Br00] by the German Informatics society (GI), in which long-lived guidelines for Informatics education were described. These guidelines are “interaction with Informatics systems”, “working principles of Informatics systems”, “informatic modelling” and “interaction between Informatics systems, human beings and society”. Finally, in the revised version (Feb. 2004) of the uniform examination requirements of secondary Informatics education [CMEC04] modelling is also one main area.

In [Br04a] it was shown that learning subjects from the OOM field fulfil the selection criteria of Schwill. OOM is one essential style of modelling in Informatics, of which the educational value was widely stressed.

4.2 Identification, structuring and testing of exercise classes

The starting point for the identification, structuring and testing of so called *exercise classes* was the identification of a lack of lesson suitable exercises for applying, practising and deepening of contents of object-oriented modelling. In contrast to this in professional textbooks a big variety of such material exists. So, the existing material was analysed, and a method for developing corresponding material for secondary Informatics education was justified, applied and evaluated [Br04a, b]. This method not necessarily needs to be applied to professional material. In principle it can easily be adapted to fields of secondary Informatics education, for which already material exists. For the development of a competence model especially the analysis of Informatics cores, subjects and types of exercises combined with the development of exercise classes is of relevance. In the following, essential steps of this method are summarized (also see figure 1).

1. **Selection of textbooks**

   The starting point was given by the selection of standard text books on OOM with the prerequisite to contain exercises. The textbooks of Rumbaugh et al. [Ru91] and Balzert [Ba99] were selected.

2. **Selection of exercises**

   Since the contained exercises addressed different target groups, namely Informatics students or computer scientists, criteria to select or to transform unsuited into suitable exercises for secondary education were developed. In the first run of the method the criteria *concepts of secondary Informatics education*, *emphasis of modelling*, *language independency* and *complexity* were justified and applied. It turned out that the complexity of exercises should not already influence their selection for a collection, because this restriction is unnecessary. Not before the design of exercises for concrete learning groups and their learning histories, assumptions about suitable levels of demands can be taken. Therefore, the criterion of complexity was no longer applied in the second run. Altogether more than 320 exercises were analysed.

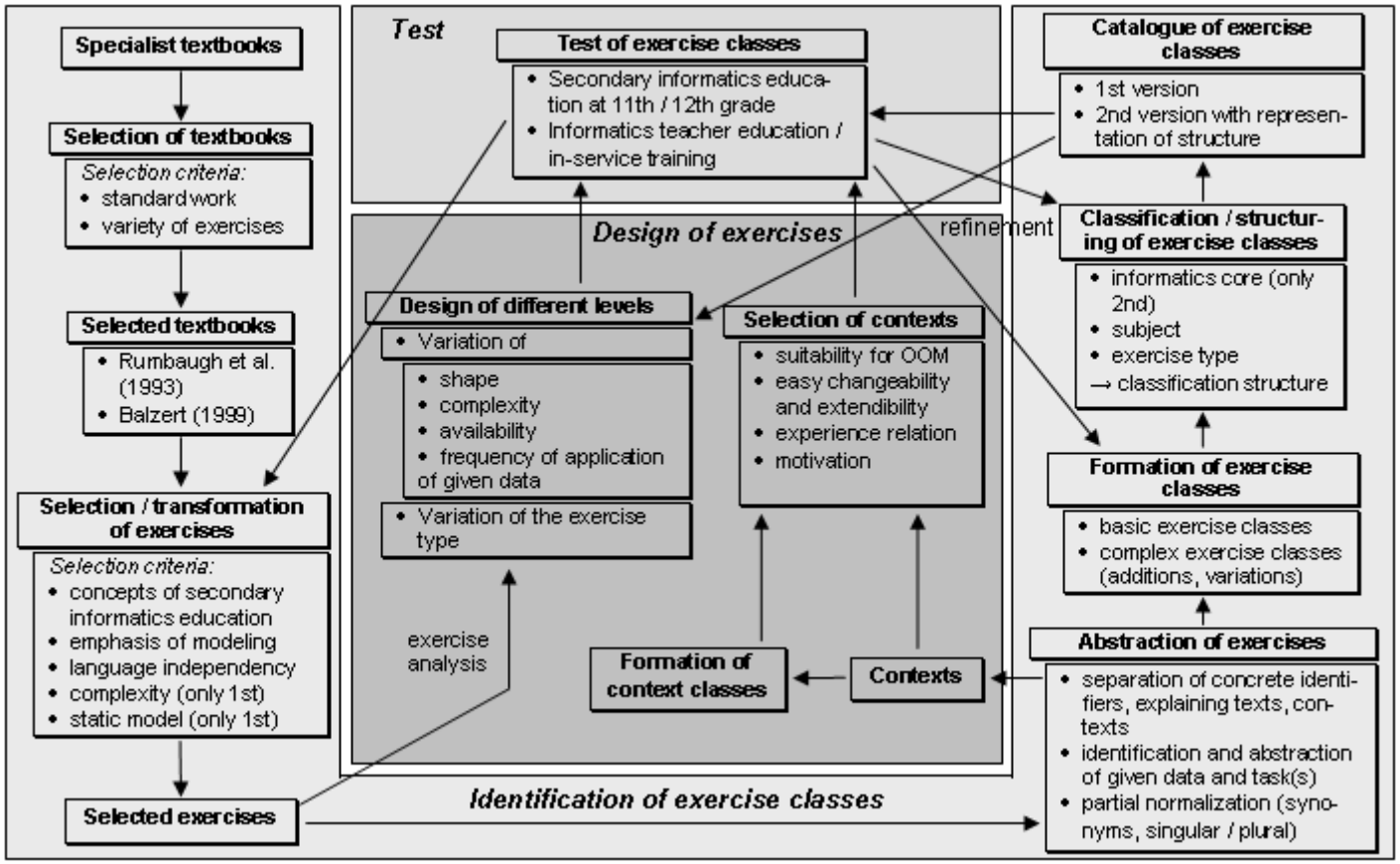


Figure 1: Development of exercise classes

3. Abstraction of exercises and development of exercise classes

After the selection the exercises were abstracted into so called exercise classes by separating them of explaining texts, real world contexts and identifiers. Every exercise class combines some given information (texts; figures, e.g. diagrams) with a task (basic exercise class). If for a task following additional tasks or alternative tasks exist, the exercise class is a complex one. It was necessary to identify different concept definitions (e.g. object diagram) within the analysed textbooks and to consider these definitions in the formation of exercise classes.

4. Structuring exercise classes

The next goal was to structure the exercise classes in a collection to simplify the access for teachers and learners. Therefore, the identified exercise classes were classified in view of their characterising dimensions *Informatics core*, *subject* and *exercise type*. In the field of the Informatics core exercises on the *static model*, the *dynamic model* and the *combination of both* were identified. The subjects of exercises were differentiated into *exercises on concepts of object-orientation*, *exercises on model elements* and *exercises on model.* Moreover, the exercise types *knowledge question, comprehension question, description task, assignment task, specification task, arrangement task, discussion task, analysis task, comparison task, validation task, identification task, modification task, transformation task* and *construction task* were identified. These exercise types were combined with the levels of the Bloom taxonomy of cognitive learning objectives and it was explained that exercises can be designed for all cognitive levels of demands. A collection of exercise classes was developed in two versions. At first, a preliminary study took place on the basis of exercises only on the static model to investigate the soundness of the concept. These exercise classes formed the basis of the tests in secondary Informatics education as well as in Informatics teacher education. The evaluation showed the soundness of the concept and the demand for exercise classes also on the dynamic model as well as on the combination of static and dynamic model and therewith initiated a second run.

5. Design of exercises with exercise classes

The findings of the tests in Informatics teacher education led to the development of a method for the design of exercises with exercise classes. Therefore, the selected exercises were structurally analysed with regard to the criteria *shape*, *complexity*, *availability* and *frequency of application of given data* and it was justified, to what extend, by their variation in combination with a suitable choice of exercise types, levels of demands in exercises can be modelled. Since besides exercise classes motivating contexts are necessary for the development of exercises, within the process of abstraction of exercises the separated contexts were analysed and the criteria *suitability for OOM*, *easy changeability and extendibility*, *every day life reference* and *motivation* for their suitability for les-

son usage were derived.

6. **Testing of exercise classes**

   The first version of the exercise classes was the basis for three empirical case studies in secondary Informatics education in grades 11 and 12 of two grammar schools in the Dortmund area in Germany. On the basis of these exercise classes, worksheets with OOM exercises were developed in cooperation with the teachers of the Informatics classes and a written questioning of the learners to grasp the educational success in the non-cognitive field was planned. All classes were sit in in spring 2002 at the processing of the exercises at two resp. three dates each. The anonymised solutions of the learners were collected and analysed afterwards. The learners were given prepared solutions instead. In the end of the case studies, the learners were asked to fill in prepared questionnaires. Altogether the soundness of the conception was shown. As important findings with regard to the further development of the conception a refinement of the inner structure of the exercise classes to simplify the selection for teachers, an extension to the field of dynamic modelling and its combination with the static model and a provision of additional and alternative tasks to better prestructure the way of solution of complex exercises were identified. Within Informatics teacher education it turned out that the student teachers had difficulties in creating exercises for different levels of demands because of a lack of experience. Therefore, the development of a method for creating exercises for different levels of demands was initiated. In in-service teacher trainings the broad acceptance of the teachers turned out, but also the demand for concept-oriented in-service teacher trainings in this field, since for many teachers OOM is also a new teaching subject.

4.3 Evaluation of Informatics cores, subjects and exercise types in view of a competence level model

By the analysis of ***Informatics cores***, ***subjects*** and ***types of exercises*** on object-oriented modelling (see point 4) basics for setting up levels of demands in a competence level model are given.

With regard to the ***Informatics cores*** it can be realized that in the modelling process the static model usually is constructed before the dynamic one. Afterwards the static and the dynamic model are developed further in an interlocked way. Published lesson experiences and results of lesson visits showed that this is also a suitable way for structuring the learning process. Nevertheless examples can be found, in which the dynamic model (e.g. use case model) served as the starting point. For the development of a competence level model no cause is given to prefer one of the two ways. It is obvious though that the combination with the respective other model represents a higher level of demands, because therewith tests of the consistency of the overall model are combined.

The ***subjects*** of exercises on OOM were classified into concepts of object-orientation, model elements and models. Concepts of object-orientation can be divided into basic

concepts (e.g. object, class) and advanced concepts (e.g. design patterns). The broad tendency of the exercises on basic concepts is on a lower level of demands than the exercises on advanced concepts. A similar relation goes for exercises on model elements resp. on models. The specification of an attribute (model element) is simpler than the specification of all attributes, methods and relations within a class model (model). Accordingly it can be justified that exercises on basic concepts resp. on model elements can be assigned to a lower level of demands than exercises on advanced concepts resp. on models.

It was shown for the identified ***exercise types*** on object-oriented modelling that they cover all levels of cognitive learning objectives according to the Bloom taxonomy [Br04a, b]. The broad tendency of the exercise classes, which belong to the lower levels according to Bloom, can be found with lower levels of demands. The same applies to the higher levels. Knowledge and comprehension questions can be assigned to all levels of demands as well as description tasks. Assignment tasks are especially suitable for the lower levels, because they take away the difficult process of identification from the learners. Specification tasks are again suitable for all levels. While on the lower levels the learners specify e.g. simple attributes, they specify on the higher levels complete models. Discussion, analysis, comparison and validation tasks are possible from a middle level of demands on and can be extended on higher levels accordingly. Identification, modification, transformation and construction tasks are possible on the lower and middle levels only, if they are formulated concretely enough and are accompanied by very illustrative material. On the higher levels, increasingly more open exercises are possible.

5 Outline of a competence level model for OOM

An important finding of the PISA study was that competence levels of a subject can be structured according to the content as well as according to important subject specific activities. In the field of Informatics it was proposed by Friedrich to structure the competences in the content dimension according to the guidelines of the overall plan for secondary Informatics education of the German Informatics society (see above) and in the process dimension according to the PISA competence levels [Fr03]. Puhlmann structured the learning of Informatics content in the form of the competence classes “application”, “development” and “decision” [Pu03]. It is an open question, if the proposed levels are suitable to structure all or only some fields of Informatics education. The following outline of a competence level model for the field of object-oriented modelling is also based upon the PISA model for maths [GMER03a], which is on account of analogies especially suitable as a basis for the informatic field of modelling, also because the PISA model is based upon the criteria for good educational standards (see section 3). For example it is quite clear that the first two criteria are fulfilled, because the OOM field belongs to the basic principles of Informatics and is among others one core field of the discipline. Publications from the didactics of Informatics were included to graduate concrete learning subjects on the basis of experiences.

Level 0: Cognitive and planning preparation of object-oriented modelling

Learners, who belong to this level, are able to make out and to classify *concrete* objects. They are able to take apart objects into pieces in a structured way and to cognitively grasp and analyse either hierarchical or tree-like structures. They have available basic planning abilities, which enable them to construct, to cognitively grasp and to manage hierarchical modularisations of plans.

All mentioned aspects are essential prerequisites for analysing and designing structures of object-oriented models. Schwill showed 2001 [Sc01] that this level usually becomes reached at the age of primary education. Level *0* was intentionally labelled so to indicate that in contrast to all following levels this preparation does not require any kind of primary Informatics education.

Level 1: Elementary object-oriented modelling

On this level, simple object-oriented modelling (e.g. of text documents in a word processing software) is carried out by the use of simplified object and class diagrams (static model). Learners, who belong to this competence level, are able to identify objects, to assign attributes and methods to them and to abstract objects of the same kind into classes. A very clear modelling subject is required as well as support of the learning process by suitable figures, tables or learning media (e.g. exploration modules [Br04a]).

The modellings carried out serve for building up the technical Informatics language on object-orientation on the one hand and as mental models of the modelling subjects on the other. Voß showed 2003 [Vo03] that learners in lower secondary education manage the learning process of the application of word processing software better, if they analyse and develop object-oriented models (reduced class diagrams) of text documents dealt with, instead of only relying on the online help documents included in the software.

Level 2: Object-oriented modelling and conceptual linking

Learners on this competence level are able to combine different concepts of object-oriented modelling to solve problems, if texts or figures guide the solution process. They are able to describe and to analyse simple given object-oriented models as well as to modify and to extend given partial models within limited scope. The learners on this level are also able to assign terms given in term lists to the categories object, class, attribute, method or relation. The independent identification of concepts in texts or figures is possible, if these materials support corresponding assignations.

In [Br04a] it was shown that learners in upper, without an educational background in Informatics from lower secondary education are able to manage such exercises. However, according to the level of demands, it seems appropriate to assign such exercise types to lower secondary education. It is a problem that in Germany still no binding Informatics foundation for all learners in lower secondary education exists.

Level 3: More extensive object-oriented modelling on the basis of demanding concepts

Learners on this competence level are able to build up object-oriented problem solutions (e.g. a static and dynamic model) over several interim steps. More open modelling exercises are managed in which among various designing possibilities suitable ones need to be chosen.

In [Br04a] it was shown that reaching this level requires competence in the steps of the object-oriented problem solving process. Learners in upper secondary education, for who the process was structured by given tables or figures (e.g. partial models), managed it without significant problems. Learners, for who the process was only structured by the declaration of steps, showed difficulties, if the steps were not practised enough before.

Level 4: Advanced object-oriented modelling and assessment of models

Learners, who can be assigned to this competence level, are able to cope with very open formulated modelling exercises, in which object-oriented models must be developed after a very thorough analysis of exercise texts. They have good command of the necessary core of an object-oriented modelling language (such as UML), of essential steps of an object-oriented process model as well as of advanced object-oriented concepts (e.g. design patterns) and are able to systematically apply the steps without further structuring help. Essential components of the problem solution process are informatic explanations as well as reflections about the modelling process itself.

Schulte showed 2004 [Sc04] that learners in upper secondary education are able to systematically apply steps of an object-oriented modelling process to solve given problems and that this way they develop more sophisticated mental models of software development than it is achieved in traditional Informatics education.

6 Conclusion and further work

In this paper a method for systematic analysis of exercises was presented and it was shown, to what extend the results of the analysis of exercises on object-oriented modelling can be used to prepare a competence level model, which is needed for the development of educational standards. An outline for a competence level model based upon the PISA maths model was presented. To validate it, systematic tests combined with empirical research are necessary. Moreover, the different approaches for learning and teaching object-oriented modelling at the levels of lower and upper secondary education as well as of early higher education need to be combined to a continuous spiral curriculum for secondary Informatics education under consideration of the special school organisational conditions for the subject Informatics, to refine the model proposed here therewith.

References

[Ba99] Balzert, H.: Lehrbuch der Objektmodellierung. Spektrum, Heidelberg, 1999.

[Ba02] Baumert, J.; Artelt, C.; Klieme, E.; Neubrand, M.; Prenzel, M; Schiefele, U.; Schneider, W.; Schümer, G.; Stanat, P.; Tillmann, K.-J.; Weiß, M. (eds.): PISA 2000 - Die Länder der Bundesrepublik Deutschland im Vergleich. Zusammenfassung zentraler Befunde. Max-Planck-Institute for Educational Research, Berlin, 2002.

[Br00] Breier, N.; Fothe, M.; Friedrich, S.; Hubwieser, P.; Koerber, B.; Röhner, G.; Schubert, S.; Seiffert, M.: Empfehlungen für ein Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen. Supplement to LOG IN 20 (2000) 2, pp. I-VII.

[Br04a] Brinda, T.: Didaktisches System für objektorientiertes Modellieren im Informatikunterricht der Sek. II. Dissertation, Faculty of Electrical Engineering and Informatics, University of Siegen, 2004.

[Br04b] Brinda, T.: Integration of New Exercise Classes into the Informatics Education in the Field of Object-Oriented Modelling. In: Education and Information Technologies 9 (2004) 2, pp. 117-130.

[CMEC03a] Standing Committee of the German Federal Ministers of Education and Cultural Affairs (ed.): Entwicklung und Implementation von Bildungsstandards. Bonn, 2003.

[CMEC03b] Standing Committee of the German Federal Ministers of Education and Cultural Affairs (ed.): Bildungsstandards im Fach Mathematik für den Mittleren Schulabschluss. Resolution from Dec. 4th, 2003, Bonn, 2003.

[CMEC04] Standing Committee of the German Federal Ministers of Education and Cultural Affairs (ed.): Einheitliche Prüfungsanforderungen Informatik. Resolution from Dec. 1st, 1989 in the version of Feb. 5th, 2004, Bonn, 2004.

[Fr03] Friedrich, S.: Informatik und PISA – vom Wehe zum Wohl der Schulinformatik. In [Hu03]; pp. 133-144.

[GMER03a] German Ministry of Education and Research (ed.): Zur Entwicklung nationaler Bildungsstandards. Eine Expertise. Public relations department, Bonn, 2003.

[GMER03b] German Ministry of Education and Research (ed.): Vertiefender Vergleich der Schulsysteme ausgewählter PISA-Staaten. Public relations department, Bonn, 2003.

[Hu97] Hubwieser, P.; Broy, M.; Brauer, W.: A new approach to teaching information technologies: shifting emphasis from technology to information. In [PS97], pp. 115-121.

[Hu03] Hubwieser, P. (ed.): Informatische Fachkonzepte im Unterricht. Köllen, Bonn, 2003.

[PS97] Passey, D.; Samways, B. (eds.): Information Technology. Supporting change through teacher education. Chapman & Hall, London, 1997.

[Pu03] Puhlmann, H.: Informatische Literalität nach dem PISA-Muster. In [Hu03]; pp. 145-154.

[Ru91] Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W.: Object-Oriented Modeling and Design. Prentice-Hall, New York, 1991.

[Sc97] Schwill, A.: Computer Science Education based on Fundamental Ideas. In [PS97], pp. 285-291.

[Sc01] Schwill, A.: Ab wann kann man mit Kindern Informatik machen? Eine Studie über informatische Fähigkeiten von Kindern. In: (Keil-Slawik, R.; Magenheim, J., eds.): Informatikunterricht und Medienbildung. Köllen, Bonn, 2001; pp. 13-30.

[Sc04] Schulte, C.: Lehr-Lernprozesse im Informatik-Anfangsunterricht – Theoriegeleitete Entwicklung und Evaluation eines Unterrichtskonzepts zur Objektorientierung in der Sek. II. Faculty of Electrical Engineering, Informatics and Maths, University of Paderborn, 2004.

[Vo03] Voß, S.: Objektorientierte Modellierung von Software zur Textgestaltung. In [Hu03], pp. 211-223.

Service Didactics / Dienstleistungsdidaktik

Volker Claus

Institute of Formal Methods in Computer Science
Faculty 5, University of Stuttgart
Universitätsstraße 38, D-70569 Stuttgart, Germany
claus@informatik.uni-stuttgart.deAbteilung

Abstract: This paper asks for a faster introduction to Computing Science for the growing number of students who have to learn information processing as a complementary science or for a foundation degree (UK). Nowadays the human lifespan limits more and more the fundamentals which are necessary for the respective profession. Therefore the skills and the essential knowledge must be restricted to the principal subject, and all other knowledge has to be taught without deep explanation. Complementary sciences and even areas of the major subject will be trained by "drill-and-practice" but in such a way that a future insight in the scientific principles always remains possible and that more topics than nowadays have to be dealt with. This methodology of teaching and learning is characterised by:

- conveying extensive knowledge and skill (using recipes for training),
- following certain paradigms of the major subject,
- training the operation and the use of systems, and explaining their prospects and risks,
- teaching some underlying theoretical foundations and rules (in a reduced manner).

These methods are called "basic reciptique" (Grundrezeptik). It is a central part of the "service didactics" which have to be developed for optimizing the efficient, quick and serious usage of a science in another context. Service didactics must elaborate a minimal but stable scaffolding of the field, design excellent supervised courses, develop models for teaching, evaluations and assessments etc. An important technique will be the "virtual laboratory" for intensive learning in limited time. Each laboratory will be built up and updated by scientists from different faculties. In future the courses of more and more studies will be oriented towards this basic reciptique. Computing / Computer Science seems to be a good candidate for investigations and the European Bologna Process with its reordering of the educational system can immediately be used for many experiments.

1 Basic Reciptique for Service Didactics

1.1 Initial position. According to education policy people are equipped with the necessary skills and working methods by the time they leave school. All they need then are general qualifications and vocationally oriented knowledge. Instead, German universities concentrate on teaching the rudiments of a science – often erroneously described as the "theory" of a science – and therefore use a large part of the students' education for contents which are not relevant for the future profession. Therefore new qualifications and contents closely concerned with practice are called for. First-year students would be delighted and there would be positive side effects such as a reduction of the length of courses of studies and the establishment of a system for lifelong learning.

The reality for more than half of the first-year students of informatics is like this at German universities: lack of orientation, "consumer attitude", and the widely held opinion that a mere presence and collecting of hand-outs are enough effort for the degree or bachelor. To take measures against this and also to even out the different previous knowledge, the basic courses mainly consist of conveying structural insights and basic principles. Since there are not enough lecturers, these courses are generally crowded. This situation might not be an obstacle for motivated students of informatics as long as there are enough additional practical courses. But what about the less motivated students or those who need informatics only as a complementary science? There is not much use in confronting these students with abstract and formal concepts which they might not understand on account of insufficient previous knowledge in maths and which therefore don't seem to have any relevance for an application of informatics. What they need are courses with extensive tutoring, a definite objective and high expenditure of energy to comprehend methods, sequences and techniques of working and to understand the "realistic" application of informatics. Obviously, such courses were a chance for every science to communicate its meaning to the students. Nevertheless, even any minor field of study endeavours to impart a deep insight in its methods and way of thinking und that is why contents are taught which are futile for a complementary science.

Why is this the case? Why does the university ignore the reality of its first-year students? And there are even models like "mathematics for business studies" in which representations and easy proceedings are dealt with and no one tries to familiarize the number of students to the science of mathematics. In contrast to this the lectures in informatics that are held for students of other fields of study follow a basis-orientated approach, that a) is not conveyed convincingly, b) cannot be tested correctly and c) wastes time that could be used for the sensible application of methods. It is this complacent attitude ("We also have to teach you something that we definitely loathe and that only shows you how out-of-date contemporary applications are.") that has been depicted so strikingly in the character Teacher Lämpel by Wilhelm Busch and that we have to get rid of undeniably.

1.2 Conclusion: We need double didactics:

a) Didactics for the new generation of informatics (We won't deal with this aspect in this paper),
b) Didactics for other fields of study that require informatics as a complementary science

First, didactics b) might be interpreted as a stunted form of didactics a) and its evolution might therefore be put in the hands of the respective group of people. That would be utterly wrong and would not produce anything new. A new approach is called for. One, which is developed without missionary eagerness on a high and efficient standard: "*service didactics*". Examples for application – which characterize the students' picture of informatics – and the methodology are to be taken from the main subject and in laboratory applications both scientific areas are brought together. At the same time the fundamental ideas and inner structure that underlie informatics have to be issued correctly in a background programme, with the emphasis on the background aspect. This mixture of recipe-like imparting, consolidation with application scenarios, bringing out the usefulness and constructive employment as well as conveying a clear scientific basis in the background is called "basic reciptique" in this paper and is yet to be developed and backed up in detail. (The accompanying scientific superstructure can be called basic reciptology.)

1.3 Comment on the techical terms: "*Reciptique*" and "*reciptology*" are no common words. Reciptique means imparting of knowledge and behaviour patterns with the help of a schematic and recipe-like proceeding. Reciptology refers to the methodology of the development and implementation of such recipe-like conveying. While the term "reciptique" has been unusual so far, the word "reciptology" has been used every once in a while for the schematic reproduction of misunderstood statements, thus in the sense of babble that has been learned by heart. (Reciptique is not to be mistaken for receptorique; the latter describes the ability to receive information by means of receptors, at the same time it can also stand for a system of receptors as in the skin or a monitoring system.)

1.4 Characteristics of intelligence tools: We can use gained knowledge to elaborate our self-esteem and our concept of the world we live in – entirely without any economic or personal benefit. Especially because "things as such" have an inner coherence and follow principles that can be subject to research, the knowledge that serves no purpose, forms the centre of academic education. This is the criteria every respective main subject has to satisfy. For the benefit of humankind we convert knowledge into products, production processes, workflows and services and by that we add purpose to an item that serves no purpose. In doing so the dimension of subjects is added – in addition to comprehension and methods of employment – to the knowledge by means of the increasing multidisciplinarity. With the explosion of knowledge in the respective subject areas, with the ever new scopes of application and with the miscellaneous basic insights an all-embracing university education can't be realised in a limited time-span of five years anymore. Thus the courses have to be restricted to a few main subject areas. Everything from other fields of study should be accessible to the students in the form of a tool box which is supposed to be of a logical structure instead of being unsystematic. This means that an ever-increasing part of the knowledge and skills have to be practised in intensive exercises. They are created somewhat according to the concept of "quick and dirty", whereupon the function of the basic reciptique consists of creating the "dirty" in spite of the "quick" as respectably as possible, if only not to interfere with lifelong learning (LLL). After all, computer scientists have to be able to communicate in teams with users in their job, as well as users have to comprehend basic concepts of computing solutions.

The tool box has little relation to the conventional handicraft. It consists solely of virtual and mental aspects. It serves as an intelligence intensifier, and we will call a tool for informatics from now on "intelligence tool". The intelligence tool box has to be so well equipped that a lot of future problems can be solved with its help. The different universities will define the future time frame differently, and develop measures to update old intelligence tool boxes and to supplement it, for example by re-engineering. As a general tendency future graduates are able to convey considerably larger areas than today. The deepened knowledge will share its place in the brain of experts with an ever growing amount of diffuse knowledge about a constantly increasing intelligence tool box. As a result, the knowledge without a purpose, which is now understood as a surety for the augmentation of creativity, will recede in the area of complementary sciences (The exception proves the rule here as well.) or, on a high standard, will be accessible only to a small number of people. The consequence for the education in informatics will be to enable students to use intelligence tool boxes as an everyday instrument, but also to impart the logical structure that lies behind the tool boxes and the sensible employment of the tool boxes.

1.5 Comment on a *feasible* implementation: For the teaching the contents, the methods, the intelligence tool boxes, the forms of the lectures, the virtual laboratories and the forms of testing have to be constituted. The contents have to be developed with the experts for the main subject and accommodated for the laboratories. They are taught in lectures that need a lot of tutoring in the beginning, but require more and more independency of the participants in the course of the semester. A typical course on data structures, which now takes up four hours in a week and is taught in four hours of lecture and two hours of exercise, *could* then be changed into four hours: one hour for a lecture that forms the guideline, one hour of a closely connected practical course in which new content is reviewed and tested in small teams, one hour of a laboratory course and one hour for exercise in which the solutions of individually prepared problems are discussed. The course is then complemented with single lessons on handling the used intelligence tool box.

If a course were structured in such a way, it could be adjusted to a practical and a laboratory course in natural sciences: It takes place in a computer lab; one or two students work on each computer; the directions to the practical course describe the exercises; in the laboratory the students illustrate and elaborate the scenarios; they write reports on the exercises and their solutions; the exercises emphasize the employment of the intelligence tool box which requires the computer workstation; the practical course takes place as a tutorial at the university (and not at home); the tutors can check on the students' knowledge by means of the discussions in the computer lab at any time. The exercises refer to the problems that the students have dealt with in the practical course and focus on formalisms and comprehensive principles. There could even be a shared discussion forum. The preparation of the course and its yearly update are time-consuming, they should be conducted by tutors and advanced students. The expenses and expenditure in time will increase considerably compared with today's situation, because parts of the lecture are substituted by tutorials and practical courses.

2. An Example (for Details see Attachment)

2.1 So what is new in this concept? In how far does this concept differ from practical training, that is widely spread with its trial-and-error method and its drill-and-practice techniques?

There are two differences: On the one hand we want to communicate the reasonable employment of informatics in the main subject, which is managed with the help of suitable scenarios in a virtual laboratory. That means that it is not enough to introduce and comment on a substantial library, but to formulate the demands on informatics for main subject purposes and to work on typical examples with the intelligence tool box. On the other hand, the curriculum should include the vital principles of informatics. Therefore a technically correct and yet reduced structural framework has to be developed, which can help to explain as many concepts and tools as possible. This is where the principles of school teaching get involved since instruction in schools have to consider the same aspects. Admittedly, this has to be readjusted for academic purposes.

2.2 AVL-Trees: Let's look at an example. Currently (in June 2004), the basic lecture in informatics at Stuttgart University deals with hight-balanced trees and its respective representation, characteristics, algorithms and realisations in programming language. Many students who don't study informatics or software engineering take to it and find that it is exciting and surprising. This is true only for one third of the lecture, though, because in this part the layout and manipulations are exemplified. The remaining two thirds deal with variants, analysis of rotations, the exact recalculation of balances, Fibonacci-trees, semantically correct implementation in procedures and proofs of the maximum or average depth of such trees. To sum it up, this knowledge is only a "value as such" itself: cognitive structure and insights – long-lasting models - are developed that should enable students of the main subject to apply, evaluate and enhance methods for storing and processing. But this doesn't appeal to students of the complementary science course. The exact content of the lecture and its possible modification is listed in the attachment.

2.3 The value of such contents is not relevant in practice, though. Through pre-defined classes hight-balanced structures are readily available for concrete applications (for example in connection to data bases), and no one has to comprehend their underlying theory and formalisms or their secret realisations. No user has to know these structures and can still use them for his problems. From the point of view of the main subject these two thirds of the content are mainly a waste of time; at the utmost, some principles and "parameters" (time response, storage space overhead, initialisation,…) are relevant to apply the existing intelligence tools from the already browsed intelligence tool boxes (classes, library programmes, methods available in the net, own additions etc.).

And still, *each* student has to learn this basic knowledge that is only relevant for theoreticians and those responsible for the implementation of library software and object-oriented classes. This is based on the assumption that an evaluation and therefore a reasonable employment isn't possible without the background information. But is this also true for other people than computer scientists? And even if it is true, what about the expenditure of time? The concerned students take a more practical position: They expect an overview with some characteristics and a general understanding for employment scenarios. However, they don't want to implement these systems but adopt standardisations which have already been tested. So why do we impose our wisdoms on them? Why don't we offer a practical and applicable – and yet well-founded – version to non-computer scientists when we claim that informatics is optimally applicable? A solution could be our service version of informatics.

3. The Meaning of a Service Didactics

3.1 Didactics is often conceived as imparting contents and methods to students. This abbreviated assumption often serves as the reason for the separation of the department of didactics from the scientific departments. But didactics have a much deeper impact: It explains the structure of a subject and is the actual foundation for Humboldt's ideal of the unity of research and teachings.

The person who researches should also teach, and only those who are dedicated to research should be able to teach. This is the reason for the privileges of universities as for example the conferral of a doctorate or the teaching load that is explicitly reduced compared to schools. But is this theory on the unity of research and teaching correct? Max-Planck institutes, the Fraunhofer society and industrial research institutes barely offer systematic teaching and employees of the DFG or graduate colleges are not supposed to teach either.

The thesis on the unity of research and teaching can only be applied to professors of universities where they are supposed to educate the highly qualified up-and-coming academics. This calls for a detailed overview on the actual state of affairs, therefore researchers are called for in this area. The taught knowledge cannot be a mere collection, though, since then it can't be conveyed in a reasonable amount of time. This requires a systematic system of a subject, an elaboration of principles, an analysis of the fundamental methods, a description of techniques for problem solving and the synthesis etc. All these objectives belong to the field of functions of the department of didactics. Consequently, didactics have to be considered as a central part of each academic subject. (Why this is not the case in reality would be the topic of another paper.)

3.2 Insertion: Science and the relevance of didactics. To become a science, an area has to meet at least four prerequisites:

- It has to generate new contents and a new methodology,
- it has to invent a new and incomprehensible language,
- it has to contain didactics,
- it has to develop reflection (and preferably also self-mockery).

(A reference to reality and a utility are often present as well, but also a science which serves no purpose could be imaginable.)

Informatics has already become a science in the 1970s, even if the didactics and the self-criticism have only been dealt with by a handful of scientists. The aspect of incomprehensibility, that is part of each science, has been shaping well in informatics and in its areas of application. To compensate for the trend to isolation didactics are called for that have to meet several conditions:

- Clarification of basic terms (e.g. fundamental ideas) and compilation of typical and plausible examples (paradigms),
- systematic development of the subject and organisation of the respective methods, influences on other subjects,
- evaluation of the contents and subareas (in terms of difficulty, utility, employment, impacts,…),
- explanation of difficult and incomprehensible parts,
- concepts for teaching and examination of contents, methods and techniques and the formulation of teaching methods (for different ages and target groups),
- preparation of curricula and frameworks, goals of the education,
- an over-all concept "from kindergarten to retirement" would be ideal,

- to this service didactics could be added, in order to meet the claim for interdisciplinarity and solidarity (in the sense of mutual assistance of sciences), and also didactics for further education (based on didactics for adult education) to react to the fast exchange of knowledge.

3.3 Although our concept of didactics argues for a certain independence, the academic departments of didactics normally concentrate on teacher training. Thus, they fulfil a concrete role in society. Generally speaking, a didactics without a purpose is not customary and for a service didactics there are no academic institutes available. However, *there are a lot of reasons for basic reciptique*:

- Ever-growing insights met with constant or decreasing time for education need a better preparation of the learning matter and a reduction to basic and useful elements.
- The interdisciplinary co-operation requires reduced frameworks of each subject as an easy access for the mutual dialogue.
- The employment of informatics accounts for recipe-like behaviour and manuals for the intelligence tools that are getting more complicated all the time.
- Typical examples and scenarios serve as an aid for orientation and evaluation.
- In a world full of tools good comprehensive principles and guidelines for as many instruments as possible are called for.
- The developed courses are an ideal basis for further education and LLL. An all-embracing system for further education is essential for all jobholders who are supposed to work productively until the age of 67.
- The necessary feedback to informatics will be increased.
- Eventually, even general working methods can be conveyed with the help of this concept; the main reason for poor performance in the first semesters is insufficient working methods.

3.4 The argument of the discrepancy between the limited time for education and the increasing amount of the content of teaching and in addition to this also the growing interdisciplinarity demand the development of new courses which are based on the basic reciptique. Since the duration of grammar school education will be shortened to twelve years, the qualification of first-year students will degrade in spite of aptitude tests. This fact intensifies the pressure on complementary sciences to educate fast and orientated towards the respective employment. New forms of an easy access to a field of study combined with a component of foundations to ensure LLL – namely the basic reciptique – will therefore be of an ever-increasing importance. We don't support a thoughtless easy access, but an education that is build on scientific working methods and also imparts these methods. Furthermore it should present a reasonable employment by elements of a virtual laboratory and that enables the participants to communicate with students of informatics or even to change into informatics as a main subject quite easily.

This is a didactic challenge: How can the intellectual content and also as many ideas as possible be conveyed into the students' heads with the help of subject matter that is recipe-like imparted? The main difficulty might be the co-operation between the different sciences: Professors have only a limited conception of the subject matters of other sciences and still have to collaborate to develop a self-contained system of courses with new methods of presentation and exercise. But maybe this is the chance to foster interdisciplinarity, to reduce reservations and to present the subject matter to the users in a more attractive way.

3.5 We are convinced that the current "Bologna"-process (combined with the twelve-year education) will also boost this trend. If the future bachelor is not a intermediate diploma with additional skills, but a university-entrance diploma with job-related knowledge, complementary sciences have to be quite compact and the main content has to be conveyed in a better way to keep the standard. That is why now is the perfect point of time to conceive and advance a basic reciptique for informatics.

4. The Academic Implementation

4.1 The first problem: absence of money. From our point of view the trend will lead to a demand for a basic reciptique. Since there is no money for new initiatives *all* members of the university will be asked to acquaint themselves with the new concept free of charge and to deliver the service without professional knowledge. We advise explicitly against such a development, because it is more expensive in the end or will be proved to be counterproductive. To exemplify this please read the following paragraphs a and b.

a. Improved teaching. For many years a "strengthening of teaching" has been demanded to enhance the result of education. This refers to two aspects: Firstly, universities are to put more effort in teaching rather than research, and secondly, teaching has to be improved. *Improved teaching* means better preparation and presentation of the respective contents, which is said to account for easier, faster, deeper etc. learning. In practice better teaching does not automatically account for better learning, better insight or shorter education, though. A reason for this is for example the idleness of humankind: the better prepared a subject matter is, the less the appeal gets to immerse in the subject matter and the faster the learner is convinced that he / she has understood everything the first time around. Therefore, "improved teaching" has often proved as a double burden: distinctly more effort for the development of the teaching material (in informatics the factor five is quite frequent) and also more effort in tutoring to convince the students of the depth of the content and the inherent ideas. Unfortunately, the result has often been less time for research and thus a negative long-term effect on the quality of education.

b. Informatics as a school subject. In the 1980s no study courses for prospective teachers were established and consequently, no teachers for informatics have been trained. Hence, the school subject informatics has been taught by teachers of other subjects. This is why a lot of first-year students started their studies with wrong expectations of informatics, which resulted in a high drop-out rate for this course of study. The schools have been in charge of the organisation, but the teachers haven't had the time to really learn the basics of informatics. This aberration might have caused more harm to the national economy than the courses of study for future teachers would have cost. By the way, a result of this political failure is that now the universities are blamed for the drop-out rates.

On account of experiences like these universities would be well advised to claim a financial guarantee before they start work in this field of activity. Unfortunately, this also refers to reasonable and essential measures as the development and analysis of a basic reciptique. It is assumed that even the best arguments won't get any reactions.

4.2 The second problem: the capacities. Presumably, there are a lot of computer scientists at the universities who would support service didactics and according courses. What we have learned from the past, though, is that useful things (as for example didactics for informatics itself) need decades for their establishment.

Delays and hindrances like these are mostly based on arguments about capacity. With a budget freeze innovations can only be installed at the expense of the already existing. Chairs have to be relocated and capacity relevant courses from the core curriculum of informatics have to be given up at the cost of new courses (which might be only for optional subjects). No head of a faculty would agree to such plans (for structural reasons), unless new posts can be made available from the outside or already cancelled contracts can be saved.

At the same time the teaching load plays a major role. With eight or nine hours in the week there's only little time to tackle innovative re-orientation or even realise reasonably. A solution would be to further it with theses or with a set of lectures by different speakers, which would take up a lot of time, though.

4.3 The third problem: There are also doubts concerning the content. Someone might argue that with courses like that wrong ways of thinking might be enhanced, future developments of informatics missed and the graduates of such courses might forget their knowledge even faster. This is why for a lot of computer scientists these useful, but at the same time undemanding courses are out of the question. Critics also mention that other departments don't approach the department of informatics either: No lecture of economics or engineering meets the requirements of informatics. Technological universities are especially careful, if a newly provided professorship does not fit in perfectly with a special branch of science.

Another point of interest in this discussion is the creation of the bachelor-degree. Those who will be working in the bachelor education are less qualified than those of the master degree. Therefore, a lot of the university staff fears that they might lose their academic reputation when they teach basic reciptique. Maybe these graduates will be worse off – financially and legally – than those who received a bachelor degree.

4.4 There are a lot of reasons in favour of service didactics, but also a lot against an organised support of the basic reciptique by the universities. Therefore nothing will be done. But this should not keep didactic departments from carrying out investigations, as long as they approve of the respective teaching and learning methods.

The objective of such investigations is the conception and outline of a basic reciptique for informatics. Which areas should be covered by the new courses and virtual laboratories and how can they be installed in a high quality? According to which pattern should they be arranged? How will they be interlinked with a main subject and how much are the expenses to implement them? How can examinations be held efficiently and in a time-saving way? Which role does the teacher take and which parts can be automatised? In how far can the contents be used for further education and propaedeutic courses for grammar schools?

5. Concluding Remark

Two central ideas have disappeared from the contemporary discussion about the improvement of education: serving no purpose and gaining maturity. Future students are supposed to complete their degrees fast and with job-related qualifications in order to quickly obtain an economic significance. This is comprehensible in regard to the age pyramid, but from our point of view the longed-for effect might not be achieved in the long run.

Again and again we meet students who have been indecisive for some time and suddenly they get motivated and immerse themselves into their studies. We see people who are only able to understand ideas with the help of abstract presentations that are not connected with any application. There are also students who attend lectures out of pure interest and not because of their examinations. We know that people need time to pervade a subject matter entirely and to develop self-confidence. A year that other people might call lost can be gained quite quickly with multiple value. Each society needs a lot of people like that.

Thus, basic reciptique can also serve the purpose to provide the freedom to gain maturity which will mainly take place in the main subject. It is such a superior objective that we aim at, although we know that universities (in contrast to professors) don't respond to these arguments. But we haven't given up hope that in the long run there will be a deeper insight.

Attachment 1: AVL-Trees (presentation)

raw structure of content, see 1.5.

a) Lecture

Duration of the lecture: Approximately 84 minutes. The numbers before each topic indicate the necessary minutes. Five minutes of each lecture are needed for wiping the board, beamer-handling and establishing silence and attention.

5	Definition AVL-tree (hight-balanced tree, special case binary trees)
9	Examples as well as two examples of binary trees, that are no AVL-trees only because of one leaf; colloquial explanation of the rotation (vertical postponement of nodes)
14	The four types of rotations with the resulting balances
8	Part of a new programme for new calculation of the balances, with a little bit of verification
4	Strategies of search, insertion and deletion
10	Deletion in detail with the problems of going upwards
6	Comments on the programming language (also recursive procedure)
7	Insertion (Repetition?): definition and characteristics of Fibonacci numbers
11	Proof of maximum depth of an AVL-tree with n nodes; measured depths in practice
7	Fibonacci-trees
3	Hight-balanced trees that are not weight-balanced

b) Corresponding exercises

The exercises deal with several concrete examples (student data file, stamps and classifications of characteristics, CD-ROM collection: advantages/disadvantages of the AVL-structure); sorting with AVL-trees and comparative measurements with other search trees; going through the operations at insertion and deletion individually and formulating them in the programming language; impacts, if the inorder successor is always chosen when deleting; generalisation of trees with balances between -2 and 2 (or from –k to +k). The exercises are selected in such a way that the exercise teams can cope with them in 50 minutes (with the exception of one challenging or theory accentuated exercise). The time for preparation and for processing the AVL-exercise is an estimated 135 minutes.

c) corresponding materials in the net

Older concepts are described in chapter 3.2.4 “Introduction to Informatics” (transparencies 111-140):
http://www.informatik.uni-stuttgart.de/fmi/fk/lehre/ss04/info2/default.htm
Exercises are on page eight on the same website.

Under the keyword “AVL-tree” there are a lot of elaborations on this subject matter, but the best descriptions can be found in appropriate textbooks.

Attachment 2: AVL-trees (from the Point of View of Basic Reciptique, Suggestion for Trial)

a) Lecture with an Integrated Practical Training (135 minutes if it is realised with the integrated course)

The lecture takes place in the room for the pratical training. There is a maximum of 16 participants present. The professor lectures for 15 minutes followed by 30 minutes of work at the computers guided by tutors. Afterwards there are 20 minutes of a lecture followed by another 70 minutes of work at the computers.

a1 Preparation in the book for practical training: Each participant has to read six pages on AVL-trees and examples before the lecture starts. (Content: characteristics of an AVL-tree, several examples, process of insertion with two rotations and the request to discover two more rotations.) We assume that everyone knows that for the lecture. Working with a class library (for example in Java) is also familiar for the participants.

a2 Lecture: 15 minutes. Establishing silence and attention will not be necessary here; wiping the board and handling the technical equipment can be dealt with in the other parts of the course. The numbers before each topic indicate the necessary minutes.

2 Definition AVL-tree (hight-balanced binary search tree)

4 Examples for insertions with the two missing rotations

4 Illustration of the four types of rotations and their effects with the help of examples

2 A weight-balanced tree that is not hight-balanced

3 Example from the application (student data file or dictionary or available stock or…)

a3: Teamwork of the participants in teams of two: handling of AVL-trees (30 minutes).

The class AVL-tree is examined. A small prepared data file has to be included in an AVL-tree. The result is tested for depth. In this tree elements are deleted. Larger prepared data files have to be included and stored once in a linear list and once in an AVL-tree. Times for the search and the transaction of deletion are compared experimentally (this requires prepared Java simulation tools which can visualise effects). The results have to be described and interpreted. (Variants have to be construed where necessary).

a4 Lecture: 20 minutes

12 The basic operations for deletion and implementation

3 Precise formalism

5 Formulating some characteristics with this (maximum depth, runtime, memory requirements)

a5: Teamwork of the participants in teams of two: (70 minutes): modification or drawing up of a programme to determine the maximum depth. (If the participants haven't finished: test it for next time). Maybe also tests to the field of "deletion" (here choosing inorder-predecessors and –successors, tests are actually carried out).

Students are expected to spend two more hours per week at the computer outside the tutored time.

b) Complementary exercises (45 minutes present in the course)

Normally, two to three exercises in each unit. Regarding AVL the students have to implement a concrete problem from the area of their main subject. Another exercise deals with basics (for example AVL-trees that are as thin as possible). A third exercise could be about sorting trees with the help of AVL-trees. The time for individual preparation and the exercises at home are about 90 minutes. During the course the students present their solutions. This takes about 30 minutes. The remaining 15 minutes of the time present at the course can be used for exercises or problems within the practical training.

c) Comments

Lectures are inappropriate for basic reciptique. This concept demands a mixture of short lecture-like parts and longer phases of practical training. The background is subject to consideration in the course and is presented by a student for the whole group.

Other ways of proceeding are imaginable, for example a long analysis of one problem extending over several weeks. This requires extensive preparations, but which can be repeated in a slightly alternated form over the years (around five years). For this time the course should be organised by one department. Variations have to be incorporated anyway, as electronic solutions can easily be handed down from one generation of students to another.

Of course there are already experiences with „compact courses“, but they don’t feature the claimed high percentage of work in the laboratory.

The capacity of such a course is a four-hour course for exercises with a group of 15 students.

The total expenditure for the students is:

Lecture, practical training, course for exercises: three hours altogether.
Additional time at the computer: two hours.
Workload at home in a team and/or alone: three hours.

Total expenditure of time: eight hours each week. With 15 weeks in a semester this amounts to 120 hours. This correlates with four ECTS-points. (For comparison: The usual course consisting of four hours of lecture and two hours of a course for exercises is equivalent to nine ECTS-points, but admittedly, it is more extensive. An average reduction of the ECTS-points to 50% to 75% should be accomplished or the subject matter should be extended with further contents of the intelligence tool box. But we advise caution with the realisation: Presumably, other main subjects will be enthusiastic about it.)

For the realisation we should also take into consideration a different concept: More advanced students serve as tutors for the younger students.

Automatic Time Measurement for UML Modeling Activities

Ira Diethelm[1], Leif Geiger[2], Christian Schneider[2], Albert Zündorf[2]

[1]Gaußschule
Löwenwall 18a
38100 Braunschweig

[2]Fachgebiet Softwareengineering
Universität Kassel
Wilhelmshöher Allee 73
34121 Kassel

(ira.diethelm | leif.geiger | christian.schneider | albert.zuendorf)@uni-kassel.de

www.se.e-technik.uni-kassel.de

Abstract:

In order to improve learning and teaching processes in computer science, we need to analyze current processes qualitatively and quantitatively. Such an analysis may finally allow to evaluate empirically the effects of new approaches and of changes to the process in comparison to previous processes. First of all, the learning effects may be measured using usual examination schemes. However, for deeper insights, the measurement of process results should be correlated to measurements during process execution. This paper outlines the current state-of-the-art in automatic time measurement in CASE tools and what may be achieved in the near future. This is done with respect to empirical studies for learning and teaching processes.

1 Introduction

In his PhD Thesis, Carsten Schulte has proposed detailed protocols of a proband's activities during UML modeling as a means for empirical studies in the area of didactics of computer science, cf. [Schu03]. Carsten Schulte used

- video protocols of the class room at all,
- video protocols of the screen contents of each proband's computer during exercises,
- audio tapes protocolling probands' discussions, and
- internal command protocols of the employed CASE tool.

After the lessons, Carsten Schulte and his team had to evaluate all these protocols manually. For the CASE tool usage, they did a replay of each session and in a raster of a minute, they categorized the probands' activities according to the topic under work and

according to the kind of activity performed (e.g. coding, bug fixing, discussion, ...). As one sees easily, such an evaluation of session protocols is extremely tedious. In order to facilitate empirical studies based on such protocols, this paper explores the possibilities and restrictions of automatic session protocol evaluation via CASE tools.

2 Automatic time measurement

In principle, it is pretty simple to add automatic protocol features to a certain CASE tool. Most CASE tools actually protocol all user interaction already e.g. for undo/redo or for recovery functionality. Thus, all that needs to be done is adding time stamps and logging of all the operations.

Note, this kind of automatic time measurement has a systematic fault, since it just measures editing activities. The times when the proband does not edit but he is thinking about a certain problem or he is discussing a topic with some team mate or he is just out for a break or he is in a meeting or he is interrupted by a phone call or he finds a brilliant solution to a problem during sleep at night, all these non-editing activities are not covered. Covering more of these non-editing activities needs other observation techniques that probably are able to enhance the measurement. However, we have no idea, how to automate the evaluation. Luckily, some recent empirical studies give hints, that in UML projects the amount of editing activities seems to be closely related to the amount of non-editing activities. If this holds, the automatic time measurement could be a reasonable indicator of the overall effort for UML based modeling. Accordingly, this paper assumes that automatic time measurement is a valid means for empirical studies on modeling activities.

Note, a simple protocol of time stamped user commands provides only limited information for empirical studies on modelling activities. To provide substantial value, the protocol must allow to retrieve detailed information of the part of a UML model that is changed, how it is changed, and in the optimal case why it is changed. Ideally, the changes are related to specific kinds of tasks and it is later on possible to relate them to specific elements of the edited UML specification. We will show how such information may be obtained and how this information may be exploited at the example of the Fujaba CASE tool.

The Fujaba CASE tool has a plug-in called CoObRA that adds undo/redo, recovery, and versioning functionality to it (see [Schn03]). CoObRA is an acronym for Common Object Replication frAmework. Basically, Fujaba employs a dedicated object structure, the so-called meta-model to represent the UML diagrams edited by a user. During editing e.g. a class diagram the user adds new objects to the internal meta-model, removes objects from the meta-model or changes the values of certain attributes of certain meta-model objects. All these operations are protocolled at this level of detail together with detailed timing information.

Note, in Fujaba layout information is stored as part of the logical meta-model and thus the corresponding operations are also covered by the CoObRA protocol.

Since certain operations results in a large number of changes to the internal object structure of the Fujaba CASE tool, our protocol groups all changes into so called user commands.

This raises the level of abstraction for the analysis while it still maintains exact information about the modified data.

3 Exploiting the change protocols

3.1 Session replay

Provided with detailed undo/redo information it is for example possible to revert a whole user session and to re-execute it step by step, in slow motion, fast-forward, or even in backward mode. This could be exploited for manual analysis of user sessions as in [Schu03].

3.2 Relating changes to specification elements

For further analysis of change protocols, it is mandatory, to identify the edited specification parts. This might be a certain fraction of a class diagram or e.g. an activity diagram modeling the behavior of a certain method. Note, sometimes the protocol data might not properly identify the edited parts of the specification, e.g. if internal statistical data is targeted. However, due to our experiences, in almost all cases it is very simple to identify the edited diagram element and to relate it to a certain part of the overall specification. This could be done on a coarse grain level e.g. per method body of on a fine grain level e.g. down to the different parts of a sequence or of an activity diagram.

3.3 Relating changes to project phases

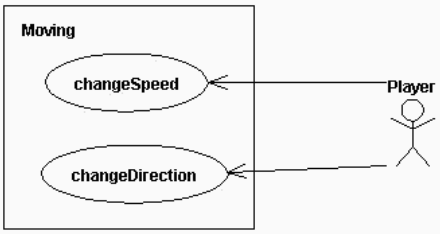
If the user follows a certain process, it is even possible to relate our protocol data to different kinds of activities like requirements definition, analysis, design, implementation, or maintenance. In our courses, we use Fujaba with the so-called Fujaba Process ([DGZ04a, GSZ03]). Fujaba supports this process by providing a document centered view of a project handbook that guides the user through the development process. In this view, the user starts in a dedicated chapter of the project handbook by editing use cases, cf. Figure 1. For each use case, Fujaba automatically adds a pre-formatted section for a textual use case description. This is then filled, manually. (The example is taken from a guided tour created for one of our courses at University of Kassel. It deals with a very simple rule in a board game called Mississippy Queen, where one travels a changing river with a steamer).

4. Analysis Phase

In the following subchapter we place the top level package diagram for use cases.

4.1. Description of Moving

<to be filled>



4.1.1. Description of usecase changeSpeed

Scenario changeSpeed1

Start situation: A new player gets the turn.

Invocation: The game asks the player to change the current speed of his steamer.

Step 1: The player increases the speed three times.

Step 2: The player decreases the speed one time

Step 3: The player stops changing the speed and the coal is decreased by one

Result situation: Speed has increased by two and coal has decreased by one.

Figure 1: Requirements definition in the Fujaba Process

A special user command turns such a textual use case description into the outline of an UML interaction diagram allowing to elaborate the use case description. In our case these are so-called story boards, a combination of UML activity and UML collaboration diagrams, cf. Figure 2.

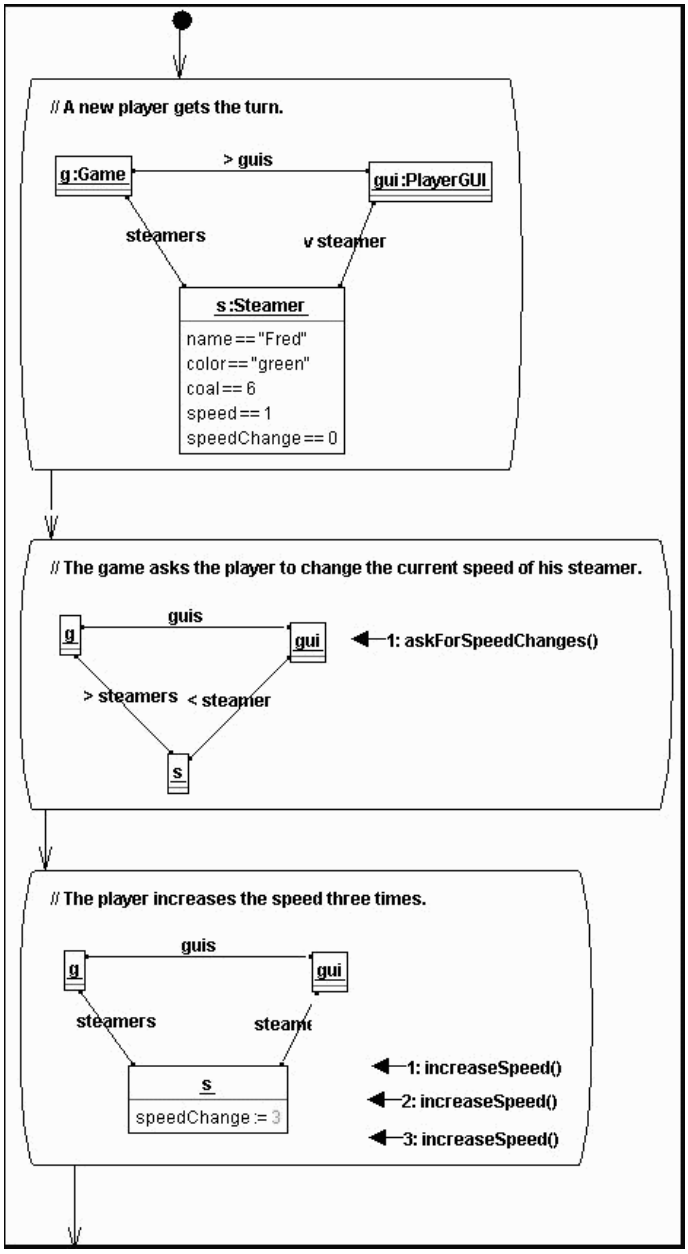


Figure 2: Analysis with story boards

From such a story board, specific user commands derive a class diagram and a test specification. The class diagram already includes declarations for all methods employed in the scenarios, cf. Figure 3.

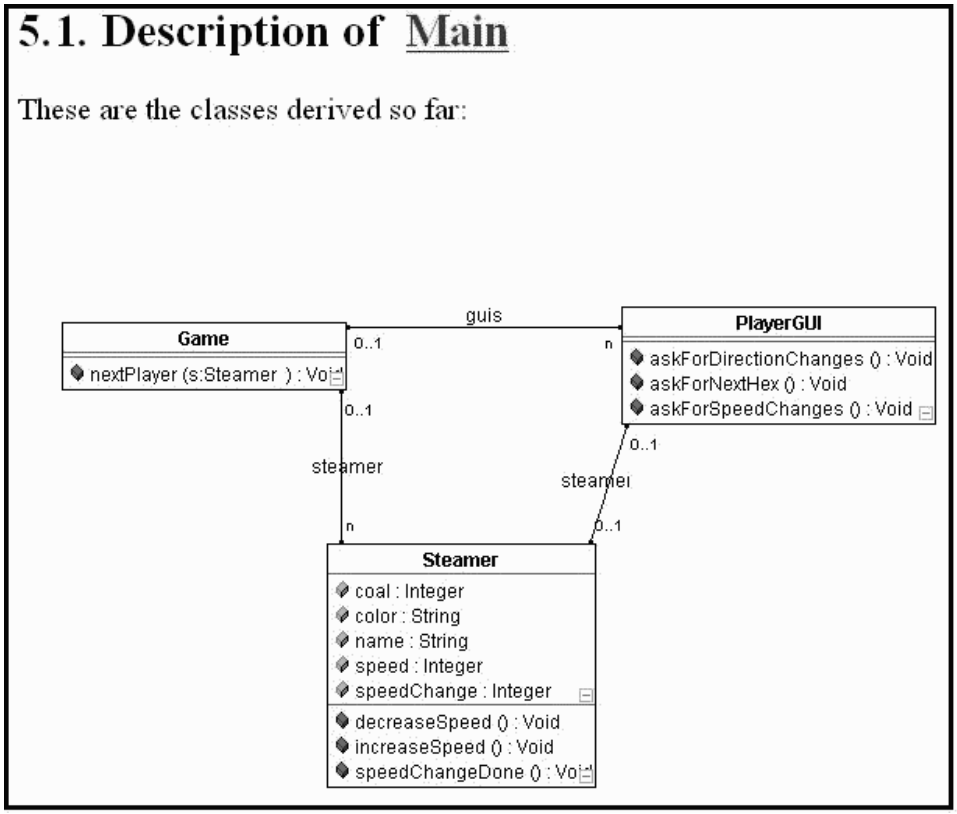


Figure 3: Class diagram derived from the story board

Now, the user has to implement the method bodies appropriately, cf. Figure 4. Once the functionality is provided, the user generates code from his specification, compiles it and tests it against the test derived from his story boards.

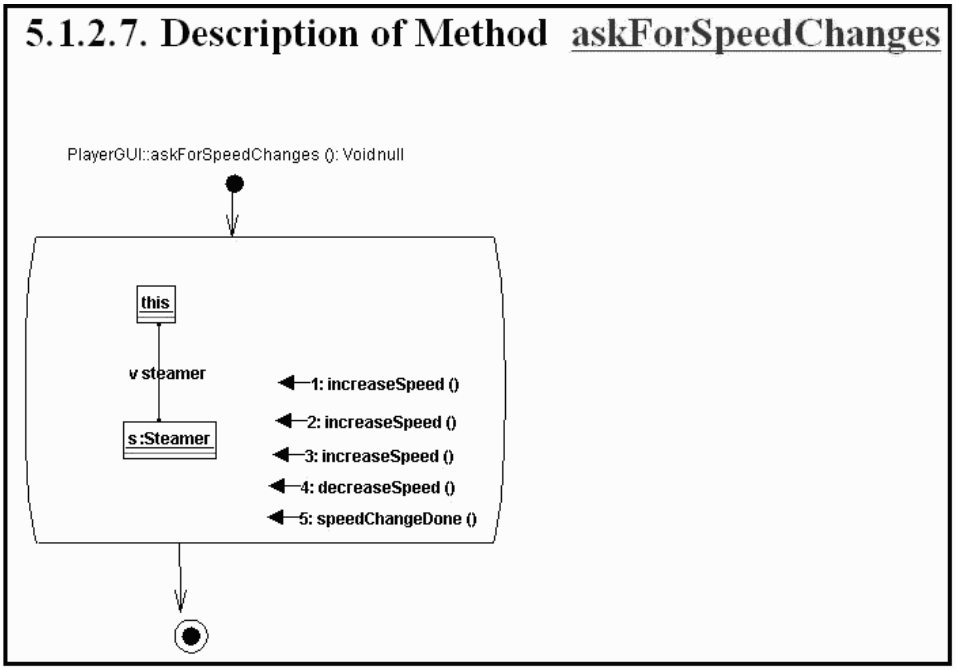


Figure 4: Method body specification

Following this process allows to relate editing activities to project phases: editing a use case or a textual use case description is considered as a requirements activity. Editing a story board belongs to the object oriented analysis phase. And editing a method body means implementation effort. Finally, any activity after successful compilation and after running the first test may be considered as a testing, bug-fixing, and maintenance activity.

Note, the Fujaba process is an use case driven and an iterative process. This means, the developer realizes one use case after the other. Thus, after implementing the methods employed in a certain use case, all testing and bug fixing activities are related to the same use case until the developer starts editing another use case.

3.4 Relating changes to tasks

As already discussed, the Fujaba Process may be used in an iterative way and by project teams. In this case, different team members may work at different use cases. Each team member may work on just one use case at a time. As outlined above, in Fujaba it is still possible to relate most editing activities to specific use cases. In the case of editing a textual use case scenario, this is trivial. The same holds for a story board, since a story board always elaborates a certain use case. In case of method bodies the situation is not always clear. However, in the Fujaba Process each use case describes a certain system

functionality that is realized by a dedicated method. Editing this method is clearly related to the corresponding use case. Similarly, the story board elaborating a given use case may employ some additional methods. If these methods have not yet been used in other use-cases they may be related to the current use case. Finally, we derive automatic tests from each story board. Bug-fixing activities caused by failure of such a test may also be related to the corresponding use case. Thus, in most cases we are able to relate editing activities to dedicated steps in the realization of a dedicated use case, even in an iterative, multi user process.

Note, due to our detailed protocol data it might be possible to analyze which parts of a specification are modified in response to a failed test. If only a method body is edited, it was an implementation problem. If the class diagram is changed, it was a design problem. If even a story board or a use case scenario needs to be adapted, it is an analysis or requirements problem, respectively. This might be related to the overall effort for fixing the bug. This may allow to study the question, whether in iterative processes the assumption still holds that bugs in early phases cost a magnitude more than bugs in later phases.

3.5 Summing up editing times

Until now, we have just related editing activities to specific tasks in the modelling process. In addition, we have some experiences in summing up the times for these editing activities. As already mentioned, Fujaba's internal change protocol provides time stamps for all editing activities. Usually, these time stamps show phases of intensive editing where subsequent editing steps have very short time distances (some seconds) followed by certain gaps, where no editing activity is recorded (for several minutes). As discussed in the introduction, the tool is not able to guess what is going on during these gaps. The user may be thinking or working on the problem with pencil and paper or discussing it with his team mates. Or the user may just take a break. As discussed, we just measure the editing activities and hope that they resemble the over all modelling effort, closely.

Based on the time stamps of our activity protocol, there are multiple ways to sum up the time spent on the different tasks on a project. A simple scheme might e.g. assume a minimal time required for a single editing activity e.g. 10 seconds and a maximum time between two editing activities that is not considered as a break, e.g. one minute. Accordingly, if we record only a single editing command, we add the 10 seconds to the time spent on the corresponding task. Second, if we do not record editing for more than e.g. one minute, we assume that the user takes a break. In that case we might add the time span from the first activity after the previous break until the last activity before the new break plus 10 seconds to the corresponding task. If the task changes during a sequence of activities e.g. between step a and b, we might cut the interval in the middle between step a and b.

Using such an approach, it might be possible to measure the time spent on a specific task with some realistic precision. This precision might be adjusted by empirical studies collecting more precise time data with alternative (manual) means.

If such an automatic time measurement delivers reliable data, there are multiple applica-

tions of such an approach in the area of software engineering and project management. For example, statistical data collected in this way from several projects might be used as a basis for effort estimations and project planning for new projects. Similarly, time data collected during project execution might be used for project tracking, cf. Figure 5. Each time, a certain task is completed, the tool might relate the measured time spent on that task with the time estimated for that task. Such a comparison may allow to indicate phases of good progress as well as delays for certain tasks that may need management intervention.

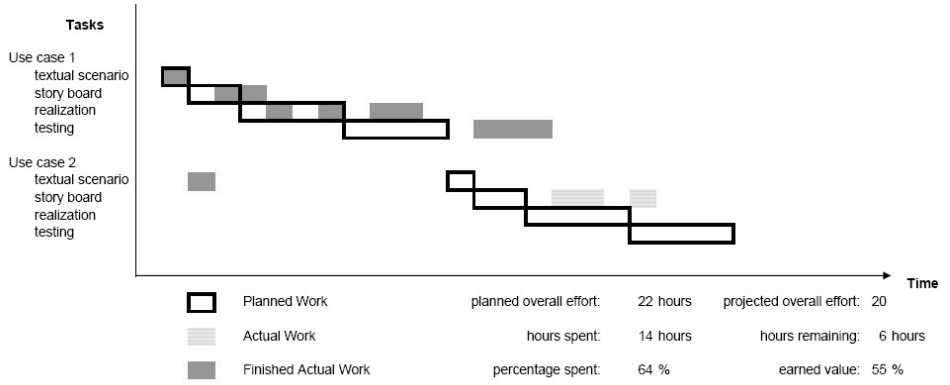


Figure 5: Possible exploitation of automatical time measurement

The example in Figure 5 outlines some possible project plan view based on such an automatic time measurement. The hollow bars indicate planned efforts for two use cases. These effort estimations could be derived from earlier projects. The green bars show actually spent time related to tasks. Note the gaps in these bars that might be caused by phases of thinking and discussions or e.g. by lunch and coffee breaks. The grey bars indicate time spent on task that are not yet completed. In this example, the estimated over all effort is 22 hours. The measurement of the actually spent time sums up to 14 hours so far. Thus, in this example already 64% of the project budget are consumed. Summing up the percentage of completed tasks results in only 55%. However, only 11 hours (half of the budget) have been spent in order to complete 55% of the tasks, thus in this example the project seems to be slightly ahead of schedule. Such situations could be reflected in an adjusted projection of the required overall effort and of a projection of the time required for completion.

Such an exploitation of the automatic time measurement is especially of interest for the area of software engineering. However, we are confident, that similar analysis mechanisms could also provide valuable input for more general empirical studies on modelling activities.

3.6 Editing patterns

In our work with the Fujaba environment in class rooms and by supervising student projects we frequently observe typical editing patterns. For example, during editing a story board the students frequently detect the necessity of an additional object within their story board while editing some later activity. Adding an object to the story board in this phase requires to go back to the first activity, to add the object there and then to copy this change forward to the next activities step by step until the student reaches his former point of editing, again.

Such editing patterns are very interesting from an analysis point of view since they indicate situations or points in time when the student has discovered a misconception in his solution. This might e.g. give hints for insufficient group discussions on the scenario.

Fujaba provides some functionality for pattern detection within static source code. Currently, this functionality is extended towards analysis of program execution traces. Similar techniques might be usable for the analysis of editing patterns, too.

4 Summary

This paper outlines the automatic protocol features of the Fujaba CASE tool. This automatic protocol features enable us to replay user sessions and to relate editing activities to different project phases and to dedicated use cases. This is supported for iterative projects with multiple developers working on a common project in parallel. On this basis, numerous other analysis mechanisms may be realized.

We propose to use these automatic protocol evaluation features of Fujaba to automate protocol evaluation in empirical studies like the one of [Schu03] .

References

[DGMZ02] I. Diethelm, L. Geiger, T. Maier, A. Zündorf: Turning Collaboration Diagram Strips into Storycharts; Workshop on Scenarios and state machines: models, algorithms, and tools, ICSE 2002, Orlando, Florida, USA, 2002.

[DGZ02] I. Diethelm, L. Geiger, A. Zündorf: UML im Unterricht: Systematische objektorientierte Problemlösung mit Hilfe von Szenarien am Beispiel der Türme von Hanoi; in Forschungsbeiträge zur ”Didaktik der Informatik“ - Theorie, Praxis und Evaluation, GI-Lecture Notes, pp. 33-42, 2002.

[DGZ04a] I. Diethelm, L. Geiger, A. Zündorf: Systematic Story Driven Modeling, a case study; Workshop on Scenarios and state machines: models, algorithms, and tools, ICSE 2004, Edinburgh, Scottland, 2004.

[DGSZ04b] I. Diethelm, L. Geiger, C. Schneider, A. Zündorf: Measurement of modeling abilities; Concepts of Empirical Research and Standardisation of Measurment in the Area of Didactics of Informatics (CERSMADI), Dagstuhl, Germany, 2004.

[Fu02] Fujaba Homepage, Universität Paderborn, http://www.fujaba.de/.

[GSZ03] L. Geiger, C. Schneider, A. Zündorf: Integrated, Document Centered Modeling in Fujaba; 1st International Fujaba Days, Kassel, Germany, 2003.

[Hu00] P. Hubwieser: Didaktik der Informatik - Grundlagen, Konzepte, Beispiele; Springer Verlag, Berlin, 2000.

[Hu98] Watts S. Humphrey: Introduction to the Personal Software Process; Addison-Wesley, Amsterdam, 1998.

[KNNZ00] H. Köhler, U. Nickel, J. Niere, A. Zündorf: Integrating UML Diagrams for Production Control Systems; in Proc. of ICSE 2000 - The 22nd International Conference on Software Engineering, June 4-11th, Limerick, Ireland, acm press, pp. 241-251, 2000.

[life02] life^3-Homepage, Universität Paderborn, http://life.uni-paderborn.de/.

[Schn03] C. Schneider: CASE Tool Unterstützung für die Delta-basierte Replikation und Versionierung komplexer Objektstrukturen; Diploma Thesis, Corolo Wilhelmina zu Braunschweig, Braunschweig, Germany, 2003.

[Schu03] C. Schulte: Lehr- Lernprozesse im Informatik-Anfangsunterricht; PhD Thesis, University of Paderborn, 2003.

[SN02] C. Schulte, J. Niere: Thinking in Object Structures: Teaching Modelling in Secondary Schools; in Sixth Workshop on Pedagogies and Tools for Learning Object Oriented Concepts, ECOOP, Malaga, Spanien, 2002.

[Zü01] A. Zündorf: Rigorous Object Oriented Software Development; Habilitation Thesis, University of Paderborn, 2001.

Measurement of modeling abilities

Ira Diethelm[1,2], Leif Geiger[2], Christian Schneider[2], Albert Zündorf[2]

[1]Gaußschule
Löwenwall 18a
38100 Braunschweig

[2]Fachgebiet Softwareengineering
Universität Kassel
Wilhelmshöher Allee 73
34121 Kassel

(ira.diethelm | leif.geiger | christian.schneider| albert.zuendorf)@uni-kassel.de

www.se.e-technik.uni-kassel.de

Abstract:

This paper discusses the difficulties of measuring modeling abilities within examinations. Modeling abilities are inherently difficult to measure since they imply cognitive processes that may not become evident in the result of a written examination. In addition, for a given problem there exists a wide variety of valid models that may just differ in the employed modeling language, technique, or paradigm. The models may just differ with respect to the aspects of the problem that are covered. Or the models may differ in the level of abstraction that has been chosen, e.g. UML level or code level. Even for a given modeling language and for clearly identified aspects that are to be covered and for a given level of abstraction there are still many possible solutions for a given problem that are difficult to compare and where it is difficult to judge their relative quality. This paper will mainly raise questions related to these problems. However, in addition we will describe a specific solution employed at the University of Kassel for grading the modeling abilities of 3rd term students.

1 Introduction

Modeling abilities are of major interest in the discussion of educational standards. Accordingly, measuring techniques for modeling abilities need to be created, evaluated and established. Generally, there are two different basic approaches: measuring the progress during the modeling process or evaluating the result. In the following article we focus on the results.

Many problems in this area are caused by the wide variety of valid solutions for one given problem. Different modeling languages may be used, different levels of abstraction may be chosen and there may actually exist multiple valid solutions for the same problem. In

this paper we do not present a solution in general but we show one possibility we found for measuring special UML models.

In chapter 2 we point out the initial situation and assumptions we made. Then we discuss the difficulties in measuring modeling abilities in general in Chapter 3. Therefore we rely on the measurement of mental models as a basis for information processing and problem solving. We also specify some requirements on measurement techniques that are concluded from discussion in this chapter.

Furthermore, in Chapter 4 we show some difficulties that we identified during our lessons in software engineering at the University of Kassel. In this course, we evaluated the modeling abilities of the students (not their knowledge in UML) on the basis of a homework consisting almost only of UML diagrams. To measure the amount and quality of the functionality modeled within a homework, we used so called *norm activities* that we describe in Chapter 5. Norm activities allow us to measure the size and quality of an UML interaction diagram (story boards and story diagrams).

In chapter 6, we resume and reflect which requirements are covered by our solution and which are not. We conclude with future work to be done.

2 Initial Situation and Assumptions

Any discussion of the measurement of modeling abilities requires a sound definition of the term modeling ability, first. In this paper, we consider modeling ability as the ability to capture an existing or described context and to create a mental model illustrating the given context as suitable as possible. Furthermore it requires to describe this mental model with a well-known and suitable modeling technique to make it accessible and assessable to others.

Our interpretation of mental models is based on [LWS96], where the authors prove that human information processing, thinking and concluding often takes place in entire models, which structures resemble the conditions of the given context.

In the area of software engineering, models are frequently implemented using a standard programming language and the size of such a program model is measured with function points or lines of code. From such a size measurement one derives the costs for the production of the model. Obviously, code size is no suitable measure for the students modeling abilities. In our opinion, today's programming languages deal with too many technical details and have a too low level of abstraction. In our opinion, a higher level of abstraction that is closer to the way of human thinking is needed in most aspects of education. Besides, we would like to be able to measure modeling abilities also with tests, which do not require an executable program and do not require too much effort to be suitable.

In addition, the length of an executable program is no measure for its quality. Compared with the size of a sample solution, longer solutions should be considered as worse compared to smart short solutions realizing the same functionality.

Teachers at school have very diverse evaluation criteria. In most cases an exercise is given

requiring a solution method and a result, e.g. the answer to a question. The work on the exercise is documented by the student and evaluated by the teacher. Usually points are assigned to the basic approach, the solution method and the result. The tasks are assigned to certain difficulty ranges and several of such tasks form a test. For example, the tasks of the German „Abitur“ have to be specified in this way (see [KMK04]). However, such an evaluation scheme measures the ability to apply a pre-defined solution method to an appropriate problem.

In our opinion, good modeling abilities will frequently result in a wide variety of solutions for a given problem where each solution has its own value and quality. Thus, an evaluation scheme for modeling abilities should not assume a certain standard solution but it should provide freedom for alternative solution approaches.

3 Difficulties in General

As discussed, the common test schema requires a fixed solution strategy and thus is not applicable to computer science. It differs from mathematics and other natural sciences since in computer science, for most modeling problems, there are many solutions to an exercise where each solution may be as good as the other. Thus, the measurement of modeling abilities requires flexible evaluation schemes that are able to deal with a wide variety of different solutions.

Second, a problem may be modeled at different levels of abstraction. One may use a fairly coarse grained UML model, e.g. only a class diagram, or a very fine grained level, e.g. a fully implemented program in a standard programming language. Solutions at such different levels of abstractions are not easily compared or graded with a single evaluation scheme.

Similarly, different schools or different teachers usually teach many different modeling techniques. This creates the problem of comparing and grading solutions to a problem that are described using different modeling languages. To overcome this problem, we need to find a small set of common modeling languages that unifies currently used modeling techniques and that allows to compare different solutions more easily. Ideally, for wide range examinations like a „Zentralabitur“ examination or a PISA test, all students should have similar experiences and skills in the modeling techniques employed in the examinations.

But it doesn't appear neither realistic nor meaningful to demand, that only one or two modeling techniques should be taught nation wide, just in order to be able to establish a simple evaluation scheme. Usually, educational standards should not be effected by evaluation methods, but vice versa. However, common nation wide educational standards would be of great value, anyhow. And if this is achieved, evaluation would benefit from it, too.

As an alternative to a specific modeling language, written or verbal natural language could be used to describe a solution to a given modeling problem. This could be used to seize and evaluate the mental model of students, cf. [HBB00]. However, [HBB00] points out that natural language is only suitable for the measurement of mental models under certain conditions, since some knowledge is difficult to verbalize. In our opinion modeling abili-

ties in the area of computer science raise a similar problem. In addition, the evaluation of a textual description requires an individual interpretation by the examiner. This creates problems for the comparability of grades given by different examiners.

[HBB00] also discusses free graphical representations of information, which they prefer in comparison to textual descriptions. Additionally [HBB00] points out the problems of unrestricted graphical representations: unrestricted graphics have to be interpreted by an examiner also. Still, [HBB00] assumes graphical descriptions to be a comparatively intuitive approach for learning if they have pre-defined meanings for the used symbols.

4 Our Modeling technique: Story Driven Modeling

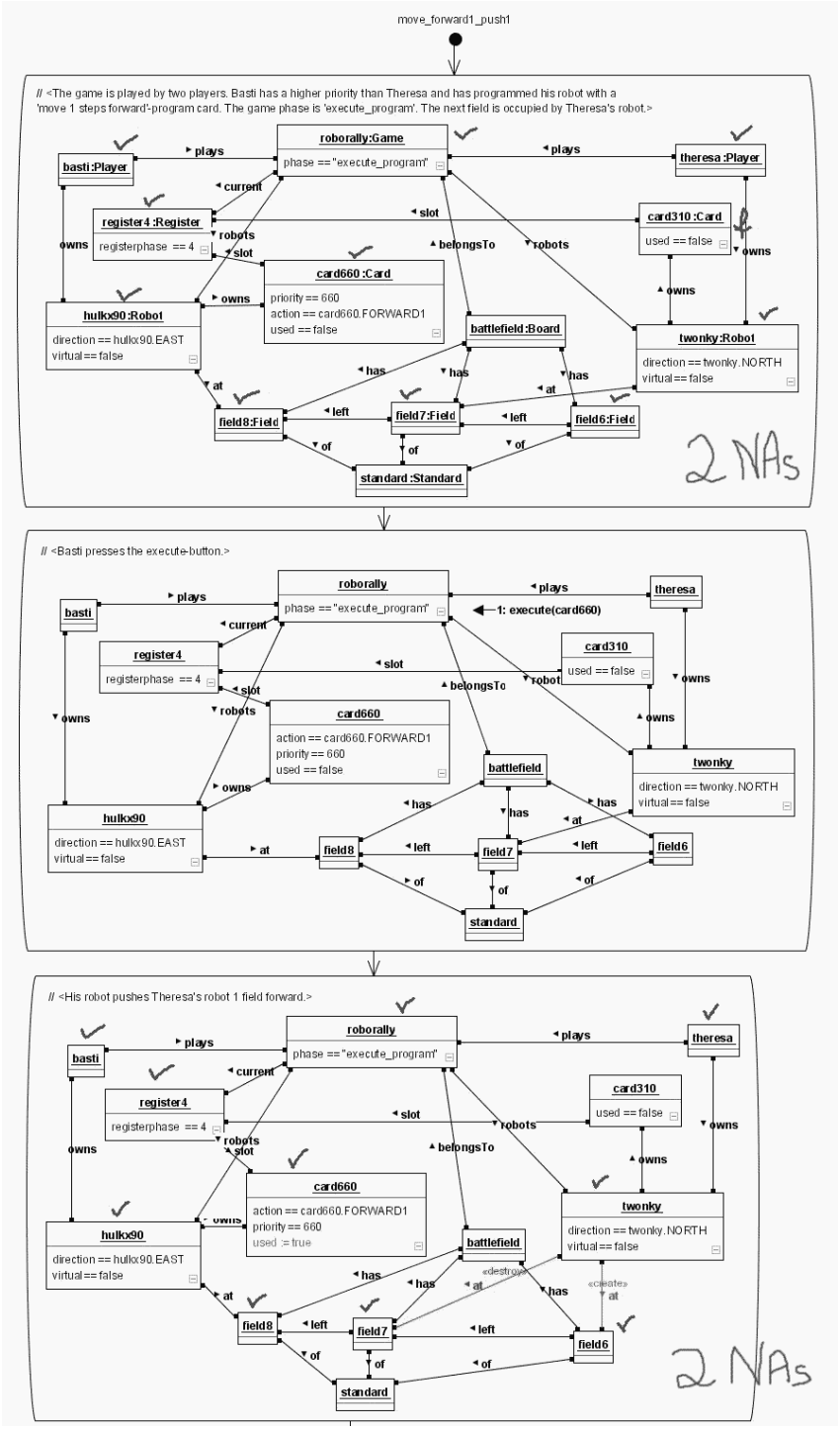
We have developed a tool supported software process called Fujaba Process (FUP) which we teach in our courses at school as well as at the University of Kassel [DGZ04]. The underlying modeling technique is called Story Driven Modeling (SDM). In this paper, we report our experiences with evaluating SDM models, created by our students as a homework for a UML lecture.

The students had to model the board-game „Roborally“ using our CASE tool Fujaba [Fu02] and the Fujaba Process. Since the Roborally game consist of many rules where some are very complex, the students did not have to model the complete game, but only parts of it. Since the students could freely decide which parts of the system they liked to model and how they wanted to model the functionality, we had to find an approach to compare and evaluate these models.

To illustrate our evaluation criterions, we will take a closer look at the FUP and the UML diagrams used in this process:

The FUP starts with the identification of usecases within the problem domain. For every usecase one or more textual scenarios (descriptions of example runs) are written. Then the developer has to translate these textual scenarios into so called story boards. A story board is a sequence of object diagrams, which shows the evaluation of the object structure comic-strip alike. Figure 1 shows a story board for a scenario where a robot moves and pushes another robot. The example used here is originally taken from a student group of our Roborally project.

The embedded object diagram in the first activity in figure 1 models the start situation of this scenario. Note, the corresponding textual description is automatically copied as comment into this activity. Our students have modeled the initial object structure using an object *roborally* of the class *Game*, which represents the functionality of the game. Two *Player* objects *basti* and *theresa* are linked to the game via a *plays* link. The robots are located next to each other on the fields *field8* and *field7*. The robot *hulkx90* has a card which tells him to move one field forward (modeled by the action attribute of object *card660*).


move_forward1_push1
// <The game is played by two players. Basti has a higher priority than Theresa and has programmed his robot with a 'move 1 steps forward'-program card. The game phase is 'execute_program'. The next field is occupied by Theresa's robot.>
basti :Player
roborally:Game
phase == "execute_program"
theresa :Player
register4 :Register
registerphase == 4
card310 :Card
used == false
card660 :Card
priority == 660
action == card660.FORWARD1
used == false
hulkx90 :Robot
direction == hulkx90.EAST
virtual == false
battlefield :Board
twonky :Robot
direction == twonky.NORTH
virtual == false
field8 :Field
field7 :Field
field6 :Field
standard :Standard
plays
current
slot
robots
belongsTo
owns
has
at
left
of
2 NAs
// <Basti presses the execute-button.>
basti
roborally
phase == "execute_program"
1: execute(card660)
theresa
register4
registerphase == 4
card310
used == false
card660
action == card660.FORWARD1
priority == 660
used == false
hulkx90
direction == hulkx90.EAST
virtual == false
battlefield
twonky
direction == twonky.NORTH
virtual == false
field8
field7
field6
standard
// <His robot pushes Theresa's robot 1 field forward.>
basti
roborally
phase == "execute_program"
theresa
register4
registerphase == 4
card310
used == false
card660
action == card660.FORWARD1
priority == 660
used := true
hulkx90
direction == hulkx90.EAST
virtual == false
battlefield
twonky
direction == twonky.NORTH
virtual == false
field8
field7
field6
standard
«destroy»
«create»
2 NAs

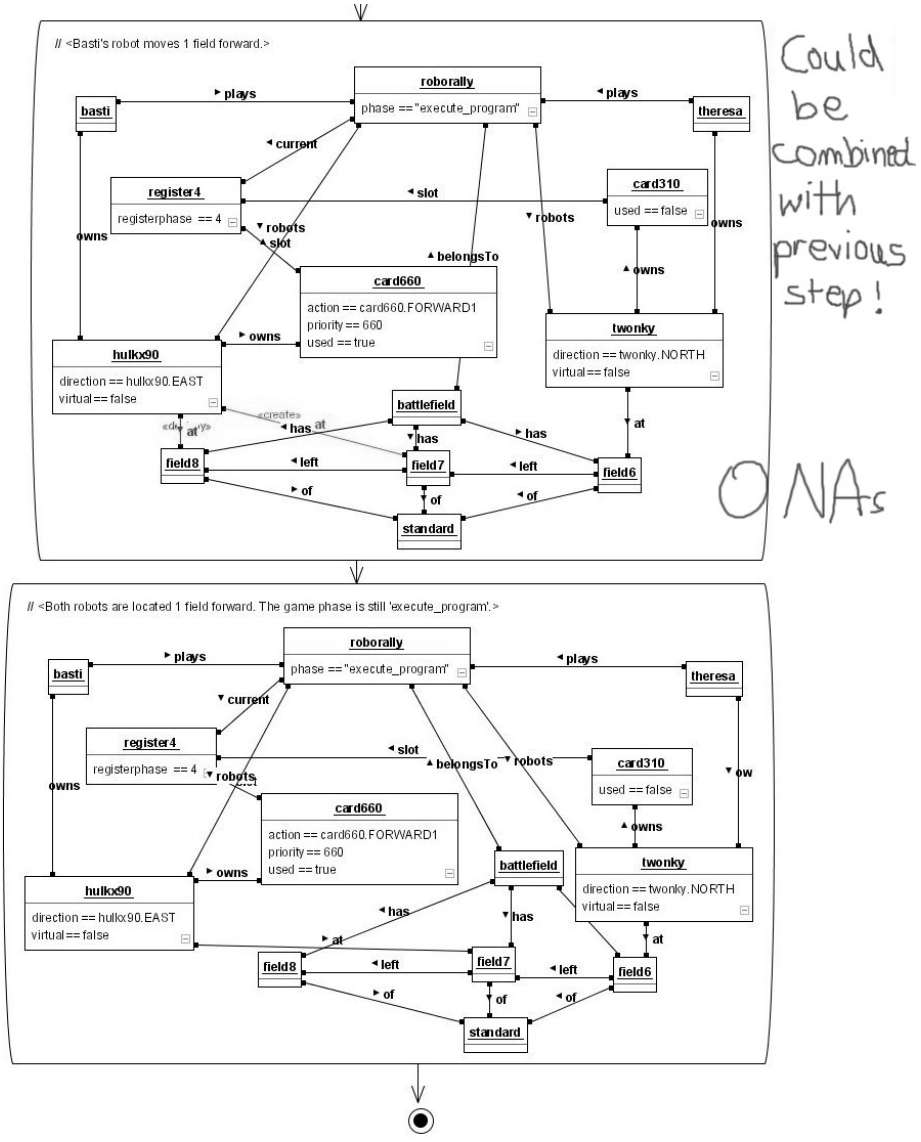


Figure 1: Story board for scenario *move_forward1_push1*

The next activity always models the invocation of the scenario. In FUP this is always a method call. Our students have modeled this by sending the *roborally* object a *execute(card660)* message.

In the following activities, the developer has to model several steps, which describe the changes done to the objects structure during this scenario. The third activity of figure 1 shows the pushing of the robot *twonky* to field *field6*. Note, that we model creation and

destruction of links / objects using ≪create≫ and ≪destroy≫ markers. The card object *card660* is also marked as used by setting the *marked* attribute to *true*.

In the next activity the pushing robot *hulkx90* is also moved one field forward to field *field7*. It is not obvious here, why our students have chosen to make this a sole step. This action could as well have been executed in the activity above.

The last activity always models the result situation which has to be reached if the scenario is successfully executed. Here, the robots *hulkx90* and *twonky* have both moved one field forward and the card *card660* has been marked as used.

From the story boards, the main class diagram may be derived, automatically. The developer may refine the class diagram by adding inheritances, changing cardinalities etc.

We also suggest a systematic approach how to derive the behavior specification (here: the method bodies) from the story boards (see [DGZ02, Zü01]). Method body specifications are modeled using so called story diagrams. A story diagram is a UML activity diagram with UML collaboration diagrams embedded into the activities. The activity diagram specifies the control flow whereas the collaboration diagrams model the changes done to the object structure. Figure 2 shows such a story diagram.

The story diagram of figure 2 models the behavior of the method *doIHaveToPush()* of class *Robot*. This method is a helper method needed by our students to implement the pushing of robots as specified in the story board of figure 1. The method returns *true* if there is a robot which must be pushed in front of the robot on which the method was called, and *false* otherwise.

The first activity checks wether the virtual attribute of the object of class *Robot* on which the method has been called (called *this* in Java and Fujaba) is set. If this check succeeds the activity is left via the *[success]* transition. In this case, the method is left and returns false since virtual robots do not interact which other robots in the Roborally game. Otherwise the activity is left via the *[failure]* transition.

The collaboration diagram in the second activity tries to identify the specified object structure. The matching is started at the *this* object. From there the *robots* link is followed. If a object of class *Game* is found, it is called *roborally*. From the *roborally* object a robot is searched using the *robots* edge, which has an *at* link to the object *nextField*. This robot is then called *robotX*. Note, the object *nextField* is already known to the system since it has been passed as parameter. In Fujaba known objects are visualized by omitting the class name after the object name. Such objects are called „bound".

If this object structure analysis fails, the activity is left via the *[failure]* transition and again *false* is returned. Otherwise the *[success]* transition is taken to the third activity. Here it is checked if the attribute *virtual* of the bound object *robotX* (known from the object search in the previous activity) has the value *true*. If yes the method returns *false* and *true* otherwise.

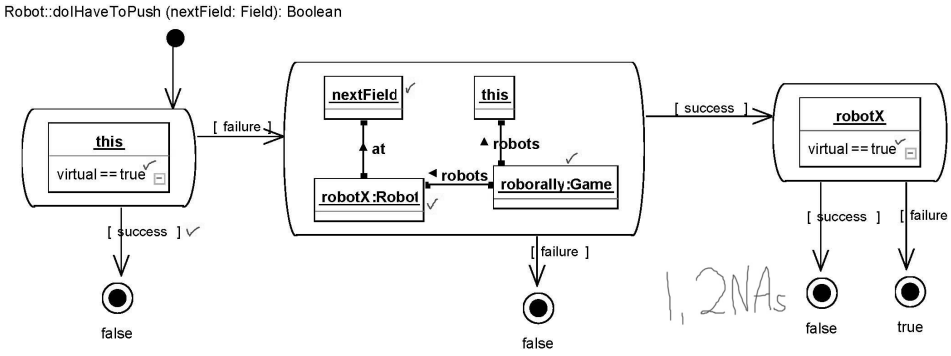


Figure 2: Story diagram for method *doIHaveToPush()*

From the class diagrams and the story diagrams the Fujaba CASE tool automatically generates executable Java code, which may be compiled and then tested using our object browser DOBS.

From the story boards the Fujaba CASE tool generates also automatically JUnit test specifications ([Gei04]). This generated tests check wether or not the implementation (modeled by story diagrams) covers the scenario specified by the corresponding story board. Using this test generation enables us to verify easily which scenario has been completely modelled and which not. But we still have no information about the complexity of a particular scenario and of the quality of the model.

At the end of the Roborally project, we had to evaluate the results presented to us by our students. This brought up the following questions:

- How should we evaluate story boards and story diagrams? Just giving points for every object, link, attribute condition etc. would just measure the size of the diagrams and would not take the real complexity of the modeled system and of the quality of the model into account.
- How can we measure functionality of systems that do not have a user interface and are therefore not testable by humans?
- How can we evaluate pure models that do not have an implementation?
- How can we measure the complexity of a scenario / method specification?

5 Our Solution: Norm Activties

We introduced the term of a „norm acitvity“ (NA) to evaluate the modeling abilities of our students. An NA is a group of five objects within a story board or story diagram which are involved in a non-trivial change to the object structure. Such changes may be changes to

attribute values, creation or destruction of links or objects and messages sent to objects. Objects not needed for these changes as well as symmetries do not count for NAs.

For passing the Roborally project, each of our students had to model 10 NAs with story boards and another 10 NAs with story diagrams within roughly 2 weeks of work. To be able to count NAs of story boards and story diagrams easily and repeatably, we set up a list containing criteria on how to count and how to identify symmetries:

Criteria list for story boards:

- If one step of a story board has about 5 objects that are needed for a sound realization the step counts as 1 NA.
- For a multiple of 5 objects the steps counts more NAs according to the factor (10 needed objects → 2 NAs).
- If a story board has no significant new elements, it is not or only partially counted.

  This criterium is needed to identify symmetries between story boards.
- Invocation and result situation are not counted.

  This is because the invocation is mostly symmetrical to the start situation and the result situation contains the same objects as the previous steps.
- If one could describe multiple steps in one step, the NAs are counted only once.

  This again identifies symmetries.
- Trivial story boards count less or even no NAs (e.g. only one changed attribute).

  In this case, the scenario was to simple and does not fulfill the requirements for containing NAs.
- For start activities we count all objects that are necessary for the subsequent steps. Objects that are not counted in any of the subsequent steps are not counted in the start situation either.

  This criterium facilitates to identify the part of the start situation that models only the context of the scenario but that is not used later on.
- Parts of a story board which do not fulfill the requirements description do not count.

  Here, the correctness of the model affects the counting of NAs.

Criteria list for story diagrams:

- Each object that is needed for a sound realization counts 0.2 NAs.

  This does again lead to 1 NA for a group of five objects.
- Branches, Loops and For-Each-Actvities count 0.2 NAs.

  Since control-flow plays an important role when modeling behavior, constructs creating control-flow are counted as well.

– Methods that do not work (no green JUnit bar) are not counted.

  If a method has not passed the tests automatically generated from the story boards, it does not cover this specific scenario and so this method is erroneous within the small part of the modeled system.

– Unusable parts of methods count nothing.

  Our students tend to model branches that are never reached. Of course such parts of a model are not counted.

– Redundant structures of any type are not counted.

  This criterium should avoid symmetrical parts to be counted more than once.

– If one can model multiple story pattern in a single one, only one story pattern is counted.

  This again identifies symmetry.

– Case differentiations with very small differences are counted only once.

  Some of our students modeled huge switch-case constructs with only little differences between the different cases (e.g. move by one, move by two, move by three fields). Of course, such constructs are only counted once.

– The *this* object does not count.

  This is because the *this* object is needed in most cases to start an object structure analysis and does not stand for any modeling effort.

Applying the criteria above to the story board of figure 1 leads to a total of 4 NAs. We start counting NAs with the first step which is the third activity in Figure 1. As mentioned in chapter 4, the first and the second step (activities three and four) of the story board can be combined. So, according to the fifth criterium of the list above, every object is only counted once. Combining the two steps would result in figure 3.

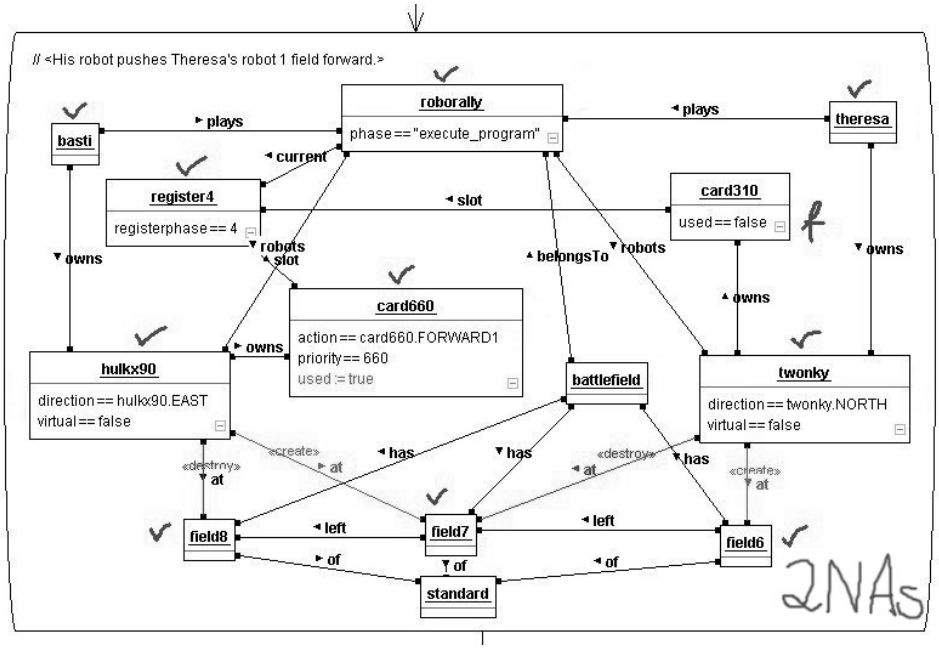


Figure 3: Combination of step 1 and 2 of figure 1

Every object with a red check mark is part of an NA. The object *battlefield* does not count since it does not take part / is not important for the changes to the object structure. The object *card310* might be important for choosing who is next because according to the Roborally rules, the card with the highest priority is played next. But because the students have not modeled a priority of *card310* (using the priority attribute) this object is useless here and therefore does not count. The object *standard* is not part of an NA as well. Of course, the information on the type of field on front of a robot, if it contains walls or holes is important for moving. But here the students have modeled an *instance of* relationship explicitly as a link. Inheritance would have been the better choice here and that is why the *standard* object is not needed here and does not count for NAs. So this step contains 10 objects counting for NAs. Since we have four changes to links and one attribute value assignment, these changes are non-trivial even for 2 NAs and therefore this step counts the whole of 2 NAs. Invocation and result situation do not count for NAs, so since this step is now the only one, only the start situation still has to be counted. According to criterium seven of the above list, we count the elements of the start situation that are employed in later steps. In this case this also sums up to 2 NAs. We end up with a total of 4 NAs for the whole story board. This means that two story boards of this complexity and one simpler one would be sufficient to pass the project concerning story boards.

For understanding the evaluation of story diagrams, we have a look at figure 2. By negating the attribute assertions, the three different branches can be combined to one. This would result in the story diagram in figure 4, which has obviously the same behavior as the one of figure 2. So, we use this diagram for counting NAs.

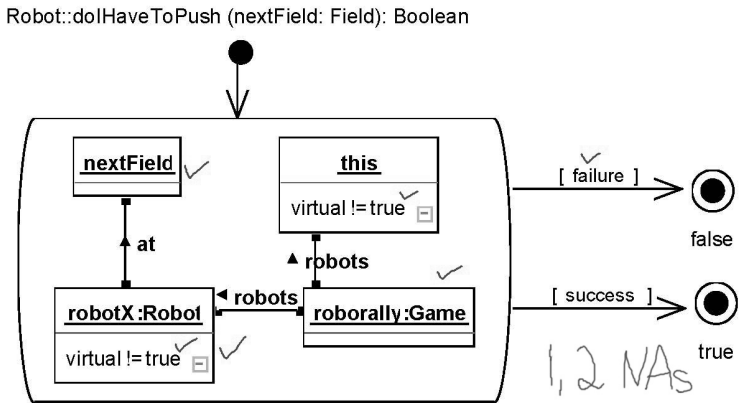


Figure 4: Simplified version of figure 2

Every object except the *this* object and every attribute assertion counts 0.2 NAs. The branching condition again counts 0.2 NAs. So this diagram would have a total of 1.2 NAs.

Using the so counted NAs enables us to measure the size of the model created by our students while ignoring symmetries and useless parts of their models. To also measure the quality of their models, we made the following considerations:

> The size of a standard solution to some modeling task may be considered as a measurement for the complexity of that task. This allows to measure the complexity of the modeling tasks addressed by a homework and enables a comparison of the quality of the homework with respect to the standard solution.

Concluding, standard solutions provide

- a measure for the complexity of the task on the one hand
- a measure for the weight of errors or unworked task on the other hand

The comparison of size (in NAs but also in pages or LOC) of the standard solution with the solution of our students gives a hint of the quality of the presented solution. Comparison with the standard solution can easily answer the question: Was a long winded model chosen that needs much more and more complex branches and contains much „Cut-Copy-Paste" code?

Concluding, we found a way to measure complexity of models created with our modeling technique. We also made some progress in measuring the quality of such models. We found out that the intuitively indicated rating by the teacher and our rating by norm activities gave similar results. Another interesting observation was that for our modeling technique models of static contexts, e.g. class diagrams, are less interesting and of less effect to the measurement than models of dynamic processes.

Our approach also allows the evaluation of partly modeled systems which do not need to be runable. It allows the evaluation of pure models which do not need to be created using a CASE tool. Such models also may only be specified e.g. on paper.

Concluding, we think, that using NAs is a step forward to achieve objective measurement of models, but lot of work still has to be done.

6 Summary and future work

As discussed, there are many open problems for the measurement of the modeling abilities of students. In our opinion, the following steps should be executed in order to improve this situation:

- modeling abilities should be measured at a higher level of abstraction as provided by e.g. programming languages. Today's programming languages deal with too many tiny technical problems that are not related to modeling abilities. In addition, the creation of an executable model for an even modest real world problem in an usual programming language requires just to much effort to be applicable for common examinations.
- we need to measure modeling abilities and not modeling language skills. Thus, we need an intuitively usable modeling language that does not require several month of training and that does not restrict the solutions to a given problem.
- we need to be able to measure the complexity of a modeling problem in order to be able to classify examination tasks into categories of difficulties.
- equipped with such an ideal high level modeling language, we still need a simple evaluation scheme that allows to grade students independent from the examiner's personal interpretation of the provided solution.

Currently, we deal with these problems in our courses at the Gaußschule Braunschweig, a secondary school, and at the University of Kassel using an adapted cut-out of the UML modeling language. This language provides a reasonable level of abstraction, however it still deals with too many technical details. In general our language still requires about two month of learning, thus it still needs to be simplified. Our first experiences in measuring the size of sample solutions in order to judge the complexity of an examination task are promising, but this needs further research. Our evaluation scheme for student projects still lacks repeatability and simplicity. However, the measurement results reflected our individual quality impressions of the solutions and the evaluation scheme was systematic enough to be understood by our students and to create the impression of a fair and reliable scheme independent from personal interpretations. Still, this scheme needs to be facilitated.

References

[Ba98] Helmut Balzert: Lehrbuch der Softwaretechnik 1, 2. Aufl., Spektrum Verlag, Heidelberg , 1998

[DGZ02] I. Diethelm, L. Geiger, A. Zündorf: UML im Unterricht: Systematische objektorientierte Problemlösung mit Hilfe von Szenarien am Beispiel der Türme von Hanoi; in Forschungsbeiträge zur "Didaktik der Informatik" - Theorie, Praxis und Evaluation; GI-Lecture Notes, pp. 33-42 (2002)

[DGZ04] I. Diethelm, L. Geiger, A. Zündorf: Systematic Story Driven Modeling, a case study; Workshop on Scenarios and state machines: models, algorithms, and tools, ICSE 2004, Edinburgh, Scottland, 2004.

[Gei04] L. Geiger: Automatische JUnit Testgenerierung aus UML-Szenarien mit Fujaba, Diplomarbeit vorgelegt bei Albert Zündorf, Universität Kassel, 2004

[Fu02] Fujaba Homepage, Universität Paderborn, http://www.fujaba.de/.

[GI00] GI - Fachausschuss „Informatische Bildung in Schulen": Empfehlungen für ein Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen; Gesellschaft für Informatik, 2000

[HBB00] S. Hillen, K. Behrendes, K- Breuer: Systemdynamische Modellierung als Werkzeug zru Visualisierung, Modellierung und Diagnose von Wissenstrukturen; in H. Mandl, F. Fischer (Hrsg.) Wissen sichtbar machen - Wissensmanagement mit Mapping-Techniken, Hogrefe Verlag, Göttingen 2000

[Hu00] P. Hubwieser: Didaktik der Informatik - Grundlagen, Konzepte, Beispiele, Springer Verlag, Berlin, 2000.

[KMK04] Kultusministerkonferenz: Einheitliche Prüfungsanforderungen in der Abiturprüfung Informatik, Beschluss vom 01.12.1989 i.d.F. vom 05.02.2004

[LWS96] G. Lüer, S. Werner, U. Sass: Repräsentation analogen Wissens im Gedächtnis; in D. Dörner, E. van der Meer (Hrsg.) Das Gedächtnis, Hogrefe Verlag, Göttingen, pp. 75-125 (1996).

[SN02] C. Schulte, J. Niere: Thinking in Object Structures: Teaching Modelling in Secondary Schools; in Sixth Workshop on Pedagogies and Tools for Learning Object Oriented Concepts, ECOOP, Malaga, Spanien, 2002.

[Zü01] A. Zündorf: Rigorous Object Oriented Software Development, Habilitation Thesis, University of Paderborn, 2001.

Essential Ingredients of Literacy in Informatics

Ludger Humbert and Hermann Puhlmann

Abstract: In 2003, a discussion about literacy in informatics was initiated in Germany. Its aim was to coin the literacy concept in the sense of OECD-PISA for the domain of informatics or computer science. To illustrate the intended concept, a few sample test items were published along with an explanation of which competencies they ask for. This proved to be a very fruitful approach towards stimulating the discussion in teacher training seminars.

With the experience of these discussions and further test items in mind, this article endeavours to strengthen the underlying theory of literacy in informatics. We claim that education which yields literacy in informatics must enable young persons to explain and understand what we call the phenomena of informatics, i. e. the appearences and consequences of informatics in every day life which need not necessarily be labelled as such at first glance. It may even be that the phenomena are a good starting point for informatics education similarly to what was proposed by Wagenschein for physics education and by Freudenthal for mathematics education. Looking at the phenomena leads to a process of modelling, another central issue in informatics education. Here, a careful distinction between modelling as a process and various modelling techniques in computer science has to be made. While the former is a creative process of thought, the latter may be useful tools for this process and play a role similar to various calculi in mathematical modelling.

Of course, in addressing central ideas like phenomena and modelling, informatics in school will have to deal with the more formal aspects of informatics such as algorithms as well, and a certain degree of instruction in computer and software usage will also be needed. Returning to the initial approach of formulating test items, these subcategories will have a place in their own right, since items addressing them are useful in analyzing various degrees of informatical literacy.

1 Informatics and PISA

In 2003, and inspired by OECD-PISA 2000 [OE01, DBKN+01], a discussion about literacy in informatics was initiated in Germany. OECD-PISA defines the concept of literacy in the domains of reading, mathematics and the natural sciences. A large number of test items answered by thousands of 15-year old pupils in the participating countries served as the basis for a comparison of the achievement of different educational systems.

An interesting aspect of this approach is that PISA dismissed the principle of former international studies, such as TIMSS, to restrict test items to topics from the common denominator of the national curricula of the participants. In other words, test items from PISA may happen to be unfamiliar to parts of the test population as long as the required competencies can sensibly be asked for from 15 year olds.

In [Pu03] it was argued that radically neglecting national or — in the case of Germany — regional curricula will be the only way of achieving an assessment that measures young persons' abilities in informatics. A curriculum-oriented approach has to be ruled out because the curricula are too different from each other. Therefore, the PISA-approach was transferred to the domain of informatics, the concept of literacy in informatics was defined, and sample test items were presented.

These test items proved to be particularly stimulating for discussions in teacher training seminars as well as in university seminars on didactics of informatics. As a result, new test items were designed in these seminars. The discussion focused on the question which aspect of literacy in informatics is covered by an item. It turned out that items which started with a real-life situation were very popular. As they are most likely to test young people's capacity to use knowledge in informatics in order to meet real-life challenges, those items are very much in the spirit of PISA.

The aim of this paper is to strengthen the theoretical foundation of literacy in informatics beyond the ideas from [Pu03]. There, three classes of competencies were defined, referring to different kinds of informatics-related occupation: "application" which focuses on the qualifications needed to use soft- and hardware, "construction" which covers the skills needed to create new informatics systems[1], and "decision" for the qualifications needed to decide and reason about the use of informatics systems and its consequences. In this article, the aspect of understanding occurrences of informatics in the world — so-called phenomena — is considered instead. The goal of measuring literacy in informatics is underlying the discussion, hence sample test questions come along with the theory.

The paper is organized as follows: We start with an example of a phenomenon, formulated as a test item, followed by a general discussion of phenomena and their role in literacy in informatics. Thereafter we turn to the process of modelling which is closely related to understanding phenomena. Finally, it is argued that questions addressing more isolated knowledge and skills should also be part of a test instrument, and the construction of tests for different purposes is briefly considered.

2 Understanding phenomena: the heart of literacy

If the aim of education is to enable young persons to take part in society in an active and responsible way, which is a main aspect of literacy in the definition of PISA, then the touchstone or the heart of literacy with respect to informatics is whether someone understands the occurrences of informatics in everyday life and society. It will be insufficient to merely have some internal knowledge of informatics that cannot be linked to the world.

We call the occurrences of informatics *phenomena of informatics*. This section gives examples, three categories of phenomena and a discussion of their relevance in teaching and assessment.

[1] See the Appendix for a definition of the term "informatics system".

An example

Let us start with a real-life example[2], a situation taken from the checkout of a supermarket. A photograph shows a bottle of lemonade which is being pulled over a sheet of glass, i. e. the scanner of the checkout. Figure 1 shows part of the text that comes with the photograph.

Another question gives an informal algorithm (i. e. in natural language) for the process of checking out and asks for a suitable extension to include a "buy two, get one free"-offer. Finally, there is a question on how the storekeeping and the checkouts can be combined. Here, the required data structure is to be modelled.

Joanna is buying a bottle of lemonade in a large supermarket with many checkouts. The cashier pulls the bottle over a sheet of glass. There is a "beep", and the price of the bottle is displayed.

Question 1:
Where can you find the price of the bottle? Mark *all* correct answers.

◯ The price is included in the barcode on the label of the bottle.

◯ For each barcode, the price is stored in each of the checkouts.

◯ The price for each barcode is stored in a central computer system.

◯ The cashier enters the price using a keyboard.

Question 2:
The supermarket wants to sell the lemonade as a special offer at reduced prices.
The former price of EURO 0.98 is reduced to EURO 0.78.
What has to be done? Mark *all* correct answers.

◯ The bottles need new labels, stating the new price.

◯ A large number of bottles of this lemonade has to be stocked up.

◯ The price must be updated in every single checkout.

◯ The new price must be entered into the central computer system.

◯ The cashiers need to learn the new price by heart.

Figure 1: Test-Items "Supermarket checkout"

The example of the supermarket shows that there is a lot of informatics in everyday life. The seemingly simple question of where the price is quoted may lead to the discussion of the lookup in a database with barcode-price-pairs. As the questions indicate, a connection to networked computing, algorithms and data modelling is also feasible.

[2]This example was developed by Andreas Schwill and the authors.

Three categories of phenomena

The fact that teachers like and invent test items of this kind shows that teachers understand and appreciate the notion of literacy as defined in PISA, i. e. the ability to complete tasks relating to real-life, depending on a broad understanding of key concepts, rather than the possession of specific knowledge [DBKN$^+$01, p19]. Teachers realize that these real-life settings are well known to their pupils. They believe that pupils should be able to cope with these tasks, even if their actual lessons do not yet foster the real-life approach. We will return to the impact this may have on informatics education below.

A closer look at various test items featuring real-life situations suggests that informatics is incorporated in three different flavours. We therefore distinguish the following *phenomena of informatics*:

1. Phenomena that are directly related to informatics systems. They occur when a person consciously uses an informatics system, such as a word processor. A part of the display on the screen, a feature of the software or a certain behaviour of the system may often be happily ignored without loosing the basic functionality of the system. A deeper understanding of the fact (based on knowledge in informatics), i. e. the ability of reasoning about and explaining the phenomenon, however, makes using the system easier, more efficient and maybe even more pleasant.

2. Phenomena that are indirectly linked with informatics systems. They occur in everyday situations whenever informatics systems are involved without being apparent at first glance. Some of these phenomena can even be quoted without reference to an informatics system (remember the example above: "Where can you find the price?"). The connection is then made by analyzing the phenomenon.

3. Phenomena that are not connected to informatics systems but have an inherent informatical structure or suggest informatical reasoning. Examples in the area of sorting and searching, which happen to be major tasks not only in informatics, abound. A person without informatics education may well be able to cope with these phenomena, she may even develop pieces of theory of informatics from a phenomenon, albeit in a less formal way.

Although sample phenomena were briefly indicated in this list, we will now discuss each kind of phenomenon on the basis of a more explicit example.

1. A phenomenon known to young persons is that it cannot be guaranteed that a short message (SMS), sent with a mobile phone, arrives at the mobile it is sent to. Although the loss of an SMS may not happen very frequently, any particular instance of this phenomenon can be extremely annoying, and it is only natural to ask why this can happen.

   This phenomenon is directly related to the informatics system "mobile phone". It matches our intention in various ways: First, it is relevant to pupils of secondary school as they often make extensive use of the short message service. Second, addressing this issue opens an important part of informatics, namely the question of

networking and protocols. Finally, and closely linked with the importance of the networking issue, the informatics addressed is not only important for informatics as a scientific discipline, but it allows to make connections to other occurrences of informatics in the pupils' lifes, such as linking computers locally in "network parties" or the various uses of the internet.

2. As indicated, the supermarket scenario from above is a phenomenon of the second type. Although it is clear that computers are involved in modern checkouts, the shopper does not need to be aware of this. He may have a naïve understanding of the checkout being a large electronic calculator, and there is no immediate need for questioning this. But what if the price quoted at the shelf is different from that on the receipt? Will it be wrong at all checkouts? Is it all right if the complaining customer is reimbursed and no further action is taken?

   All of this leads to an analysis of the background, and, depending on the depth of analysis, one has to deal with database systems, networked computers or algorithmic representation of processes. Again, these are important parts of informatics, the phenomenon is relevant in everyday life, and the issues addressed are not isolated but open for links to other occurrences of informatics. Furthermore, it must be stressed that the combination of background knowledge from informatics and the skill of applying this knowledge to the checkout-example is the kind of qualification that is essential for full participation in society.

3. For the third kind of phenomenon we pick up the idea of sorting and searching, which is important in many situations. Imagine someone playing a card-game such as bridge. Once the cards are dealt out, every player has to sort the cards in her hands. Players have different strategies for this, which amounts to defining different orderings on the set of cards. In the case of (younger) children with small hands, one can also observe different "algorithms" for sorting the cards, e g. by using stacks for every colour, sorting within each stack and then merging the four stacks by concatenation.

   Analyzing situations like this appears to be valuable in the learning process because pupils can (re-)invent parts of informatics. They may achieve results which are in fact "state of the art" in informatics, namely in those cases where the common sense solution is what is being used. Or they may find a good starting point for further discussion.

   An example for this, and another phenomenon of the third kind, is the travelling salesman problem, the relevance of which is easily understood by pupils. Often a first "common sense" solution is the "next neighbour"-strategy. This is easily revealed to lead to different results depending on the starting point. Thus, a discussion on other strategies, correctness issues (does the strategy lead to an optimum?) and complexity may follow.

The importance of phenomena in the education process is not a new idea. Wagenschein [Wa76] pointed out that studying phenomena drives the process of learning in physics, and

Freudenthal [Fr83] claimed that the learning and teaching of mathematics has to start with the phenomenology of mathematical structures.

In a similar way we propose to take phenomena as starting and focal points in informatics education in school. As the examples demonstrate, phenomena allow a wide range of issues in informatics to be addressed, and they foster mental links between different areas. Of course the phenomena have to be carefully chosen in order to establish concepts of informatics, and after an initial analysis one has to free oneself from the particular setting of the phenomenon in order to address the informatical contents in a broader way. This may lead to doing informatics seemingly for the sake of itself for a while. It would however be splendid to see (possibly another) phenomenon thereafter that can be explained with the newly acquired knowledge.

There are, admittedly, also critics of the phenomena-driven approach (e g. [Mu01]). One of their major points is that there are deep results e g. in physics which are not accessible by phenomena but by strict adherence to theoretical, maybe abstract reasoning. This may also be the case in informatics. Were this only restricted to informatics for the specialist, we might ignore it. But e g. results from theoretical informatics about complexity and computability are enlightening for everyone since they may correct wrong beliefs about the possibilities of computing. In so far, we admit the need of parts of a curriculum that can hardly be accessed by phenomena. However, even in these cases one might find phenomena that work as openers for the field. (Consider the next-neighbour strategy for the travelling salesman problem which by no means addresses the P vs. NP issue.)

We argued that phenomena are desirable as an element of informatics instruction. Independently, and in the context of this article, more importantly, we claim that—as indicated above— understanding phenomena is a core element of literacy. What is really wanted as an outcome of the educational system (and what PISA defines as literacy) is that young people are able to use their knowledge and skills in everyday situations and that they can make wellfounded judgements in questions where information technology is involved, be it for their personal use or in the context of society. For this, formal knowledge or the ability to reason within the scientific framework of informatics will not suffice. It is essential to make the connection to the occurrences of informatics, i. e. phenomena, if these goals of education are to be achieved. Therefore, any instrument that tests competencies of pupils in informatics should have a large portion of phenomenon-based test items.

3 Modelling skills

Modelling as a process

A person with a high degree of literacy in informatics will understand phenomena on the basis of a comprehensive knowledge in informatics and high skills in connecting this knowledge and the real-life occurrences of informatics. In many cases this will not only involve finding the connection but there is some work to do on both sides, the side of informatics and the side of real life. Together, theses pieces form a *modelling process*.

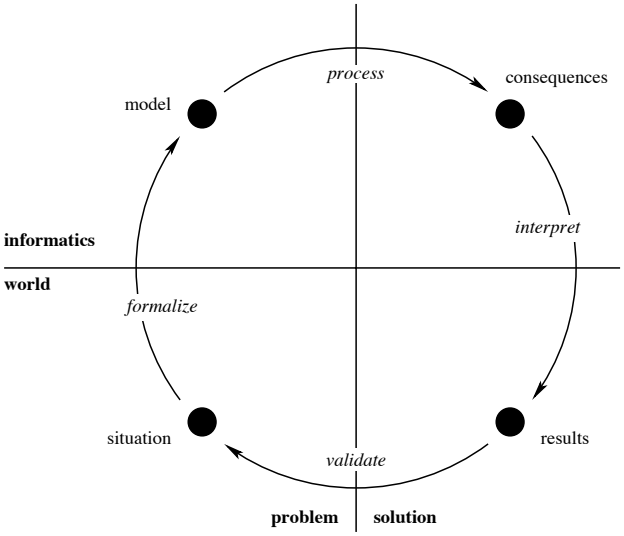


Figure 2: The process of modelling in informatics

The process of modelling is well described in [DBKN+01, p143] for mathematical modelling, and it is very much the same for modelling in informatics (cf. Figure 2, adapted to informatics from [DBKN+01, p144]): Starting on the side of the "world", one has to formalize the situation, i. e. translate a real-life situation into the language and thinking of informatics which yields an informatical model. Within the side of informatics, the model is processed further, be it by implementing it on a computer and running a program or by some sort of reasoning about the model. In either case, a result in the language of computing is achieved. This must be interpreted within the original setting from the "world", leading to a solution for or answer to the real-life situation. Finally, this solution must be validated with respect to the initial problem, and possibly one has to do these four steps once again in order to improve the solution.

Ideally, the cycle of modelling includes the choice of the instrument or technique for the step of formalization. After all, phenomena have no label attached saying e g. "use an entitiy-relationship-diagram to understand me". In order to choose an appropriate modelling technique, a person must have a large repertoire to choose from. This is where the necessity of learning and teaching different techniques of formalization arises.

As a matter of fact, many of them happen to have the word "model" in their name, and the word itself is widely used in informatics (cf. [Th02] for a survey). Often it is connected with special techniques of mapping "the world" to a formal system. Both, the formal system and the specific representation of "the world" within the formal system, are called a model (e. g. the entity-relationship-model as a system using boxes and diamonds with their respective semantics and a concrete ER-diagram for, say, the situation of a public library). Unless one wants to create a new modelling technique, the knowledge of techniques like this is needed to model aspects of the world, but bear in mind that the word "model" does

not need to occur in the technique's name. Sometimes simple things, such as the rule of three in mathematics will do the job without having an impressive name. So, within an informatics lesson, the achievement will not depend on the formalization technique being called a "modelling technique". What is important is that the modelling techniques are used to highlight the modelling process. So a foundation for general modelling skills can be laid, and a piece is added to the repertoire of techniques the learner will later depend on.

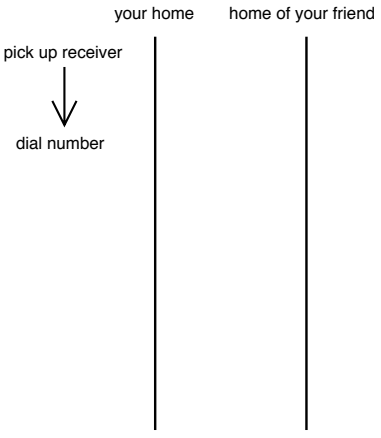
Addressing specific modelling techniques — an example

Even with a boost in informatics education, it will be unrealistic that everyone knows about every single technique of formalization and can do the rest of the modelling process in every case. But with the seperation of activities from Figure 2 we can pick out parts of the modelling process. We can construct test items that ask if someone can do the processing of a model that is given. Or we can ask for a suitable formalization of a situation in the world without going on to the processing etc.

Another way of restricting the complexity of a question is by explicitly giving the intended formalization technique (though it excludes those who do not know this particular method but could have chosen a different one).

Phone call to a friend

You want to make a phone call to your friend. There are several actions needed to do so, such as picking up the receiver, dialling the phone number, stating your name etc. The actions take place at either of the two locations "your home" and "home of your friend".



Question:

Think about the actions needed from picking up the receiver to finishing the call. Complete the sequence diagram by filling in the actions and connecting them through arrows.

Figure 3: A modelling task

In the example it is assumed that persons being tested know about sequence diagrams. Then the task is to find appropriate actions at the two locations and the sequence in which the actions take place. There is not "the" right solution because the granularity of actions may differ. A coding scheme for the evaluation should reflect this and group possible answers accordingly. Of course, one can think of variations of this test item, making the item harder or easier. It will probably be easier if the actions are explicitly given and only the correct sequence must be constructed. A more demanding version might ask for a graphical representation of the actions needed to complete a phone call without referring to sequence diagrams or giving the initial diagram.

We do not declare ourselves in favour of one of these variants. The choice will depend on the role of the item within a test. In order to test the knowledge about the particular modelling technique of sequence diagrams, one may give more details. A variant with less details given focuses on the ability of modelling a situation regardless of the technique used.

4 Knowledge and skills in formal techniques

Full understanding of a phenomenon will often be paired with a good command of modelling as a process. A restriction by explicitly stating the intended modelling technique shifts the focus from the problem to testing whether this particular technique is known and mastered. In order to get a diagnostic instrument for the outcome of an educational system with respect to informatics, one might consider further restrictions: one can ask mere knowledge questions and test the skills in formal techniques that are needed as a small sub-part of some modelling process.

Examples of knowledge questions

Knowledge questions may ask if someone is familiar with the terminology of informatics or if he is able to choose an appropriate application to complete a task:

Knowledge: What is a login and why/and for whom is it useful?

1. Which data is transfered when you log in a computer system?
2. Give an example of a "good" and a "bad" password. Explain your choice.

Choice of application: You want to share the solution to your maths homework with a friend who lives too far to simply drop in on. How can a computer help you?

Testing skills in formal techniques

Skills in formal techniques are needed at various steps of the modelling process. They also build the foundations for continuing education, in particular for those who aim for a profession in the field of informatics. Examples for questions addressing these skills are:

Programming skills: Encode the algorithm shown in the Nassi-Shneiderman-diagram in a programming language of your choice. (The question would have to show a diagram.)

Computer usage skills: You typed a text with 4375 characters. How can you find out, how large it will be when saved to a disc?

5 Consequences for test-construction

Literacy is not something a person has or has not. Instead, there are various degrees of literacy, and one aim of a test instrument may be to determine the degree of literacy in informatics a person or a whole test population has. It may even be that literacy in informatics is a multi-dimensional concept. E g. there might be the two dimensions "ability to link formal knowledge to real-life situations" and "comprehensive knowledge within informatics". While the former seems to be the kind of literacy needed to navigate through life, the latter is certainly needed for those who want to continue their informatics education and possibly earn their living in the field of informatics. However, it may also turn out that the two aspects are closely linked in that whoever excels in one aspect does so in the other. The decision on the dimensionality needs empirical data and cannot be made at this point.

A second aim of a test instrument may be to get information about the educational system. In the case of informatics, it would be highly interesting to compare the curricula (if informatics is part of the curriculum) with the abilities of the test population. Will there be isolated pieces of knowledge or is the knowledge interrelated? Do pupils have competencies as intended by the curricula or do their competencies exceed the curricula in some cases? And of course: what is the *de facto*-status of informatics education within a system, does it build a good foundation for further development?

Depending on the primary intention of a test, the aspects of phenomena, modelling, and knowledge and skills in formal techniques will play a different role. This should be reflected in the test. As a rule of thumb, the share of "knowledge and skills"-questions may be larger in tests for a single class and a short period of time, where the test aims at examining if the class has learned a specific part of the curriculum. If the test covers a longer period of time, any informatics curriculum should be expected to contain some modelling and, hopefully, phenomena. So the focus of the test should shift to these aspects in this case as well as in the case of a larger test population where a common curriculum cannot be guaranteed.

For large scale tests such as OECD-PISA, further aspects have to be considered: in order to measure different degrees and possibly different dimensions of literacy in informatics, the test items must cover a wide range of difficulties. They must be well distributed between different classes of competencies (such as "application", "construction", "decision", cf. [Pu03]) and different aspects of literacy.

In the light of the discussions we had in teacher training seminars, one of the most no-

ble tasks will be the construction of phenomena-oriented items which require low reading competency and address some central aspect from informatics. The authors welcome any contribution to this and will be happy to discuss further ideas towards setting up an evaluation scheme for the achievement of informatics in school.

Appendix

Informatics and informatics systems

In this article, the term "informatics" is used in the sense of "Informatik" in German or "Informatique" in French, although it is not yet widely used in the English language, where the words "computing" or "computer science" are more common. Furthermore, we use the term "informatics system" as defined in [CS01, p301] (translation by the authors):

An *informatics system* is the specific combination of hardware, software and networking facilities needed to solve some application problem. The term includes those non-technical issues and their solutions that arise from embedding the system into the application area, in particular questions of system design, user training, security and consequences of using the system.

Informatics in this setting is the scientific discipline addressing the construction and design of informatics systems.

References

[CS01] Claus, V. und Schwill, A.: *Duden „Informatik": ein Fachlexikon für Studium und Praxis*. Bibliographisches Institut. Mannheim, Leipzig, Wien, Zürich. 3. Aufl. 2001.

[DBKN+01] Deutsches PISA-Konsortium Baumert, J., Klieme, E., Neubrand, M., Prenzel, M., Schiefele, U., Schneider, W., Stanat, P., Tillmann, K.-J., und Weiß, M. (Hrsg.): *PISA 2000: Basiskompetenzen von Schülerinnen und Schülern im internationalen Vergleich*. Leske + Budrich. Opladen. 2001.

[Fr83] Freudenthal, H.: *Didactical phenomenology of mathematical structures*. volume 1 of *Mathematics Education Library*. D. Reidel Publishing Company. Dordrecht. August 1983.

[Hu03] Humbert, L.: *Zur wissenschaftlichen Fundierung der Schulinformatik*. pad-Verlag. Witten. März 2003. zugl. Dissertation an der Universität Siegen `http://www.ham.nw.schule.de/pub/bscw.cgi/d38820/` – last visited: 30^{th} May 2004.

[Mu01] Muckenfuß, H. Retten uns die Phänomene? Anmerkungen zum Verhältnis von Wahrnehmung und Theorie. August 2001. `http://www.ph-weingarten.de/homepage/faecher/physik/muckenfuss/gebiete/didaktik/vortrag/Rettung.pdf` – last visited: 15^{th} May 2004.

[OE01] OECD (Hrsg.): *Lernen für das Leben. Erste Ergebnisse der internationalen Schulleistungsstudie PISA 2000*. OECD. Paris. 2001. OECD – Organisation for Eco-

nomic Co-operation and Development http://www.pisa.oecd.org/Docs/Download/PISA2001(deutsch).pdf – last visited: 26^{th} May 2004.

[Pu03] Puhlmann, H.: Informatische Literalität nach dem PISA-Muster. In: Hubwieser, P. (Hrsg.), *Informatik und Schule – Informatische Fachkonzepte im Unterricht INFOS 2003 – 10. GI-Fachtagung 17.–19. September 2003, München*. Number P 32 in GI-Edition – Lecture Notes in Informatics – Proceedings. S. 145–154. Bonn. September 2003. Gesellschaft für Informatik, Köllen Druck + Verlag GmbH.

[Th02] Thomas, M.: *Informatische Modellbildung – Modellieren von Modellen als ein zentrales Element der Informatik für den allgemeinbildenden Schulunterricht*. Dissertation. Universität Potsdam Didaktik der Informatik. Juli 2002.

[Wa76] Wagenschein, M.: Rettet die Phänomene! (Der Vorrang des Unmittelbaren). *Scheidewege*. 6(1):76–93. 1976.

Learning Process' Evaluation in Vocational Schools for the IT Sector's Training Occupations

Dietmar Johlen

Oskar-von-Miller-Schule Kassel
Weserstr. 7
34125 Kassel
Germany
d.johlen@ovm-kassel.de

Abstract: The PISA study's results led to a broad discussion about the improvement of the German educational system. Reforms to increase the quality of education have to be undertaken. The system of vocational education and training (VET) is affected by this situation.

About 70 percent of an age group passes through vocational school. The competences the PISA study has analyzed are essential for successfully completing a training occupation. The vocational school has to make an effort to compensate deficits of general schooling. This is especially important for participants of purely school-based forms of training who were not able to receive a training place yet.

Currently, the future of the dual system of VET is called in question. Accepting a training place appears less attractive for school-leavers. In consequence training companies encounter increasing difficulties finding suitable candidates for their training places. At the same time the overall number of training places is decreasing which is partly due to the ailing German economy. In order to improve the dual system's attractiveness new training occupations, e.g. in the fast growing IT sector, were successfully established. Several reforms, e.g. the concept of learning areas (Lernfeld-Konzept), strengthen activity-orientation and aim to prepare the dual system for the future. The concept of learning areas transfers a substantial amount of the curricular responsibility to the vocational school. In order to develop learning situations the teachers at vocational schools have to decide which competences they want to strengthen and how they want to evaluate the learning process.

This paper presents the concept of learning areas for the IT sector's training occupations. The scenario-approach is introduced, which represents a methodical-didactic reference system for the development and execution of instruction. From this starting point the evaluation of the learning process is discussed.

Research findings:

- The scenario-approach puts the concept of learning fields in precise terms.
- The scenario-server representing the development of the scenario-approach serves as an environment to accommodate the learning process' evaluation.

1 Introduction

The framework curricula (Rahmenlehrpläne) of the IT sector's training occupations are based on the concept of learning areas (Lernfeld-Konzept) since 1997. However the implementation of the concept of learning areas cannot be regarded as completed. The organization of classes with respect to learning areas still represents an important challenge for various areas in vocational schools. This includes replacing subject-systematic instruction by action-systematic instruction with respect to the procedures in the appropriate action fields of practice. Thus the co-operation between vocational schools and training companies is becoming more important. This co-operation facilitates the arrangement of authentic learning situations providing experience in a particular action field. The concept of learning areas requires a far closer co-operation among the teachers in the team, in order to organize and co-ordinate learning-area-spreading instruction. Since the framework curricula (see e.g. [RLP97]) do not describe the learning areas in great detail it is left to the teacher team to fill the learning areas with contents.

In this paper the scenario-approach is introduced, which serves as a methodical-didactic reference system for the development and execution of instruction in the IT sector's training occupations. The scenario-approach integrates instruction contents in the learning areas into a continuous scenario covering the whole duration of the traineeship. The advantages of the scenario-approach are listed below:

- Suggestion to arrange the framework curricula into concrete learning situations.
- Reference to a minimum complexity for the instruction examples.
- Common platform for agreements among colleagues in the team.
- Platform for the co-operation between vocational school and training company.
- Conceptual frame for the organization of further training for teachers.
- By giving the scenario a real life image the conclusion is given for contents which are organized in an action-systematic way.
- Increasing steadiness of the learning process in block instruction.
- Providing connections for learning situations in various fields of the training.

The following chapter describes briefly the concept of learning areas and analyzes the requirements in order to arrange learning situations for a particular learning area including the education assignment (Bildungsauftrag) and the qualification assignment (Qualifizierungsauftrag) of the vocational school. Chapter 3 presents the scenario-approach with the example of the virtual enterprise bitwerk AG . The learning process' evaluation is discussed in chapter 4. A conclusion is given in chapter 5.

2 Concept of learning areas

The concept of learning areas is to be regarded as one possible option in order to overcome the transfer problem. The transfer problem concerns the difficulties which arise

with knowledge acquired in a subject-systematic way and applied on practice-relevant situations.

Action competence is to be promoted in the VET and action-oriented instruction to be worked in the curriculum by putting the concept of learning areas into action. The learning area concept proceeds from action fields of practice, which are of vocational, social and personal relevance. These action fields are transformed by a didactical adaptation and reflection into learning areas for the vocational school. Within this process the education assignment (Bildungsauftrag) of the vocational school is to be considered. It requires the promotion of the personality development towards social responsibility in connection with the acquisition of qualification for a vocational activity, which is required on the job market (qualification assignment of the vocational school). The connection of these two VET goals creates a basis for the development of vocational identity as component of social integration [Ba00]. The education assignment works as a bridge between action field and learning field. On this basis it is the task of the schools to arrange given learning areas in the form of precise learning situations [BS98]. So the transfer of learned contents in practical applications is to be facilitated, in other words "from knowledge to ability" ("Vom Wissen zum Können") [Sl00].

In May 1996 the Conference of Education Ministers (Kultusministerkonferenz) decided on a common basis on framework curricula with respect to learning areas [KMK96]. This includes the framework curriculum for the information technology specialist (Fachinformatiker) [RLP97] and the information technology and telecommunications system electronics technician (IT Systemelektroniker).

The transfer of parts of the curricular responsibility to the schools during the arrangement of the learning areas has profound consequences. Rejecting the subject-systematic organization of instruction and dissolving of subjects leaves a gap, which has to be filled by an action-systematic structuring. New scientific contents must be transferred into the action-oriented context. Teachers must make themselves familiar with the action fields of practice. For this a close co-operation between the school and the training company creates the basis. Instruction in the learning areas takes place in teacher teams. Working out suitable instruction scenarios, which promotes instruction in learning areas, contributes to the team formation. [BS98] and [Sl00] have suggested a set of questions to facilitate the evaluation of learning situations in respect of the concept of learning areas.

The framework curricula for the IT training occupations do not contain methodical preferences. Nevertheless it is stressed that action-oriented forms of instruction should be considered appropriately [RLP97].

3 The scenario-approach

Instruction contents in the learning areas are oriented towards the whole duration of the traineeship at a continuous scenario by the scenario-approach. The scenario takes place within the virtual enterprise bitwerk AG. The bitwerk AG represents an enterprise,

which consists of a group of affiliated companies (see Fig. 1). The structure is developed in such a way that the learning situations of the different learning areas necessary for instruction can be illustrated on it. Therefore the enterprise covers typical legal forms in the affiliated companies, which are of importance for the business portions of the training. The succession and alignment of the learning situations during the training occupation result from the development of the bitwerk AG in the scenario. The teacher team takes up definite events or processes on the basis of the common scenario of the bitwerk AG and deals with them from the perspective of the respective learning area. At the beginning of the training occupation the scenario focuses on adapting a single desk computer of the bitwerk AG. The requirements necessary for this work place are to be considered with the adaptation. The assumption of an acquisition of an enterprise by the bitwerk AG is regarded as an example for the third year of the training occupation. Here e.g. in the learning area 10 (Betreuung von IT-Systemen) a strategy is compiled for the adjustment of the different IT infrastructures.

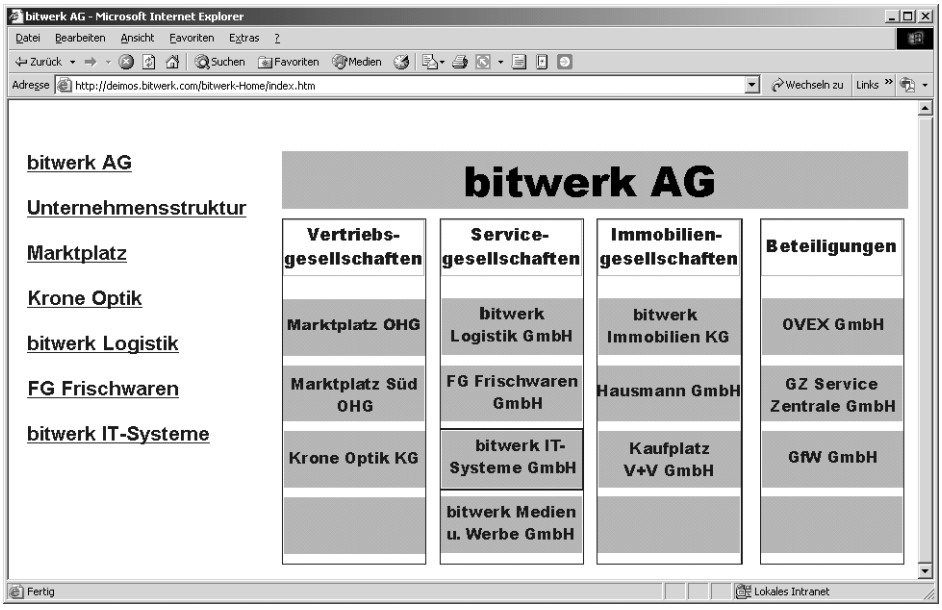


Figure 1: Homepage of the virtual enterprise bitwerk AG. The learning situations, which are subject to a particular instruction, are situated within the bitwerk AG.

The concept of the scenario-approach is illustrated briefly in the following at a precise learning situation for the training occupation information technology specialist.

Situation: To respond faster to problems with the IT infrastructure within the bitwerk AG the “IT Systeme GmbH” – a service company of the bitwerk AG – plans a ticket hotline to receive problem messages.

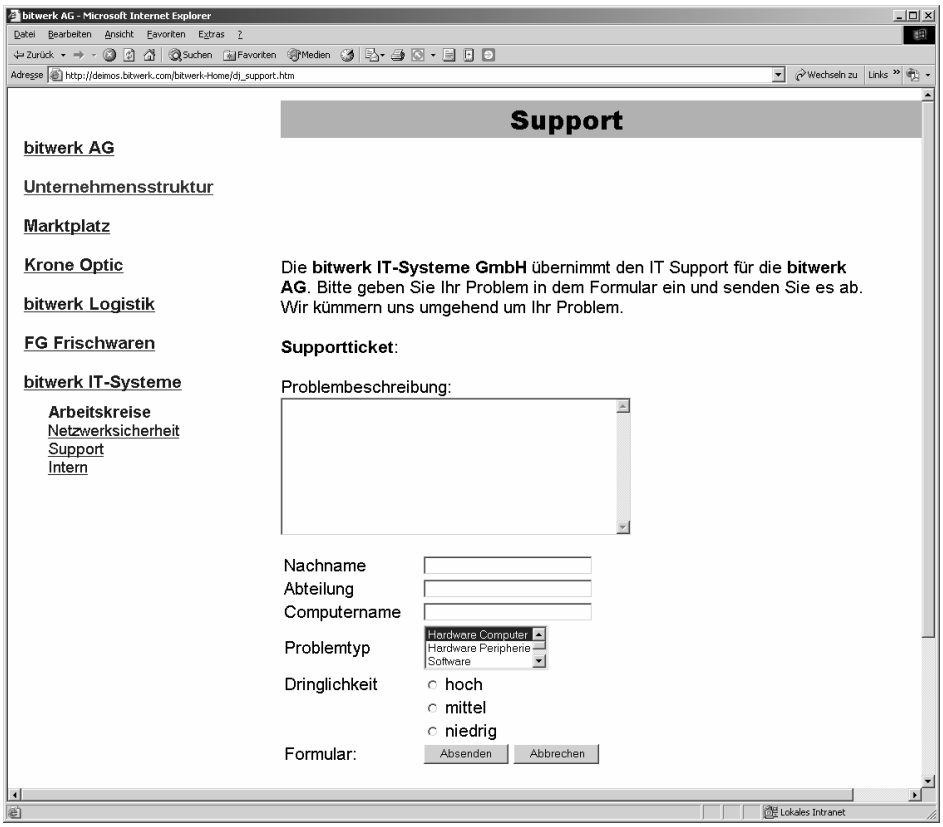


Figure 2: bitwerk IT Systeme GmbH's support ticket input form.

This assignment, which is worked on at the end of the first year of training occupation, is approached from the point of view of different learning areas. In the learning area 6 (Entwickeln und Bereitstellen von Anwendungssystemen) designing a graphical input mask is emphasized in connection with the preparation of programs to the processing of the received support tickets in a database (see Fig. 2). An action-oriented approach to learning area 6 is discussed in detail in [Jo03] and [Jo04]. In learning area 4 (Einfache IT-Systeme) the support tickets are analyzed and categorized. For the problem categories the students prepare standardized solutions to solve the problem. In the business-based learning areas e.g. the accounting of services between parts of the bitwerk AG's enterprises are brought up for discussion. The students take into account the amount of time necessary for the particular support work. The learning area English focuses on a translation of the support ticket input form for English-speaking coworkers of the bitwerk AG.

The scenario arranges a particular learning situation within a superordinate context. In this way the scenario forms a reference system for the students, in which they move. Into this reference system they also insert their solutions. For the presented example the design of the webpages has to follow the style guides of the bitwerk AG. As a further condition the selected script language for the processing of the input forms must be sup-

ported by the used webserver that has already been set up within the bitwerk AG. Therefore the scenario prevents unwanted arbitrariness when working on the assignments without imposing too tight constraints on the students. This leads to more lively discussions during presentations.

The scenario-approach goes beyond a mere description of the scenario. The IT infrastructure of the bitwerk AG, which contains e.g. domains, databases and webservers, exists fully functional on scenario-servers[1] (see Fig. 3). The scenario-servers represent thereby the informatization[2] of the scenario-approach. A scenario server permits to make available a part of the IT infrastructure in an exactly defined condition as it is needed by the current state of the scenario. The transition between states of the scenario is possible within a few minutes. Therefore the scenario-server permits in particular the change of a scenario from one lesson to another, which proves to be quite attractive in everyday school life. In practice it is useful to divide the virtual IT infrastructure into two sections: One section for the constantly available basis services (e.g. sign on server, webserver) and one section, in which the currently regarded portion of the scenario develops. Thereby the basis services of the scenario are decoupled by possible influences of the momentarily worked on scenario portions. These services are available at any time.

The agreement in the teacher team on a common scenario forms the basis for further arrangements to precise learning situations. The common scenario supports the mutual understanding for contents of the respective learning areas in the teacher team by a uniform level of information. Thereby overlappings between the learning areas becomes easily recognizable. The scenario supports the arrangement of the learning areas, which are only roughly outlined in the framework curriculum. Contents presented in class are situated within a superordinate scenario and are therefore much more integrated on this basis. In block instruction the scenario-approach leads to an increasing steadiness of the teaching and learning processes, since repetition portions can be reduced. On the basis of the scenario requirements of the learning areas' minimum complexity can be noted down, which gives an orientation to teachers during the arrangement of a learning area.

The informatization of the scenario-approach in the scenario-server guarantees a simple distribution of the scenario. Teachers and students are able to run the scenario, which is familiar to them from school, on different hardware and operating system platforms at home or at the training company. The scenario-server is attractive in particular for preparation and reinforcement of instruction units. Tedious and time consuming installations of IT infrastructure that can only be operated on one single location at school can be made available in the scenario-server in a reproducible and distributable way. Thereby opening the possibility for students to work on a learning situation of the scenario beyond the scope of the instructions at school. The solutions to the assignments of particular learning situations are integrated in the scenario-server and transferred back to school. The scenario server allows students to actively participate in the arrangement of

[1] Software products such as VMWare and VirtualPC allow operation of multiple operating systems virtually on a single computer. Per virtual operating system, which is to be operated on this computer, are approx. 100 MB to 200 MB of main memory and 1 GB up to 2 GB hard disk capacity additionally needed.
[2] Informatization - utilizing information technology.

the instructions. In this triad of planning-judgement-decision making the students in particular take over responsibility [Ka02].

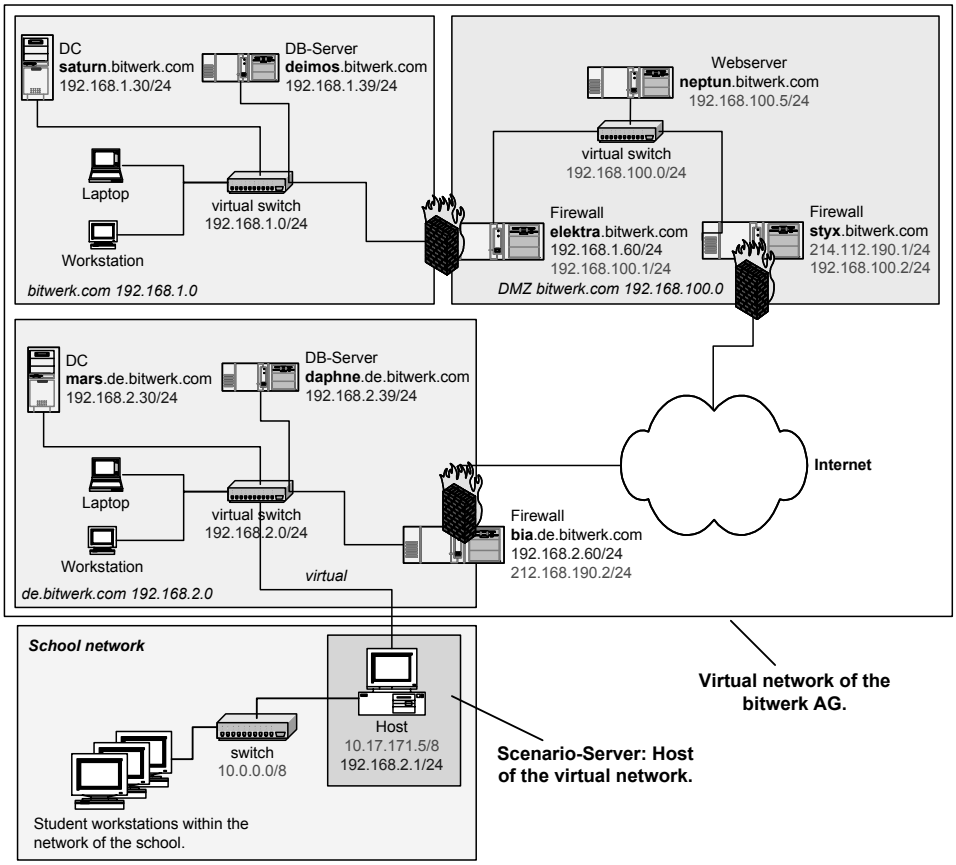


Figure 3: Scenario-server hosting a typical, virtual IT infrastructure of the bitwerk AG as part of the school's network (DC - domain controller, DB - database).

The scenario-server forms a common platform for the teacher team. The instruction projects are developed on this basis. The scenario-server supports the specialization of the teachers within one teacher team. Installations by individuals within the virtual bi-werk AG infrastructure can be used by all team members due to the distributable nature of the scenario server. Thus the scenario-approach is suitable in particular for the teachers' further training. A common scenario, which couples the offered course modules contentwise and points out overlappings to other course modules, supports the team formation at the schools. The use of scenario-servers in the further training modules opens the possibility that the installations and configurations made at the further training locations are available after the further training modules. Thus the difficulties can be reduced, which arise after a further training course, if the subject has to be converted on the hardware and operating system platform of a particular school. The szenario-

approach is currently successfully used by the IT Akademie Hessen in some modules for application development. The IT Akademie Hessen is responsible for the further training of teachers of the vocational schools in Hessen.

4 Learning process' evaluation

The concept of learning areas requires a new type of examination in the vocational schools as well. Assignments have to be embedded in a context which corresponds to the particular learning area. The scenario-approach is suitable to serve as a context. The scenario-server is in particular attractive for the learning process' evaluation. It provides an easily distributable infrastructure that is largely capable of modelling most of the IT infrastructures of interest. The desired state of this infrastructure is well defined and reproducible. During an assignment the students work on a particular section of the scenario. Problems of this type will be called scenario-oriented-problems. Scenario-oriented-problems are categorized by the coordinate system in Fig. 4.

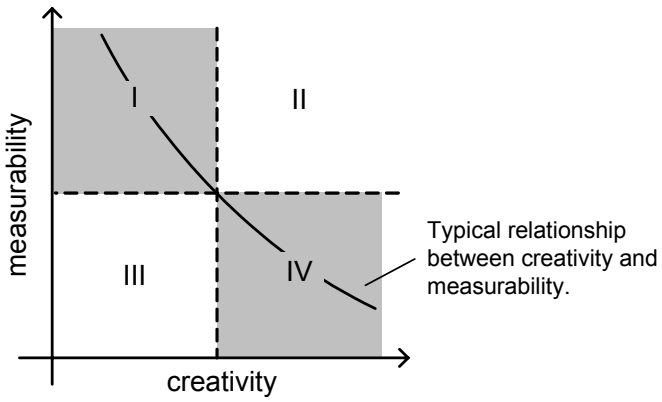


Figure 4: Coordinate system for categorizing scenario-oriented-problems. They are divided into type I-IV.

The axes of the coordinate system characterise the degree of creativity a particular problem leaves to the student and the ease of measurability with respect to uniqueness of the solution. Measurability is understood here in particular in the sense of automating the process of evaluating solutions. This coordinate system yields 4 types of scenario-oriented-problems. Only type I and type IV are of practical importance. The following section briefly illustrates one type I and one type IV problem.

Scenario-oriented-problem type I
A group of users has to be created in the domain de.bitwerk.com (see Fig. 3). The user details are listed in the table *user* which is part of the database *guest* located on the database server *daphne.de.bitwerk.com*. The domain user account details and the profile settings have to be set up according to the entries in table *user*.

Scenario-oriented-problem type IV
The "IT Systeme GmbH" wants to extend the ticket hotline. To avoid abuse each user has to enter a valid username and password. In order to facilitate processing a support assistant receives exclusively messages that have been issued at the site where this support assistant is located. The domain administrator has suggested a solution that stores all user data only once within the domain.

The example of the type I problem yields a single valid solution. It helps the student to find out his level of understanding for a particular subject e.g. creating user accounts. Therefore type I problems are processed by an individual student. The solution is evaluated automatically e.g. by the scenario-server. An automated web-based test system[3] has been implemented within the scenario-server that supplies type I problems and checks the answers automatically.

A type IV problem yields a variety of different possible solutions. In general a solution to a type IV problem cannot be called right or wrong. It rather meets a certain requirement more or less. This type of problem is suited to promote action competence. It encourages judging and decision making. In contrast to type I problems type IV problems are assigned to groups of students. Assigning the same problem to several groups increases the competition between the groups and facilitates the evaluation among the groups. In general it is not possible to automatically evaluate the solutions.

5 Conclusion

The first experiences implementing the scenario-approach show that the team's formation process has been supported. The scenario-approach put the concept of learning fields in precise terms. The common reference system of the scenario-approach simplified the mutual arrangement to precise learning situations. The number and duration of common learning situations have to be increased slowly in the beginning, thus avoiding an excessive demand of the teacher team. In order to monitor the learning progress type I problems have been used on a regular basis during a type IV problem. The students pointed out that this procedure helped them to find out problems immediately by the direct feedback of the automated evaluation.

The employment of the scenario-server reduced lengthy installations and configurations, thus reducing the work load of the participating colleagues. The simple way of passing on the scenario to interested colleagues increased the acceptance of the scenario-approach. The students made use of the scenario-server and arranged parts of the structure to meet their needs. They did not feel restricted by the constraints imposed by the scenario.

[3] MIDAS – Manageable Integrated Database Assessment System.

A concluding assessment of the scenario-approach cannot be given at this point. Until now the scenario-approach is used in the first year of training.

References

[Ba00] Bader, R.: Stand der wissenschaftlichen Forschung zum Lernfeld-Konzept. (http://www.uni-magdeburg.de/ibbp/bp/downloads/Lernfeld-Konzept.pdf), 2000.

[BS98] Bader, R.; Schäfer, B.: Lernfelder gestalten – Vom komplexen Handlungsfeld zur didaktisch strukturierten Lernsituation. In: Die berufsbildende Schule (BbSch), 50 (1998), 7-8.

[Jo03] Johlen, D.: Handlungsorientierter Zugang zur objektorientierten Programmiertechnik mit der Metasprache UML für die IT Berufe. In: Die berufsbildende Schule (BbSch), 55 (2003), 9.

[Jo04] Johlen, D.: Arbeitsbuch Anwendungsentwicklung. Stuttgart: Holland + Josenhans, to be published.

[Ka02] Kath, F.M.: Paradigmenwechsel auch in der Fachdidaktik – Wunsch oder Realität? In: Die berufsbildende Schule (BbSch), 54 (2002), 4.

[KMK96] KMK – Sekretariat der Ständigen Konferenz der Kultusminister der Länder in der Bundesrepublik Deutschland: Handreichungen für die Erarbeitung von Rahmenlehrplänen der Kultusministerkonferenz für den berufsbezogenen Unterricht in der Berufsschule und ihre Abstimmung mit Ausbildungsordnungen des Bundes für anerkannte Ausbildungsberufe. Bonn, 1996.

[RLP97] KMK – Sekretariat der Ständigen Konferenz der Kultusminister der Länder in der Bundesrepublik Deutschland: Rahmenlehrplan für den Ausbildungsberuf Fachinformatiker/Fachinformatikerin. (Beschluss der Kultusministerkonferenz vom 25. April 1997), 1997.

[Sl00] Sloane, P.F.E.: Lernfelder und Unterrichtsgestaltung. In: Die berufsbildende Schule (BbSch), 52 (2000), 3.

Informatics and Standards at an Early Stage

Peter Micheuz

Institut für Informatik Systeme
Universität Klagenfurt
Universitätsstraße 65-67
9020 Klagenfurt
Austria
peterm@isys.uni-klu.ac.at

Abstract: Since the beginning of the school year 2002/2003 almost all comprehensive secondary schools in Carinthia/Austria have offered the subject Informatics in the first two grades to an extent of an hour per week. The preliminary results of this project will be discussed in this paper, including the outcome of an online survey which dealt with various organisational structures and the teachers' attitudes towards Informatics in the first two school years. The results of another online survey will be shown in an overview regarding the basic conditions at schools and the informatic pre-knowledge of all the pupils involved. The main objective of this project is to define a minimal standard by means of a democratic process of all schools involved.

1 General informations about the project

In most federal states of Austria there is no subject Informatics in comprehensive secondary schools in the first two years. Informatics, if at all, is taught in an integrative manner in other subjects. The Austrian ministry of education leaves it to the schools to install Informatics at the expense of other subjects. Actually this does not happen very often. Due to a nationwide reduction of two lessons a week one year ago, many initiatives to introduce Informatics as a new subject have been cancelled as well. As a consequence it is even harder to introduce Informatics now.

Since the beginning of the school year 2002/2003 almost all comprehensive secondary schools in Carinthia/Austria have offered the subject Informatics in the first two grades to an extent of an hour per week. About 15 schools, 85 teachers and 2000 pupils between 10 and 12 years are involved in the project which is financed by an initiative launched by the Carinthian local school administration. Additional hours had to be remunerated to allow the splitting of the first and the second classes into smaller groups. These groups are taught in special Informatics classes to an extent of an hour per week.

After one year another project organized by the pedagogical institute (PI) in Carinthia was started to evaluate Informatics at the first two grades of secondary school. The main objectives were to reinforce networking initiatives between the schools and teachers involved and to install a standardized curriculum. For this purpose meetings of the school coordinators were arranged and a web based document management system was installed to support the contact between the school coordinators and the exchange of experience.

A specific result of this network and evaluation was the definition of a standard that most of the pupils should achieve after two years of Informatics. The curriculum which was developed consists of operationalized teaching objectives and should cover about 2/3 of the teaching time available. Moreover a pool of exercises and relevant informatic problems appropriate for this age-group were collected from the schools and should support both teachers and pupils in achieving the standard.

This paper shows some results of this project including the outcome of empirical research which was carried out at the beginning of the project. One survey dealt with the various organisational structures of the Informatics classes. Furthermore general information about the subject matters was collected and the teachers' attitudes were investigated. Results of another online survey will be shown in an overview of the infrastructure at schools and the pupils' previous knowledge of Informatics.

Furthermore this study should point out that especially at the transition of primary to secondary schools an attempt for standardization is beneficial to avoid a further digital divide among the pupils at an early age.

2 General statistics and empirical data collected by the project coordinators at the schools

This chapter provides important information on statistical facts regarding the preconditions at the participating schools. Thirteen project coordinators held meetings at their schools and collected data which was finally put into an online database.

13 of 15 possible comprehensive secondary schools, called "Gymnasium", in Carinthia are taking part in this project. In eleven schools the subject Informatics is compulsory, in two it is offered on a voluntary basis as an addition to all other subjects. If Informatics is compulsory for an hour a week (or for two hours a week in a secondary school) – it has the same status in the canon as all the other subjects taught in the first and second forms of the Gymnasium. This inclusion in the canon resulted in the reduction of the following subjects by an hour. German (4), Handicrafts (4), Biology (3), Physics, Geography, History, Music, Sports in each case (1).[1] These reductions were agreed by the particular school forums consisting of representatives of pupils, teachers, and parents.

[1] The numbers in brackets indicate the number of schools.

Almost 60 groups in the first forms, just as much groups in the second forms (altogether about 2000 pupils) are involved in this project. On an average these groups consist of 16 pupils. In more than 50% of the schools more than 16 students are in a group due to the relatively high number of pupils in the classes of the first two forms of the schools.

75% of the Informatics classes are regularly held in the morning.

About half of the schools organise their Informatic lessons periodically. Thus Informatics is taught to an extent of two hours in succession every fortnight.

The denotation of the subject is not standardized. In about half of the cases it is called "Informatics", followed by "Introduction to Informatics", "Information technology" and "Basic education in Informatics".

85% of the Informatics lessons take place in the computer lab. Only a quarter of the pupils have to share the PC with a colleague.

Teachers' remarks on Informatics classes

85 teachers with different education (or training) and training courses in Informatics are involved. About half of them completed extended courses in Informatics and acquired certifications such as the (Advanced) ECDL. More then 70% have long-term experience in teaching Informatics in upper grades.

Their motivation is to a great extent a strong personal interest in helping young people master the fourth cultural technique and to awake and reinforce their interest in this field.

At this point it should be mentioned that in some schools the newly generated subject Informatics had to be covered by teachers who are not necessarily interested in teaching a subject they are not trained in and which they are not in favour of.

The following numbers show the teachers' satisfaction with their teaching of Informatics:

- very satisfied 42 %
- satisfied 34 %
- not satisfied 8 %
- no comment 16 %

This corresponds to a great extent to the way school administrations assigned the lessons in Informatics to the teachers. In more than 70% of the cases these assignments are based on agreements, personal wishes and the qualifications of the teachers. The rest of the lessons is allocated rather involuntarily and concerns teachers who did not have a formal training in Informatics.

Apart from relatively large groups, a very lively atmosphere during the lessons together with the practical work on PCs sometimes makes teaching very strenuous and by far more demanding than in other subjects. Due to relative inhomogeneous groups preparations for Informatics lessons are extraordinarily intensive. On the other hand the pupils' high motivation and interest can make up for time consuming preparations. Teachers also appreciate the fact that Informatics allows integrating new ideas spontaneously into their lessons. For an inventive teacher the subject Informatics offers a wide range of opportunities.

Forms of organization

Especially at the beginning an hour of Informatics a week cannot be considered sufficient to cover all the difficulties in establishing the same technical working conditions for all students. Problems such as logging in and out together with a range of other technical problems might reduce the productive time of an hour enormously.

Therefore "two hour-lessons" every fortnight are favoured by several teachers. However, this approach also bares the risk of more lessons being cancelled because of holidays, teachers' training or illness. The period between Informatics lessons might also be too long.

Another criticism to this approach is the students' resistance to practise typewriting at home.

It is not surprising that in all schools most of the lessons consist of a mix of instructional and constructive phases. About 2/3 of the schools involved support project oriented and group work. Individual work (with worksheets or online practising), cross-subject projects and self study on the basis of E-Learning material contribute to the wide range of didactic approaches in Informatics classes. The students are assessed by active participation, tests, homework and (sometimes) final tests.

Number of computer labs at the schools involved

- two in 3 schools
- three in 4 schools
- four in 3 schools
- five in 2 schools

Ratio of PCs per total number of pupils

- less than 1:30 in 2 schools
- between 1:25 and 1:14 in 2 schools
- between 1:12 and 1:9 in 4 schools
- between 1:8 and 1:6 in 5 schools

Ratio of freely accessible PCs in relation to the total number of pupils

- less than 1:100 in 5 schools
- between 1:100 and 1:50 in 6 schools
- between 1:50 and 1:30 in 1 school
- between 1:30 and 1:25 in 1 school

The schools are providing

- a personal login 100%
- personal webspace 67%
- a personal email address 77%

In a third of all schools a special preparation for the ECDL (European Computer Driving License) is offered and in almost 40% of the schools these two initial years are followed by further lessons in Informatics in the next two years.

Software and tutoring software besides MS-Office

Paint Shop Pro, Corel Draw, Paint, typewriting programs, Dreamweaver, Photoshop, Goldfinger, German-, English- and Italian- tutoring software, Bit Media ECDL, Encarta, Interaktiv durch Österreich, Capella, Tippmaster, ECDL-CD, Slovenian tutoring software, etc.

What is being taught in Informatics lessons?

An internal curriculum is predominant at schools (> 90%). Only one school makes use of a school book, in all the other schools teachers work with material they prepared themselves. The Internet very often serves as an excellent source for suggestions and useful material. Additionally E-Learning material, exchange of documents/material between colleagues, computer magazines and books on Informatics are widely used.

Topics covered in the period September 2002 until December 2003

- Input (keyboard/mouse) 16% (min: 2%, max: 47%)
- hardware 4% (min: 2%, max: 6%)
- operating system(s) 5% (min: 2%, max: 11%)
- file management 8% (min: 3%, max: 14%)
- word processing 19% (min: 13%, max: 30%)
- spreadsheet 11% (min: 0%, max: 25%)
- presentation 12% (min: 3%, max: 18%)
- graphics/picture editing 5% (min: 0%, max: 8%)
- communication 10% (min: 3%, max: 25%)
- other topics 10% (min: 0%, max: 13%)

This table discloses very well the different views on the importance of the various topics. Above all the handling of the keyboard is regarded as being extremely important and therefore is trained to a great extent of the time. Obviously there are comparatively big differences at the schools regarding the weight of the various topics.

3 Results of an online survey among the pupils

From January to February 2004 an online survey was carried out among pupils participating in this project. The questionnaire consisted of about 20 items. Three of those, which will be discussed shortly, caused a lot of interesting reactions:

- Did pupils have previous knowledge about computers after primary schools?
- What are the (hardware) preconditions at home and how intensive is the pupils' computer use at home?
- What are the pupils' attitudes towards the subject Informatics so far?

Fortunately the pupils' feedback was extremely high with 1800 of about 2000 taking part in the survey and therefore the result can be regarded as very reliable. The most interesting results of this survey are presented in the following charts and are grouped by the schools involved thus making meaningful comparisons possible.

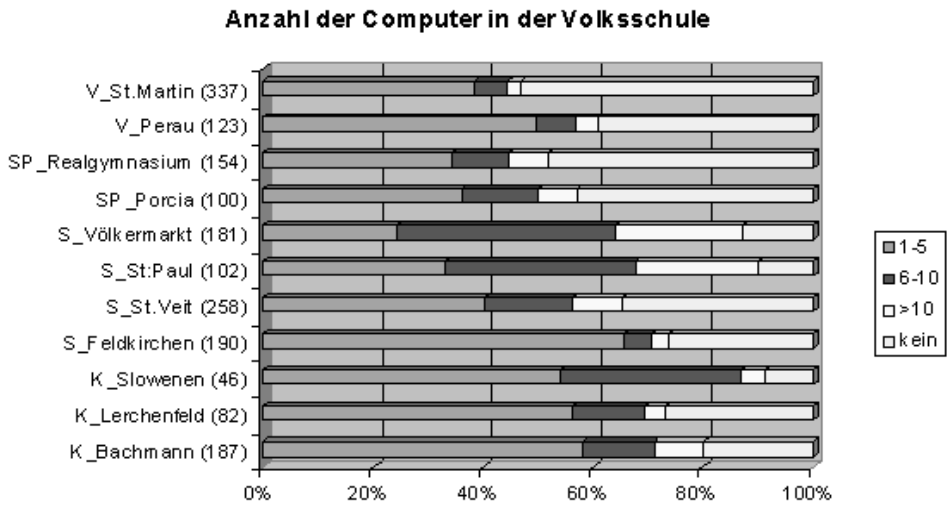


Chart 1: Number of computers in primary schools

This chart clearly shows that the average number of computers in primary schools in Carinthia varies in the particular school districts. Since most schools are still short of PCs the assumption that pupils are systematically introduced to Informatics is not tenable.

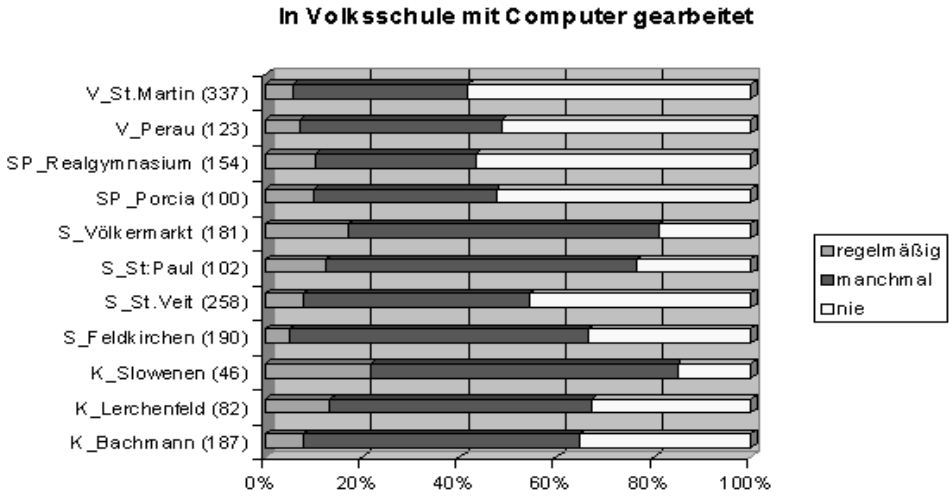


Chart 2: Working with computers in primary schools

This chart supports the thesis that very few pupils regularly worked with computers at primary school (about 10%). However, about 40% of pupils did have experience with computers at primary schools.

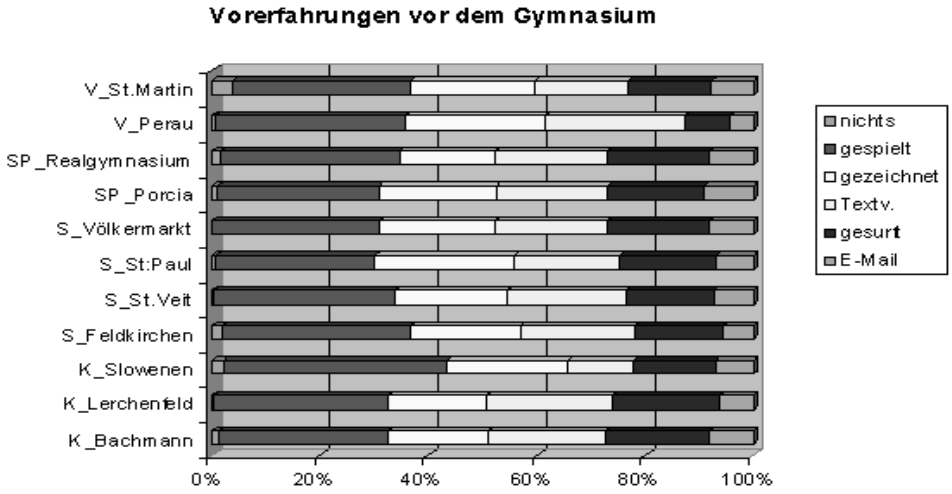


Chart 3: Experience with computers before attending the Gymnasium

Nevertheless, outside school 80% of the pupils gained experience with computers by playing games (30%), word processing (20%), using the Internet (20%) and e-mailing (10%).

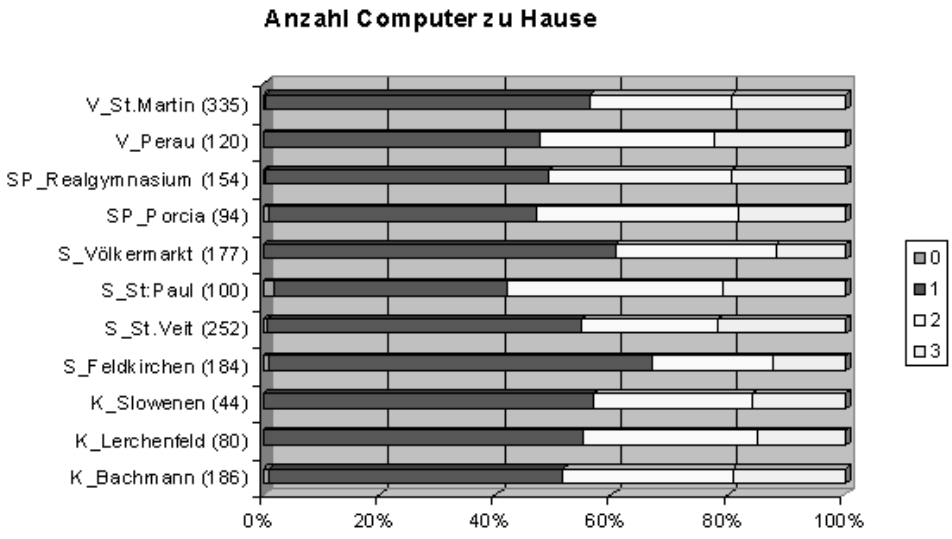


Chart 4: Number of computers at home

This chart impressively shows the saturation with computers at pupils' homes. The percentage of pupils with no access to computers at home has reached less than 1%!

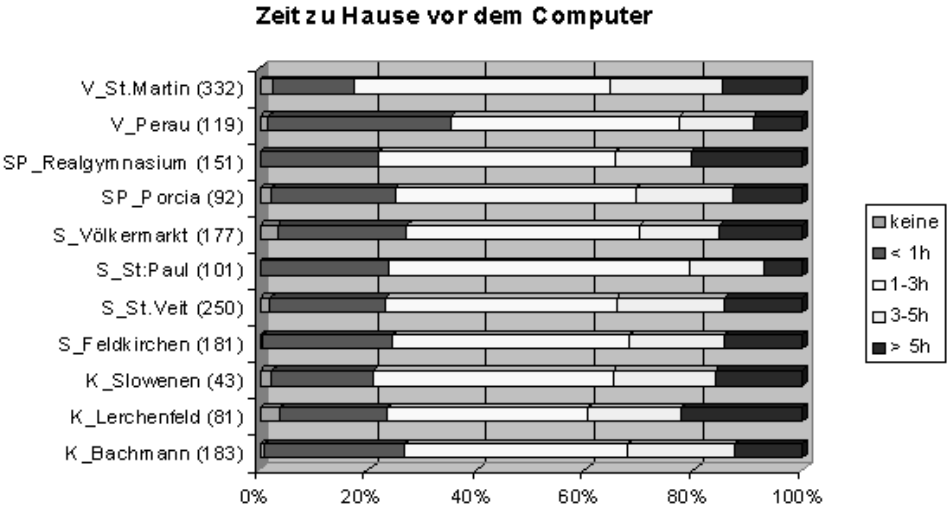


Chart 5: Time spent with the computer at home

How much time do pupils spend in front of the computer in general? This graphic shows that the average time pupils use computers at home lies between 1 and 3 hours a week.

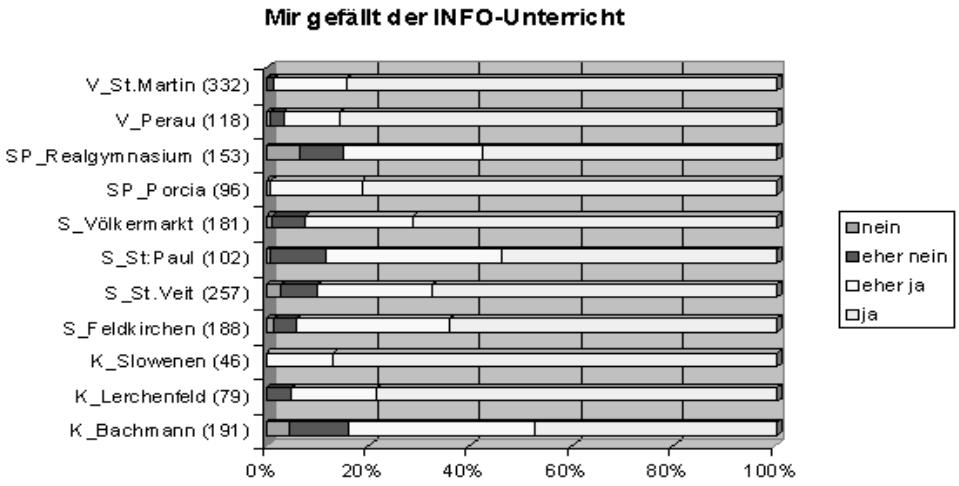


Chart 6: I like Informatics

This reveals the fact that the pupils like the subject Informatics above-average.

A further analysis shows that there is a difference in this attitude between boys and girls. In general boys like informatics a bit more than girls.

There is also a difference between the first and the second form. In the second form, after one year of Informatics, the enthusiasm (of boys and girls as well) for this subject decreases significantly.

4 Finding the standards

The third and major step of this project was to establish a standard curriculum. As mentioned above present teaching of Informatics differs from school to school to a large extent. Since this situation is not tenable the call for standardization can no longer be ignored.

In order to get rid of this situation of diverging teaching objectives all schools involved in this project presented their specific curricula by putting all the detailed learning items into a structured online database. Afterwards the project team categorized and clustered these items. Finally, in a teamwork process the curriculum took concrete form.

The main teaching objectives of this minimal standard are as follows:

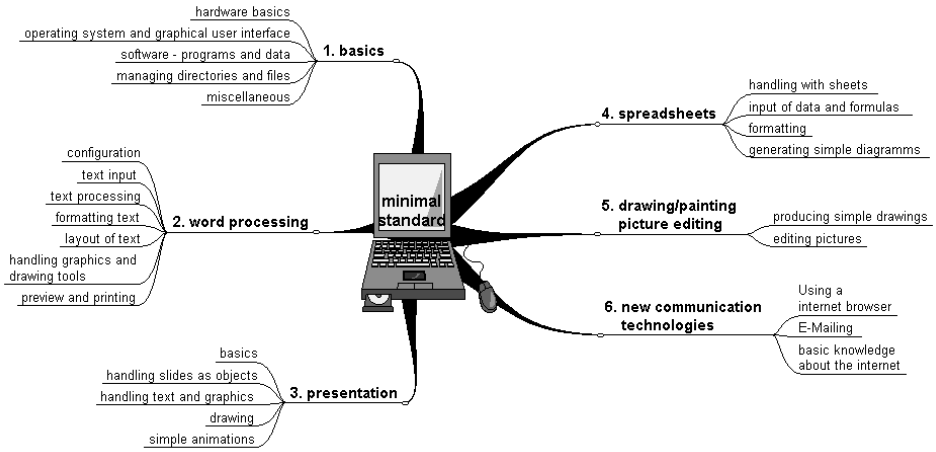


Figure 1: The categories of the minimal standard

Figure 1 shows the main items of the minimal standard. There is of course a further classification where the teaching objectives are operationalized which means that they can be assessed and tested.

This definition of a standard for 12-year-old pupils is not really revolutionary. Nobody would expect this. But it is a compromise of many teachers who are involved in this project. The decision about this curriculum was unanimous at the end and it was preceded by constructive discussions. Nevertheless especially two aspects are worth mentioning. First there was a slight uncertainty whether "typewriting" should become part of the standard or not. At last the coordinators came to the conclusion to abandon its integration into the definition of the standard. Of course typing skills are important, but they should be acquired in training courses outside Informatics classes. However, the knowledge of some main keys on the keyboard should be part of the standard.

The issue of spreadsheets led to a discussion as well. In the end all participants were convinced that the basic handling of spreadsheets is essential and indispensable especially with regard to applications of other subjects such as maths and geography.

On the website http://www.schulinformatik.at you can have a look at the curriculum and the syllabus respectively

5 About curricula, syllabi and standards.

This syllabus can be understood as an orientation guide for both teachers and pupils Nowadays teachers are very often confronted with diffuse curricula which leave them alone with the decision what to teach. Experienced teachers might not have problems with such a curriculum, but in most cases, especially when teaching a dynamic subject such as Informatics, standards with operationalized learning and teaching objectives are often regarded as a desirable orientation guide.

Apparently many Austrian teachers were very happy when the idea of the ECDL (European Computer Driving License) emerged and the first syllabi of this international certificate became generally known. For the first time they had something concrete at their disposal and could teach the pupils according to an internationally accepted curriculum.

The theoretical background of a detailed definition of standards can be deduced from a didactics which is orientated on operationalized teaching objectives and can be found in an article of Christine Möller [Gu02]. This didactical approach assumes that

- the finding of the teaching objectives should be a task for people in charge of the curriculum,
- the emphasis should lie in a clear description of these targets, that means in a precision which can only be achieved if the performance of the learner as well as the content by means of which the performance (knowledge, skills, competence) can be assessed is determined distinctly,
- precise or operationalized targets are a necessary (but not always sufficient) precondition for an adequate choice of teaching methods,
- the success of the learning and teaching process can only be verified by means of the teaching objectives

This didactical approach is prescriptive in a sense that it should provide both teachers and learners with concrete methods for planning, organising and assessing their lessons.

The planning of lessons which orientates towards teaching objectives can be identified by a determinable behaviour of the learners on the basis of concrete guidelines. There should also be a precise classification of the goals into specific categories. This process of operationalizing is completed only if detailed teaching objectives are embedded in a set of fundamental ideas and some superior objectives [cf. Eigenmann/Strittmacher 1971].

When the planning phase is finished teachers have to find the appropriate methods to support the pupils in reaching the teaching objectives. This is not an easy task but it is easier to plan when the objectives are clear and well defined.

The choice of appropriate methods is up to the teacher and nobody can relieve him from this duty.

The last but very important part in this didactic approach is assessments. Assessments should prove whether the learning process was successful or not. The task of constructing adequate and valid tasks and problems according to the curricular goals is very demanding and time consuming. But it is necessary in order to evaluate the result of every learning process.

The advantages of the didactic approach described are obvious:

- Transparency
  Concrete teaching objectives provide an informative basis for pedagogical argumentation and are results of an appropriate choice of the subject-matter. They make understandable the objectives of a curriculum.

- Assessing
  Teaching objectives provide for clarity (for all the people involved in the learning process) and can be a basis for a fair assessment system. At this point it should also be mentioned that focussing exclusively on operationalizing holds the danger of an uncontrolled mechanization of education.

- Efficiency
  Assuming concrete teaching objectives are the basis of an adequate learning organisation it can be deduced that the learning situation is clear and unambiguous which means that there is a chance of positive reinforcement for teachers and learners.

Therefore this didactic approach is a very efficient instrument of constructing a desirable behaviour in a sense that the students know what is expected from them and that they have the chance to achieve these objectives.

These theses lead to the issue of "standards" which is currently being discussed intensively not only in Austrian schools. Standard (in the context of school and education) is just another word for focussing on the desired learning results of the pupils [Do04]. From the seven criteria for educational standards [Kl03] three should be pointed out here:

- Cumulativity
  Educational standards refer to competences which have been acquired by a learner so far. Therefore they aim at cumulative, systematically networked learning.

- Commitment for all
  The minimal standards should be universally valid and can be expected to be achieved by (almost) all pupils at a certain age in all types of schools.

- Realizability
  The learning objectives should be achievable with realistic effort.

At the moment in Austria a nationwide definition of standard items is in progress, but until now restricted to the subjects German, Mathematics and English. The ministry for education plans introduce the handling standards for selected schools. In a first step the target-group are 14-year-olds. In Austria this is the transition between "Unterstufe" and "Oberstufe", when many pupils change the school type.

At this point the question arises why to establish a (at the moment only regional) standard in the field of Informatics and IT at such an early stage. The answer is based on the following theses and presumes that informatic competence is an essential part of a general education in form of a fourth cultural technique.

- Standards support the introduction of a mandatory subject Informatics and therefore should be based on a mandatory set of concrete teaching objectives.
- Standards foster the learning process, provide for a solid basis, and help prevent an early digital divide among the pupils.
- The age of 10-12 is appropriate for a first systematic education in Informatics.
- Standards provide for a solid fundament with respect to the use of ICT in other subjects as well as in forthcoming E-Learning environments

6. Resume and perspectives

Some main objectives of the project which have been described in this article have been achieved already. These are

- intensive discussions about Informatics at schools and a broad exchange of experience,
- an extensive and detailed representation of the status quo with regard to the realisation of education in the field of Informatics in the 1st and 2nd form of the Carinthian comprehensive secondary schools,
- the definition of a manageable set of clear teaching objectives as an orientationfor both teachers and pupils.

But this is not more than a first step in the right direction. Defining the teaching objectives is only half of the truth. The challenge for the near future is to ensure that the process of achieving the standards will be successful. In terms of the saying "the journey is the reward" demanding efforts still have to be taken to find appropriate strategies to fulfil the high expectations.

In the meantime measures have already been taken to provide a representative pool of material, appropriate exercises and assignments. Almost all schools involved revealed their collection of (partly unstructured) learning material for covering many items of the standard. This useful data will be categorized and structured and will be put at disposal for the schools in the form of an internet platform at the beginning of the school year 2004/05. It will be up to the didactical skills of the teachers to make appropriate use of this learning material.

The last step which completes the process of standardization, assessment, is the most important. Assessments give feedback about the competences the pupils (should) have acquired after two years of instructions in Informatics. At the moment this step is in progress.

"A standard is something set up and established by authority as a rule for the measure of quantity, extent, value, or quality [Ma04]". A far as marking is concerned a remarkable diagram as another result of the online survey among the pupils is shown next.

In the meantime measures have already been taken to provide a representative pool of material, appropriate exercises and assignments. Almost all schools involved revealed their collection of (partly unstructured) learning material for covering many items of the standard. This useful data will be categorized and structured and will be put at disposal for the schools in the form of an internet platform at the beginning of the school year 2004/05. It will be up to the didactical skills of the teachers to make appropriate use of this learning material.

The last step which completes the process of standardization, assessment, is the most important. Assessments give feedback about the competences the pupils (should) have acquired after two years of instructions in Informatics. At the moment this step is in progress.

"A standard is something set up and established by authority as a rule for the measure of quantity, extent, value, or quality [Ma04]". A far as marking is concerned a remarkable diagram as another result of the online survey among the pupils is shown next.

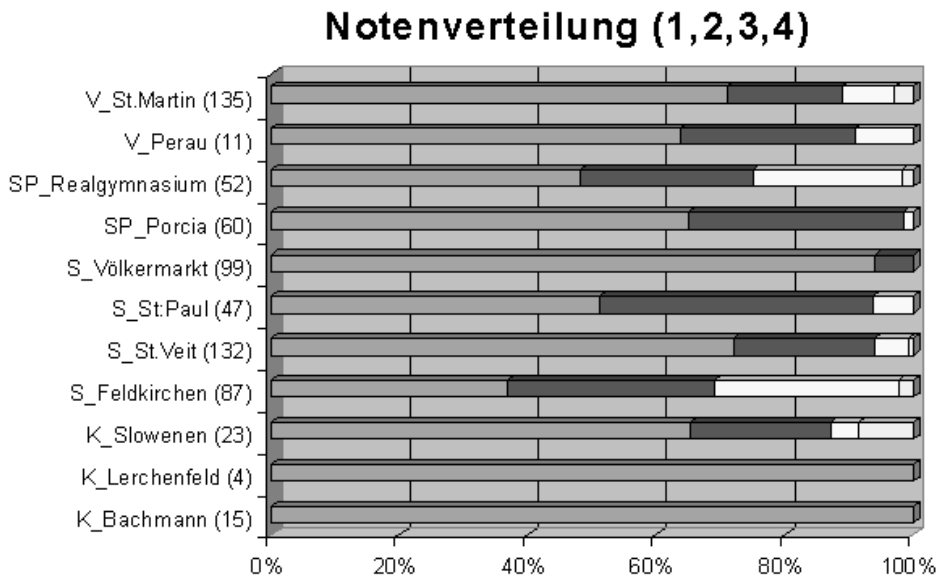


Chart 7: The distribution of the marks given in the subject Informatics in the 1th form

In Austria students are graded between 1 (very good) and 5 (insufficient). This chart obviously reveals the fact that that the teachers grade up because the marks are generally very good.

"It must be pointed out that the marks and the (real) competences often do not correspond [Bi04]". It can be assumed that Chart 7 is not really meaningful with regard to the knowledge and skills of the pupils. But the thesis that standards will lead to a better balanced distribution of marks has not been supported yet.

References and Web links

[Gu02] Herbert Gudjeons, Rainer Winkel (Hg.), Didaktische Theorien, Bergmann+Helbig Verlag, p 75 – 92

[Do04] CD Austria, Standards in der Schulinformatik, Cristian Dorninger, p 4-6

[Kl03] Eckhard Klieme et al., Zur Entwicklung von Bildungsstandards – Expertengutachten für die Kulusministerkonferenz, Bonn, 2003

[Ma04] The Marriam-Webster Online Dictionary http://www.m-w.com (June 2004)

[Bi04] http://www.bildungsstandards.de (June 2004)

The Contribution of Computer Science to Technical Literacy

Eckart Modrow

Max-Planck-Gymnasium
Theaterplatz 10
37073 Göttingen
emodrow@astro.physik.uni-goettingen.de

Abstract: The idea of general education is, among other things, to bring pupils in contact with different ideas, topics, methods, activities, and areas of work independent of their social or family context. In this sense school serves as a "test living", where pupils can try out themselves in different fields. These attempts may fail here without any serious consequences. It is hoped that some of these tests will be successful, so that young people at the end of their school career can choose from a positive variety of life perspectives. If an important aspect is missing at school, this aspect cannot be tested there. But the pupils know very well that then the personal risk of a failure in that case is being shifted to the time after school. A positive decision for this aspect becomes difficult.

In High School technical thinking and acting hardly have any room. Since an ever growing number of students of an age group pass through High School, they hardly come in touch with technology-related topics. Consequently later they often ignore the enormous field of technical professions. If the spectrum of school subjects is to be extended by a new subject, which can establish a close link to technology, this subject should have its place in grades 7-9, because there the important predecisions for the later professions are made. The subject "computer science" is able to take over this task very well, because, due to the universality of its tools, it can use these tools without any additional equipment in many different areas.

The following contribution examines how the term "technical general education" can be concretized. On the basis of some examples the consequences for instruction are explained.

1 Introduction

In all subjects the selection of contents predominantly takes place due to subject-immanent considerations. Usually it is too little considered that these contents only are stones in a larger puzzle, which as a whole should result in a general education that, among other things, has the job to bring pupils in contact with different ideas, topics, methods, activities, and areas of work independent of their social or family context. In this sense school serves as a "test living", where pupils can try out themselves in different fields. These attempts may fail here without any serious consequences. It is hoped that some of these tests will be successful, so that young people at the end of their school career can choose from a positive variety of life perspectives. If an important aspect is missing at school, this aspect cannot be tested there. But the pupils know very well that the personal risk of a failure in that case is being shifted to the time after school. A positive decision for this aspect becomes difficult.

2 Technical literacy

At first we have to clarify how we have to understand "technical literacy" or "technical general education". If we consult Klafkis[1] well-known definition for general education, we learn, "general" means

- that all members of the society have access to contents of the education system,
- that the whole of the human possibilities is addressed, thus the person as a whole,
- and that education takes place by means of key problems, typical for the epoch.

The second aspect means that the pupils should find out the full width of their possibilities, and so surely an overview about the different sciences is needed. In High School technical thinking and acting hardly have any room. The huge field of technical professions and engineering sciences has no related subject. Physics could actually take over the task, but both at university and school it is handled as a pure basic subject, nearly without connection to current technology (see below). Since an ever growing number of students of an age group pass through High School, they hardly come in touch with technology-related topics. Consequently later they often ignore the field of technical professions. So the first aspect of Klafkis definition also is affected: High school students in Germany usually do not belong to the disadvantaged social groups. But they obtain an education which omits substantial ranges of a technical world.

It is trivial that the third aspect of Klafkis general education is fulfilled with IT-problems today. Similarly as in former times the technical application of electrodynamics substantially accelerated the introduction of the school subject "physics", the extreme social meaning of IT-systems will force a compulsory school subject "computer science". Actually it should have had.

---

[1] [Kla85] S. 17

The other definitions of general education usual in the field of didactics of computer science also fit well to a technical variant. Among the criteria of Bussmann and Heymann[2] particularly the "*preparation for future life situations*" and the "*construction of a conception of the world*" are relevant. The "*stabilization of the ego of a pupil*" can be interpreted similar as Klafki definition. In the "*recommendations for a concept regarding the IT-education at schools*" of the GI[3] the points "*principles of effect of IT-systems*" and "*problem solving with IT-systems*" particularly fit.

How should the specific technical aspect be understood? A technical literacy in my view must cover three ranges:

- It has to produce **knowledge** about basic technologies, their applications and effects. In this sense it considers Klafkis "key problems".
- It has to obtain **technical thinking**, a goal- and product-oriented way of working. Here the gaining of knowledge is considered not so much as a way to understanding, but more instrumentally for reaching a purpose. The necessary knowledge has to be found independently. Not the contents are primary, but the kind of their acquisition and their utilization.
- It has to produce and strengthen **team ability**, in order to use individual knowledge and talents in cooperation with others to reach the common goal.

So, technical literacy has goals which traditionally are not straight in the centre of High Schools: "*Different from the natural sciences, where the laws of nature form the centre of interest, technology is arranged and created directly by humans. Thus the scientifically oriented canon of subjects cannot produce eo ipso technical literacy.*"[4] Particularly the individual marking opposes contrarily to team work, and independent learning can be found only in beginnings. The output orientation of the new education standards also will not change anything, because knowledge namely is more applied, but the kind of knowledge is given. More interesting are the planned core-curricula, which fix only a part of instruction and so leave place for extensions – at least they should. Since the idea is not completely new, I doubt that it will be carried out in the desired sense. Experience unfortunately teaches that on the one hand the "cores" are much too extensively defined and on the other hand the cores are taken for the whole by the teachers because in times of central school-leaving exams the tasks of the exams will refer necessarily to these cores. We have to look for another place for technical literacy.

Thus, if technical disciplines in their application orientated way of handling knowledge and their heuristic working method should be noticed by the pupils as a possible field of their later job, then a subject with a close link to technology should worry about it. Computer science as the only technology-oriented subject in High School would be outstanding suitable for this because in its lessons exactly this working method can be tested. To be effective, this subject should have its place in grades 7-9, because there the important pre-decisions for the later professions are made - less as positive decisions

---

[2] [Bus87] roughly translated
[3] [GI01] as well
[4] [VDI02] as well

than as negative: Subjects are voted out. A "recruiting campaign" later does not reach the majority of pupils.

3 Time distribution to the subjects

Let us have a look on the distribution of time for subjects to find out, where the key topics in High School are. For example we choose the new timetables in Lower Saxony for the shortened system with twelve years:

field	timetable 1 (197 hours)			timetable 2 (208 hours)		
	subject	hours		subject	hours	
A	DE	24		DE	23	
	1. FS	23		1. FS	22	
	2. FS	20		2. FS	20	
				3. FS	19	
	MU	10		MU	9	
	KU	10		KU	9	
	sum:	**87**	**44,2 %**	sum:	**102**	**49,0 %**
B	GE	11		GE	10	
	EK	9		EK	9	
	PO	8		PO	9	
	RE/WN	12		RE/WN	12	
	sum:	**40**	**20,3 %**	sum:	**40**	**19,2 %**
C	MA	24		MA	23	
	BI	11		BI	10	
	CH	8		CH	7	
	PH	9		PH	8	
	sum:	**52**	**26,4 %**	sum:	**48**	**23,1 %**
others	SP	12		SP	12	
	Verfg.	1		Verfg.	1	
	Ags	5		Ags	5	
	sum:	**18**	**9,1 %**	sum:	**18**	**8,7 %**

Expanding field A …

… chargeable to field C.

We can see that about half of the time is reserved for the languages, music and arts, less than a quarter for math and natural sciences.

What do pupils learn there? In a very rough manner we can say:

They learn

- in field A to produce, understand and (literarily) interpret texts,
- in field B to understand and recognize relations in texts and data,
- and in field C to find data and recognize relations.

Pupils thus work predominantly interpretively with texts and they analyze and evaluate data. Constructive work we find actually only in field A. That changed a bit with the new media, because e.g. within the B-field presentations etc. often are produced about special topics and current developments. But these products also have to be regarded as "text production" in a wider sense. Instrumental knowledge is hardly acquired.

So in this traditional canon of subjects only the natural sciences physics and chemistry are candidates as a place for technical questions.

PH/CH	17	**8,6 %**	PH/CH	**7,2 %**

The entitled amount of time for them is small, compared e.g. with the foreign languages.

4 The motivation of teachers and pupils

Even in the natural sciences the conditions for technical learning normally are not given. The basic studies e.g. of a physicist, which corresponds to the studies of a physics teacher in a large degree, is concerned exclusively with physical fundamentals, thus with finding of and handling with physical regularities. These are in their original form applicable to idealized situations, but not to material technical problems. A normally trained physics teacher does not understand anything about technology at all. As a result material technology hardly emerges in his lessons - if the secrets of the iron does not regard. Above all technical thinking as mentioned above does not arise. A physicist acquires knowledge in order to understand connections. Whether something and which of this knowledge can be applied anywhere is subordinate. He has completely different priorities as someone, who wants to solve a technical problem and evaluates knowledge in regard to a possible solution of the problem. The experience of technical problem solving is not obtainable in physics lessons, and even with a reorientation of the teachers no time for technical literacy would be present within their small portion of the timetable, because the natural sciences have to achieve their own goals.

Completely different is the situation for computer science teachers. The nearly always insufficient, usually completely missing training prevents a theoretical orientation of the subject, and also the motivation both of teachers and pupils comes from another direction. Peter Berger[5] writes: "*In the innovative school subject `computer science´ innovation takes place at present less from inside, by the innovative teacher, who finds a new paradigm of instruction and learning - but rather from outside, by a new paradigm, that `finds its teacher´ and forces him, also the quite traditional one, to use increasingly innovative patterns.*" Pupils are different motivated than in other subjects as well. At an interrogation of pupils in higher level courses[6] rather clear tendencies showed up:

- Pupils in computer science courses classified themselves as little careful, industrious, ambitious and planning. They resemble therein with pupils in math or physics courses and are, compared with those, still under the average. In the subject however they approved themselves a substantially better work attitude.
- Within the fields of creativity and social learning the self-assessment is similar to the language courses.
- Extremely pronounced is the self estimated ability and the expectation in instruction concerning independent, problem oriented learning.

As examples the following partial results may serve. I asked among other things for the self-assessment about creativity and the desire for independent work, both generally and in the selected subject:

	IN	MA	PH	DE	EN
creativity generally	65 %	69 %	57 %	74 %	71 %
creativity in the subject	82 %	46 %	62 %	79 %	57 %
difference:	**18 %**	**–23 %**	5 %	5 %	–14 %
independent work gen.	29 %	38 %	57 %	68 %	71 %
independent work i. t. subject	65 %	46 %	67 %	58 %	62 %
difference:	**35 %**	8 %	10 %	–11 %	–10 %

The wish for problem oriented lessons, which permits independent learning with effects beyond school was obvious:

"I would like to select tasks for myself, which becomes part of the lessons."					
	IN	MA	PH	DE	EN
generally	63 %	46 %	55 %	68 %	62 %
concerned to the subject	**88 %**	38 %	57 %	74 %	62 %
difference:	**25 %**	–8 %	2 %	5 %	0 %

[5] [Ber98] roughly translated
[6] [Mod03]

	IN	MA	PH	DE	EN
"Contents should follow from problems"					
generally	69 %	77 %	90 %	79 %	90 %
concerned to the subject	**81 %**	**100 %**	**90 %**	**79 %**	**90 %**
difference:	**13 %**	**23 %**	0 %	0 %	0 %
"I would like to find out contents for myself e.g. from books or from the Internet."					
generally	31 %	31 %	29 %	47 %	71 %
concerned to the subject	**75 %**	23 %	48 %	53 %	62 %
difference:	**44 %**	–8 %	19 %	5 %	–10 %
"In lessons I would like to win suggestions for work in addition to the lessons."					
generally	**38 %**	54 %	67 %	79 %	71 %
concerned to the subject	**81 %**	69 %	76 %	74 %	71 %
difference:	**44 %**	15 %	10 %	–5 %	0 %

It is obvious that in computer science – followed from the difficulties in teacher training - aspects of application should be more important as e.g. theoretical aspects. Computer science teachers and pupils "want to do something", and the appropriate tools are available for them. Due to the universality of these tools, they are applicable without any additional equipment to very different fields, in times of financial bottlenecks a real advantage.

5 The contribution of computer science to technical literacy

In computer science pupils

- may be acquainted with valid hard- and software-models of computer science systems and/or learn to develop and test them. The understanding for developments in a world, shaped by technology, is supported, e.g. by work in and with networks, as well as their social effects. It has to be noted, that this is one of the few fields in which schools have experiences since many years.
- gain experience in instrumental acquisition and employment of knowledge. The fields of appliance can be selected freely from a wide range due to the universality of the tools and methods of computer science. Here experiences in independent work are much more easily possible than in other subjects. Due to the enormous efficiency of the tools, first and inefficient beginnings of problem solutions are realizable and testable, so that independent problem solving becomes possible as standard requirement without excessive demand to the pupils.
- partial are active in work-sharing phases within a team - if lessons are organized accordingly.

6 Result

- In the face of the development of information technologies, technical literacy has to be an obligate goal for all school forms, especially for High Schools. Because the integration of information-technical basic formation into schools has failed, a special subject is required.
- The position of information technology as crucial fundamental technology makes it possible to select computer science as technical reference subject. The proximity of the subject to the engineering sciences, its result- and product-oriented methods and the power of its tools make instruction possible on different, self-selected problems and with experiences in teamwork, independent learning and problem solving. And that without additional expenses.
- The proximity of computer science to technology offers the chance to extend the spectrum of High School subjects decisive by obligate computer science instruction. So the renouncement in contents of technical computer science in the new EPAs is a crucial error, in my view. Computer science, which relies alone on algorithmically, theoretical and socio-political aspects of the subject, gives up thoughtlessly the possibility of adapting the High School education to current conditions.

References

[Ber98] Berger, Peter: Informatische Weltbilder
LOG IN 3/4, 1998

[Bus87] Bussmann, H. / Heymann, H.-W.: Computer und Allgemeinbildung
Neue Sammlung 1 1987

[GI01] Gesellschaft für Informatik: Empfehlungen für ein Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen
Beilage LOG IN 2, 2001

[Kla85] Klafki, Wolfgang: Neue Studien zur Bildungstheorie und Didaktik
Beltz 1985

[Mod03] Modrow, Eckart: Pragmatischer Konstruktivismus und fundamentale Ideen als Leitlinien der Curriculumentwicklung
Dissertation 2003

[VDI02] VDI-Ausschuss „Ingenieurausbildung“: Technische Bildung in die Schule!
2002

Creating Proper Media Objects for Computer Supported Learning-Environments

Olaf Scheel

Research Group Didactics of Informatics
University of Paderborn
Fürstenallee 11
D-33102 Paderborn
olasch@uni-paderborn.de

Abstract: In the last three years the research group of Didactics of Informatics at the University of Paderborn has carried out a project called MuSofT (Multimedia in der SoftwareTechnik). The aim of the MuSofT project is to produce multimedia learning objects for teaching and learning software engineering. The educational objectives are achieved by means of case studies, especially the model of a high rack storage area. Thus, the idea of an Informatics Learning Lab (ILL) occurred.

The ILL is an interactive web-based multimedia exploration platform to enable constructivist types of blended learning. Students use learning objects in a self-organized learning process in an open collaborative learning environment. This paper describes a method to create learning objects for this scenario and a concept for an empirical study to evaluate their quality:

After deciding about objectives and the learners' roles in a given technical context of the ILL we have to focus on the construction of learning and media objects. Media objects can be constructed on different levels of abstraction from the socio-technical information system of the case study: real world scenario, physical model (here LEGO Mindstorms) or software model. For all levels of abstraction the ILL provides students with different types of media, which should enable them to gain comprehension of relevant facts and structures of the ILL. We also have to distinguish between different types of encoding: symbolic (dealing with signs and symbols e.g. in a text), drawing (abstract mapping of facts in a chart) and picture (lifelike mapping). These types of encoding are cut into two different areas: respectively static and dynamic types of information representation at the different levels of abstraction.

In this grid of abstraction levels, encoding types and also granularity many different types of media objects are possible. But which are the proper ones to support learning software engineering effectively?

1 The Informatics Learning Lab (ILL)

1.1 Aims and Reasons

The teaching at universities is dominated by teacher-centred lectures which are composed in a domain-specific way – without (or with less) applied context. But this traditional way of learning has several disadvantages which can be attenuated by the use of E-Learning:

- Gruber, Mandl and Renkl asserted in 2000 (by referring to the TIMSS II study[1]) that knowledge without embedded context is inactive. It exists in an abstract way, can be accessed in examinations - but not in practice.

  "*[…] Das gewissermaßen 'in vitro' erworbene Wissen kann zwar im universitätsanalogen Kontext, in dem es erworben wurde, genutzt werden, etwa in Prüfungen; in komplexen, alltagsnahen Problemsituationen gelingt die Wissensanwendung jedoch oft nur unvollständig oder überhaupt nicht. Damit kommt es zu einer Kluft zwischen 'Wissen und Handeln'.*" [GM00]

- Students have different preparatory training - maybe because of different school education or timetables at university. But lectures don't assist individual ways of learning.

- The individual attendance of a huge group of students in a traditional way is costly.

Thus reasons for the introduction of E-Learning at universities could be:

- to adjust the different knowledge of students in learning groups (That implies the question in which areas adjustment will be necessary.),

- to create applied knowledge which can be transferred in a new context,

- to enable self-directed learning and to teach responsibilities for this way of learning,

- to save money while teaching huge groups of students (This aspect demands the existence of adequate learning materials and platforms. Because nowadays E-learning is often more expensive than traditional learning. – One starting point for blended learning concepts.).

These are the principal aims of our Informatics Learning Lab in the domain of software engineering.

---

[1] http://www.timss.mpg.de/

1.2 Learning Processes in the ILL

The ILL (Informatics Learning Lab) is an exploration environment to enable self-organized learning processes in learning communities. In consideration of the pedagogical position that learning is an active process – not passive acquisition of knowledge, students can interact with the learning material, teachers and other students. Thus the didactical (models and roles, objectives, selection of content), the organizational (methodical concept, integration of media, interaction between learning groups) and the technical context (learning platform, groupware, CMS, digital media) could be a target of an empirical study. This paper will mainly describe a study about the selection of content - more specific the selection of media and learning objects - and its implementation in a learning process according to constructivist learning theories.

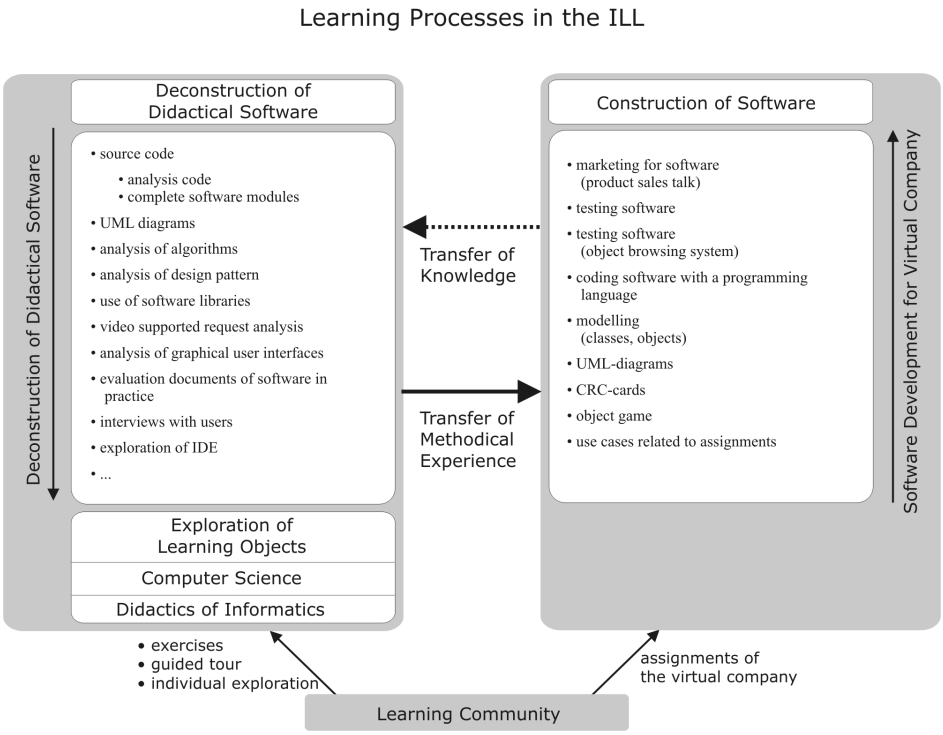


Figure 1: Learning Processes in the Informatics Learning Lab.

Core of the ILL are several case studies, which are dealing with different aspects of software engineering: *school kiosk*, *media player*, *computer game 'Ursuppe'* and a *high rack storage area* (HRSA) have been completed up to now. Any case study consists of didactical software which has to be examined by the students. After and during this the acquired declarative and procedural knowledge has to be used in a constructive way by creating a new or changing the old information system.

The different learning phases in the ILL consequently are [Ma03] (in the scenario HRSA):

- "foundation of a virtual company with students as the owners, assignment to build an automated commissioning unit;
- exploration and deconstruction of the physical and software model (guided, self-directed supported by LOs);
- modelling a software model with CRC-Cards and UML;
- exploration of the modelling concept of the Mindstorms model, comparison and assessment with regard to the model concepts created by the students;
- acquiring a deepened knowledge of the three perception models by using open and closed LOs (source code, technical functionality);
- operating re-engineering tasks related to the Mindstorms model (variation of sensors, different types of racks),
- exploration of the communication protocol used by the bricks and of the layered architecture of the software (using LOs);
- cooperative construction (modelling, encoding, assembling Lego components) of the commissioning unit by the students, transfer of knowledge on different levels, self-directed use of LOs according to their needs of support;
- presentation of the product, quality assessment, reflection on the learning process and self-evaluation regarding the achievement of objectives."

2 Learning Theories and the ILL

Blömeke asserts with a view of constructivist learning theories: "Konkret bedeutet dies, dass durch ein Ausgehen von authentischen Aufgaben, die Einbeziehung authentischer Kontexte, die Einnahme multipler Perspektiven und Modelllernen der Wissenserwerb optimiert werden kann." [Bl01] This is a reference to cognitive apprenticeship, situated cognition, anchored instruction and cognitive flexibility. The ILL tries to fulfil the consequences of these sub-theories.

The cognitive apprenticeship by Collins, Brown and Newman [CB89] demands the descriptive demonstration by the tutor (*modeling*), increasing activity of the student (*scaffolding*) - simultaneous with decreasing activity of the tutor (*fading out*), until the role of the tutor is limited to guidance (*coaching*). It is important for the success of this approach that articulation and reflection take place at every step of this procedure. The role of the tutor in the ILL is partially adopted by the learning objects: at first by closed objects with strong structuring of students activities - later by open ones for self-directed work and learning.

The case studies of the ILL build up anchors for authentic situations in terms of the Vanderbilt Group [Va94]. Learning in the ILL is problem-based, opened for new solutions. The variety of media objects for the same subject allows individual views in terms of the purpose of cognitive flexibility [Sp92].

3 Media in the Informatics Learning Lab

Media in the ILL can be (among others) classified by granularity, coding type and level of abstraction. Here the categories coding type and level of abstraction should serve as a grid for media objects, which can be bundled in complex learning objects of higher levels. These higher LOs can be classified by granularity - or rather by their function during the learning process.

3.1 Types of Media Objects

The ILL consists of several content modules which are covering different aspects and objectives of software engineering. To categorize a single media object of the case study high rack storage area you can differentiate between coding types (according to Tulodziecki [TH02]), level of perception and abstraction.

This classification of media and learning objects in categories should assist to choose proper media objects for the construction of learning objects for a special subject to be learned. The single media objects symbolize different views on a socio-technical information system – for example on a high rack storage area. If a media object is identified as improper for a learning subject, it is possible to replace it by the help of this grid of categories with an object that deals with the same subject – but with another view on the system. Currently, media objects for several subjects (e.g. communication between the RCX-units of a LEGO Mindstorms model of the HRSA) exist. – Others have to be created or modified for this empirical study.

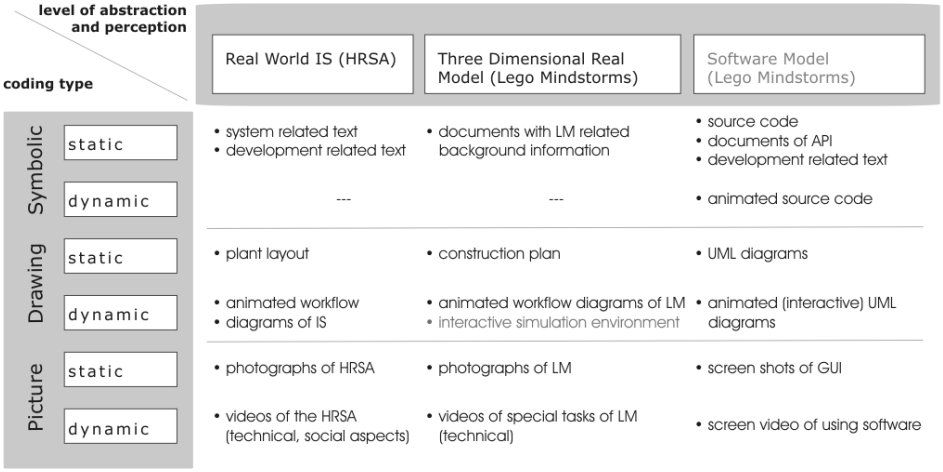


Figure 2: Media objects of the case study *high rack storage*.

3.2 Granularity of Learning Objects

Learning materials in the ILL should be reusable - not only for teachers but first of all for students. To manage learning materials in a technology-supported learning environment we have to build discrete chunks of these materials, the learning objects[2]. But learning objects exist on different aggregation levels with a different influence on methodology, didactical concepts and learning theories.

The LOM-Standard (Learning Objects Metadata) [LOM02] also describes aggregation levels for Learning Objects, but the levels 1-4 only refer to abstract terms like *lesson, course* and *set of courses*. The description of the specific role of these levels is missing. More reasonable is the consideration of a student's interaction with the learning material. According to Koppi/Hodgson [KH01], Kassanke [Ka03] and SCORM [ADL04] we have five different levels of learning objects granularity[3]:

Level 1: Raw Asset

This can be defined as the smallest unit of media fragments with potential use in an educational context. They are multi- or monomodal, but have no inherent educational directives. Raw assets have the highest level of reusability in other didactical, organisational or technical contexts. Examples:

---

[2] „Learning Objects are defined here as any entity, digital or non-digital, which can be used, re-used or referenced during technology supported learning." - IEEE Learning Technology Standards Committee http://grouper.ieee.org/LTSC/wg12/index.html

[3] The question about metadata is connected with granularity and hierarchy of learning objects. Butt his isn't topic of this publication.

- A picture of a high rack storage area.
- A video of selling goods in a school kiosk.
- An animation of a rack feeder.

Level 2: Learning Asset

A learning asset is a raw asset, or maybe assets, in an educational context with a small thematic range. The atomic elements of the learning assets are ordered in a sequence, but not supplemented with a task or an exercise. They are passive objects. Learning assets are low affected by learning theories or pedagogical decisions. The reusability in other contexts is high. Examples:

- A picture of a high rack storage area with a description of the capacity of the storage and of the functions the several parts have.
- A sequence of videos from a school kiosk with the reference to use cases.
- An animation of a rack feeder with object diagrams of the subunits.

Level 3: Task or Exercise

A task or exercise demands activities by the students with small thematic range. It can include raw or learning assets to foster these activities. The problem-based tasks or exercises are usually affected by pedagogical decisions because of the order of elements, the character of the initiated activity or the feedback method. As a consequence of this, level 3 LOs have low reusability in other contexts. According to the ideas of constructivism, especially cognitive apprenticeship, and concepts of blended learning it seems necessary to distinguish between open and closed LOs - depending on their position in the learning process.

Closed learning objects are small CBT/WBT-units. They include a problem based task, learning materials to deal with this assignment and a guided tour through these materials - usually finished by a test to verify the learning outcomes. The students are not allowed to choose their own way of learning. These closed learning objects are qualified to approximate the previous knowledge of the students at the beginning of a blended-learning course.

Open learning objects on this level include exploration assignments. At the beginning of the learning process students get an assignment. In order to solve the problem (and respectively achieve the objectives) they have to explore different topic-related learning materials, generate their own answers or methods of solution and discuss them with other students and the tutor. The quality of the learning process and the achievement of learning objectives should be improved by a process of evaluation. These learning objects are suitable for the workshops and seminars in a blended-learning process, but not in their earlier phases.

Examples:

- An animation of a rack feeder with a task asking the students to identify functional subunits resp. classes with methods and attributes.
- A multiple choice test about concepts of OOP.
- A request to add a class of a user manager to the model of a school kiosk software.

Level 4: Learning Module

A learning module contains one or more tasks, learning or raw assets. It describes one topic with all its pedagogical parameters, including objectives, methodical decisions, previous knowledge of students, sequencing, didactical, organisational and technical context. The complexity is high, the reusability in other contexts low. This is the highest level of aggregation with impact on didactical decisions. The cohesion of the subunits is given by the wider topic of the module. Didactical concepts like cognitive apprenticeship and blended learning are visible in the sequencing of lower LOs on this level. Examples:

- A module about Java coding in a LEGO Mindstorms high rack storage.
- A module about design patterns in a school kiosk software.
- A module about database design in an online news agency.

Level 5: Thematic Web

A thematic web bundles different learning modules with different topics to a curriculum for a predefined graduation - e.g. all courses to reach a master degree at university. The reusability in another context is minimal.

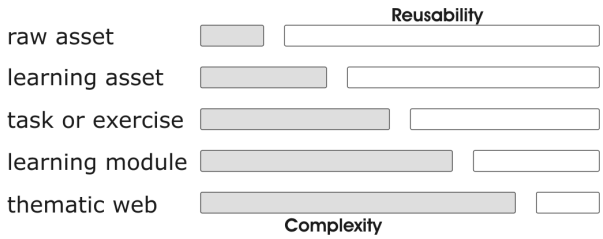


Figure 3: Reusability vs. complexity of LOs.

Reusability as the opposite to complexity is important for the adaptability of the LOs in the evaluation process.

4 Concept of Evaluation

The basic concept of the ILL was subject to a first evaluation during a course at the University of Paderborn in summer 2003. This was a preliminary study - close to the grounded theory from Glaser and Strauss [GS67] - in order to prepare a second one which will be described beneath. Topic of this evaluation was the testing of the instruments and the searching for adequate research questions.

Because of the small groups of students and the impossibility to pre-produce all media and learning objects it is necessary to make the empirical study in various passes. Each part of the study will be followed by a phase of material production for the next part. Currently we are estimating three years of research - including specification of research questions, production of media and learning objects and interpretation of results.

The main research interest is to find out which type of media support (abstraction level, encoding type, sequence and structure of learning objects) will be necessary to optimise students learning success. The components of the study will be questionnaires, group discussions, screen videos, guideline oriented interviews, product analyses, screen videos and also the observation of the students' activities (process analyses).

4.1 Instruments

Questionnaires and Interviews

One objective of the ILL – and above all a condition for the collective work at the case studies – is the adjustment of the students' different practical and scientific knowledge of the students. A grading test has to be made at the beginning of the course to evaluate the previous knowledge of the students in the different relevant areas of software engineering (e.g. practical experiences with UML, coding or design patterns). The students will be sorted by experience levels (beginner, advanced learner, expert) in this area. Assigned to these levels of experience the system will offer the students (mostly) closed learning objects to approximate previous knowledge.[4]

At this early state of the survey the areas of students' deficits are unclear. Because of that it is necessary to add a guideline-oriented interview on this questionnaire at the first rounds.

After that it might be useful to search for patterns (like UML-type, source code type) in order to gain a simple classification of learners. The ambition is to ascertain which type of student can benefit the most from which types of learning objects.

[4] This is a problematical procedure because of lack of time to produce missing media and learning objects.

When the students have finished the course we have to introduce another questionnaire to check the achievement of objectives. Such a questionnaire can (mainly) proof declarative knowledge. To test procedural knowledge a process evaluation is necessary. The questionnaire should also clarify the students' satisfaction and self-assessment in regard to the learning process.

Product Evaluation

During most of the phases of the learning process the students are expected to produce something (e.g. a program or class diagrams). Thus, concrete products, which can be analysed with criteria of professional software engineering, are supposed to exist. These outcomes can conveniently be used to detect (individual) deficits even during (not only after) the course. If such deficits exist new learning objects have to be created - or rather existing objects have to be modified for later rounds. This modification can be made by exchange of media objects about the same subject areas or by converting lower level learning objects in objects of level 3 or higher.

Process Evaluation

Product analyses can't create insights in the process of its creation. But procedural knowledge will be visible in the students' working process.

Screen videos of the students' work with computers can represent a part of the learning process. So in each instance two students are expected to talk about their activities in front of a computer and about each of their simple clicks on the screen.[5] This communication is more important than the mute screen image. A software like Videograph[6] is useful for a precise category-based analysis of the screen videos.

The computer-supported process analysis should be completed by the conscious observations of the tutors.

4.2 Research Questions and Realisation

As mentioned above the evaluation should take three years. The first year should serve the production of learning objects, the expected types of students and a first pass for testing research questions, instruments of evaluation, tools and the learning platform[7]. The second and the third pass will each be realised in a course one year later. Each time only one parameter of the learning design will be changed. We don't want to evaluate the role of the tutors' activities in these courses, because we suppose that the 'fading out' will be the same in each pass.

Research questions which have to be stated more precisely at this time are:

---

[5] The tutor also should stimulate the communication between students because of the methods of cognitive apprenticeship.
[6] http://www.ipn.uni-kiel.de/aktuell/videograph/htmStart.htm
[7] We will use a sTeam server. http://steam.upb.de/en/

- How must learning objects be built for such a scenario in the computer science education at university? How must we integrate the LOs in a learning process which is organised in orientation according to the concept of cognitive apprenticeship?
- Are closed learning objects only convenient at the beginning of the learning process or also in a later phase?
- Which coding types and levels of abstraction are eligible to enhance the learning success in regard to the learning issues (case study) and the common educational objectives of university courses in software engineering?
- Does the insertion of interactive animations and videos assist the learning outcome? - The creation of animations (e.g. in Flash) and videos is very complex and expensive. A pass with only static drawings and pictures without animations and videos will show the difference.
- Which case studies are appropriate to increase the acquisition of knowledge in the area of software engineering? How do they assist the domain-specific and general transfer [Eb96] of knowledge?

References

[ADL04] Advanced Distributed Learning Initiative: SCORM 2004 Content Aggregation Model Version 1.3. http://www.adlnet.org/index.cfm?fuseaction=rcdetails&libid=648, last visit: 2004-08-01, 2004.

[Bl01] Blömeke, S.: Zur medienpädagogischen Ausbildung von Lehrerinnen und Lehrern. Folgerungen aus der aktuellen lern- und professionstheoretischen Diskussion. In: Medienpädagogik, http://www.medienpaed.com/00-2/bloemeke1.pdf, last visit: 2004-07-01, 2001; pp. 6.

[Eb96] Eberle, F.: Didaktik der Informatik bzw. Einer inofrmations- und kommunikations-technologischen Bildung in der Sekundarstufe II. Sauerländer, Aarau, 1996; pp. 201-208

[CB89] Collins, A., Brown, Newman: Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In Resnick, L. B. (eds.) Knowing, learning, and instruction. Hillsdale,. NJ: Erlbaum, 1989; pp 453-494.

[GM00] Gruber, H.; Mandl, H.; Renkl, A.: Was lernen wir in Schule und Hochschule: Träges Wissen? In: Die Kluft zwischen Wissen und Handeln - Empirische und theoretische Lösungsansätze. Göttingen 2000; pp. 11-26.

[GS67] Glaser, B.G.; Strauss, A.L.: The discovery of grounded theory. Chicago 1967.

[Ka03] Kassanke, S.: Ontologiebasierte Strukturierung von Lernobjekten in der Domäne Operations Research/Management Science und Einbettung in ein hypermediales Lernsystem - Konzeption und Implementierung. Paderborn 2003.

[KH01] Koppi, A.J.; Hodgson, L.: Universitas 21 Learning Resource Catalogue using IMS Metadata and a New Classification of Learning Objects. In: Proceedings of ED-MEDIA 2001 - World Conference on Educational Multimedia, Hypermedia and Telecommunications, Tampere, Finland, 2001. Tampere, 2001; pp. 998-1001.

[LOM02] IEEE Learning Technology Standards Committee: Draft Standard for Learning Object Metadata, http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf, last visit: 2004-07-01, 2002.

[MS04a] Magenheim, J.; Scheel, O.: Using Learning Objects in an ICT-based Learning Environment. In: Proceedings of E-Learn 2004 - World Conference on E-Learning in Corporate Government, Healthcare & Higher Education, Washington, 2004 (about to be published).

[Ma03] Magenheim, J: Demands on Digital Media in an Informatics Learning Lab – Medial Aspects of an interactive Learning Environment for Software Engineering. In: Proceedings of the 7th World Multi-Conference on Systemics, Cybernetics and Informatics SCI 2003, Orlando, 2003.

[MS04b] Magenheim, J.; Scheel, O.; Integrating Learning Objects into an Open Learning Environment - Evaluation of Learning Processes in an Informatics Learning Lab. In: Proceedings of WWW 2004 - The Thirteenth International World Wide Web Conference, New York 2004. New York, 2004; pp. 450-451.

[Sp92] Spiro, R. J. e.a.: Cognitive flexibility, constructivism and hypertext: Random access instruction for advanced knowledge acquisition in ill-structured domains. In: Duffy, T.; Jonassen, D. (eds): Constructivism and the Technology of Instruction. Hillsale, NJ, Erlbaum, 1992.

[TH02] Tulodziecki, G.; Herzig, B.: Computer & Internet im Unterricht. Medienpädagogische Grundlagen und Beispiele. Cornelsen Scriptor, 2002.

[Va94] Cognition and Technology Group at Vanderbilt: Multimedia environments for enhancing student learning in mathematics. In: Vosniadou, S.; De Corte, E.; H. Mandl (eds): Technology based learning environments. Psychological and educational foundations. Berlin. Springer, 1994; pp. 167-173.

An Empirical Study of Introductory Lectures in Informatics Based on Fundamental Concepts

Markus Schneider

Institut für Informatik
TU München
Boltzmannstr. 3
85748 Garching
markus.schneider@in.tum.de

Abstract: Before carrying out an empirical study in the area of didactics, one has to clarify the quantity to be measured. Often it is measured to which extent a student has acquired the basic concepts of the respective field. However, in Informatics this is a problem, since the question of the basic concepts is still discussed controversially.

This paper first analyses the introductory lectures in Informatics given from 2000 to 2003 at the Technische Universität München, extracts the learning targets and relates them to the most established fundamental concepts. The resulting sequence of concepts represents one dimension of the matrix of measurable quantities. The other dimension represents the complexity of the respective problem. By relating the elements of this two-dimensional taxonomy to the problems of the final exams of the considered lectures, one get the possibility to evaluate these lectures conceptually. This evaluation is done using the points the students achieved in the respective problems. Thereby, the analysis is performed both for the students as a whole and for male and female students separately.

1 Introduction

The first academic year at the university is perhaps the most crucial phase of the study of informatics. Here most of the students decide whether to abort or to continue the study until reaching an academic degree and the abortion rates in this phase is much higher than in the rest of the study. Therefore, for the first academic year a well-founded didactical concept is necessary not to lose motivated and engaged students yet at the beginning of the study.

Analysing the learning success of three lectures “Introduction to Informatics I/II” the presented empirical study is a contribution therefore.

The three lectures were given at the Technische Universität München: Academic Year 2000/2001 [Br00], Academic Year 2001/2002 [Br02], and Academic Year 2002/2003 [Kn02]. The author of this paper has accompanied these lectures and organized the tutorials associated to the lectures. During the three years, the number of students decreases from about 950 beginners in 2000 to about 350 beginners in 2002. Since the failure rate per academic year is about 30%, the number of students attending the lectures two- or more times is high.

The methodical basis of the here presented study is on the one hand similar to the strategy of the PISA-study (e.g.: [Ku02]); the evaluated problems are classified by their complexity. On the other hand, a classification concerning the learning targets is used. The basis of this scheme is related to the fundamental concepts proposed by Schwill [Sc93]. The resulting two-dimensional matrix of measurable quantities represents the core of the study.

2 The structure of the considered lectures

First, the considered lectures have to be analyzed concerning their contents. The contents define the learning targets and these targets will be related to the above-mentioned fundamental concepts.

The tables 1-3 present the raw structure of the here considered lectures and show the contents of the lectures "Introduction to Informatics I" and "Introduction to Informatics II" in the 3 academic years from 2000 to 2003. Thereby, the order of the subjects represents their temporal order.

Academic Year 2000/2001

Principle of the lecture: The students are confronted from the very first with motivating systems having moderate or high complexity. Thereby object modelling techniques act as a guideline and offer the possibility to decompose this system into small subsystems. Implementing these small subsystems, the students are confronted with the various program paradigms; but also with theoretical topics, like program verification, semantics of recursive functions etc..

Table 1: Academic Year 2000/2001

Lecture	Subject	Concepts	Program Language
Introduction to Informatics I	Object-oriented modelling of systems	Object, Class, Aggregation, Inheritance, Interface	Java
	Algebras	Abstract algebra, concrete algebra, signature, algorithm, text rewriting system	
	Boolean algebra	Boolean algebra	
	Term rewriting system	Term rewriting system, interpretation, correctness	

Lecture	Subject	Concepts	Program Language
	Functional programming	Functional modelling, recursion, conditional expression, correctness, semantics, fixed point theory	
	Imperative/ OO-programming	Assignments, loop, imperative Programming embedded in OO-techniques, array	
	recursive data structures	Sequence, tree, algorithms on recursive data structures	
	Object-oriented programming	Inheritance, abstract class, polymorphism	
Introduction to Informatics II	UML	Class diagram, sequence diagram, use cases	Java
	Software Engineering	Design-Pattern: strategy, adapter, composite-pattern	
	OCL	Detailed design, contracts, AVL-Tree	
	Predicate logic	Predicate logic	
	Program-Verification	Hoare-Calculus, correctness	
	Exceptions	Exception handling	
	Event-oriented programming	Model view controller, observer	
	Automata and formal languages	DFA, Chomsky hierarchy, pumping lemma	
	Machine-oriented programming	v. Neumann architecture, abstract machine-oriented language, basic control structures	

Academic Year 2001/2002

Principle of the lecture: In some aspects, the basic concept of this lecture is dual to the one of the academic year 2000/2001: Firstly, the student develops small and simple systems using Boolean algebra and text rewriting systems and avoiding technical questions. In the course of the year, the student implements system using first the functional paradigm and then the imperative or object-oriented program paradigm. Theoretical questions like verification or semantics are discussed in the context of the respective paradigm.

Table 2: Academic Year 2001/2002

Lecture	Subject	Concepts	Program Language
Introduction to Informatics I	Information and Representation	Interpretation of Information, Boolean Algebra, Interpretation of Terms, Sequence, Formal Language	Gofer
	Algorithms and Algebras	Algorithm, text rewriting algorithms, algebra, algebraic specification, term rewriting systems, important algebraic structures	
	Program Languages	BNF, syntax, semantics	
	Functional programming	Functional modelling, recursion, basic recursive algorithms on numbers and sequences, semantics, correctness	
	Imperative programming	Statement, loop, procedure, Hoare-Calculus, array, reference	Pascal
Introduction to Informatics II	Recursive data structures	Sequences, stack, tree, algorithm on recursive data structures	
	Object oriented programming	Class, object, inheritance, polymorphism	Java
	Coding and Information	Coding techniques, information theory, security	
	Combinatorial and sequential circuits	Normal forms of Boolean functions, arithmetical circuits, combinatorial circuits and DFA	
	Computer architecture and machine-oriented programming	v. Neumann architecture, abstract machine-oriented languages, basic control structures, techniques of addressing, recursive data structures	MI

Academic Year 2002/2003

Principle of the lecture: Again, the principle of this lecture is to pass from simple to complex systems: First, small systems are developed using declarative program languages, whereas complex and object-oriented systems are discussed at the end of the academic year. The design of this lecture has some remarkable features: First, the fundamental principles of the functional program paradigm are introduced, to use these programming techniques for a practical discussion of the topics, cryptography, algebras, automata, etc. Spiral like, the theoretical aspects of these topics are discussed in the second part of the lecture.

Table 3: Academic Year 2002/2003

Lecture	Subject	Concepts	Program Language
Introduction to Informatics I	Overview on functional programming	Functional modelling, recursion, pattern matching, recursive data structure: sequence, trees	OCaml
	Information theory	Information theory, entropy, coding-trees, Huffman coding	
	Cryptology	LZW-algorithm, CRC	
	Algebra	Algebras, text-rewriting system, term-rewriting system	
	Predicate logic	Predicate logic, logic programming, BNF	Prolog, OCaml
	Formal languages and Automata	Chomsky hierarchy, DFA, NFA, Pumping Lemma	
Introduction to Informatics II	Chomsky hierarchy and Automata	PDA, stack-oriented programming, Turing-machine	Postscript, Forth
	Correctness of Functional programming	Partial and total correctness, noetherian induction	OCaml
	Recursive data structures	Trees, AVL-Trees, Algorithm on recursive data structures, B-trees	
	Theory of functional programming	Fixpoints of functions and data structures, Lambda-calculus	
	Object-oriented modelling	Foundation of object modelling, UML	
	Imperative programming	Assignment, Loop, Arrays	
	Object-oriented programming	Class, object, inheritance, polymorphism, design-pattern	
	Correctness of imperative programming	Hoare-Calculus	

Lecture	Subject	Concepts	Program Language
	Machine-oriented programming	v. Neumann architecture, stack, abstract machine-oriented languages, basic control structures	MI

Common Contents of the three lectures

Whereas the order and the intensity of the respective topic vary from lecture to lecture, all three lectures have common topics:

- Abstract and concrete algebras
- Text- and term rewriting systems
- Functional programming in theory and practice; verification of functional programs; semantics of functional languages
- Imperative programming and its verification using Hoare-Calculus
- Object-oriented modelling using UML, and object-oriented programming
- V.Neumann architecture and machine-oriented programming
- Combinatorial and sequential circuits
- Formal languages, Chomsky-hierarchy and Automata
- Information theory and coding

3 Fundamental learning targets and fundamental concepts

To get the fundamental learning targets from the above-mentioned topics, these topics are structured following the fundamental concepts of Schwill.

Fundamental concepts of Informatics

The basis of Schwill's classification is the central task of Informatics: The process of software development. Analysing this process, he derivates the following fundamental concepts:

- Development of algorithms:
  - Design paradigms: Branch and Bound, Divide and Conquer,
  - Concepts of programming: Recursion, Iteration, Indeterminism,
  - Sequential/concurrent processes
  - Evaluation: Verification and complexity
- Structured partition:
  - Modularisation: Top down method, bottom up method, specification, abstract data types,
  - Hierarchical Structures: Trees, Compilation,
  - Detection of orthogonal structures
- Languages
  - Syntax
  - Semantics

The fundamental learning targets of the lectures

Relating the above-mentioned topics to this catalogue of concepts, we propose the following relation:

Table 4: Fundamental concepts and lecture topics

Topic	**Fundamental concepts**
Abstract, concrete algebras	Modularisation
Text-, term rewriting systems	Concepts of programming
Functional, imperative, object-oriented and machine-oriented programming	Concepts of programming, syntax, semantics
Verification of functional-, imperative programs	Evaluation
Object-oriented modelling, UML, modelling of automata	Modularisation, Hierarchical structures;
Information theory, coding	
Formal Languages, Chomsky hierarchy and Automata	Languages
Combinatorial and sequential circuits	

The topics “Information theory”, Combinatorial, and sequential circuits cannot related directly to the proposed concepts. Here, an extension of the concept catalogue or a more differentiated taxonomy seems necessary.

The relation “OO-modelling, .., modelling of automata” □ “Modularisation, Hierarchical structures” seems unusual; one would expect a concept like “modelling”. However, such a concept would be too unspecific, since “modelling” includes all modelling-techniques from graphical modelling to the modelling of algorithms by concrete implementations.

It is natural to assume, that the fundamental concepts in the above topic-concept relation define the fundamental learning targets of the lectures. Therefore, after the first academic year the lecturer of the considered lectures expects from the students a basic knowledge and practical abilities in the following areas: **Structured partition, text and term rewriting systems, Concepts of functional, imperative, object-oriented and machine-oriented programming, Evaluation, Information theory and (formal) languages.**

The matrix of measurable quantities

The evaluation of the lectures is performed using the points the students achieved in the problems of final exams. Since the topics of the problems are associated to the various learning targets, one gets a one-dimensional classification of the problems with regard to the learning targets. Furthermore, the complexity of the problems varies; so, we expand the one- dimensional classification by a second dimension representing the degree of complexity. Each element of the resulting two-dimensional classification is associated to a set of problems and the statistical analysis is carried out for all such sets. Therefore, the two-dimensional classification scheme defines a matrix of measurable quantities.

Table 5: Number of marked problems per category

		Complexity		
		Low	Intermediate	High
Learning Target	Structured Partition	804 (162/642)	2068 (424/1644)	0
	Text-, term rewriting systems	703 (136/567)	570 (95/475)	0
	Functional Programming	800 (161/639)	1631 (1293)	544 (87/457)
	Imperative Programming	800 (161/639)	0	703 (136/567)
	Object-oriented Programming	254 (53/201)	0	0
	Machine-oriented Programming	153 (27/126)	101 (26/75)	0
	Program-Evaluation	0	1165 (235/930)	0
	Languages (Formal)	565 (110/455)	543 (105/438)	0

Table 6 shows the mean value of the points the students achieved in the problems of the various categories; these values are given relative to the maximal reachable points. As above each element of the matrix gives the overall mean value; the mean value for female/male students is given in brackets. The detailed results for the standard deviation can be omitted for the following tables 6-9, since it reaches for all calculations values of about 30%.

Table 6: Mean value of points per category

		Complexity		
		Low	Intermediate	High
Learning Target	Structured Partition	75% (74%/76%)	44% (39%/46%)	-
	Text-, term rewriting systems	57% (48%/59%)	41% (28%/45%)	-
	Functional Programming	57% (50%/59%)	36% (24%/39%)	28% (19%/31%)
	Imperative Programming	46% (40%/48%)	-	26 % (10%/29%)
	Object-oriented Programming	56%(52%/57%)	-	-
	Machine-oriented Programming	56%(52%/57%)	38% (38%/37%)	-
	Program-Evaluation	-	38% (39%/38%)	-
	Languages (Formal)	54% (51%/55%)	40% (28%/42%)	-

The overall results

The (relatively) highest results have been achieved for the learning target "Structured Partition". For the learning targets related to program paradigms (i.e. text-, term rewriting systems, functional programming, … machine oriented programming) we have mean values of about 55 % (low complexity), 38% (intermediate complexity) and 27% (high complexity). It is worth noting, that these values are lowest for imperative programming, whereas the functional paradigm shows better results.

The results for the more theoretical topic "Formal Languages" are in the same interval. The problems on "Program-Evaluation", a rather mathematical topic, reach a mean value of 38 % for problems with intermediate complexity.

To discuss these results it is useful to take into account that a student fails the final exam, if his total amount of points is less than 40 % of the maximal reachable points. Assuming that the average student ought to be able to solve problems with intermediate complexity at the end of the first academic year, one recognizes, that about the half of the students do not achieve the learning targets related to the "Concepts of Programming". The same is valid for the "theoretical" learning targets "Program-Evaluation" and "Formal Languages". Only the "Structured Partition" exceeds definitely the limit of 40%.

The results for the individual lectures

So far, the results for the three lectures as a whole have been presented. Such an analysis was possible, since all three lectures are based on the same learning targets, although the didactical principles of the lectures differ. Therefore, the question arises, whether the above outlined results change dependent on the didactical principle of the lecture. Within the framework of the here considered data such a statistical analysis is not meaningful, since the number of data of the individual lectures gets too low.

Without statistical precision, a basic tendency can be described: The results given in the previous paragraph seems to be valid also for the individual lectures! No principal differences are recognizable!

The results for female/male students

Table 6 shows the numerical results for female/male students in brackets. Dependent on the learning target the results differ more or less. With respect to the learning target "Program-Evaluation", the results show minor differences; male and female have nearly the same mean value. Looking at the learning target "Structured Partition", the differences gets more significant; dependent on the level of complexity the mean values of the female students are 2% - 7% less than the one of the male students. Greatest differences are recognizable for the learning targets "Formal Languages" and "Concepts of Programming": They come for problems with low complexity to 5% - 9% and for those with intermediate or high complexity to 10%-20%. The result for the learning target "Machine-oriented programming" and intermediate complexity differs from that tendency; but since we have only 101(26/75) marked problems of this category, this value is perhaps less significant.

5 Conclusions

Independent on the didactical principle of the lecture, the evaluation of three lectures "Introduction to Informatics I/II" shows the existence of fundamental didactic problems. Greatest problems arise teaching the major learning target of the first academic year, the various concepts of programming. The comparison of the results of the functional and imperative paradigm shows, that the students in the first semester have greatest problems with the imperative paradigm. On the other hand, problems concerning text- or term rewriting systems are better solved than those concerning functional programming are. What may be the reasons of these results?

The syntactic complexity of a program paradigm increases from text/term rewriting systems, over functional to imperative programs. (The semantic complexity behaves perhaps reverse.) The above results suggest, that this higher syntactic complexity is the problem for the students. Algorithms in functional program style are very close to the natural description of a solution, whereas the imperative solution is often shadowed by technical details. This suggests teaching the various program styles in the order of increasing syntactic complexity. Further, it is important to consolidate a program concept, so that the majority of the students have real practical experience with the paradigm. Therefore, the imperative and the object- oriented program paradigm ought to be discussed not until the second semester.

Undoubtedly, a moderate strategy in teaching programming concepts is necessary to increase the learning success. But also the absolute values for the learning targets related to the programming concepts are not satisfying; an average mean value of 38% for problems with moderate complexity is very (too?) low! Are there fundamental problems in teaching programming concepts in the framework of the traditional lecture?

It is a basic principle of learning psychology that the learning success results to a high degree from the self-activity of the student. This is valid especially for the learning of the various programming concepts. However, in the traditional structure of the lecture, the student has a passive role; often the tutorials associated to the lectures do not demand sufficient self-activity from the student. Therefore, it is evident that the student cannot solve programming problems of the final exams having intermediate or high complexity.

Therefore, it is necessary to work out lecture models, which support the students in their self- activity. Such models are currently prepared at the Technische Universität München.

The differences between female and male students particularly in the area of programming concepts are the other major problem resulting from the above analysis. Dependent from the degree of complexity male students reach 10% - 20% higher values than the female students.

What might be the reasons for these differences ?

From studies on school informatics (e.g.: [Fi98]) one knows: The average female student have less precognition on program languages or computer application than the average male student does. Obviously, this fact results in different programming abilities in the first academic year at the university. The only possibility to even out these differences is to support the self-activity of all students especially during the first academic year. Therefore, all efforts point to the same measure: The self-activity of the students has to be supported resolutely! Adequate teaching models incorporating this guiding idea will be discussed in future works.

References

[Br00] Brügge B.: Einführung in die Informatik I, http://atbruegge27.informatik.tu-muenchen.de/teaching/ws00/Info1/.

[Br01] Brügge B.: Einführung in die Informatik II, http://atbruegge27.informatik.tu-muenchen.de/teaching/ss01/Info2/.

[Br98] Broy M.: Informatik, Springer Verlag 1998

[Fi98] Finck N.: Koedukation im Informatikunterricht – Ein Erfahrungsbericht. In: Hyper-Forum für Informatik und Schule, Universität Potsdam, 1998

[Kn02] Knoll A.: Einführung in die Informatik I, Einführung in die Informatik II, http://www6.in.tum.de/info1, http://www6.in.tum.de/info2

[Ku02] Kunter et al.: PISA 2000, Dokumentation der Erhebungsinstrumente, Max Planck Institut für Bildungsforschung, 2002

[Sc93] Schwill A.: Fundamentale Ideen der Informatik. In: Zeitschrift für Didaktik der Mathematik, 1993/1

Empirical Studies as a Tool to Improve Teaching Concepts

Carsten Schulte

Pelizeaus Gymnasium
Gierswall 2
33102 Paderborn
Germany
carsten@uni-paderborn.de

Abstract: Alarmed by the outcome of the PISA-Study in informatics education a discussion about empirical measurements has emerged. What instruments should be particularly developed for this subject?

The ideas discussed in this article are based on the debate on interaction between media and methods and in addition about the general aim of such approaches.

We argue for a combination, which relates learning outcomes to the learning and teaching process itself. This type of empirical studies tries to improve learning environments and can be a valuable addition to empirical studies measuring and comparing learning results.

1 Introduction

In context with PISA, measurements are likely to be seen as a way of comparing educational systems in different places as a whole, by questioning a representative sample survey of all learners in a given group. By ranking the measurements it becomes obvious which ways of learning and teaching are more advantageous than others. But unfortunately, as the discussion about PISA shows, things aren't as easy as that because changing things means adaptation since it is not possible to copy an entire approach. But how to choose those 'things', which are responsible for the effects? Therefore one need not only to know which concepts is the bests but why and how they are working. For example in every 'programming'- 'software development'- or 'algorithmic problem solving'-class a programming environment is used. The teacher chooses languages and tool support in order to meet curriculum needs and in hope to improve learning effectiveness. But how can he be sure to make the best choice (see [McI02])?

Questions like this arouse in all areas where computer is used to support learning and therefore are subject in many studies. A possible research design is to compare two groups interacting with two different tools with a pre- and a post-test revealing which group learned more. But results of different studies are often contradictory, which is due to this type of research design, in which the computer is used as a learning tool or media. Clark concluded in 1983 “media do not influence learning under any conditions” (quoted from [Ko94]). Kozma answered in 1994, “if there is no relationship between media and learning it may be because we have not yet made one” [Ko94, p.7].

The consequence is to study the use of the learning tools more closely. The general research question shifts from searching the best media to searching effective learning environments in which teaching methods, media / tool usage and learning activities are effectively combined. Sometimes this idea is being marketed as blended learning.

The conclusion for empirical measurement is to supplement pre-post designs with instruments to measure the interaction between learners and the learning-tools (or: media). These studies aim to find out and describe successful 'learning patterns': effective user-tool interactions that result in meaningful learning processes.

In the next part of the paper two related instruments will be described: Log file-analyses and the categorization-based examination of screen-videos. Thereafter problems of interpreting results will be discussed and it is argued for a theory-based interpretation.

2 Instruments

The idea of interaction analysis isn’t new. Flanders’s concept uses a fixed schema in which in a fixed interval the observer marks the actual category, which might describe the interval properly (or: that fits the interval). This process can be supported with tools allowing coding videotaped lessons afterward. In these tools, the video can be played repeatedly and stopped at will. Examples of such tools are nud*ist, aquad, catmovie...

2.1 Log file-analysis of a software-development process

Lab-phases or small projects seem to be an important learning method and therefore worthwhile to be studied in order to find out how this method can be used to stimulate learning.

Log files are a means to protocol user actions with the tool. Usually the given command, a time-stamp and the user-input (texts,..) are logged. A Log file gives a summary of the development process. Compared to an analysis of the result (the developed software), a log enables the researcher to gain insight into the failures made and corrected, the difficulties which costs much time to overcome, etc. To do this, the log must be analyzed and interpreted.

I will give an example from a study with two novice-courses in two secondary schools; students were about 16-17 years old. After some month of introduction the task was given to develop software in a group of about four to six persons, which enables two users to play the game memory.

The program was developed using Fujaba. It is an UML-based tool that generates code from class and activity diagrams. The tool logs user actions: Class- or method-diagram, the name, source-code statements, variable declarations, compiler invocations and debugging sessions. The logs were visualized using a simple schema: From left to right the time-flow is given, and from top to bottom the user-actions are shown: On top actions with class-diagrams, then with methods, followed by compiler and debugger-invocations. In each category the names of the classes or methods are listed: the earlier a name is used, the higher it is drawn (see figure 1).

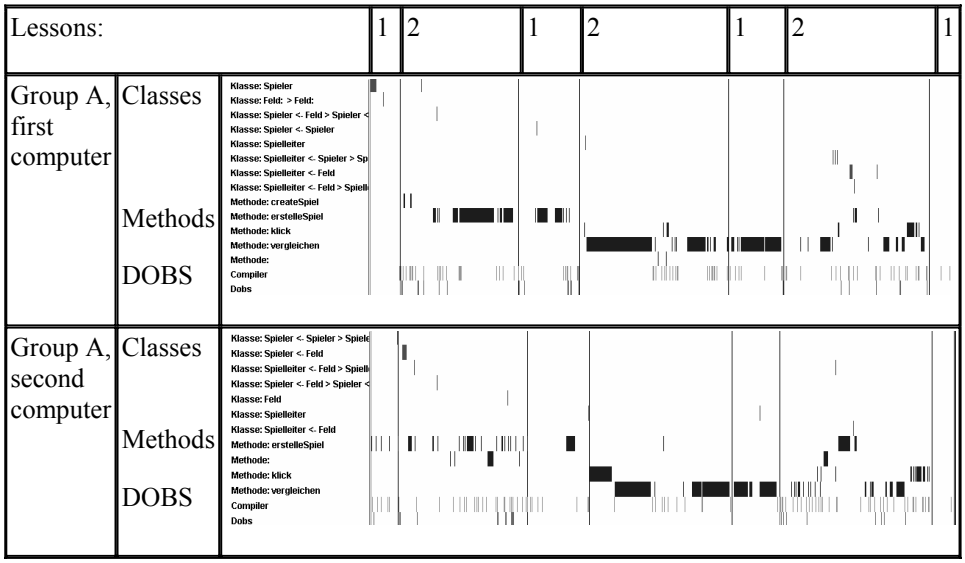


Figure 1: Visualisation of a Log file

This means that a development process starts with implementing a class-model, followed by the implementation of the methods and ends with a debug session would result in a diagram with different blocks from top left to lower right. If there are corrections or additions on the class model being made later, then a mark will appear that is more right than another in the class-section. This way the log-file shows difficulties with the original class-design. It also shows, what methods e.g. took most time to be produced and how many debug-session were necessary. For example the diagram shows, that the class model was quite stable, while two methods consumed most of the development time. An additional analysis of the project shows, that these two methods implement most of the functionality. The group solves implementation difficulties by sticking to the original design, without considering alternative designs, which might have led to shorter method-bodies. A conclusion for the learning process could be to give hints on refactoring strategies, e.g. to split complex methods.

The figure also shows a way to handle the collected log data. It is an aggregation of several thousand items, sorted by time (x-axis) and type (y-axis). Therefore it is easier to make use of the data. Of course, log files are objective and reliable by nature, but what about validity? As log files report user inputs they do not report intentions, they don't even distinguish between purposefully input and simple typing errors. So log files are raw data, which have to be interpreted to make use of the information contained in them. And this results in problems about whether the interpretations are valid.

In general, it seems necessary to distinguish between interpretations as 'normative' and reporting 'possibilities'. Consider the given example. It shows how a team of students was programming a piece of software. And as such, we see a possible way of accomplishing the given task (as we have shown there successful examples). By comparing with other groups the given example could be evaluated as more or less successful. The used time could be the indicator for ranking the successful groups.

Another way of using the results is to compare the empirical development pattern with a normative pattern. This way of using log files has the advantage that in either case some kind of outcome will be generated: maybe the empirical patterns will stick to the normative one, or they won't. Is the measured pattern closely related to the normative one, and then it is being interpreted as a good result.

In the given example a normative pattern may be that expert groups solve easy programming tasks by defining a class structure, implementing methods and a final test run. Therefore a successful pattern would be build out of blocks from the upper left corner to the lower right (see figure). But, what if a group uses some kind of test-driven approach so that in the given figure often entries in the lower line (using debugger) can be found? Due to the given normative frame this approach would be seen as unconventional.

So implicitly made assumptions about good habits in software development should be made explicitly. Such, the norms for interpreting the empirical data are open and can be proven, also.

In a process of empirical studies these norms should be getting more and more accurate – that is one of the main intentions of a theory based approach discussed in the over next section. But before that some conclusive remarks on log files and then the supplement of this instrument with screen videos should be given.

Analysis and interpretation of log files rely on implicit or explicit assumptions about 'good' development process for novices and therefore couldn't be separated from the processes taught to the students, which therefore should be evaluated, too. Second, a process useful for a given task might not be working in another problem domain. Third, in a learning and teaching process – which software development processes are in our context – errors may be a good opportunity to learn.

So it might be useful to support log files with an instrument allowing a closer look on the aims and intentions learners are influenced by while working on their software development task. Such an instrument is to videotape their work including discussions between group members.

2.2 Category-based evaluation of screen-videos

With capturing tools like camtasia screen videos are recorded automatically capturing the screen the learners see and, additionally, their utterances and discussions.

On the one hand, this instrument can supplement log files. For example movements of a mouse cursor aren't logged but they contain information, also: e.g. a programming task might last longer than estimated simply because students weren't accustomed to the IDE and had to search menus for commands to use.

On the other hand it supplements information with the things students say or discuss during work. With this information it is possible to give a more detailed interpretation of the log file patterns. From the utterances it should be possible to say whether there was a plan or an idea the students where following or there wasn't.

In order to lead them to think aloud in a natural way, one can let them work in groups or pairs on one computer.

In the given example, students often discussed their designs, leading to pauses, but only very seldom one member of a group engaged in 'trial-and-error'-programming, although a first impression of the log files might look like trial-and-error. There are several methods to make videotapes searchable for patterns. Most of them rely on transcribing the video. For example all utterances can be transcribed and therefore be searchable.

	Working Style	
	Number	**%**
Trial and error	2031	17,2%
Clear intention	4600	38,9%
Hint from outside person (teacher, other group)	2263	19,1%
Hint from an older project	100	0,80%
Group discussion	2837	24,00%

Figure 2: Working styles

Alternatively the use of fixed codes allows using statistical operations to analyze the data collected by videotaping. The codes represent categories that might be interesting. Here, for example, a code could mean 'purposeful change of a given method body' vs. 'change made by random, just to explore what happens'. Having coded all videos one can just count them.

In this case, all categories were coded by a fixed interval of 10 seconds. This interval-based scheme allows counting out. Alternatively one can count them turn-by-turn. This was done with the log files (see figure one), therefore showing the length of a measured category more accurately. Note, however, that it is more useful to code all data by one of the two possible methods in order to make them statistically comparable. Coding can be done in real time, while transcribing takes two to three times longer.

3 Interpretation of the results

What might be fruitful results of such studies? For example a set of different programming styles: Styles A, B, C. These styles then are mapped to the quality of programs indicating that lets say style A seems to produce better programs then style C. Romero et.al. have chosen this approach to gain a model of program comprehension and debugging expertise, using log files, screen recordings, verbalizations and a category-based coding strategy. To interpret the results: „a cluster analysis can allow us to categorize groups of programmers according to their displayed strategies and to compare this categorization with their performance data. This categorization can also be complemented with the findings of the quantitative analysis. In this way, a model of program comprehension and debugging expertise in terms of behavior and strategy can be empirically derived“ [RBC04, p. 10].

Such models or sets of different styles are useful for a teacher to describe the level of expertise of each learner. Maybe the teacher even can see specific learning needs. Thereby results of empirical studies are used as tools to describe learner types and to develop specific learning concepts for them.

But this means to use such a ‘model of program comprehension’, or a set of empirically derived programming styles in a normative way: aim of a learning environment then is to train learners to follow style B …

And, a learning environment is seen as successful if it ‘produces’ students that follow a certain style regarded as expertise style.

There seem two objections to this kind of interpretation (or: use of) empirical studies: The direct mapping to a learning environment assumes that one approach fits for all students, but there isn’t such thing as the best teaching method for all [Bl03]. Second, such models or ‘type systems’ describe empirically observed patterns which maybe not the best possible patterns.

Another problem is, how to use such comparisons to develop more successful learning environments? It may be useful to supplement these results with another method of interpretation so that the captured data could be mapped closer to the learning process itself.

4 Conclusion: theory based approach

So far, empirical studies are described here as a means to categorize, mark, evaluate or grade a learning environment. But this means, that the empirical results are only indirectly helpful to improve a given learning environment / teaching concept.

So a method to use empirically studies of the types described above more directly to improve a teaching concept should be useful and will be outlined in this section. The idea is to use a given learning theory as a guideline for developing and evaluating a learning environment. This guideline helps to establish a closer connection between certain results and specific aspects of the teaching concept – and thereby it is possible to develop fine-grained improvements.

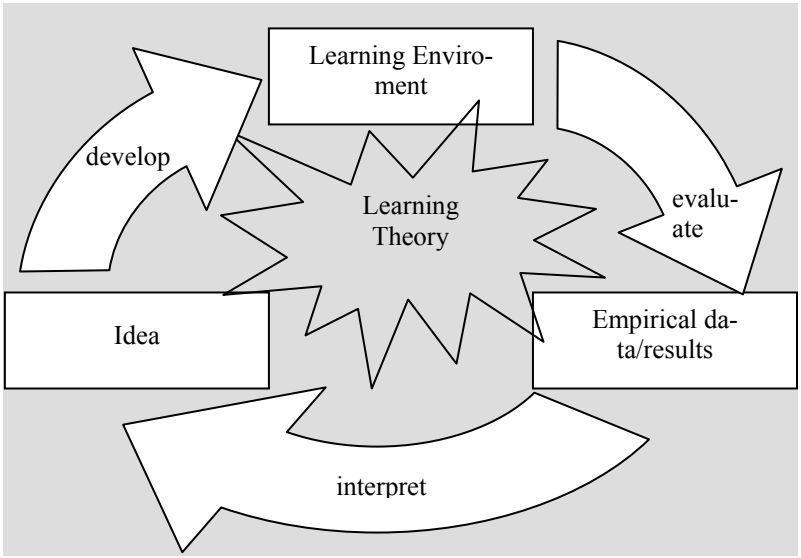


Figure 3: theory based approach

In order to get specific handles to improve an empirically studied learning environment, the studied learning environment is based on a learning theory, which predicts certain results. For example the theory gives hints how to develop a learning environment: for example to situate learning in order to support students ability to use gained knowledge for 'real' problems. So in the learning environment a real world example is used to situate the tasks. Given a cause-effect-structure like this, the empirical data can be used to compare results to the intended ones: If students fail to use knowledge to solve a real problem in a test, the learning environment has failed to situate learning. Maybe the given examples should be changed, …

The general approach is to look for differences between given and estimated results, which can lead to a closer look. Thereby it should be possible to relate certain failures or flaws to certain aspects of the learning environment.

Of course, the results have to be interpreted. In the example above maybe to change the example isn't the right solution. Again, here the theory should help, because it stresses certain aspects of the learning environment as especially important. These crucial aspects should be regarded while designing the instruments for the empirical evaluation. And, of course, even a step earlier: while designing the environment itself. Using this general approach, a detailed comparison between intended and empirically measured results should be possible.

I will try to demonstrate this approach be discussing another example, based on the empirical study already mentioned in earlier sections. The example is the evaluation of a learning environment for introducing students to object oriented modeling based on cognitive apprenticeship (CA). CA suggests introducing novices to a topic like it is done in apprenticeship: One idea is to let students participate in solving realistic tasks with the help of an expert which let students solve easy sub-tasks on their own. The expert demonstrates how to solve and use them in the context of the realistic task. It's crucial to avoid working on isolated sub-tasks without the given context and to give novices an overall understanding of the subject domain. In the context here, novices should participate in the developing of an object-oriented program. So they should understand the difference between classes and objects. Given an overall understanding and a training in using tools, programming language and modeling technique students should be able to develop a software on their own – if, as CA claims, they have gained a general understanding which enables them to use knowledge and skills together in order to solve the task.

Empirical results show that students were able to solve the task, but they weren't able to explain the difference between classes and objects. Mean score in question 'Explain the concepts 'class', 'object' and their difference) was 0.5 from a possible range from 0-2 points.

So it seems, they didn't understand the technology they have used. Maybe they solved the task just by trial-and-error. But (see table working style) trial-and-error was used far less than a purposeful working style. In this case, learning theory helps to explain the empirical data: In the learning environment graphical (UML-based) tools and programming language were used. This lead to a merely visual understanding of object oriented concepts. Classes and objects were visually described. An examination of the videotaped lessons also showed that verbal explanations of teachers and students often used informal language. For example the word 'class' and 'object' were not consistently used. According to Meyer's theory of learning with multimedia, which is based on Paivio's dual coding theory, pictures and word lead to different mental models [MMR00, figure 1]. In this article, Moreno and Mayer derive some principles for the instructional use of multimedia, which can be adapted to this case: students learn better, when verbal information is presented auditorily as speech rather than visually (Modality principle); students learn better, when visual information is presented simultaneously to verbal information (Redundancy principle). As a consequence, the learning environment was (very slightly) changed according to these principles: So in the next course, students were regularly asked to explain visualizations (UML diagrams or visual source code) to the class. The teacher took care that the students made correct use of the terms (especially class and object).

After the course the same questionnaire was given to the students. Mean score (question class and object) changed from 0.5 to 1.7 points, which can be regarded as a significant improvement.

So, empirical studies can and should be directly used to improve the quality of learning environments. To do so, standardized instruments are helpful to compare results.

References

[Bl03] Blömeke, Sigrid: Lehren und Lernen mit neuen Medien – Forschungsstand und Forschungsperspektiven. Unterrichtswissenschaft 1, 57, 2003.

[Ko94] Kozma, Robert: Will media influence learning. Reframing the debate. Educational Technology Research and Development 2,42, 1994.

[McI02] McIver, Linda: Evaluating Languages and Environments for Novice Programmers. PPIG 2002. (www.ppig.org)

[MMR00] Moreni, Roxane; Mayer, Richard E.: A Learner-Centered Approach to Multimedia Explanations: Deriving Instructional Design Principles from Cognitive Theory. IMEJ, 2,2, 2000. (http://imej.wfu.edu)

[RBC04] Romero, Pablo; du Boulay, Benedict; Cox, Richard; Lutz, Rudi and Bryant, Sallyann: Dynamic rich-data capture and analysis of debugging processes. PPIG 2004.

[Sc04] Schulte, Carsten: Lehr-Lernprozesse im Informatik-Anfangsunterricht. Theoriegeleitete Entwicklung und Evaluation eines Unterrichtskonzepts zur Objektorientierung in der Sekundarstufe II. Dissertation, Paderborn, 2004.

Philosophical Aspects of Fundamental Ideas: Ideas and Concepts

Andreas Schwill

Institut für Informatik
Universität Potsdam
August-Bebel-Str. 89
14482 Potsdam

schwill@cs.uni-potsdam.de
http://www.informatikdidaktik.de

Abstract: We consider the term „idea“ from a philosophical point of view. In particular we are interested in the concept's origins, its relevance to human thinking and in particular in its pedagogical value for computer science lessons in schools as well as universities. Since the concept of fundamental ideas in computer science has been seized, extended, and reviewed by other authors and applied to lessons, often with a different understanding of the defined terms, we wish to explain some of the objectives of the approach in more detail and in particular clarify the relation between concept and idea in order to provide a common understanding of the relevant notions.

1 Introduction

In this paper we extend long-term considerations on fundamental ideas of computer science first published in [Sc93] (see [Sc94] for a revised version in English and [Sc97] for an extended abstract). While the earlier papers focus mainly on the definition and background of fundamental ideas, here we try to consider the notion „idea“ from a philosophical point of view. In particular we are interested in the concept's origins, its relevance to human thinking and in particular in its pedagogical value for computer science lessons in schools as well as universities. Since our concept of fundamental ideas of computer science has been seized, extended and reviewed by other authors [B98, Mo03a, Mo03b], applied to lessons, and included into curricula [BEL02] often with a different understanding of the defined terms, we furthermore wish to explain some of the objectives of the approach in more detail. In particular the notions of concept and idea are often mixed-up. As a consequence sometimes concepts, like the Turing machine, or subjects of computer science, like date bases, are erroneously denoted as ideas. So this paper attempts to clarify the relation between a concept and an idea in order to provide a common terminology of the relevant notions.

For the rest of the paper we assume that the reader is familiar with [Sc93] resp. [Sc94] or the relevant sections in [SS04].

2 Philosophers on ideas

By an idea one often denotes a plan, a thought, an imagination or an

> „object of a non-sensory intellectual perception, in which its nature may be recognized“[1] [Enzyklopädie der Philosophie].

In particular in philosophy the concept of ideas has a long tradition. Later it has been reconsidered in the field of education [B60]. But while philosophical papers often try to make the notion as precisely as possible, many pedagogical papers on fundamental ideas show a considerable deficit, at least for a formal scientist, if it concerns an exact clarification of properties and attributes of ideas. A typical approach is to give few examples for ideas and one or two criteria. As to the author's knowledge the philosophical contribution to clarify the notion and pedagogical relevance of ideas has not been analyzed so far. Even in an earlier paper of Schweiger [Sc92] these aspects are omitted although the title might suggest the contrary.

In the following we want to summarize the most important philosophical considerations of the notion of an idea, since they give interesting indications for the pedagogical value of an idea in computer science education.

The notion of an idea dates back to Plato and has been reconsidered later by Descartes, Locke, Hume and Berkeley, however with a different meaning. It was not until Kant who returned to the Platonian notion of idea. In the following we wish to analyze the main properties of ideas as far as they help clarify the notion.

2.1 Plato (427-347 BC)

Plato has the most abstract vision of ideas. Besides reality they form a class of its own, a cosmos of pure objects (a higher reality) located at a celestial place, i.e. independent of human thinking. Every real object is just an imperfect copy of the ideas behind it. Typical Platonic ideas are the idea of a circle, of a chair, of justice, of the good, of beauty which only have imperfect copies in reality. The function of ideas is normative: they provide guidelines that humans might approach.

All Platonian ideas are innate and thus the basis of human perception. Every human being, at the time of birth, has the chance to have a short look into the heaven of ideas.

---

[1] „Gegenstand einer nicht-sinnlichen intellektuellen Anschauung, in der sich dessen Wesen zu erkennen gibt.“

Then perception occurs not by acquisition of a new idea but by recalling ideas acquired earlier.

In summary: Ideas are certain abstract ideal imaginations of objects that are not available in reality but that act as models for human behavior or real objects and thus define objectives which humans try to achieve approximately. This normative aspect which we consider very important for didactic issues will be discussed again later.

With Descartes and later Locke, Leibniz, Hume and Berkeley ideas lose their cosmological attitudes; yet, the term „idea“ eventually develops to denote every act of thinking and becomes a synonym for „concept“. It was Kant who once again distinguishes between these two terms.

2.2 Descartes (1596-1650)

Also Descartes considers ideas as images but in contrast to Plato he exchanges archetype and image:

Plato: ideas → real objects.
Descartes: perceived objects → ideas.

Now real objects are no longer imperfect copies of a Platonic idea which are already (prenatally) in the (sub-)conscious mind but conversely ideas are images of objects perceived by the conscious mind. Furthermore Descartes significantly extends the meaning of idea and uses „idea“ as a superordinate concept for all objects in the mind, i.e. „everything ... that is directly perceived by the mind“. By admitting that the mind is able to perceive and to develop new ideas by its own Descartes negates that all ideas are innate. He distinguishes between innate ideas, perceived ideas and ideas that are developed by the human him- or herself. Here the innate ideas may not be considered a fixed collection of objects of the mind but as abilities for acquiring and developing ideas (Fig. 1).

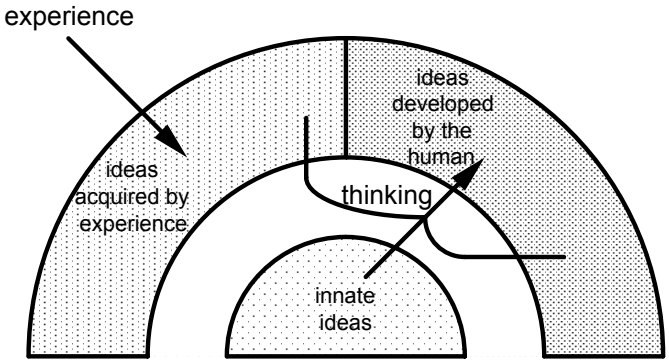


Fig. 1: Ideas and their role according to Descartes

2.3 Locke (1632-1704)

A definite breach of Plato's doctrine of ideas is done by Locke, a co-founder of empirism. In [L] he analyses

> „the original of those ideas, notions, or whatever else you please to call them, which a man observes, and is conscious to himself he has in his mind; and the ways whereby the understanding comes to be furnished with them" [L: I,1,3].

and distances himself from Plato by defining „idea" as a notion that

> „serves best to stand for whatsoever is the object of the understanding when a man thinks" [L: I,1,8].

Here „thinking" is very widely used and not only covers the intrinsic operations of the mind but also sensory perceptions, senses of pain, remembrances and concepts, for instance warmness, movement, or color.

Another novelty of Locke's is the absolute rejection of innate ideas. In fact for him the mind is a „white paper, void of all characters, without any ideas" [L: II,1,2], a tabula rasa in which all experiences are imprinted as in a wax tablet. These experiences are subdivided into two classes, *sensations* and *reflexions*. In this framework knowledge is acquired based on „the perception of the agreement or disagreement of any of our ideas" [L: IV,3,1].

Furthermore Locke develops a detailed classification of ideas. He distinguishes between simple elementary and complex ideas. Softness, warmness, color are *simple ideas*. *Complex ideas* are subdivided into ideas of *substance*, *relations*, and *modes*. Another classification concerns the operations by which the mind is able to combine simple ideas to complex ideas: *composition*, *comparison*, and *abstraction* (Fig. 2).

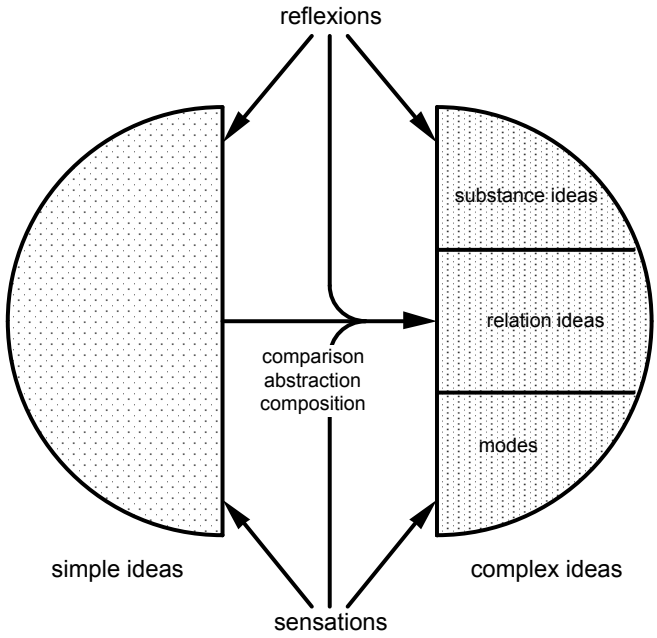


Fig. 2: Ideas and their operations according to Locke

Summarizing Locke's wide use of the notion of ideas containing any operations of the mind is not suitable for our educational purposes. On the contrary with respect to our didactic objectives we wish to distinguish sharply between scientifically correct and formalized concepts on a high intellectual level and ideas which underlie, explain, and motivate the concepts and are supposed to be more simple and easier to learn. For our considerations we can profit from Locke's observation that all ideas are based on a certain set of simple ideas and hence may be structured according to their complexity. Of particular interest are the operations of composition and abstraction which may lead to a hierarchical structure of more and more complex and abstract ideas.

2.4 Leibniz (1646-1716)

The proof that there are no innate ideas is not always coherent in Locke's work and refuted by Leibniz who defends Descartes' conception against Locke. Furthermore he restricts Locke's widely used term of idea. Leibniz considers an idea not to be a certain act of thinking but an ability[2]; traces of impressions in the mind are not ideas[3]. Leibniz

[2] "Die Idee besteht für mich nicht in einem bestimmten Akt des Denkens, sondern in einem Vermögen, so daß wir die Idee eines Dinges haben können, selbst wenn wir nicht wirklich darüber nachdenken, doch bei gegebner [sic] Gelegenheit darüber nachdenken können." [Le]

[3] "Spuren von Eindrücken in unserem Gehirn sind keine Ideen." [Le]

distinguishes between ideas and concepts but classifies concepts as part of the ideas. He also accepts the existence of innate ideas.[4]

2.5 Hume (1711-1776)

Hume's merits consist among others of a more detailed analysis and classification of processes in the mind as well as a detailed structuring of the processes that guide the development of ideas.

While Locke refers to all contents of the mind as ideas, Hume distinguishes between impressions and ideas, where impressions are vivid and strong and ideas are weaker and less vivid. They arise

- from a sensory perception (*sensation*), e.g. by hearing, smelling, tasting, seeing, for instance the impressions of loud, hot, bright,
- from *reflexion*, e.g. emotions, for instance the reflexions of joy, pain, hate,
- by recalling ideas (*memory*) acquired earlier, for instance remembering an earlier incident.

Impressions are either *simple* or *complex*, i.e. composed from simple impressions. An example of a simple impression is the color yellow, while the impression of an apple is complex and may be subdivided into simple elementary impressions like red, round, sweetish smell.

Ideas are derived from impressions, but they lose their strength and vividness during that process. In analogy to impressions they may be subdivided into simple ideas and complex ideas where there is a bijective relation between simple impressions and those simple ideas which they are originally derived from. Complex ideas are not preceded by corresponding complex impressions. Rather they are established using *imagination* that has the ability to combine simple and complex ideas in order to obtain new ideas or to reorganize ideas. So there is a causal relation between ideas and impressions in the Aristotelian sense that there is nothing is in the mind what has not been in the senses before (Fig. 3).

Hume admits the existence of innate ideas. For instance, reasoning from experiences in the past to incidents in the future is based on an innate idea: otherwise creatures would have very small chances to survive if this idea were only established during a longer process of experience and reflexion.

---

[4] "In dieser Weise sind uns die Ideen und Wahrheiten eingeboren als Neigungen, Anlagen, Fertigkeiten oder natuerliche Kraefte, nicht aber als Taetigkeiten, obgleich diese Kraefte immer von gewissen, oft unmerklichen Taetigkeiten, welche ihnen entsprechen, begleitet sind." [Le1]

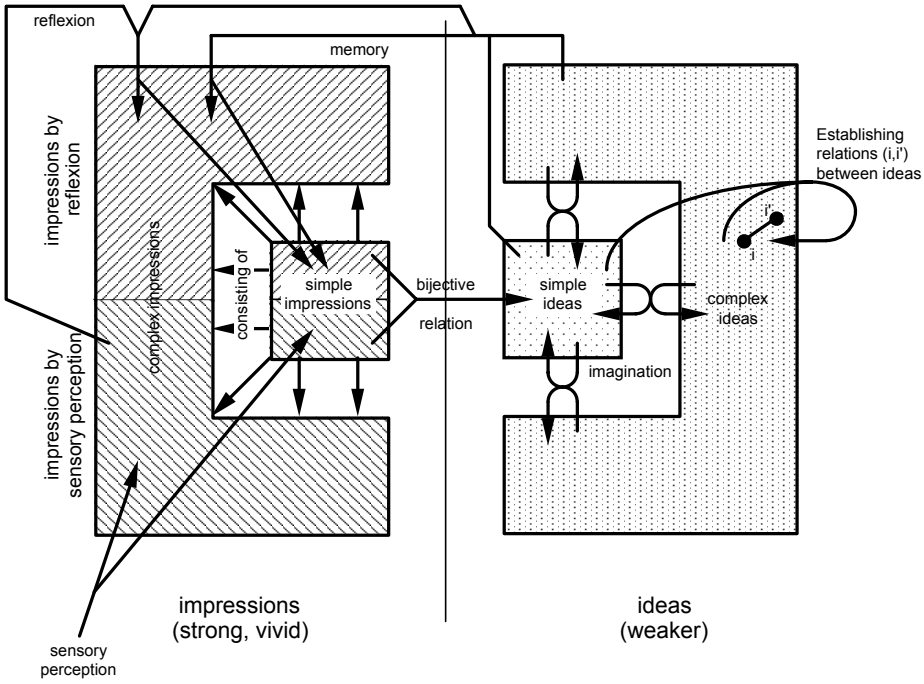


Fig. 3: Operations on impressions and ideas according to Hume

Conclusion: For our purposes Hume's considerations are useful only in two respects: On the one hand ideas may be – as also shown by Locke – distinguished according to their complexity and structured hierarchically where the number of compound operations imagination has to perform may serve as a measure of complexity or hierarchical level. Moreover all ideas rely on certain basic ideas. On the other hand ideas developed earlier are included into processes of the mind by reflexive self-perception. We may assume that they control and influence this process to some degree. We remark that Hume as well as Locke unfortunately do not distinguish between an idea and a concept.

2.6 Kant (1724-1804)

Kant's analysis of human cognitive processes gives us major innovations and precisions of the notion of ideas and their function. In the following we focus on his main work „The Critique of Pure Reason" [K].

2.6.1 Overview on „The Critique of Pure Reason"

Kant continues the rationalistic considerations of Leibniz. Let us return to Locke's metaphor that all experiences are engraved into a wax tablet that is completely empty when the human is born. Leibniz had already refuted Locke by argumenting that the wax

tablet has to have a particular structure in order to be able to store experiences at all, since experiences can only be made if they are carved into the tablet. Signals that reach the tablet on another way, be it optically or acoustically, are not recorded and not turned into experiences. So there has to be a certain match between signals that carry experiences on the one hand and the sensors as well as the perceptional processes of the mind on the other hand in order to make experiences at all. These properties that sender and receiver have to have in common are analyzed by Kant in the „Critique of Pure Reason" concluding that it is not experience that determines perception but conversely that experience to a large extent is a product of our mind. All structures that we may find in our experiences were prescribed by the mind:

> „The understanding does not derive its laws (a priori) from, but prescribes them to, nature.".[5]

Hence, objective perception is not possible since all experiences are formed and modified by the mind before they become perceptional material. According to Kant the structures that enable and guide perception are divided into three levels with increasing distance from the objective reality:

- the *pure forms of perception*[6] space and time which influence the perception of every object. So the mind does not perceive the real objects (the „Ding an sich") but only their appearances which have been modified by pure forms of perception.

- the *pure* concepts of understanding[7], so-called *categories*, like quantity, causality, possibility, necessity, which form the notional framework for every form of human thinking; they are pure forms for constituting experiences.

- the *pure concepts of reason*, so-called *transcendental ideas*, like soul, world, God, which as idealized objectives establish the methodology for extending knowledge. This aspect will be analyzed in more detail as it is interesting for our educational purposes.

2.6.2 Kant on ideas

Kant turns away from his predecessors' and contemporaries' notions of ideas and returns to the Platonic concept but with some major modifications. First he carefully distinguishes between ideas, perceptions, imaginations, notions, concepts etc. He assigns ideas to the category of representations and classifies as follows [K, B376/377]:

- A representation with consciousness is a *perception* (perceptio).

---

[5] "Der Verstand schöpft seine Gesetze nicht aus der Natur, sondern schreibt sie dieser vor" [K1, §36]

[6] „reine Anschauungsformen"

[7] „reine Verstandesbegriffe"

- A perception which relates solely to the subject as the modification of its state is *sensation*.
- An objective perception is *knowledge* which may either be an intuition or a concept.
- An *intuition* is singular and relates directly to an object.
- A *concept* is indirectly related to an object in terms of an attribute that several objects may have in common. We may distinguish empirical and pure concepts.
- A pure concept has its origin in the understanding alone and is called *notion*.
- A concept that exceeds the possibility of experience is called an *idea* or concept of reason.

Kant defines „idea" explicitly at many different places in his works, namely as

- a concept in such a perfection that cannot be found in experiences,[8]
- a necessary concept of reason which has no corresponding object in sensory experience.[9]

So ideas are results of pure thinking and cannot be found in experience at least not in the imagined form but solely as imperfect copies, so were the Platonic ideas.

2.6.3 What is the methodological relevance of the Kantian ideas?

Every human cognition begins with a sensory experience which is modified by the pure forms of perception (time and space). Afterwards these experiences are structured conceptionally along the question „what is" using the pure concepts of understanding (categories). Finally, the pure concepts of reason (ideas) organize the acquisition of concepts of understanding along the question „why holds ...?" or „how are things related?", and they define the objective for the mind to search for a maximal unity (systematic) of the perceived material, thus setting a direction for this search. So roughly (Fig. 4):

- pure forms of perception modify perceptions to intuitions
- categories organize intuitions towards concepts

---

[8] „Eine Idee ist nichts anderes als der Begriff von einer Vollkommenheit, die sich in der Erfahrung noch nicht vorfindet. Z. E. die Idee einer vollkommnen [sic], nach Regeln der Gerechtigkeit regierten Republik!" [K2, VIII, 196]

[9] „notwendiger Vernunftbegriff, dem kein kongruierender Gegenstand in den Erfahrungen gegeben werden kann" [K, B384]

- ideas guide the mind to extend concepts towards a total uniformity.

Which ideas are, as driving forces, responsible that we aim at a maximal uniformity and systematic of all insights? This question has to be answered for any science and the results have to be carried over to school and university curricula in order to teach students a correct view of the respective science. For computer science a first attempt is contained in [Sc93,SS04].

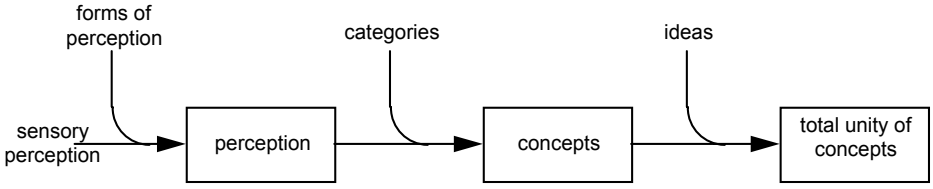


Fig. 4: Cognitive processes as seen by Kant

2.6.4 The regulative function of ideas

Kant has analyzed very carefully the use of ideas of reason versus categories of understanding. While concepts help understand facts by constituting knowledge (by definitions, theorems, proofs), the function of ideas is *regulative*. They guide the mind to extend its knowledge, by searching for suitable experiences, towards objectives that are described by ideas. However, a problem arises if one tries to obtain results about the ideas themselves, i.e. not using them regulatively but constitutively like concepts. Considering ideas as objects instead of objectives and trying to draw conclusions or making proofs about them, inevitably leads to contradictions. Kant proves that by showing that both the assumption of the truth of an idea as well as its negation lead to a contradiction. The reason for this lies in the attempt to obtain results on ideas that lie beyond experience, an inherent property of ideas as shown earlier in this paper.

Example: Let us consider the idea of finding a primal incident initiating the beginning of time (big bang theory). Assuming time has a beginning as well as it has not leads to a contradiction, because we try to use the idea of finding the first causal incidence in a constitutive way as an assertion instead of using it in a regulative way as a concept of desire.

In summary: Ideas are idealized imaginations which objectives are attached to that may not be experienced. However, they guide the human impulse to research and instruct the mind to extend its knowledge towards these objectives possibly without ever reaching them. In [SS04] we have denoted this property of ideas the *goal criterion*.

Now we wish to consider three aspects of this criterion in more detail, since they give hints why ideas have a particular relevance for science and education:

- the *methodological aspect* of ideas is oriented to science. It covers the property of ideas to set up rules, principles, methods and schemes for acquiring knowledge. This aspect seems particularly relevant for computer science that appears to be an „engineering science of the mind“ or a „science of humanities and engineering“[10] [B74] or an „application-oriented science of methodology“[11] [CS01] and always combines research on its objects with research and further development of its methods. So what are the methods of computer science in terms of fundamental ideas?

- the *psychological aspect* covers the motivating attributes of ideas: Their dynamic and process-oriented character is the driving force that activates humans to do research. So, if based on ideas, lessons become more transparent and meaningful for students because they gain along with scientific knowledge answers to questions like „what do I wish to achieve?“ or „where do I want to go?“. So their activities obtain a direction and may be understood as steps on a scientific way towards an objective.

- the *normative aspect* covers current objectives of scientific research and makes a contribution to clarify the paradigm of computer science in the sense of T.S. Kuhn [K62] who defines a paradigm to be a consensus of the scientific community, i.e. a collection of solutions to concrete scientific problems that the community has come to accept. Computer science, even with a historical background of some 50 years, still develops dynamically and has not elaborated a generally-accepted paradigm[12]. On the contrary continuous paradigm changes are announced. The normative aspect of fundamental ideas will be discussed elsewhere in more detail.

3 Ideas and concepts

Several times in our previous considerations we have implicitly distinguished between idea and concept. Now we wish to separate these terms more carefully. In particular for the didactic purposes in [Sc93,SS04] this separation is essential. Consequences of a mix-up of these two terms may be seen from aberrations in mathematics education in the past. In the sixties approaches to redesign mathematics education based on J.S. Bruner's concept of fundamental ideas [B60] lead to a fundamentalist reorientation towards Bourbaki's mother structures and eventually to the integration of set theory in primary school. This development which may retrospectively be regarded as failed based on a misinterpretation of Bruner's principle to focus on structures of science and on an incorrect identification of concept and idea.

The following table summarizes main aspects of ideas and notions.

---

[10] „Geistes-Ingenieurwissenschaft“ or „Ingenieur-Geisteswissenschaft“

[11] „anwendungsorientierte Methodenwissenschaft“

[12] On the other hand P. Wegner [W83] concludes that for an interdisciplinary science as is computer science peaceful coexistence of several paradigms is in fact a hint for scientific maturity.

concept (in German: Begriff)	**idea (in German: Idee)**
Concepts cover the permanent aspects of an object and its essence; they describe what an object is.	Ideas postulate a certain essence of objects; they describe what an object should be.
Concepts emerge from objects by grasping their essence, i.e. by abstraction of essential properties of different single objects and unification.	Ideas precede objects as an eternal perfect model. Objects (even if imperfect) emerge by concretisation of ideas.
Objects a concept is derived from originate from experience.	Objects, which an idea is underlying, occur only as imperfect copies in experience.
Concepts are more general and therefore poorer, less concrete and less varied with respect to their meaning, since they are abstractions of all single objects that belong to the concept.	Ideas are richer and more perfect than the single objects imperfectly cloned from ideas.
Concepts are static and product-oriented. They are snapshots of a learning or research process but do not clarify this process, its starting point or its aim.	Ideas are dynamic, vivid and describe a methodical process, i.e. a path that is paved with ideas, as well as origin and objective of this process.
Concepts are operands of thinking.	Ideas determine which operations of thinking on which operands are performed.
Concepts assert facts (constitutive aspect).	Ideas express (possibly unsolvable) tasks to establish facts (normative/regulative aspect).
Concepts structure cognitive material on the level of understanding by unifying and organizing a multitude of experiences.	Ideas control the process of understanding on the level of reason and determine how and in which direction to extend knowledge.
Without concepts science is not possible, since one cannot establish a systematic relation between the surge of single possible and real experiences.	Without ideas science is possible but the will to do it is missing. Only ideas give the motivation to create concepts, to aim at insights and to extend them in a certain direction.

4 Conclusions

In this paper we have contributed to a unique and common terminology in the field of fundamental ideas of computer science, a task that is inherent for doing science but far from being done in the field of didactics of computer science or even in computer science itself. We have compared the two notions „idea“ and „concept“ and collected characterizing properties from philosophical works over the centuries which hopefully may clarify some of the recent discussions and interpretations of this didactic approach and lead to a correct use of the terms in the didactic discussion. But while the technical effort is tremendous the results appear tiny and unimportant, yet the approach shows that gaining standards in terminology, not mentioned standards in educational objectives or curricula, is a long-term process in particular for a non-formal science like didactics.

References

[B74] Bauer, F.L.: Was heißt und was ist Informatik? IBM Nachrichten 1974, 333-337.

[BEL02] Bieber, G.; Ebner, R.; Lösler, T.; Schwill, A.; Thomas, M.; Vollmost, M.: Rahmenlehrplan Informatik – Wahlpflichtbereich – Sekundarstufe I, Ministerium für Bildung, Jugend und Sport Brandenburg, Wissenschaft und Technik-Verlag, 2002.

[B60] Bruner, J.S.: The process of education, Cambridge Mass. 1960.

[B98] Baumann, R.: Fundamentale Ideen der Informatik – gibt es das? In (B. Koerber, I.-R. Peters, Hrsg.) Informatische Bildung in Deutschland – Perspektiven für das 21. Jahrhundert, LOG IN Verlag GmbH Berlin 1998, 89-107.

[CS01] Claus, V.; Schwill, A.: Duden Informatik, Bibliogr. Institut 2001.

[K] Kant, I.: Die Kritik der reinen Vernunft, 1781.

[K1] Kant, I.: Prolegomena zu einer jeden künftigen Metaphysik die als Wissenschaft wird auftreten können, 1783.

[K2] Kant, I.: Über Pädagogik, 1803.

[K62] Kuhn, T.S.: The structure of scientific revolutions, Chicago 1962.

[Le] Leibniz, G.W.: Was ist eine Idee? (Quid sit idea?), 1678.

[Le1] Leibniz, G.W.: Neue Abhandlung über den menschen Verstand (Nouveauz Essais sur l'entendement humain), 1704.

[Lo] Locke, J.: An essay concerning human understanding, 1690.

[Mo03a] Modrow, E.: Pragmatischer Konstruktivismus und fundamentale Ideen als Leitlinien der Curriculumentwicklung am Beispiel der theoretischen und technischen Informatik, Dissertation, Universität Halle 2003.

[Mo03b] Modrow, E.: Fundamentale Ideen der theoretischen Informatik. In (P. Hubwieser, Hrsg.)Informatische Fachkonzepte im Unterricht, Lecture Notes in Informatics 2003, 189-200.

[SS04] Schubert, S., Schwill. A.: Didaktik der Informatik, Spektrum-Verlag 2004.

[Sc92] Schweiger, F.: Fundamentale Ideen - Eine geistesgeschichtliche Studie zur Mathematikdidaktik, Journal für Mathematikdidaktik, Heft 2/3 (1992), 199-214.

[Sc93] Schwill, A.: Fundamentale Ideen der Informatik, Zentralblatt für Didaktik der Mathematik 1 (1993) 20-31.

[Sc94] Schwill, A.: Fundamental ideas of computer science, EATCS-Bulletin No. 53 (1994) 274-295.

[Sc97] Schwill, A.: Fundamental ideas - Rethinking computer science education, Learning and Leading with Technology 25,1 (1997) 28-31.

[W83] Wegner, P.: Paradigms of information processing. In: The Study of Information (F. Machlup, U. Mansfield, eds), Wiley 1983, 163-175

GI-Edition Lecture Notes in Informatics - Seminars

Vol. S-1: Johannes Magenheim, Sigrid Schubert (eds.): Informatics and student assessment – Concepts of Empirical Research and Standardisation of Measurement in the Area of Didactics of Informatics, GI-Dagstuhl-Seminar 2004

The brochures can be purchased at:

Köllen Druck + Verlag GmbH
Ernst-Robert-Curtius-Str. 14
D-53117 Bonn
Fax: +49 (0)228/9898222
e-mail: verkauf@koellen.de