ABSTRACT
In order to validate suitable methods for investigating learning processes, we are currently conducting a case study, exploring the mental models of novice students in the field of object oriented modeling and programming. Until now we have asked the students of an introductory course of lessons to draw concept maps at three points of time. Additionally we conducted a small exam, where the students should implement some of the most important concepts. It turned out that a substantial learning progress could be observed by evaluating the concept maps and that there is a strong correlation between the quality of the maps and the success in the exam.

Categories and Subject Descriptors
K.3.2 [Computers and Education]: Computer and Information Science Education. Computer science education.

General Terms
Measurement, Human Factors, Languages.

Keywords
Learning process, mental model, concept maps, object-orientation, objects first.

1. INTRODUCTION
Facing the task to evaluate a recently introduced compulsory subject of Informatics in our state, we are looking for research results and methods that enable us to explore relevant learning processes systematically. Unfortunately, despite the multitude of past and current research activities, the didactical research in Informatics (or Computer Science) is still far behind traditional school subjects like Mathematics or Biology. There are many publications about specific aspects of teaching and learning Informatics, but little enough of them contain transferable results that might form a systematic theory of Informatics education, as Malmi et al. stated 2010: “However, after decades of research, we still have only a vague understanding of why it is so difficult for many students to learn programming, the basis of the discipline, and consequently of how it should be taught.” [17]. One could suppose that it might be easy to adopt results and research methods from other subjects like Mathematics, but unfortunately it turns out too often that things are very different in our subject domain. As an example we refer to the results of a SIGCSE 2007 working group [6], that argued that the hierarchy of learning outcomes is quite specific in Informatics and thus there is a need for a specific taxonomy (based on [2]).

In the Computer Science education research (CER) we are just taking the first steps towards a systematic investigation of learning processes by developing competency-based models [16], whereas e.g. in Mathematics such models were developed already in the early 80ties [20]. A necessary precondition of a competency-based model is the knowledge of the cognitive structures that are underlying the competences, see e.g. [28]. In order to explore and validate evaluation methods for these cognitive structures and to develop instruments for systematic research, we closely evaluated a CS1 lecture for students of engineering. We choose this lecture because its learning content is quite similar to a core part of the subject of Informatics, namely the part of OOM/OOP in the 10th grade. As our state administration is very restrictive in approving inquiries at schools, we prefer to develop these instruments at universities. Hereby we can reduce the activities in schools to absolutely necessary projects.

The course of lessons was investigated very closely concerning the knowledge that the students are supposed to learn. To this purpose we asked the students to draw concept maps at different points of time during the course. Up to now we have evaluated three generations of these maps. Additionally we asked them to complete a small exam test, also anonymously.

2. THEORETICAL BACKGROUND
Our teaching concept follows the object-first approach that was introduced in about 10 years ago as a reaction on problems students faced in writing their first object-oriented programs [4], [8]. We have presented a quite radical version of this approach for the compulsory subject of Informatics in Bavaria [9], [10], similar to the approach that was presented by [7]. Recently Ehler and Schulte [5] have compared the Objects-First and the Objects-Later approach empirically.

Particularly in the science education there are many research activities that use concept mapping techniques in order to investigate cognitive structures (regarded as “mental models”), see [15], [27]. The students are asked to draw a graph, whose nodes represent concepts and whose edges symbolize associations between these concepts, e.g. “is part of”. There is a variety of measures for the assessment of concept maps (as graphs) and many corresponding research results that validate these measures, e.g. [1], [14], [25], [26].
According to [23], mental models are changed by assimilation or accommodation. Assimilation means including new information into an existing mental model by activating an adequate schema or by adjusting it by means of accretion (accumulation of new information) or tuning (change of single components). If these processes are not successful, new information can only be accommodated by the process of reorganization, which is realized by constructing a new mental model. In [24] he describes closely, which methods might be applied to explore or assess mental models.

Concerning the evolution of mental models, we refer to the Conceptual Change theory, which was discussed in detail by [21], comparing two competing theoretical perspectives regarding knowledge structure coherence: knowledge-as-theory vs. knowledge-as-elements perspectives. Following the first one the student’s knowledge is “most accurately represented as a coherent unified framework of theory-like character”, following the latter it is “more aptly considered as an ecology of quasi-independent elements”. The usage of concept maps offers direct access to the “ruggedness” of the knowledge of a student [14].

Nevertheless is crucial to remember that a concept map does not represent the knowledge of its author directly. Correctly it has to be regarded merely as an externalization of this knowledge that might be influenced by the motivation to draw an extensive map, current attention to a specific knowledge area or many other external influences [19].

Concerning the representation of subject domain knowledge of object-oriented programming, there was an interesting proposal by [22], who propose to organize it in Truths, which are collections of concepts, operational skills and assessment criteria”.

3. THE COURSE OF LESSONS

For our investigation we chose a current running course that is introducing freshmen of engineering (major in geodesy) into the fundamentals of object-oriented programming (OOP), attended by about 40 students. The course comprises two weekly hours of lecturing and two more hours of practicing in groups of 20 students. It runs over one semester (15 weeks usually) and is taught in German language.

As pointed out in [12], there is a fundamental didactical dilemma in teaching OOP: On the one hand modern teaching approaches postulate to teach in “real life” context [3], which means to pose authentic problems to the students. Therefore it seems advisable to start with interesting, sufficient complex tasks that convince the students that the concepts they have to learn are really helpful in their latter professional life. On the other hand, if we start with such problems, we might ask too much from the students, because they will have to learn an enormous amount of new, partly very difficult concepts at one time, as discussed in [13].

We solved this problem following a “strictly objects first” strategy [7], by distributing the learning objectives over the parts of the course that precede the “serious” programming-part and thereby avoiding to confront the students with too many unknown concepts at the point they have to write their first program. Principally we suggest the students to look at an object as a state machine [11]. In order to realize this in a student oriented way, the students need to be able and to understand a simulation program of a typical state machine, e.g. a traffic light system.

In order to understand the results of our surveys, we have to summarize the curriculum of the course:

Chapter 1 – Modeling. Informatics: main subject areas, typical working methods; Functional modeling: data flow diagrams; modeling techniques in Computer Science.

Chapter 2 – Object Oriented Modeling. Objects in documents: object, class, attribute, method, class card, object card; artificial languages: grammars, BNF; states of objects: state, transition, state diagram, real and program objects; object diagram, association, class diagram, multiplicity of associations, compound objects: creation of objects as values of attributes.

Chapter 3 – Algorithms. The concept of Algorithm: programming languages, class definition: definition and declaration, signature of methods, access modifier, attribute declaration, definition of methods; structure of Algorithms: graphical representation of algorithms, structural components of algorithms, nesting of components, input and output of algorithms; properties of Algorithms: terminating, deterministic, determined.

Chapter 4 – Object Oriented Programming. Definition of classes: structure of object oriented programs, definition and declaration, signature of methods, access modifier, attribute declaration, definition of methods; assignment statement, ring exchange, assignment in constructor methods, encapsulation, equality; translation of computer programs, compiler vs. interpreter, execution of programs, course of events of a program; communication by methods: input, output, side effects, local and global variables/attributes; creating objects at runtime, constructor method, references, removal of objects; implementation of Algorithms: structure elements in programming languages: sequence, conditional statement, repetition; arrays, index...

Chapter 5: State Modeling. Finite automatons: triggering and triggered action, finite automaton, state chart; Implementation of automatons: switch statement; conditional transitions: complete state modeling, implementation of conditional transitions

Chapter 6 – Interaction and Recursion. Implementation of associations: unidirectional, bidirectional, multiplicities 1:1, 1:n, m:n, association class; sequence charts: calling of methods, sequence charts; Recursive algorithms: linear and cascading recursion;

Chapter 7: Generalization: Sub- and superclasses, specialization, inheritance; implementation of specialization, overriding of methods, generalization, class hierarchies; polymorphism: calling methods of foreign classes, abstract classes

4. SUBJECT DOMAIN KNOWLEDGE

In order to remove all irrelevant information, we have summarized all learning elements that we expect the students to know after the course by reducing the slides and the textbook for the course to a list of statements without any examples or explanations, looking e.g. like the following:

“If an attribute is marked as private, only objects of the same class are allowed to read or write its value.”

These statements (called knowledge elements) filled about 13 pages of text. To derive a list of concepts that should be presented to the students before they started to draw their concept maps, we reduced these statements to by the following steps: using the feature word frequency in the module MaxDictio of the software package MaxQDA (www.maxqda.de), we produced a list of key-words of the text. Then we sorted this list alphabetically and case-sensitive and removed all words with a lower case letter at the beginning. In the German language this condition assures that the deleted words are not nouns. Finally we removed all remaining
not-nouns, which were nevertheless written upper case (e.g. because they opened a sentence). In the next step we reduced all words to a standard form (singular nominative) and removed all variations or abbreviations of the same noun. From now on we had to respect the semantics of the text in the given context. We separated combinations of nouns that have their own meaning in our context (in German e.g.: Attributwert -> Attribut, Wert), combined words that have no meaning on their own (garbage_collection -> garbage_collection) and removed all words, whose meaning was too general (Informatics, Model), too specific or too technical (Pascal, RAM), all proper nouns and finally all purely didactical, organizational and pedagogical keywords. The whole process turned out to be quite objective and reproducible.

Afterwards we re-imported the resulting list of words into MaxQDA, coded and categorized it following the rules of qualitative research [18], obtaining finally a list of 40 concepts (CL) from these categories: aggregation, algorithm, array, assign statement, association, attribute, class, condition, conditional statement, creation, data encapsulation, data type, data, default value, execution, function value, function, generalization, identifier, inheritance, initialization, input parameter, interface, method call, method, object, operation, polymorphism, program, reference, repetition, specialization, state, state machine, state transition, structural elements, subclass, transition, value, variable.

This CL was presented every time, when the students were asked to draw the concept maps. The concept maps should be drawn using exclusively these 40 concepts (not all of them, but no other ones).

5. DATA COLLECTION

At the beginning of the course, every student was asked to provide some personal information. Afterwards we asked them to draw a first concept map ("pre-test", shortly PT). To ensure anonymity, we identified each student by a number given to them randomly at this pre-test. In the subsequent tests, the students were asked to give this number, so we could assign the maps or exercises to the other artifacts of a student.

Altogether we collected three generations of concept maps at three distinct times along the course. The pre-test PT was done by 39 students before the course started. The first mid-test (MT 1) was done by 38 students after 4 weeks, right after chapter 2 had been taught to the students.

![Exemplary concept maps of MT 1](image)

A small exam was completed by 26 students after 7 weeks, where the students should implement some of the most important concepts: assign statement, attribute, class, data encapsulation, data type, identifier, initialization, method call, method, object, program, value.

At this time the course had finished chapter 4 almost completely (with the exception of arrays). Finally, one week after the exam, a second mid-test (MT 2) was completed by 19 students. All but the last test MT 2, were conducted using paper and pencil. For MT 2, the students used the free graph editor yEd (see www.yworks.com), starting from a template containing the CL. This has been done partly, to counter an increasing "laziness" of the students when drawing the maps (since they had to redraw all the edges already drawn previously). After MT 2, there were 17 students left that took part in every test and the exam. Thus, in the subsequent analysis, we concentrated mostly on the data of those 17 respondents.

In the exam the students were asked to write the definition of a simple class City with given attributes for name, population, area and a simple method that calculates the population density out of these attributes. Additionally the attributes should be initialized with given values. For the evaluation all mistakes that were made by the students were described marked with a unique identifier, counted and assigned to one of the concepts of CL. This test was completed by 26 students, who made altogether 105 mistakes. The distribution of the mistakes over the concepts was the following: Assign statement 26%, method 12%, method call 12%, class 11%, initialization 10%, data type 8%, program 8%, data encapsulation 6%, value 5%, attribute 3%.

The most frequently occurring mistakes were pure syntactical: String constant without quotation marks 13%, String type written lower case 8%, missing semicolon at end of line 8%, incorrect string concatenation in output method 8%.

6. DATA ANALYSIS

In order to evaluate the concept maps, we let a working student redraw the maps on paper using yEd. By this way we could preserve all information that was contained in the original sheet. Afterwards all named edges of the graphs were transferred from yEd to MS Excel as quadruples with the following components: map-id, from concept, to concept, name of the edge.

Then we normalized the names of the edges in the following steps: all verbs were transformed to a standard form (first person singular indicative), all isolated prepositions and articles were deleted, all auxiliary verb were removed, isolated nouns or adjectives were deleted, all multiplicities were removed ("some", "many" etc.). In the remaining set of different edge names (PT: 116, MT 1: 95, MT 2: 106) the edges were marked with their frequency. We also counted all names with more than 1 occurrence. The number of those stayed nearly constant over all three surveys: (PT: 36=31%, MT 1: 35=37%, MT 2: 29=27%). Thus, apart from single occurrences, the students used about 30 different names to express the associations between the concepts.

In the next step all associations were scored by the lecturer of the course with points (0 point for "totally wrong", 0.5 points for "partly correct" and 1 point for "totally correct"). The sequence of the proportions of the totally correct edges proved that the students have really gained some relevant knowledge: PT: 33%, MT 1: 48%, MT 2: 65%.

Static aspects mostly deal with the properties of a concept map when treated as a graph. The measures are usually taken for a single graph, thus we can analyze single concept maps as well as
deriving a distribution over a group of students. Two very simple measures are the number of edges in a concept map (as an approximation for its “complexity”) and the number of (weakly) connected components, leaving out isolated concepts. Both are used and validated, for example, in [14], the latter is called ruggedness. Additionally, we can calculate the average score (using the edge scores 0, 0.5, 1 described above) over all edges of a given map.

Table 1: Ruggedness (R), edge count (EC), average score (AS).

<table>
<thead>
<tr>
<th>Test</th>
<th>R</th>
<th>EC</th>
<th>AS</th>
<th>R</th>
<th>EC</th>
<th>AS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PT</td>
<td>1.7</td>
<td>8.5</td>
<td>0.5</td>
<td>0.96</td>
<td>5.08</td>
<td>0.14</td>
</tr>
<tr>
<td>MT1</td>
<td>1.8</td>
<td>12.0</td>
<td>0.6</td>
<td>0.98</td>
<td>5.72</td>
<td>0.19</td>
</tr>
<tr>
<td>MT2</td>
<td>2.7</td>
<td>17.7</td>
<td>0.8</td>
<td>1.7</td>
<td>5.93</td>
<td>0.11</td>
</tr>
</tbody>
</table>

Table 1 shows a summary of those three measures. When correlating the ruggedness and edge count, no (or only a very weak negative) effect shows. This implies that the measures are somewhat independent. Static analysis reveals how heterogeneously the group of students is at a given point in time. We can observe a large spread (standard deviation) in the edge count for every test. Additionally, we can observe that the spread in the scores is the largest at MT 1. Given that this was the first “real” test (where the course had already begun) we see, that some students pick up the new material quickly whereas others seemingly have major difficulties in creating a mental model for the presented concepts. Obviously, from a teachers’ point of view, one hopes for this discrepancy to diminish as the course progresses. This aspect directly leads to the dynamic aspects.

Dynamic aspects reveal how the mental models of the students change over time. This is especially interesting when contrasted with the material the students have been presented with in the meantime. Looking at Table 1 again - but this time concentrating on the vertical differences - the mean of the edge count clearly shows an increasing trend, while the standard deviation remains nearly constant. This shows that the maps are getting more complex, on average, which is to be expected. Interestingly, however, the average ruggedness also increases. We could interpret this result as in favor of the “knowledge-as-elements” theory [21]. However, this might also have more profane reasons. For example, the students might be too lazy to redraw the complete map and just focus on “new” edges, thus increasing the ruggedness. Or they might not see the connection between the different chapters yet. The average correctness of edges also increases over time – again, this is to be expected and shows that the students are indeed learning.

In order to relate the evolution of the maps of a student to the learning contents of the course, we need some measure of importance (or saliency) of the concepts of CL for a given time frame (e.g. for the material taught between two tests). There is a lot of research about how this might be done automatically. For this experiment, we simply used the occurrences of the concepts in the corresponding parts of the list of the knowledge elements of the course (see section 4 above). We were expecting that the concept maps change more at those concepts that were mentioned more frequently in the covered knowledge elements. Given two concept maps drawn by the same student at different times, we counted the edges connected to each concept in both maps, then calculated the difference between those lists. The sign of the result determines whether edges have been added or removed to the concept. If the label of the edge should be incorporated in the process, we can for example rely on the manual scoring of the edges. By this way we produced a list of concepts with assigned numerical values that indicated the change of the number of associations to or from this concept, which we correlated to the measure of saliency of the concepts.

When examining the results for all students that took part in all three tests, we get the correlations shown in Table 2. For several values of k, the k most salient concepts are correlated to the corresponding measure of change of those concepts (averaged over all students that took part). As we can see, we get very good correlations for a small to medium number of concepts, all with p<0.05. If we keep k increasing, the effect gets weaker. This is to be expected, as the chapters of the course are usually focused only on a small number of concepts, where also the biggest changes in the map are to be expected.

Table 2: Correlations between changed edges and saliency

<table>
<thead>
<tr>
<th>k</th>
<th>PT–MT1</th>
<th>MT1–MT2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.87</td>
<td>0.99</td>
</tr>
<tr>
<td>5</td>
<td>0.86</td>
<td>0.85</td>
</tr>
<tr>
<td>7</td>
<td>0.84</td>
<td>0.81</td>
</tr>
<tr>
<td>9</td>
<td>0.82</td>
<td>0.75</td>
</tr>
</tbody>
</table>

In the PT1-MT1 case the 9 most salient concepts (in decreasing order) were: object, class, value, property, function, program, association, data, state.

If we add up all the edge scores in a single concept map, we get a single score for a concept map, which we can correlate by taking these values for PT, MT1 and MT2. We get 0.49 and 0.44 when correlating PT to MT1 and MT2, respectively (Spearman’s rank correlation). Correlating MT1 and MT2, we get 0.64. All p-values were below 0.04. So, whether a student got a low or a high score in MT1, this generally will be reproduced in MT2. Additionally, when correlating the values for MT1 and MT2 with the number of errors in our exam, we get -0.68 and -0.53 (p < 0.04 in both cases). This might indicate that there is a causal relationship between the score of the concept map of a student and his/her achievement in a test. If such a relationship could be shown empirically, it would be a major indication for the usefulness of using concept maps to investigate students’ learning processes.

7. CONCLUSION AND FUTURE WORK

We presented several measures that can be used to analyze static and dynamic aspects of concept maps, when observing students over a larger time frame. So far, the measures seem plausible, however, they need to be investigated more closely. We hope to work out an automatic scheme that creates concepts out of our course material and analyzes maps that students create over the course of a lecture. This will also include finding out more about how the repeated drawing of concept maps could be supported by tools without influencing the outcome too heavily.

8. REFERENCES
