ABSTRACT
In order to explore and 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. After abstracting and systemizing the information that was presented to the students of our introductory CS 1 course for non-majors we have asked them to draw concept maps at four points in time. Additionally, we conducted a small midterm exam, where the students had to implement some of the most important concepts and a regular final exam. We found that learning progress can be observed in detail by evaluating the concept maps.

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
The research in the field of Computer Science education is just taking the first steps towards a systematic investigation of learning processes by developing competency models [15], whereas e.g. in Mathematics such models were developed already in the early 80ies [21]. 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” [16]. Despite the current focus on competencies we are still interested in the investigation of knowledge elements, as these might be preconditions for certain competencies (following e.g. [31]).

For example, knowledge about algorithmic structures is suggested to be a precondition for programming competencies. Apparently it is crucial for learning, particularly when following modern constructivist teaching approaches, that the learners gain exactly the necessary prerequisite knowledge, because otherwise their attempts to acquire a certain competency are not likely to be successful. Thus, if the prerequisite knowledge is well known, it is possible to support these learning processes in a very efficient way by presenting it at an early stage to the students. Therefore, our long-term goal is to find out (empirically) which cognitive structures are preconditions for programming competencies.

Searching for methods that might help us to detect this, we are currently exploring and trying to validate evaluation methods for cognitive structures. To this purpose, we have closely evaluated the knowledge structures of a CS1 lecture for students of engineering. We chose this lecture because its learning content is quite similar to a central part of the recently introduced compulsory subject of Informatics, namely the part of OOM/OOP in the 10th grade, which is the focus of our educational research activities [10].

The course of lessons was investigated very closely concerning the knowledge that the students are supposed to acquire. We asked the students to draw concept maps anonymously (marked with a random unique code number) at different points in time during the course. Until now, we have evaluated four generations of these maps. Additionally, we asked them to voluntarily complete a small midterm exam and to mark their final exam with the code number of their maps, allowing us to match the maps of each student to his/her exam solutions.

2. THEORETICAL BACKGROUND
Our teaching concept follows the “objects-first” approach that was introduced about 10 years ago as a reaction to problems students faced in writing their first object-oriented programs [4]. We have presented a quite radical version of this approach for our compulsory subject of Informatics in Bavaria [9], similar to the approach that was presented later in [8]. Recently, Ehler and Schulte have compared the objects-first and the objects-later [6] approaches empirically.

Particularly in science education there are many research activities that use concept mapping techniques in order to investigate cognitive structures (regarded as “mental models”), see [14], [30]. The students have 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. [28], [1], [29]. Some years ago, Sanders et al. compared the knowledge of students in several nations using concept mapping techniques [25]. Smith and Davenport developed a graph-theoretical measure for the similarity of graphs based on neighborhood structures [7]. McClure et al. [18] validated this measure by correlating it with several scoring techniques. Hereby, they also detected that the scoring of locally correct edges using a master map is the most convincing scoring technique for concept maps.

By drawing concept maps, the students externalize parts of their knowledge. Due to the structure of the maps, this is restricted to factual and conceptual knowledge following the categorization of Anderson and Krathwohl [2]. According to [24] 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 (changing of single components). If these processes are not successful, new information will only be accommodated by the process of reor-

© ACM, 2011. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in ICER '11 Proceedings of the seventh international workshop on Computing education research Pages 77-84 http://doi.acm.org/10.1145/2016911.2016929
ganization, which is realized by constructing a new mental model. In [26] and [27] Seel describes closely which methods might be applied to explore or assess mental models.

Concerning the evolution of mental models, we refer to the theory of Conceptual Change, which was discussed in detail in [22] by comparing two competing theoretical perspectives regarding knowledge structure coherence: knowledge-as-theory vs. knowledge-as-elements. Following the first one, the students’ 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 [13].

Nevertheless, it’s crucial to remember that a concept map does not represent the knowledge of its author directly. Instead, it has to be regarded merely as an externalization of this knowledge that might be influenced by the motivation to draw an extensive map, by the focus of current attention or by many other external influences, as already stated by [20].

Concerning the representation of subject domain knowledge of object-oriented programming, [23] proposed to organize it in Trues, (Testable, Reusable Units of Cognition) which are collections “of concepts, operational skills and assessment criteria”. The drawing of concept maps usually starts with a list of given concepts that the students should arrange and connect by associations. Such a list might be formed e.g. by the 39 concepts that [3] identified as “quarks of object oriented development” by exploring relevant textbooks. Mead et al. compared several criteria for identifying very important knowledge elements and propose to search for anchor concepts that are either fundamental or integrative and transformative [19].

3. THE COURSE OF LESSONS

For our investigation we chose one of our currently running courses that introduces 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 practice in two groups of 20 students. It runs over one semester (usually 15 weeks).

As pointed out in [11], there is a fundamental didactical dilemma in teaching OOP: On the one hand, modern teaching approaches postulate to teach in a “real life” context [5], i.e. to pose authentic problems to the students. Therefore, it seems advisable to start with interesting, sufficiently complex tasks that convince the students that the concepts they have to learn are really helpful in their 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 once, as discussed in [12].

Following a “strictly objects first” approach [8], we solved this problem 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 when they have to write their first program. Basically, we suggest to the students to look at an object as a state machine [10]. In order to realize this in a student oriented way, the students need to be able to understand a simulation program of a typical state machine, e.g. a traffic light system.

Please note that the course was taught in German language, thus all the text material, the concepts and the concept maps had to be translated from German to English for this paper.

As it is not possible to understand the results of our survey otherwise, we shortly 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 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 automata, triggering and triggered action, state chart; Implementation of automata: switch statement; conditional transitions: complete state modeling, implementation of conditional transitions.

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

Chapter 7: Generalization: Sub- and super classes, 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 compare the knowledge that was externalized by the students with the knowledge they should acquire by studying the course material, we tried to find representations of the relevant information that are as compact and as formal as possible. For that purpose we have summarized all learning elements that we expect the students to know 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 form the possible nodes of the concept maps, we reduced these statements in 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 keywords of the text.
Then we sorted this list alphabetically (case-sensitive) and removed all words starting with a lower case letter. In German, this condition assures that the deleted words are all non-nouns. Finally, we removed all remaining non-nouns that were written in 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. The following steps were based on the meaning of the words in the given context. We separated combinations of nouns that have an independent meaning in our context (in German e.g.: Attributwert was separated to Attribut, Wert), combined words that have no independent meaning (garbage, collection) was combined to garbage_collection and removed all words whose meaning was too general (Informatics, model), too specific or too technical (Pascal, RAM). Finally, all proper nouns and all purely didactical, organizational and pedagogical keywords were omitted. 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 [17], obtaining finally the following list of 40 concepts (shortly called CL): aggregation, algorithm, array, assign statement, association, attribute, class, condition, conditional statement, data encapsulation, data type, data, default value, execution, function value, function, generalization, identifier, inheritance, initialization, input parameter, instantiation, interface, method call, method, object, operation, polymorphism, program, reference, repetition, specialization, state, state machine, state transition, structural elements, subclass, transition, value, variable.

Compared to the list of the 39 “quarks” that were elaborated in [3], there are 9 exact (verbatim) correspondences, 6 terms of CL have a very similar meaning to 7 of the quarks and 25 terms of our list have no apparent correspondence to any quark. The reason for the minor overall correspondence might be that the (more abstract) quarks are selected concerning their importance for object orientation as a technique, while CL represents just the basic concepts of our lecture which is not restricted to object orientation.

We asked the students to draw their maps in the following way. We presented the concepts of CL in the form of a checklist. At first the students should check all the concepts that they believed to know something about. Following this, they should draw a graph, using the checked concepts as nodes and connecting these by associations, which all should be denoted by suitable labels. For the evaluation of the maps we have removed all associations that were not labeled, assuming that these did not reflect any precise knowledge.

Following [18] the scoring technique “relational with master map” has the highest reliability and validity of the 6 strategies they have tested. It is performed by scoring every association in a student map as correct or incorrect by comparing it to a master map that was drawn by an expert, depending on a comparison of the labels that were given to the regarded association by the student or the expert, respectively. In order to get such an expert map that is as objective as possible, we derived it from the same material that we have used for the derivation of our CL. For that purpose, we coded all sentences of the list of the knowledge elements (see above) by the occurrences of one or more of the 40 concepts of CL using MaxQDA. Afterwards, we produced a list of all sentences that were marked with one or more concepts of CL, assuming that these sentences might suggest associations between those concepts. As concept maps can represent binary associations only, we checked the arity of these associations. Among the 161 sentences that contained more than one concept, we found 101 containing 2, 40 containing 3, 17 containing 4 and 3 containing five concepts. Therefore we could assume that most of the associations (63%) are binary and consequently could be represented in a concept map directly. Following this, we translated the information that was contained in these 161 sentences in associations by qualitative means. It turned out that this was not possible in some cases, e.g. because the regarded statement was structurally too complicated. We ended up with a set of 98 associations which formed our objective expert map (OEM) that was used e.g. to score the student maps.

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”, PT) using pencil and paper. To ensure anonymity, we identified each student by a code number given to them randomly at this pre-test. In the subsequent tests, the students were asked to give their code number, so we could assign the maps or exercises to the other artifacts of each student.

Figure 1. Exemplary map in the yEd graph editor.

 Altogether, we collected four generations of concept maps at four distinct points in time. As the drawing was done partly in the main lecture and partly in the tutorials (and was voluntarily), we had varying numbers of participants. The pre-test was done by 39 students before the course started. The first mid-test (MT1) was done by 38 students after 4 weeks, right after chapter 2 had been taught to the students.

A small midterm exam (MX) was completed by 26 students after 7 weeks, in which 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. 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.

At the time of the exam, the course had finished chapter 4 almost completely (with the exception of arrays). One week after the exam, another collection of concept maps (MT2) yielded 19 student maps. Up to MT1, the tests were conducted on paper. From
MT2 on, the students used the freely available graph editor yEd (see www.yworks.com and Fig. 1 for a screen-shot), starting from a template containing our list of concepts, CL. This has been done to counter an increasing “laziness” of the students when drawing the maps (since they had to redraw all the concepts and edges already drawn previously). Finally, immediately after the end of the lecture and some weeks before the final exam, there was a last test (post-test, POST), that was attended by 17 students. In the final exam (FX), 13 students gave us their code number and hereby allowed us to correlate their maps with their scoring in the exam.

6. DATA ANALYSIS
The collected data (we digitalized all maps that were collected on paper using yEd) consisted of 107 maps from students, the results of the midterm exam and the results of the final exam.

6.1 Naming of Associations
Before analyzing the students’ maps, we normalized the labels of the edges (which were freely chosen by the students) in the following way: all verbs were transformed to a standard form (first person singular indicative), all isolated prepositions and articles were deleted, all auxiliary verbs were removed, isolated nouns or adjectives were deleted and all multiplicity specifications (“some”, “many” etc.) were removed. In the next step we categorized the resulting names from all surveys following the rules of qualitative text analysis [17].

![Figure 2. Frequency of categories of association labels (PT in front, MT1, MT2, POST at back).](image)

We found that the number of (categorized) edge labels with more than one occurrence was quite low and stayed nearly constant (PT: 24, MT1: 28, MT2: 29, POST: 31) over the four surveys. Thus, apart from single occurrences, the students only used about 30 (semantically) distinct names in each survey to express the associations between the concepts. This is very interesting, as it shows that the restriction of edge labels to a predefined list, which would make the automatic collection and evaluation of maps much easier, would not result in a considerable loss of the gathered information.

Additionally, it turned out that the relative frequency of these categorized associations was quite constant over the four surveys. Fig. 2 shows the result for those concepts that were used at least in one survey for more than 2% of the edges. The most frequent categories by far were contains and its opposite form has. In contrast to this, the frequency of usage of the 40 concepts (CL) was quite different over the four surveys.

Subsequent to normalization and categorization, all associations were scored by the lecturer of the course with points (0 points for “totally incorrect”, 0.5 points for “partly correct” and 1 point for “totally correct”). This was done by comparing it to the objective expert map (see section 4) following the technique “relational with master map” suggested by [18]. The edges were scored “totally incorrect” only if the suggested relation between these two concepts was apparently wrong and “partly correct” if it did not describe the relation totally correct, but a certain aspect of it in an acceptable way.

6.2 Graph-theoretical Measures
Two very basic measures of a concept map are the number of edges (as approximation for its “complexity”) and the number of (weakly) connected components, leaving out isolated concepts. In other words, we treat the concept map as an undirected graph and count the connected components (clusters of nodes that are reachable from another). Both measures are used and validated, for example, in [13], where the latter measure 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 summarizes the results of the surveys regarding these measures.

![Table 1. Ruggedness (R), edge count (EC), correct edge count (CEC) average score (AS).](image)

6.3 Midterm (MX) and Final Exam (FX)
For evaluation, all mistakes that were made by students were closely described, provided with a unique identifier, counted and assigned to one of the concepts of CL. This test was completed by 26 students, who made a total of 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 purely 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%.

Next, we took a closer look at the relationship between a students’ concept map and his/her result in MX. To this end, we isolated the maps of MT1 of the 17 students that both drew this map and gave their code number on MX. When correlating the number of correct edges with the achieved score in the exam, we get an effect of 0.68 with a p-value of 0.002. Apparently, on average, a student with good results in the exam will also have a high number of correct associations in the concept map. This indicates that there is some correspondence between the declarative knowledge of a
student and his/her ability to apply that knowledge in the context of an examination.

As the final exam took place very short before the deadline of this conference, we had not the time to investigate possible correlations between the performance of the students in this exam and the quality of their concept maps in detail. At a first glance we were not able to find correlations between the graph measures presented in this paper and the results of the exam at any relevant level of significance, which is mainly due to the very low number of students that took part in POST and also gave their map code number at FX.

6.4 Evolution of the Students’ Knowledge

Our aim was to find concepts and associations (in the OO context) that were understood very well well or, in contrary, have caused remarkable problems to the students. Because it is very hard to keep an overview over the evolution of graphs with too many concepts, we concentrated this analysis on concepts that have a close relationship to object orientation, which was the focus of the course. Therefore, we selected those 9 concepts of CL that are contained in the list of quarks [3], added generalization, specialization and subclass, because they represent the quark class hierarchy and finally method call because it is a special form of the quark message passing. The result was the following list (OCL): aggregation, association, attribute, class, instantiation, generalization, inheritance, method call, method, object, polymorphism, specialization, subclass.

Restricted to these 13 concepts, we could draw a partial objective expert map based on the associations we had derived from the course texts for the total OEM (see section 4). Nevertheless, only 10 of these concepts actually appear in this graph, because 3 of them were not connected by the total OEM at all. The partial map is shown in fig. 3.

Figure 3. Partial objective expert map of OO-concepts

When summing the edge scores for each edge over all student maps for a given survey, we get a value that describes how well this particular association was understood. For easier analysis, we used -1, 0, 1 as score values (in contrary to 0, 0.5, 1 as above). Thus, all “partly correct” edges were ignored in the summation. From this list of edge scores we extracted those that were particularly low or high. We defined a high score to be not lower than the 3rd quartile and a low score to be not higher than the 1st quartile of the score values. Based on these limits we could draw a map each for the corresponding high scored respectively low scored associations for each survey and hereby get an overview over concepts and associations that were understood well or badly, respectively. Figures 4 (high scored associations) and 5 (low scored associations) show the development of those “landscapes” over the four surveys.

Figure 4. Frequently correct (high scored) associations

Finally, we investigated the similarity of the connections (by associations without respect to their labels and their correctness) of the single OO related concepts in the student maps compared to the expert maps. We applied the similarity measure for graphs proposed by [7]. It is built upon the set of neighbors of a given node, computed locally for each node and then – if necessary - averaged over all nodes for the whole graph. It basically measures the similarity of the set of neighbors of a concept in the two compared maps. Its value is always within the interval [0; 1]. We were interested in concepts that show a very high (respectively very low) degree of similarity to the expert maps. However, the extreme measures of 0 and 1 are difficult to interpret. Isolated nodes, for example, will always get a rating of 0 and nodes with very few edges in the expert map (like generalization) tend to produce a rating of 1 very often. Therefore we left out the nodes with exact values of 0 and 1, i.e. we don’t make any statement about the similarity of these concepts. When taking every map of every student into account, taking the average of the similarity for each of the OO related concepts - compared to the partial objective expert map - we get the ranking that is shown in table 2.
7. DISCUSSION

The score values show that the average number of correct edges per student map increases clearly up to MT2. Looking at the quartiles (not shown in Table 1) of the CEC, both the 1st and the 3rd quartile (as well as the median, consequently) show a monotonically increasing trend (from 1 to 6 and from 4.8 to 15 respectively) over the four tests. This is to be expected and shows that the students are indeed gaining relevant knowledge.

We can also observe that the spread in the scores is the largest at PT, followed by MT1. The spread in PT is easy to explain by the varying of already existing subject-domain knowledge of the freshmen. Nevertheless, an average of 3 correct edges shows, there was already some amount of knowledge before the lecture had started. Given that MT 1 was the first “real” test (where the course had already begun) the high standard deviation shows 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. Exactly this can be observed in Table 1. The only exception to the trend is a small increase in the standard deviation at the POST test, which might have been caused by an increasing laziness of some students.

Interestingly, the average score AS (edge count divided by edge score of each map) is not increasing similarly. Nevertheless, looking at the quartiles of AS, both the 1st and the 3rd quartile show a monotonically increasing trend (from 0.43 to 0.69 and from 0.64 to 0.83 respectively) over the four surveys similar to CEC. This shows again that the students are learning actually.

When correlating ruggedness and edge count, no (or only a very weak negative) effect shows. This implies that those measures are somewhat independent. We can see that the mean of the edge count in table 1 clearly shows an increasing trend, while the standard deviation is quite high overall, decreasing from PT to MT2 and increasing again at POST. 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 [22]. However, it might also have simpler reasons. For example, the students might be too lazy to redraw the complete map and just focus on “new” edges, thus increasing the number of connected components. Or they might not see the connection between the different chapters yet.

Concerning figures 4 and 5, it is very interesting to see, how the knowledge of the students is developing. Starting from nearly no relevant knowledge, the incorrect associations are growing seemingly more than the correct ones (which, indeed, stay nearly constant until the last test). That means, the students were actively drawing edges in the concept maps, yet the labels were clearly incorrect (edge score of -1). So, the students knew what the main concepts of the lecture had been before the corresponding tests and they seemingly had a mental representation of their interconnections. Otherwise they most probably wouldn’t have drawn any edges. As the material of object-orientation is rather complex and also completely new to the students, it is not surprising that, at the beginning (i.e. MT1, when they were first presented the material), there are more misunderstandings than correct associations. However, as this trend continues throughout MT2 and even into POST, we can identify clear problem areas of the covered material. For instance, the association between class and object remains a misconception throughout every test. MT1 is the only test, where this association is also present among the “correct” edges. This lends

| Similarities of the connections of OO concepts |
|-----------------|--------|
| instantiation  | 0.97   |
| inheritance    | 0.91   |
| class           | 0.31   |
| object         | 0.25   |
| attribute      | 0.23   |
| subclass       | 0.15   |
| association    | 0.13   |
| method         | 0.13   |
| method call    | 0.12   |
| specialization | 0.07   |
| generalization | 0.06   |
| aggregation    | 0.03   |
| polymorphism   | 0.01   |
itself to the interpretation, that a number of students were able to recall some factual knowledge about the relation of classes and objects at MT1, whereas in subsequent tests, they neither seemed to recall it, nor gained some deeper understanding about those two very central concepts of object-orientation.

Another interesting example is the association between class and subclass. Basically, the understanding of this association requires knowledge about inheritance, which wasn’t present in the lecture until the very last chapter 7. Still, students seemed to have a (incorrect) mental representation of the meaning that led them to draw an edge in the concept map. For example, one of the incorrect edges in MT1 between class and subclass was labeled “contains”. Interestingly enough, when the material was finally presented in the lecture (before MT2), subclass appeared for the very first time in the list of correct associations, even in the context of inheritance, but still, a large enough number of students had misconceptions (as the association class-subclass is also present in the list of incorrect associations of MT2).

Trying to locate problem areas, we are searching for concepts/associations that:
- were present in the lecture prior to the test (otherwise we cannot expect the students to know anything about it),
- are present only in the list of incorrect associations (this indicates the most troublesome areas) and
- remain that way for more than one test (so it’s clear that the misconceptions remain).

For this lecture, this leads to the concepts of inheritance, generalization and association, which, interestingly, participate in different associations in each test – clearly indicating that the students did not have a well formed mental model about it. Also, the already mentioned class-object association and the association attribute-method fulfill these requirements.

However, when contrasting the evolution to the ordered list of similarities in Table 2, we get a slightly different picture. Class and object are remarkably high in the ranking, even though their absolute values are not remarkably high. Thus, even though there are related misconceptions concerning those concepts, many students seem to have drawn correct edges around these two concepts. So, in the overall structure, students seem to have got a grasp on how the concepts are interconnected with the rest. And indeed, class is the center of both the incorrect and correct landscapes in the figures.

8. CONCLUSION AND FUTURE WORK

We presented several measures that can be applied to analyze static and dynamic aspects of concept maps, when observing students over a larger time frame. So far, these measures seem plausible; however, they need to be investigated more closely. In particular, we will explore the relation between the changes in the maps of a student and the schemata of knowledge they have learned in order to distinguish assimilation and accommodation processes. In our next step we will investigate the results of the final exam (FX) more closely in order to find knowledge patterns in the maps that are corresponding with the ability to apply the knowledge in the exam. These might be candidates for the prerequisite knowledge elements that we have addressed in the introduction of this paper.

Additionally we are developing a web-based software tool that allows drawing concept maps using any web-browser (CoMapEd), storing them together with all relevant information about the survey in a relational data base on a server and evaluating the maps using all measures discussed above. We aim to use this tool for the first time for a survey among the informatics teachers in our state that is scheduled for the end of 2011. The goal is to investigate the knowledge of the teachers in a similar way as presented in this paper. After this survey, we plan to offer access to the tool for students and teachers in schools and universities.

9. REFERENCES


