ABSTRACT
In introductory courses there are topics with a huge amount of concepts and interdependencies between them. Especially in the course materials the relevant definitions or specifications are widely spread. This paper presents a semi-automatic way of text analysis to display the structure of concepts in literature such as textbooks. Therefore we searched for text paragraphs that contain a definition or specification of a given concept. From the results we built a graphical representation, containing “definition-nodes” that show the logical structure of those concepts. We call it a Concept Specification Map. Furthermore we show an application of the method on five textbooks of introductory courses into object-oriented programming. We present the results of the analysis and draw some conclusions on their influence regarding introductory courses.

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

Keywords
semi-automatic analysis, textbooks, CS1, concept structures, graphical representation

1. INTRODUCTION
During the last few years the number of investigations of object-orientation (OO) in computer science education has risen substantially. With the help of courses for freshmen at our university we analyzed their way during introduction into programming ([9]). We found that it is difficult to find the elementary concepts within all the theories on introducing object-oriented programming (OOP) like “objects-first” or “objects-later”. Another aspect is the differentiation between novice and advanced concepts within object-oriented programming and their order in an introductory programming course. To increase clarity, we are investigating the materials the students are using at the very beginning of their “programming career”. In a first step we examined the conceptual knowledge which the students gain in our courses. Therefore we investigated concept maps the students drew during their courses ([1], [10], [11] and [12]). The second step is now an inquiry into the structure of the content of literature especially course materials like textbooks. In the first paragraph we will introduce a semi-automatic way of text analysis which leads to a graph showing all the interdependencies of the concepts in a given piece of literature. We will call it a Concept Specification Map (CSM). The following paragraphs will show the application and results of the method on five books gathered from introductory courses in several universities. Finally, we will discuss the results and give a brief overview about the impact on introductory courses and future research.

2. BACKGROUND AND RELATED WORK
Armstrong explained in [2] that:

"One reason that learning OO is so difficult may be that we do not yet thoroughly understand the fundamental concepts that define the OO approach" [2].

Facing this problem Armstrong analyzed literature to find the essence of object-orientation. To define the so-called "quarks" of OO, she reviewed 239 sources and found 88 candidates. Out of these, she listed all the concepts which appeared in more than 50% of the literature items.

Besides the work of Armstrong Meyer and Pedroni made investigations on curricula and other teaching materials which lead to the so-called "Trucs" (Testable, Reusable Units of Cognition) introduced by Bertrand Meyer in [14] and investigated by Pedroni [15]. They formed the initial idea of our work. Pedroni analyzed course materials and examined notions, a smaller unit of the "Trucs". Following the semi-automated extraction, which inspired our qualitative analysis, she put the "Trucs" and notions into relation and visualized them with a kind of map which is quite similar to ours. The difference is that the "Trucs"-maps are gathered from course design materials like curricula.

There has been a lot of textbook analysis in the last few years, each of them with another goal. An earlier study from 1988 analyzed the content of textbooks based on the topics of each book. Means [13] calculated a ratio of the frequency...
of the topics in the text related to the area the text needed in the book. He wanted to find out if there was any development of the books through the different editions, whereas we will only count the definitions respectively specifications given in the books without putting them into relation to the size of the books.

An extensive study on textbooks for introductory programming in Australia was conducted by de Raadt, Watson and Toleman [16] in 2005. They analyzed textbooks according to their structure, the given exercises and their coverage of the ACM/IEEE curriculum.

"[They] found large variations in the texts reviewed. Many texts rely on examples to fill content. Few texts attempt to comply with curricula recommendations. Few contain a continuous instruction in problem solving." [16]

In contrary to [16], we ignore structural elements and focus on the learning content of the books.

In 2010 Börsler et al. [5] evaluated programming examples in introductory textbooks, applying several criteria concerning the technical, object-oriented- and didactical quality of those examples.

3. METHODOLOGY

All the analysis done in the investigations mentioned above, result in a list of properties of the examined texts. Our method for the investigation of the concepts of a specific topic leads to graphical representation of identified "definitions", which we called Concept Specification Map (CSM). First we will show what this CSM is like and after that we will present a method for a semi-automatic analysis, gathering "definitions" respectively specifications of concepts.

3.1 Concept Specification Map

The main focus of our investigation lies in the representation of object-oriented programming and design in computer science education. Therefore, we wanted a graphical representation of concepts and their interdependencies. The result is an overview map for concepts and specifications. The concepts are gathered by a semi-automatic text analysis described down below.

In our maps the text passages we extracted were symbolized by a circle, connected by a bold line to the specified concept and by outgoing arrows to the specifying concepts (see figure 1). For better readability, the "definition"-nodes are numbered. This allows the referencing of the nodes in the text. A sample is shown in the following sentence and in figure 1.

(12) “A class is a collection of fields that hold values and methods that operate on those values”. [8]

The concepts field and method are specifying the concept class. The number in the circle corresponds to the one in parenthesis in the “definition”.

One of the origins of our work were the “quarks” that were identified by Armstrong [2]. She found eight concepts that she defined as the "quarks" of object-orientation: Inheritance, Object, Class, Encapsulation, Method, Message Passing, Polymorphism and Abstraction. For each of these concepts she gave a short definition she gathered from her text analysis. We used these definitions to build a sample Concept Specification Map which is shown in figure 2.

3.2 Gathering of Specifications

For the Concept Specification Maps we need a list of “definitions” of concepts. In most texts in scientific or educational literature there are no definitions like “object-orientation is...”, so we further speak of specifications respectively specifying or specified concepts instead of definitions. In order to find associations between the concepts we follow a semi-automatic approach. So, first we automatically extract the sentences from the text that contain one of the concepts. In addition, we extract two sentences before and after this sentence. Each element of the resulting list is manually rated with 1, if the given concept is specified in a more detailed manner. Otherwise, it is rated with 0. The 1-rated “definitions” form a new list, where specified and specifying concepts are listed for each piece of text. Then, we eliminate the duplicate entries in the passages and in the list of the concepts. Elements with the same list of concepts (specified and specifying) are put together to one combining the passages. This is done simply by concatenating the two specifications.

The following list shows a few examples of extracted text passages from the books we investigated with the method

![Figure 1: sample "definition"-node](http://www.yworks.com)

![Figure 2: CSM of the definitions of the “quarks” defined by Armstrong [2]](http://www.yworks.com)
Table 1: textbooks which are used by or recommended to the students in the investigated CS1-courses

<table>
<thead>
<tr>
<th>Author</th>
<th>Title</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eckel, B. (EB)</td>
<td>Thinking in Java</td>
<td>7</td>
</tr>
<tr>
<td>Abelson, H. und Sussman, G.J. (AH)</td>
<td>Structure and Interpretation of Computer Programs</td>
<td>1</td>
</tr>
<tr>
<td>Sedgewick, R. und Wayne K. (SR)</td>
<td>Introduction to programming in Java</td>
<td>17</td>
</tr>
<tr>
<td>Deitel, H. und Deitel, P. (DH)</td>
<td>Java: How to program</td>
<td>2</td>
</tr>
<tr>
<td>Flanagan, D. (FD)</td>
<td>Java in a Nutshell</td>
<td>8</td>
</tr>
<tr>
<td>Bishop, J.</td>
<td>Java gently</td>
<td>-</td>
</tr>
<tr>
<td>Bloch, J.</td>
<td>Effective Java</td>
<td>-</td>
</tr>
<tr>
<td>Felleisen</td>
<td>How to design Programs</td>
<td>-</td>
</tr>
<tr>
<td>Sebesta, R. W.</td>
<td>Concepts of Programming Languages</td>
<td>-</td>
</tr>
<tr>
<td>Mitchell, J. C.</td>
<td>Concepts in Programming Languages</td>
<td>-</td>
</tr>
<tr>
<td>Roberts, E.</td>
<td>The Art and Science of Java</td>
<td>-</td>
</tr>
<tr>
<td>Stroustrup, B.</td>
<td>The Design and Evolution of C++</td>
<td>-</td>
</tr>
<tr>
<td>Meyers, S.</td>
<td>Effective C++</td>
<td>-</td>
</tr>
<tr>
<td>Meyers, S.</td>
<td>Effective STL</td>
<td>-</td>
</tr>
<tr>
<td>Meyers, S.</td>
<td>More Effective C++</td>
<td>-</td>
</tr>
<tr>
<td>Gamma, E., Helm, R., Johnson, R. &amp; Vlissides, A.</td>
<td>Design patterns: elements of reusable object-oriented software</td>
<td>1</td>
</tr>
<tr>
<td>Bloch, J. &amp; Gatter, N.</td>
<td>Java puzzlers</td>
<td>-</td>
</tr>
<tr>
<td>Meyer, B.</td>
<td>Touch of class</td>
<td>-</td>
</tr>
</tbody>
</table>

above. The rating is added to each item in parenthesis. The specified concept is written in capital letters, the specifying concepts are in italic typeset and the main sentence is in typewriter typeset.

“These difficulties become even greater when we allow the possibility of concurrent execution of programs. The stream approach can be most fully exploited when we decouple simulated time in our model from the order of the events that take place in the computer during evaluation. We will accomplish this using a technique known as delayed evaluation. We ordinarily view the world as populated by independent objects, each of which has a state that changes over time. An object is said to “have state” if its behavior is influenced by its history. A bank account, for example, has state in that the answer to the question “Can I withdraw $100?” depends upon the history of deposit and withdrawal transactions.” [1]

(1-rated)

Java programs are written by combining new methods and classes that you write with predefined methods and classes available in the Java Application Programming Interface (also referred to as the Java API or Java class library) and in various other class libraries. Related classes are typically grouped into packages so that they can be imported into programs and reused. You’ll learn how to group your own classes into packages in Chapter 8. The Java API provides a rich collection of predefined classes that contain methods for performing common mathematical calculations, string manipulations, character manipulations, input/output operations, database operations, networking operations, file processing, error checking and many other useful operations. Familiarize yourself with the rich collection of classes and methods provided by the Java API (java.sun.com /javase/6/docs/api/).” [6]

(0-rated)

“These problems will likely be solved because the potential value of increased software reuse is enormous. Classes normally hide the details of their implementation from their clients. This is called information hiding. As an example, let us consider the stack data structure introduced in Section 6.6. Recall that a stack is a last-in, first-out (LIFO) data structure – the last item pushed (inserted) on the stack is the first item popped (removed) from the stack.” [6]

(1-rated)

4. APPLICATION

After developing the method we wanted to apply the method on a set of literature. Therefore we analyzed textbooks regarding the concepts of object orientation and object-oriented programming.

As described in [3], we first tried to identify textbooks that are recommended for introductory courses at more than one renowned university. The first step was the definition of a list of universities. For this purpose we started with the textbooks that are recommended by the nine technical universities of Germany for their introductory courses in computer science, where the first programming concepts are taught. For a better comparison with international universities, we restricted the list to books that are available in English. The resulting list contained only a few books. In consequence, we expanded the list of universities and added the “top-ten” of the QS World University Ranking 2011[4] for Engineering and Technology faculties. Following the analysis of the resulting 19 introductory courses, we received a list of 30 textbooks. Eleven were ignored because there was no English edition available. The remaining 19 books are shown in table[4]. The five books that are highlighted in gray color are used at more than one university. These were the books that we analyzed for the construction of our Concept Specification Map (see abbreviations in parentheses).

For every book we started with the concepts object-orientation, object-oriented programming and object-oriented design which were combined to one common starting point.

in the maps. Subsequently, each concept that was found to specify another was treated in the same manner. We ended up with a Concept Specification Map for each book that shows the interdependencies between the concepts. Please note that our maps contain only concepts that specify object-orientation directly or indirectly through several other concepts. The complete Maps are not included in this paper, because they are too large to be displayed here. They are available for download.\(^3\)

5. RESULTS

As mentioned above, we extracted text passages from the books and rated them with 1 if they contain a specification of a particular concept. The first concept we investigated in each book was “object-oriented programming” combined with “object-oriented design”. Overall, we examined 36460 text passages. This ranged from 839 passages in (AH) to 18726 passages in (DH). The ratio of 1-rated elements to all passages was 0.4% to 0.7%.

Table 2 displays, which concepts are referred to specify the term “object-oriented programming and design”. The books are ordered by the number of referred concepts.

<table>
<thead>
<tr>
<th>Abbr.</th>
<th>Book</th>
<th>Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>(EB)</td>
<td>Thinking in Java</td>
<td>class, composition, data abstraction, inheritance, message, object, polymorphism, reference</td>
</tr>
<tr>
<td>(DH)</td>
<td>Java: How to Program</td>
<td>class, composition, data abstraction, inheritance, message, object, polymorphism, visibility</td>
</tr>
<tr>
<td>(FD)</td>
<td>Java in a Nutshell</td>
<td>class, encapsulation, field, information hiding, inheritance, method, object</td>
</tr>
<tr>
<td>(AH)</td>
<td>Structure and Interpretation of Computer Programs</td>
<td>class, inheritance, state</td>
</tr>
<tr>
<td>(SR)</td>
<td>Introduction to Programming in Java</td>
<td>object, reference</td>
</tr>
</tbody>
</table>

Besides the concepts that specify object-oriented programming directly, we want to examine the concepts that have the most specifying concepts and those which specify the most concepts. In (DH) and (SR) we have class as the concept specified the most, whereas (EB) has object, (FD) has method and (AH) has state as their concept specified the most. The other interesting concepts are those which specify many other concepts. These concepts are: method/object (in (EB)), object (in (DH)), class/method/data (in (FD)) and state (in (AH) and (SR)).

The coverage ratio is another interesting aspect, we want to investigate. In this context coverage means that there is a direct or indirect connection to the concepts object oriented programming and design. In (EB) all concepts are covered, if we consider the concepts of message and message passing respectively data abstraction and abstraction to be the same. The textbook by Flanagan lacks the concepts message passing, polymorphism and abstraction. The other “quarks” are covered. In Deitel’s book only the concept abstraction has no connection to OOP. (SR) and (AH) both cover the concepts object and class. (SR) furthermore covers the concept method and (AH) covers the concept inheritance.

The last investigation, we did into the five books, was counting the “definition”-nodes and the concepts related to “object-orientation”. The number of nodes may vary from four nodes in (AH) to 41 nodes in (DH). The book that uses the most concepts (18) is (DH): object-oriented programming and design, “quarks” (class, encapsulation, inheritance, polymorphism, method, object, message) and “non-quarks” (visibility, association, composition, argument, reference, statement, parameter, information hiding, state, attribute, variable). The one with the least (6) is (AH): object-oriented programming and design, “quarks” (class, inheritance, object) and “non-quarks” (state, variable). An overview of the numbers of nodes and concepts of all the books is given in Table 3.

Table 3: Number of nodes and concepts

<table>
<thead>
<tr>
<th>Abbr.</th>
<th>Book</th>
<th># Nodes</th>
<th># Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>(DH)</td>
<td>Java: How to Program</td>
<td>41</td>
<td>18</td>
</tr>
<tr>
<td>(EB)</td>
<td>Thinking in Java</td>
<td>35</td>
<td>17</td>
</tr>
<tr>
<td>(FD)</td>
<td>Java in a Nutshell</td>
<td>28</td>
<td>17</td>
</tr>
<tr>
<td>(AH)</td>
<td>Structure and Interpretation of Computer Programs</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>(SR)</td>
<td>Introduction to Programming in Java</td>
<td>14</td>
<td>11</td>
</tr>
</tbody>
</table>

In addition to the separate analysis of the five books, we formed a common Concept Specification Map of all books. For this purpose we combined the list of all 1-rated passages and removed the duplicates, as we did when we were analyzing the books. This results in a quite complex map with 27 concepts and 98 “definition”-nodes. The three most specified, respectively specifying concepts in this conjunction map are object, class and method. Besides Inheritance and Encapsulation these concepts are the core concepts of object-orientation referring to Armstrong\(^2\).

Often the differences between two books that are both recommended in introductory programming courses are huge. The resulting maps differ in the number of concepts as well as in the number of “definition”-nodes. To show this, we present the Concept Specification Map (CSM) of (EB) in figure 4 and (AH) in figure 5.

\(^3\)http://eddi.informatik.tu-muenchen.de/ mahara/view/view.php?id=178

Figure 3: Concept Specification Map of (AH)
6. DISCUSSION

At first we will discuss the definition of object-oriented programming and design in each textbook.

Table 2 shows which concepts specify object-oriented programming and design directly. A specification does not mean that there is one explicit definition with all the given concepts in it. The specifications we examined are collected throughout the book. If we look at the first and the last book, there is a big difference in the number of concepts. For a course on introduction into object-oriented programming it is important that the concepts introduced in a corresponding textbook are related to the main topic to avoid misconceptions through misleading interdependencies.

Another thing that can be seen in table 2 is that the first four books appeal to almost the same concepts. Particularly class and inheritance are connected in all these books to object-oriented programming and design. The first three books have also object in common. These three concepts are the first three “quarks” defined in [2].

The books differ in their number of text passages. The ratio of 1-rated elements is almost the same for all books. So, the number of passages, we can use, depends on the number of text passages we find in the books. Because we start with the concept object-oriented programming and design and analyze only the concepts we found in the analysis, the number of investigated text passages is highly dependent on the number of interdependencies to the concepts of object-orientation. As can be seen in table 2 the books with the most investigated text passages have more concepts associated with object-oriented programming and design. For the method we selected to form our Concept Specification Maps it is important that we have a lot of interdependencies to object-orientation. With regard to the goal of clarifying the structure of the concepts associated to object-orientation in textbooks, this is not surprising.

Besides the aspect of giving an overview, our Concept Specification Maps represent a method to rate textbooks on their coverage ratio of the concepts of object-orientation. One first attempt for that is the coverage of the “quarks” Armstrong defined in [2]. As we have seen above, we can separate our books into two groups. The first three books shown in table 3 have a high degree of coverage of the concepts defined as “quarks”. The book written by Eckel[7] even covers all eight concepts. So we can say, that all the concepts that are relevant for the understanding of object-orientation are defined or at least specified in the book. In a book like the one by Sedgewick [18] or Abelson [1] the concept object-oriented programming and design is present but the main related concepts are not specified in the book. So, this coverage rate is of interest for the selection of a textbook for an introductory course in object-oriented programming.

If we assume the “quarks” to be the basis of object-orientation which has to be learned, the coverage ratio is also a good indicator for recommending it or not. Assuming “object-oriented programming and design” to be the central topic of a course the coverage ration shows how many concepts of the core of OOP are related to this topic in the textbook. For even better suitability we can weight the concepts on their importance for the course and build rankings of the books. As we analyzed the books without any corresponding course, our ranking results from the simple appearance in the text and is shown in table 2.

A concluding analysis of our five books focuses on the number of nodes and concepts in the Concept Specification Maps. Again the ranking of the books is based on the absolute number of nodes and concepts reflects the results of the investigations we described above. The equality of the number of concepts in the first three books (EB, DH, FD) is striking. They all consist of 17 or 18 concepts. The intersection of the lists of concepts leads to the nine concepts: OOP, method, object, class, encapsulation, inheritance, argument, parameter and variable. The last three concepts expand the list of the “quarks”. Except the first, the others cover the concepts defined by Armstrong. So we can conclude that these books have a kind of a core of object-oriented programming and design in common.

If we consider the equal number of concepts for the three books, the absolute numbers of nodes in the Concept Specification Maps show us the level of interdependency. The more connections between the concepts are given, the more different ways of specification is offered by the book. In our five books the one by Deitel[6] is the most detailed. The one with the most pages is Java in a Nutshell[8], but most of it is just a syntax reference.

7. CONCLUSION

After analyzing the five books, we consider our findings to provide an interesting method to analyze textbooks or other literature for the purpose of a detailed overview on concepts like object-orientation. For an overview of object-oriented programming and design in textbooks, the Concept Specification Maps seem to be a good tool, but the required effort is quite high. In particular, the rating of the text passages takes a long time and the ratio of the 1-rated passages is very small and therefore not very motivating.

The original idea of the Concept Specification Maps was to get an overview of the concepts which have interdependencies with other concepts of a given topic. When we were analyzing the passages and were drawing the Concept Specification Maps, we recognized some useful influence on course design.

On the one hand it is possible to analyze the materials, such as slides or other working materials of a course and compare the examined concepts with the corresponding concepts from textbooks. So our method can help to find a textbook that covers the most content of a course. It can also be used to analyze any other learning content.

Another aspect is the assessment of student knowledge. The Concept Specification Maps can show which concepts belong together and which concepts are prerequisites of others, if we assume the textbook to cover the content of a course. If we assess the concepts a student knows, we can visualize the degree of coverage of the concepts of object-orientation. Along with a method to find precedence relationships as shown in [18], we can recommend the following learning objectives.

7.1 Further research

In a next step we will try to simplify the syntax of the Concept Specification Maps (CSM). Currently, there are “definition”-nodes which connect several concepts to one concept. For a complete intersection CSM, we have to find a way to break up these “multi”-nodes into “single”-nodes where only one edge comes in and one goes out. These nodes can then be combined with each other if they connect to the same concepts.
An intersection Concept Specification Map could lead to a definition of a kind of a core of object-orientation which is covered by all books covering object-oriented programming and design to a well defined degree. Finding the final degree when we can say a book covers object-oriented programming and design will also be the content of our future research.

To get a broader view on textbooks in object-orientation we have to add more books to our list and analyze them in the way shown above. The results of the investigation have to be related in a closer way to our research on introductory courses in object-oriented programming.

8. REFERENCES