On concept map assessment methods and their application to teaching computer programming

Jeroen Keppens
School of Arts & Humanities, King’s College London
Submitted: March 2007

Introduction
An important aspect of effective teaching is careful assessment of the extent to which students have assimilated the material they were taught. Novak (2005) has devised a concept mapping methodology for this purpose. More precisely, concept mapping has been developed as a tool to assess student learning in a longitudinal to test the benefit of teaching abstract science concepts at an early age. Generally speaking, a concept map is a graph containing labelled nodes and labelled arrows between pairs of nodes. The nodes represent concepts that are identified by each node’s label. The arrows denote relationships, which are identified by each arrow’s label, between the concepts they link to one another. Figure 1, for instance, shows a sample concept map describing the main components of an implementation of a graphical user interface in a modern programming language. For example, it contains the concepts of “Layout Manager” and “User Interface Component” and a relationship describing that “Layout Managers” determine the locations of “User Interface Components”.

The development of this representational formalism was motivated by Ausubel’s assimilation theory (Ausubel, 1968). Ausubel hypothesised that learning is crucially dependent upon the learner’s preexisting awareness of concepts and their inter-relationships. This implies that any meaningful learning stems from the interaction between newly introduced information and the learner’s prior knowledge. He suggests that much of this interaction involves differentiation between alternative meanings and conflict resolution between new and old ideas. As such, the outcomes of meaningful learning may include a more precise classification of concepts and relationships, and the resolution of ambiguous or incorrect ones. As concept maps are explicit visualisations the concepts and relations between concepts that are relevant and valid to a particular domain in the view of an individual learner, they are an effective means to test Ausubel’s hypothesis.

Since their invention, concept maps have gained increasing popularity as a learning and organisational tool. In the domain of computer programming education in particular, research suggests that the development and incremental refinement and improvement of mental models of programming concepts and developed systems promotes meaningful learning (Mayer, 1981). However, the potential to assess a student’s evolving understanding of the domain of study, in this context, is often ignored (Ruiz-Primo & Shavelson, 1996). Moreover, assessment of concept maps remains a topic of ongoing research.

This essay discusses a range of different concept map assessment methods that have been proposed in the literature. It aims to examine their main feature and evaluate their suitability for teaching computer programming.
Assessment methods

Quantitative assessment methods
Quantitative assessment techniques provide a means to calculate a numerical score for a given concept map as a measurement of a student's understanding of a particular domain. The objective of such methods is to produce a total order of the different learners' understanding of the domain or numerical data that can be employed for statistical hypothesis testing. This subsection discusses a representative set of this category of methods.

Table 1: Structural scoring

<table>
<thead>
<tr>
<th>Concept map feature</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid hierarchical link between concepts</td>
<td>5 points each</td>
</tr>
<tr>
<td>Valid cross-link between concepts on different branches of a hierarchical structure</td>
<td>10 points each</td>
</tr>
<tr>
<td>Other valid links between concepts</td>
<td>2 points each</td>
</tr>
<tr>
<td>Examples of concepts</td>
<td>1 point each</td>
</tr>
<tr>
<td>Invalid concept or link</td>
<td>0 points each</td>
</tr>
</tbody>
</table>
Holistic scoring method

The holistic method instructs (expert) assessors to assign concept maps a score on a given scale, say 0 to 10, which expresses the concept mapper’s overall understanding of the domain. It does not supply any algorithm, heuristics or guidelines to calculate the score. The holistic method was originally devised by McClure, Sonak and Suen (McClure et al., 1999) as a control method to test the effectiveness of weighted average methods.

Weighted component scoring methods

Weighted component scoring methods assign partial point scores to certain concepts and/or links between concepts. The score associated with a concept map equals the sum of the partial point scores awarded to each component of that concept map. The values assigned to components may depend on their validity or the type of structure they add to the overall concept map. Two weighted additive component scoring methods have received substantial attention in the literature on concept map scoring: the structural and relational scoring methods.

The structural scoring method, devised by Novak and Gowin (1984) seeks to reward hierarchically structured knowledge. The method proposes a relatively small score for each valid link and each valid example of a concept. It rewards a substantially higher score to links that express a hierarchical relation, such as “is a kind of” or “contains” relationships. The highest scores are reserved for links between concepts that are located on different branches of a hierarchical structure. Table 1 summarises a sample structural scoring scheme.

The relational scoring method, devised by McClure, Sonak and Suen (1999), awards points to each link between concepts in isolation. Higher scores are assigned to links that are correctly labelled and ones that express a foundational relationship of the domain, such as taxonomical and causal relationships. Table 2 summarises a sample relational scoring scheme.

The closeness index

The closeness index, devised by Goldsmith, Johnson and Action (1991), is a heuristic that aims to calculate the similarity between a student’s and a teacher’s concept maps. The approach focuses on the concepts and links between concepts that two maps have in common, but it ignores the labels of the links. The closeness index of a concept c that the student’s and teacher’s maps have in common equals the number of concepts directly linked to c in both maps divided by the number of concepts directly linked to c in either map. The overall closeness index of two maps is the average closeness index over all nodes in those maps. In other words, the closeness index equals:

\[
\frac{1}{n} \sum_{c \in S \cap T} \frac{1}{n} \sum_{c \in S \cap T} \frac{n}{S \cap T}
\]

Table 1: summarises a sample structural scoring scheme

<table>
<thead>
<tr>
<th>Concept map feature</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid, but incorrectly labelled link between concepts</td>
<td>1 points each</td>
</tr>
<tr>
<td>Valid and correctly labelled link between concepts that does not represent a hierarchical, causal or sequential relationship</td>
<td>2 points each</td>
</tr>
<tr>
<td>Valid and correctly labelled link between concepts that does represent a hierarchical, causal or sequential relationship</td>
<td>3 points each</td>
</tr>
<tr>
<td>Link between concepts where no relationship exists</td>
<td>0 point each</td>
</tr>
</tbody>
</table>

Table 2: summarises a sample relational scoring scheme

<table>
<thead>
<tr>
<th>Concept map feature</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid, but incorrectly labelled link between concepts</td>
<td>1 points each</td>
</tr>
<tr>
<td>Valid and correctly labelled link between concepts that does not represent a hierarchical, causal or sequential relationship</td>
<td>2 points each</td>
</tr>
<tr>
<td>Valid and correctly labelled link between concepts that does represent a hierarchical, causal or sequential relationship</td>
<td>3 points each</td>
</tr>
<tr>
<td>Link between concepts where no relationship exists</td>
<td>0 point each</td>
</tr>
</tbody>
</table>

The Holistic scoring method

The holistic method instructs (expert) assessors to assign concept maps a score on a given scale, say 0 to 10, which expresses the concept mapper’s overall understanding of the domain. It does not supply any algorithm, heuristics or guidelines to calculate the score. The holistic method was originally devised by McClure, Sonak and Suen (McClure et al., 1999) as a control method to test the effectiveness of weighted average methods.

Weighted component scoring methods

Weighted component scoring methods assign partial point scores to certain concepts and/or links between concepts. The score associated with a concept map equals the sum of the partial point scores awarded to each component of that concept map. The values assigned to components may depend on their validity or the type of structure they add to the overall concept map. Two weighted additive component scoring methods have received substantial attention in the literature on concept map scoring: the structural and relational scoring methods.

The structural scoring method, devised by Novak and Gowin (1984) seeks to reward hierarchically structured knowledge. The method proposes a relatively small score for each valid link and each valid example of a concept. It rewards a substantially higher score to links that express a hierarchical relation, such as “is a kind of” or “contains” relationships. The highest scores are reserved for links between concepts that are located on different branches of a hierarchical structure. Table 1 summarises a sample structural scoring scheme.

The relational scoring method, devised by McClure, Sonak and Suen (1999), awards points to each link between concepts in isolation. Higher scores are assigned to links that are correctly labelled and ones that express a foundational relationship of the domain, such as taxonomical and causal relationships. Table 2 summarises a sample relational scoring scheme.
where \( n \) is the number of concepts that appears in one or both maps, \( S_i \) denotes the set of concepts directly linked to the \( i \)th of the \( n \) concepts in the student’s map, and \( T_j \) denotes the set of concepts directly linked to the \( j \)th of the \( n \) concepts in the teacher’s map.

**Qualitative assessment methods**

Qualitative assessment methods produce descriptive assessments of concept maps. Rather than aggregating concept map features into a single number, they make a synthesis of the various features and provide a descriptive diagnosis of the underlying extent of understanding. This subsection discusses a representative set of such methods.

**Linkage analysis**

Linkage analysis, devised by Liu, Don and Tsai (2005), aims to identify potential misconceptions of students by comparing the concepts each individual concept is directly linked to in a student’s and the teacher’s concept map of a particular domain. In this way, linkage analysis identifies certain symptoms that indicate potential misconceptions and may be able to suggest improvements to flawed concept maps.

For example, linkage analysis can identify potentially confused concepts. If a concept \( c_1 \) in the student’s map is linked to a set of concepts \( C \) while the teacher’s map contains a concept \( c_2 \) that is mostly connected to most of the concepts in \( C \), then the student may be confusing \( c_2 \) with \( c_1 \). In this case, \( c_1 \) is said to be a confused concept. If a student incorrectly links a concept \( c_1 \) to a set of concepts \( C \), while in the teacher’s map, a concept \( c_2 \) is connected to the concepts in \( C \), then it can be suggested to the student that \( c_1 \) may have to be substituted to \( c_2 \). Linkage analysis can also identify less obvious misconceptions. For example, when a concept \( c \) is correctly linked to other concepts in a set \( C \), but the concepts in \( C \) are incorrectly linked, then the student may have misunderstood \( c \) in the first place.

A set of algorithms has been developed to perform the above form of linkage analysis automatically for given student and teacher maps. Its purpose is to provide automated support for assessment in concept map based e-learning.

**Spoke, chain, net differentiation**

Kinchin and Hay (2000) propose to extract from concept maps, three types of substructure: spokes, chains and nets. In essence, a spoke is equivalent to a single level hierarchy, a chain corresponds to a sequence of concepts and a net denotes a substructure where a pair of concepts can be related to one another by means of different sets of concept links. Figure 2 illustrates these substructures by means of archetypical concept maps containing the same four computer programming concepts.

where \( n \) is the number of concepts that appears in one or both maps, \( S_i \) denotes the set of concepts directly linked to the \( i \)th of the \( n \) concepts in the student’s map, and \( T_j \) denotes the set of concepts directly linked to the \( j \)th of the \( n \) concepts in the teacher’s map.

**Qualitative assessment methods**

Qualitative assessment methods produce descriptive assessments of concept maps. Rather than aggregating concept map features into a single number, they make a synthesis of the various features and provide a descriptive diagnosis of the underlying extent of understanding. This subsection discusses a representative set of such methods.

**Linkage analysis**

Linkage analysis, devised by Liu, Don and Tsai (2005), aims to identify potential misconceptions of students by comparing the concepts each individual concept is directly linked to in a student’s and the teacher’s concept map of a particular domain. In this way, linkage analysis identifies certain symptoms that indicate potential misconceptions and may be able to suggest improvements to flawed concept maps.

For example, linkage analysis can identify potentially confused concepts. If a concept \( c_1 \) in the student’s map is linked to a set of concepts \( C \) while the teacher’s map contains a concept \( c_2 \) that is mostly connected to most of the concepts in \( C \), then the student may be confusing \( c_2 \) with \( c_1 \). In this case, \( c_1 \) is said to be a confused concept. If a student incorrectly links a concept \( c_1 \) to a set of concepts \( C \), while in the teacher’s map, a concept \( c_2 \) is connected to the concepts in \( C \), then it can be suggested to the student that \( c_1 \) may have to be substituted to \( c_2 \). Linkage analysis can also identify less obvious misconceptions. For example, when a concept \( c \) is correctly linked to other concepts in a set \( C \), but the concepts in \( C \) are incorrectly linked, then the student may have misunderstood \( c \) in the first place.

A set of algorithms has been developed to perform the above form of linkage analysis automatically for given student and teacher maps. Its purpose is to provide automated support for assessment in concept map based e-learning.

**Spoke, chain, net differentiation**

Kinchin and Hay (2000) propose to extract from concept maps, three types of substructure: spokes, chains and nets. In essence, a spoke is equivalent to a single level hierarchy, a chain corresponds to a sequence of concepts and a net denotes a substructure where a pair of concepts can be related to one another by means of different sets of concept links. Figure 2 illustrates these substructures by means of archetypical concept maps containing the same four computer programming concepts.
Spoke, chain and net substructures classify how well certain concepts are integrated in a learner’s mental models of the subject of study. Indeed, the substructures prescribe if and how a learner’s concept map collapses under the influence of new information that contradicts it.

In a spoke substructure, the learner has identified certain concepts that are related to a given core (or key organising) concept, but fails to identify how the former concepts are related to one another. As such, they may be unable to relate concepts to one another in situations that do not include the core concept. For example, the learner of the concept map of Figure 2(a) would be unable to specify what the attributes of a given object are, without referring to the object’s class.

A chain substructure is usually an indication of rote learning, as the sequence depicted by chain often corresponds to the order in which concepts were introduced in the lecture. Certain links in such substructures are inevitably tenuous, and may break down when confronted with new information. For example, when confronted with certain methods that do not manipulate attributes, the learner of the concept map shown in Figure 2(b) may be left wondering how attributes are related to objects.

In a net substructure, concepts are integrated with one another more strongly. Therefore, such substructures are more robust to contradictory information than spoke and chain structures. For example, the learner of the concept map of Figure 2(c) would not be expected to experience difficulty with the aforementioned sample problems. Evidence suggests that net structures indicate meaningful learning (Kinchin et al., 2005).
Qualitative simulation refers to a set of techniques devised to extrapolate the behaviour of physical systems in terms of qualitative descriptions. Like numerical simulation, it formalises system behaviour by means of mathematical models. But, the quantities that model's variables take over time are denoted using crude qualitative distinctions, such as “above zero”, “up to a local maximum” and “decreasing”. Biswas et. al. (2005) have devised a method to use qualitative simulation for the assessment of causal concept maps, i.e. concept maps in which all links describe causal relations with a specific, pre-defined semantics. The approach requires a somewhat narrowly defined concept mapping task. It involves the students in teaching an autonomous agent (i.e. an independent problem solving computer program), known as “Betty’s Brain”, about a particular type of system (such as a river’s eco-system) by defining the agent’s mental model by means of a causal concept map. The agent is then quizzed by a series of questions that require it to predict certain effects of changes in the system under investigation. If its predictions are invalid, another agent interacts with the student to explain the predications of “Betty’s Brain” and to help diagnose the error.

Concept map assessment in teaching computer programming

Evaluation criteria
Assessment methods are normally evaluated using a range of criteria. Some of these criteria, such as fairness and transparency, refer to the environment in which the assessment occurs. This essay will not consider such issues, focusing instead on the criteria that are primarily affected by the choice of assessment method and the subject of assessment. These are validity, reliability and efficiency. Table 3 summaries who this discussion can be applied to the assessment methods discussed herein.

Validity
The validity of an assessment method is the extent to which the measurements it produces are accurate reflections of what the method intends to determine. Intuitively, a valid assessment method is said to measure the right thing.

The concept map assessment methods surveyed herein appear to measure very different aspects of concept maps, ranging from the equivalences with a single concept map to broad structural features of concept maps. In empirical studies of quantitative methods, a method’s validity is normally defined based as its correlation with another assessment method (McClure et al., 1999; West et al., 2002). While this approach allows for the significance of the results to be validated, it can be flawed, especially if the validity of the method that others are compared against is not demonstrated. In this essay, a more qualitative approach will be taken by identifying whether the concept map features that the assessment methods examine are important to achieve the learning outcomes. One feature that is particularly relevant in this respect is the amount of variability of correct concept maps that the assessment method tolerates. This varies between disciplines and sometimes, within disciplines. Biglan defines a discipline’s hardness as the degree to which it contains a central body of theory that is universally accepted within its membership (Biglan, 1973). As such, methods that define valid concept maps more narrowly are more suitable for hard disciplines while methods that employ looser definitions are more suitable for soft disciplines.

The selection of application domains for the assessment methods surveyed herein confirms this hypothesis. Those that are primarily applied to disciplines classified to be hard in Biglan’s framework tend to classify relationships between concepts into correct and incorrect ones. Indeed, all of the quantitative assessment methods discussed herein have been applied primarily to (hard) science education. Those with substantial applications in soft disciplines do not impose such precise criteria. For example, the qualitative simulation approach of Biswas et. al. has been primarily applied to ecological modelling (Biswas et al., 2005), which
is a discipline that has reached little consensus regarding the theories it has developed. Clearly, assessment methods that rely on comparing a student concept map with a model or teacher concept map (e.g. the closeness index and linkage analysis) are entirely unsuitable for such disciplines. Also, techniques that rely on scoring the relevance of links between concepts may be difficult to apply validly to maps of soft concepts given that soft domain allows for more valid permutations of map structures.

Surveys suggests that most computer scientists consider their discipline to be a hard one (Clark, 2003). However, crucial sub-disciplines of software engineering, such as human-computer interaction and systems requirements analysis, are soft disciplines (Dix et al., 2004). For example, usability principles of user interfaces, such as predictability and consistency are difficult to define formally. They are recognised intuitively when confronted with an example of a user interface that exhibits these features. However, they may mean different things to different people. As such, concept maps of these aspects of computer programming correspond to mostly personal interpretations and, therefore, they may be assessed most effectively by methods that measure the maps’ sophistication rather than their similarity to a given model.

**Reliability**

The reliability of assessment methods refers to the consistency of the method between different raters. In empirical studies, it is normally defined as interrater correlation. Some of the assessment methods discussed herein have already been automated and most can at least be formalised by means of an algorithm and are, therefore, automatable. Obviously, when an assessment method is applied by a machine, reliability is perfect. Three methods, the holistic, structural and relational scoring methods can not be automated. For these methods, research suggests that the reliability of the holistic method is rather poor while that of the structural and relational scoring method varies between studies and application domains (McClure et al., 1999; West et al., 2002).

**Efficiency**

The efficiency of an assessment method is its resource-economy. The main resources required for concept map assessment are staff time for preparation of the concept mapping task and assessment and the actual assessment. Firstly, the preparation time can be substantial when a model concept map needs to be designed. Some methods, such as the closeness index and linkage analysis, require this. Other methods, such as the holistic, structural and relational scoring methods, benefit from a model concept map (McClure et al., 1999). Secondly, the assessment time can be negligible those assessment methods that are automatable. For the assessment methods that are not automatable, there is only little research into the relative efficiency of these approaches. Work by McClure et. al. suggests that the holistic and structural methods are somewhat more efficient than the relational one, though these results are specific only to one concept mapping task (McClure et al., 1999).

Surveys suggests that most computer scientists consider their discipline to be a hard one (Clark, 2003). However, crucial sub-disciplines of software engineering, such as human-computer interaction and systems requirements analysis, are soft disciplines (Dix et al., 2004). For example, usability principles of user interfaces, such as predictability and consistency are difficult to define formally. They are recognised intuitively when confronted with an example of a user interface that exhibits these features. However, they may mean different things to different people. As such, concept maps of these aspects of computer programming correspond to mostly personal interpretations and, therefore, they may be assessed most effectively by methods that measure the maps’ sophistication rather than their similarity to a given model.

**Reliability**

The reliability of assessment methods refers to the consistency of the method between different raters. In empirical studies, it is normally defined as interrater correlation. Some of the assessment methods discussed herein have already been automated and most can at least be formalised by means of an algorithm and are, therefore, automatable. Obviously, when an assessment method is applied by a machine, reliability is perfect. Three methods, the holistic, structural and relational scoring methods can not be automated. For these methods, research suggests that the reliability of the holistic method is rather poor while that of the structural and relational scoring method varies between studies and application domains (McClure et al., 1999; West et al., 2002).

**Efficiency**

The efficiency of an assessment method is its resource-economy. The main resources required for concept map assessment are staff time for preparation of the concept mapping task and assessment and the actual assessment. Firstly, the preparation time can be substantial when a model concept map needs to be designed. Some methods, such as the closeness index and linkage analysis, require this. Other methods, such as the holistic, structural and relational scoring methods, benefit from a model concept map (McClure et al., 1999). Secondly, the assessment time can be negligible those assessment methods that are automatable. For the assessment methods that are not automatable, there is only little research into the relative efficiency of these approaches. Work by McClure et. al. suggests that the holistic and structural methods are somewhat more efficient than the relational one, though these results are specific only to one concept mapping task (McClure et al., 1999).
Applications

The potential applications of concept mapping in teaching computer programming are threefold. They can be employed to assess:

- Knowledge of theoretical computer programming concepts. Programming languages employ a small number of mathematical concepts that students find difficult to understand (Hu, 2006). These concepts have very precise definitions, allowing for little room for interpretation. For example, the basic programming concepts of "class", "object", "attribute" and "method" have precise meanings and the number of ways in which they can be related to one another is limited, as illustrated in Figure 2. Given the small size of this domain, most students will be aware of most of the relevant concepts and consequently, research suggests that the symptoms of student misconception of this knowledge takes to form of subtle flaws in the concept maps, such as minor structural deviations from the correct definition (Liu et al., 2005).

We hypothesise that approaches that can compare teacher and student maps are more likely to produce useful information to diagnose student misconceptions. The closeness index and linkage analysis provide the most detailed assessment methods to perform such comparisons. Both approaches have been automated, but they require the construction of a model. However, the effort required to produce such maps is expected to be limited as most programming courses are restricted to a single language and number of distinct theoretical concepts in each programming language is relatively limited.

- Ability to use and form a synthesis of software libraries. While most modern software is large and complex, many software products use similar components. For example, most user interfaces of software use windows, icons, menus and pointers that are similar in appearance and behaviour. Such components are available through software libraries. Therefore, computer programmers need to be able to employ effectively large software libraries that implement these components. Although the relevant entities and their interactions are defined precisely for each software library in a so-called Application Programming Interface (API), effective programmers do not memorise them. Instead, they develop their own conceptual models of APIs, which enable them to quickly look up in documentation, the programming instruction for the functionality they require in their programs.

Student concept maps can reveal how effectively students will be able to navigate API documentation. Different types of programming problem require that components be combined in different ways. Hence, in order to be able to solve a wide range of problems, a programmer should be aware of alternative ways of relating API concepts to one another. It is expected that such knowledge manifests itself in the form of more sophisticated structures of API concept maps. Therefore, we hypothesise that concept map assessment methods that analyse the structural complexity of concept maps would be the most appropriate means of assessing this ability. Chain-spoke-net differentiation is designed to assess precisely these types of structural sophistication and, therefore, is expected to be a valid approach to assessing concept maps on this topic. This approach is also automatable and independent of a model concept map which suggests that it would also be reliable and efficient in this setting.

- Ability to use and form a synthesis of software libraries. While most modern software is large and complex, many software products use similar components. For example, most user interfaces of software use windows, icons, menus and pointers that are similar in appearance and behaviour. Such components are available through software libraries. Therefore, computer programmers need to be able to employ effectively large software libraries that implement these components. Although the relevant entities and their interactions are defined precisely for each software library in a so-called Application Programming Interface (API), effective programmers do not memorise them. Instead, they develop their own conceptual models of APIs, which enable them to quickly look up in documentation, the programming instruction for the functionality they require in their programs.

Student concept maps can reveal how effectively students will be able to navigate API documentation. Different types of programming problem require that components be combined in different ways. Hence, in order to be able to solve a wide range of problems, a programmer should be aware of alternative ways of relating API concepts to one another. It is expected that such knowledge manifests itself in the form of more sophisticated structures of API concept maps. Therefore, we hypothesise that concept map assessment methods that analyse the structural complexity of concept maps would be the most appropriate means of assessing this ability. Chain-spoke-net differentiation is designed to assess precisely these types of structural sophistication and, therefore, is expected to be a valid approach to assessing concept maps on this topic. This approach is also automatable and independent of a model concept map which suggests that it would also be reliable and efficient in this setting.
Table 3: Overview of features affecting the validity, reliability and efficiency of concept map assessment method

<table>
<thead>
<tr>
<th>Assessment method</th>
<th>Validity</th>
<th>Reliability</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Permitted variability</td>
<td>Objective of measurement</td>
<td>Preparation time</td>
</tr>
<tr>
<td>Holistic scoring</td>
<td>unspecified</td>
<td>unspecified</td>
<td>low</td>
</tr>
<tr>
<td>Structural scoring</td>
<td>moderate</td>
<td>map structure and labels</td>
<td>good</td>
</tr>
<tr>
<td>Relational scoring</td>
<td>moderate</td>
<td>individual relations and labels</td>
<td>good</td>
</tr>
<tr>
<td>Closeness index</td>
<td>very low</td>
<td>similarity to model map</td>
<td>automated</td>
</tr>
<tr>
<td>Linkage analysis</td>
<td>very low</td>
<td>similarity to model map</td>
<td>automated</td>
</tr>
<tr>
<td>Spoke, chain, net differentiation</td>
<td>high</td>
<td>map structure</td>
<td>automatable</td>
</tr>
<tr>
<td>Qualitative simulation</td>
<td>moderately</td>
<td>model predictions</td>
<td>automatable</td>
</tr>
</tbody>
</table>

- **Ability to construct concept maps.** Programmers are not only required to assimilate technological and mathematical knowledge. They also need to be able to learn the relevant knowledge about the domain which the software they develop is to be integrated. The crucial part of this learning process is modelling the domain knowledge. To that end, programmers routinely communicate with diagrammatic formalisms such as entity relationship diagrams, state transition diagrams and interaction diagrams. These different types of diagram can be conceived to be highly formalised versions of concept maps. Therefore, the ability to draw concept maps is a particularly useful skill for computer programmers.

Assessing software application specific concept maps is difficult. The purpose of these diagrams is not necessarily completeness, accuracy or structural sophistication. Their quality depends on how well they reflect an application's requirements. An effective way of testing this criterion is to explore the logical implications of the conceptual model for specific test cases. This approach has been proposed in the qualitative simulation based method, though for the restricted setting of causal models. Such an assessment method would match the learning outcomes of the modelling exercise and as substantial parts of the approach can be automated, with the important exception of test case generation, it is expected to be reliable and efficient as well. The development of such an assessment approach would therefore present a potentially useful piece of future research.

Table 3: Overview of features affecting the validity, reliability and efficiency of concept map assessment method

<table>
<thead>
<tr>
<th>Assessment method</th>
<th>Validity</th>
<th>Reliability</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Permitted variability</td>
<td>Objective of measurement</td>
<td>Preparation time</td>
</tr>
<tr>
<td>Holistic scoring</td>
<td>unspecified</td>
<td>unspecified</td>
<td>low</td>
</tr>
<tr>
<td>Structural scoring</td>
<td>moderate</td>
<td>map structure and labels</td>
<td>good</td>
</tr>
<tr>
<td>Relational scoring</td>
<td>moderate</td>
<td>individual relations and labels</td>
<td>good</td>
</tr>
<tr>
<td>Closeness index</td>
<td>very low</td>
<td>similarity to model map</td>
<td>automated</td>
</tr>
<tr>
<td>Linkage analysis</td>
<td>very low</td>
<td>similarity to model map</td>
<td>automated</td>
</tr>
<tr>
<td>Spoke, chain, net differentiation</td>
<td>high</td>
<td>map structure</td>
<td>automatable</td>
</tr>
<tr>
<td>Qualitative simulation</td>
<td>moderately</td>
<td>model predictions</td>
<td>automatable</td>
</tr>
</tbody>
</table>

- **Ability to construct concept maps.** Programmers are not only required to assimilate technological and mathematical knowledge. They also need to be able to learn the relevant knowledge about the domain which the software they develop is to be integrated. The crucial part of this learning process is modelling the domain knowledge. To that end, programmers routinely communicate with diagrammatic formalisms such as entity relationship diagrams, state transition diagrams and interaction diagrams. These different types of diagram can be conceived to be highly formalised versions of concept maps. Therefore, the ability to draw concept maps is a particularly useful skill for computer programmers.

Assessing software application specific concept maps is difficult. The purpose of these diagrams is not necessarily completeness, accuracy or structural sophistication. Their quality depends on how well they reflect an application's requirements. An effective way of testing this criterion is to explore the logical implications of the conceptual model for specific test cases. This approach has been proposed in the qualitative simulation based method, though for the restricted setting of causal models. Such an assessment method would match the learning outcomes of the modelling exercise and as substantial parts of the approach can be automated, with the important exception of test case generation, it is expected to be reliable and efficient as well. The development of such an assessment approach would therefore present a potentially useful piece of future research.
Conclusion
This essay has presented a survey of concept map assessment methods and examined their suitability in the context of teaching computer programming, with a particular focus on validity, reliability and efficiency of these methods. The survey has identified seven concept map assessment methods that differ considerably from one another: holistic scoring (12), structural scoring (14), relational scoring (12), the closeness index (6), linkage analysis (10), chain-spoke-net differentiation (9) and qualitative simulation (3). While application domains exist for each method that are particularly well suited to it, three applications of concept mapping in computer programming teaching have been identified and corresponding assessment method have been proposed. Firstly, the closeness index and linkage analysis were suggested as suitable assessment methods for determining a student’s understanding of a programming language’s basic concepts. Secondly, chain-spoke-net differentiation was put forward as an effective method to evaluate a student’s awareness of software libraries. Thirdly and finally, a novel qualitative simulation based approach was proposed to assess student’s model building ability.

References

Conclusion
This essay has presented a survey of concept map assessment methods and examined their suitability in the context of teaching computer programming, with a particular focus on validity, reliability and efficiency of these methods. The survey has identified seven concept map assessment methods that differ considerably from one another: holistic scoring (12), structural scoring (14), relational scoring (12), the closeness index (6), linkage analysis (10), chain-spoke-net differentiation (9) and qualitative simulation (3). While application domains exist for each method that are particularly well suited to it, three applications of concept mapping in computer programming teaching have been identified and corresponding assessment method have been proposed. Firstly, the closeness index and linkage analysis were suggested as suitable assessment methods for determining a student’s understanding of a programming language’s basic concepts. Secondly, chain-spoke-net differentiation was put forward as an effective method to evaluate a student’s awareness of software libraries. Thirdly and finally, a novel qualitative simulation based approach was proposed to assess student’s model building ability.

References