Automatic Extraction of Notions from Course Material

Michela Pedroni, Manuel Oriol, Bertrand Meyer, Lukas Angerer
Chair of Software Engineering
ETH Zurich
8092 Zurich, Switzerland
{michela.pedroni|manuel.oriol|bertrand.meyer}@inf.ethz.ch; angererl@student.ethz.ch

ABSTRACT

Formally defining the knowledge units taught in a course helps instructors ensure a sound coverage of topics and provides an objective basis for comparing the content of two courses. The main issue is to list and define the course concepts, down to basic knowledge units. Ontology learning techniques can help partially automate the process by extracting information from existing materials such as slides and textbooks. The TrucStudio course planning tool, discussed in this article, provides such support and relies on Text2Onto to extract concepts from course material. We conducted experiments on two different programming courses to assess the quality of the results.

Categories and Subject Descriptors
K.3.2 [Computers and Education]: Computer and Information Science Education – *curriculum, computer science education.*

General Terms
Theory, Standardization.

Keywords
Curriculum design, knowledge modeling, course design, ontology learning.

1. MODELING COURSES

Teaching is a highly personal endeavor. The human touch is essential: a course is not an engineering product, and will never be specified as precisely and rigorously as, for example, a computer program. Still, applying modeling techniques partly imitated from software and other engineering disciplines can provide many benefits, as evidenced by our use of TrucStudio [12], an open-source tool that produces simple teaching-specific domain models based on the concepts of *cluster*, *Truc*, and *notion*. Some of the foremost advantages of the approach are to allow instructors to define and plan their courses in a systematic way; facilitate conscious decisions on the topics to teach and those to leave out; improve the course structure by showing what notions are taught and when; support comparisons between variants of the same course (for example across different institutions); and assert that a course or curriculum complies to given requirements, for example those defined by accreditation organizations such as ABET, on the basis of sound, objective evidence.

The main issue with such a system is the amount of work required for producing the domain models. Tool support, even just semi-automatic, is essential.

Such tools exist in the context of ontologies. An ontology is a domain description that uses concepts and relations between these concepts. The teaching community has come to rely on ontologies particularly in online teaching, where they help classify "learning objects" and describe learning and teaching strategies [6].

This article proposes to use ontology learning techniques on existing textbooks or slides to identify the main concepts covered, and describes how TrucStudio takes advantage of the open-source tool Text2Onto [3] for this. In a case study to assess the approach, we processed teaching materials from two courses, an introductory one using Eiffel and an advanced one in Java, both held at ETH Zurich. The study gathers evidence on the quality of the extracted concepts with respect to the type of input (slides and textbooks).

Section 2 describes Trucs and TrucStudio in more detail; section 3 explains ontologies, ontology learning, and Text2Onto; section 4 describes our case study and analyses its results. The article concludes with ideas for future work.

2. TRUCS AND TRUCSTUDIO

Among the concepts underlying TrucStudio, Trucs and notions are units of knowledge at distinct levels of granularity, identified for their relevance to teaching the corresponding subject matter; clusters are knowledge areas gathering such units.

2.1 Trucs, notions, and clusters

A *Truc* (Testable, Reusable Unit of Cognition) captures a collection of concepts and operational skills proceeding from one central idea [9]. It is of general interest (independent of a specific course) and can be covered in one lecture or a small number of lectures. Typical Trucs for teaching object-oriented programming are "object", "class", "routine call", "argument passing", and "inheritance". A Truc description follows a standardized scheme including a summary, examples, specification of the Truc's role, possible applications, dependencies on other Trucs, benefits and drawbacks. The scheme also includes pedagogical elements: known common confusions when learning the Truc and typical assessment questions to determine whether a student has really mastered the concepts.

While Trucs capture the central teaching units of a course, finer-grained units of knowledge — *notions* — help refine the specification. A notion “represents a single concept or operational skill or facet of a concept” [12] and belongs to exactly one Truc. Its description contains a list of alternative names and a summary. A Truc may have a *central* notion, which then bears the same name. Examples of notions within the Truc “routine call” are: the central notion “routine call”, “multi-level routine call” (calls of the form o1.o2.o3.f), “unqualified routine call” (routine calls that do not specify their target).

A *cluster* is a loose collection of Trucs and/or other clusters. A Truc may belong to more than one cluster in the domain model; the set of all clusters forms a directed acyclic graph. Clusters represent knowledge areas and help keep large numbers of Trucs manageable.

The framework defines two types of relations between notions: *is-a* links and *requires* links. They make it possible to check the soundness of a teaching sequence and to detect prerequisite violations. Dependencies at the notion level contribute to dependencies at the Truc level: a Truc A *depends* on another Truc B if any of its notions *requires* a notion of B. As an example of links between notions, “unqualified routine call” is an “*is-a”* specialization of “routine call” and “routine call” requires the “return value”. As a consequence, the Truc “routine call” depends on the Truc “object”.

Notions and Trucs with their links define a two-layered graph (see Figure 1), which provides a domain model for organizing courses and curricula with TrucStudio.

This approach may be compared to such models as “Anchoring graphs” [8] and MOT [11]. Anchoring Graphs use cognitive load to identify dependency relationships between concepts. The main intent is to define a partially fixed teaching sequence. Unlike the Truc framework, this approach takes teaching decisions in the domain modeling phase. The MOT model uses six types of knowledge objects and six types of links to model knowledge structures. The strength of the Truc framework, in comparison to this knowledge modeling approach whose complexity may be necessary for the purpose of designing online education systems, lies in its simplicity.

2.2 TrucStudio

TrucStudio[1], the tool supporting the Truc framework, has two main parts: a domain modeling interface and a course design interface. The domain modeling interface shows a list of clusters, Trucs and notions on the left (Figure 1) and a manipulation interface on the right. Selecting one of the entities updates the manipulation interface, which can be either a form for editing the selected entity or a graphical view that shows a clustered notions graph (as in Figure 1). The graphical view automatically generates a layout for the graph and lets the user blend in or out any item of the graph (such as all the notions of a particular Truc, or all the *requires* links between notions). The user may also manually rearrange the nodes, export the image to a PNG file, and load or save the layout into an XML file format. The graphical view provides a restricted set of manipulating operations such as creating or removing clusters, Trucs, notions and links.

The course design interface provides a list of all courses managed through TrucStudio. A course contains a list of available time slots (e.g. every Monday 2 p.m. to 4 p.m. in the semester) and a list of lectures. A lecture presents a sequence of notions. TrucStudio provides a rudimentary soundness check for the notions covered in a lecture, reporting notions that are repeated in different lectures and violations of a notion’s dependencies. TrucStudio requires users to enter an estimate of the time needed for teaching the lecture and maps it onto the time slots to produce the course schedule.

The domain model and course design can be copied to and loaded from XML files. Also available is partial import from OWL, an ontology storage format. The computing ontology project [1], intended to support curricular planning by providing a domain description of the computing disciplines as an OWL file, is of direct interest here since a computing ontology might serve as a starting point for devising domain descriptions for courses.

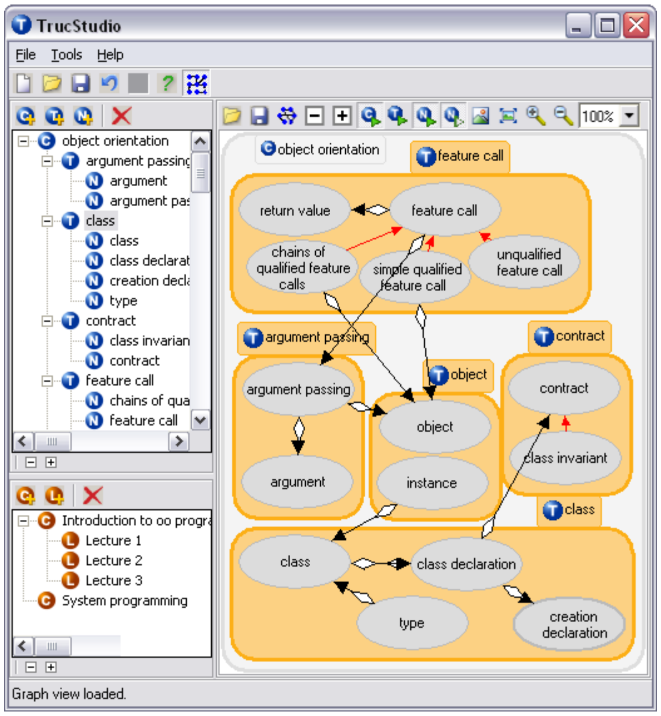


Figure 1 Screenshot of TrucStudio
(→: *is-a* link, ◇→: *requires* link)

3. ONTOLOGY LEARNING FROM TEXT

According to Gruber, an ontology is “an explicit specification of a conceptualization” [7]. The Trucs-notions framework, intended only for teaching applications, has a narrower scope than the general notion of ontology, but (because of this focus on a specific domain) a richer model within that scope.

Many researchers in e-learning advocate the use of ontologies to produce metadata information for learning objects [4], but only a few have considered applying to education *ontology learning* techniques (see e.g. [2]), which use machine learning and natural language processing to extract concepts and build is-a and other relationships from existing data, natural language descriptions [5].

Text2Onto [3], is one of the available ontology learning tools. It is open-source and targets data-driven change discovery using an incremental ontology learning strategy from text. Text2Onto can extract not only concepts and but also relations between these concepts, such as subclass-of, part-of, instance-of, equivalence.

[1] Available at http://trucstudio.origo.ethz.ch

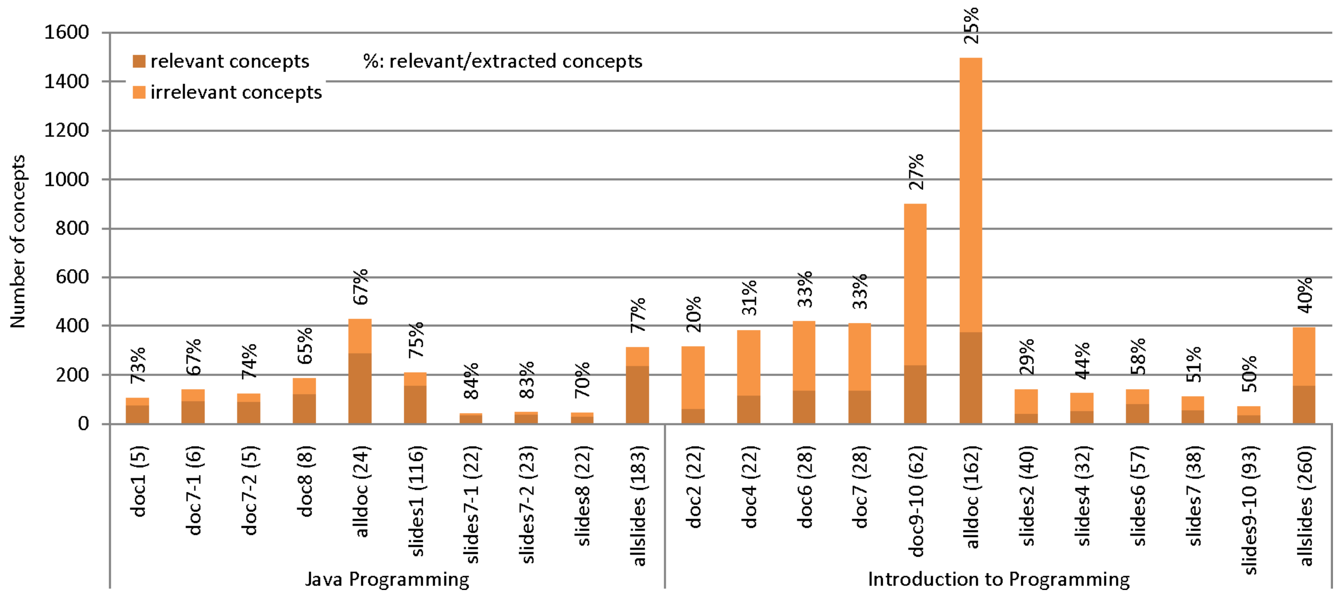


Figure 2 Relevant vs. irrelevant concepts

Preliminary tests suggested that the relation extraction mechanisms would not work well for teaching materials; so the present study focuses on concept extraction. Text2Onto provides several algorithms calculating relevance measures for each extracted concept. The tool can import documents in the usual teaching formats — PDF, HTML and plain text — and export its results in OWL, an ontology description format.

Using Text2Onto together with TrucStudio involves the following steps: (1) Call Text2Onto on a set of input documents. (2) Apply Text2Onto's concept extraction algorithm, calculating the relevance values for each of the concepts found. (3) Still in Text2Onto, export the resulting concepts into an OWL file. (4) In TrucStudio, import the OWL file and present the extracted concepts to the TrucStudio user, who can use them to define Trucs, notions and clusters.

4. APPLICATION TO TEACHING MATERIAL

The main goal of the case study was to answer the following questions: What inputs work best for the text processing (course slides or textbooks)? Can we generalize over different courses? How many concepts are extracted and thus need manual processing? What percentage of the extracted concepts are actually usable as Trucs or notions?

To answer these questions, the study used teaching materials from two courses: "Introduction to Programming" (slides[2] and accompanying textbook [10]) and "Java Programming" (slides and handouts[3]). Since the inputs to the ontology learning tool are the slides and textbooks available from the course web pages in PDF, the results are dependent on the quality of these documents.

[2] Available online at http://se.ethz.ch/teaching/ws2006/0001

[3] Available online at http://se.ethz.ch/teaching/ss2007/0284

"Introduction to Programming" is a required first-semester course for all computer science majors. It follows the "Inverted Curriculum" approach [13] using Eiffel and covers fundamental object-oriented and procedural concepts such as objects, classes, inheritance, and control structures. The teaching material has been used and refined over the past four years. Since the course faces a very diverse student body including many novice programmers, its teaching material contains many examples from a real-life domain: transportation in a city. An initial assumption is that ontology learning works better on the slides of this course than on the textbook because the textbook examples are more general and may include less immediately relevant concepts.

"Java Programming", an elective course, targets master-level CS students. Now in its second iteration, it covers advanced Java concepts such as threads, compilation, and sockets. The course page provides slides and reading material; in the slides, most code examples appear as screenshots. The reading material is a digest of the concepts covered in the course and is meant as complementary material. (In contrast, the Introduction to Programming textbook can be used independently of the course.) These characteristics suggest that the material from this course would produce better results in ontology learning.

Figure 2 lists on the x-axis the texts used as input to Text2Onto. For both courses, the study used a selected set of slides and the corresponding extracts from the reading material, with names starting with *slides* and *doc* respectively. The items *alldoc* and *allslides* show the results when Text2Onto processes all the reading extracts or all the slides of a course in a batch. Numbers in parentheses are the number of slides or pages.

4.1 Relevant vs. extracted concepts

All the listed reading extracts and slides were processed by Text2Onto, resulting in OWL files that contain the concepts and their estimated relevance. Instructors then categorized the extracted concepts as relevant or not.

Figure 2 shows the number of relevant and irrelevant concepts for each of the documents. The number at the top of each bar indicates the percentage of relevant concepts.

Extracted concepts. For both courses, the number of concepts extracted from slides is significantly lower than the number of concepts extracted from reading material. Slides generally present material in a condensed form, avoiding verbose explanations and full sentences. They reduce the text to include only the most relevant concepts and omit most of the noise found in reading extracts. As expected, the number of extracted concepts from slides is lower than from reading extracts.

The comparison of the numbers of extracted concepts between the two courses indicates that for slides the density of extracted concepts per slide is similar (for Introduction to Programming 2.7 concepts per slide; for Java Programming 1.9 concepts per slide). For the reading extracts of the Java course the density is higher (22.9 concepts per page) than for the textbook extracts of Introduction to Programming (15.1 per page). This can be traced back not only to the authors' different writing styles but also to differing purposes: while the reading extracts for Java Programming are complementary material summarizing the lectures' topics, the Introduction to Programming document is a self-contained tutorial involving thorough explanations and examples.

Relevant concepts. The higher number of extracted concepts for reading extracts (in comparison to slides) generally also results in a higher number of relevant concepts. A possible explanation is that some concepts important to an understanding of the "big picture" show up in the instructors' verbal explanations but not in the slides. Assuming that the slides as well as the reading material contain the fundamental concepts (Trucs), such missing elements can only be at the notion level. A recommendation could thus be to use slides as input if a coarse domain description suffices (providing the Trucs and most important notions), and reading material extracts if a more detailed domain description is needed.

More significant than the raw numbers of extracted and relevant concepts is the percentage of the extracted concepts that are meaningful for domain modeling. This measure reveals that for both courses the concepts extracted from slides are much more likely to be relevant than those extracted from reading material. Extracting the concepts on smaller documents provides more accurate results than using the entire set of slides or reading extracts as input to the ontology learning tool, but comes at the tradeoff of a much higher number of concepts to process.

The comparison of relevant concepts between the courses shows that for both the slides and the reading extracts the material of the Java course produces higher percentages. This result is also reflected in the density of relevant concepts per page or slide. For the slides of Introduction to Programming the number of relevant concepts per slide is 1.2 as opposed to 1.5 concepts per slide for Java Programming. The reading material exhibits the same tendency in an extreme form: the reading extracts from Java Programming contain 16.0 relevant concepts per page, while the Introduction to Programming textbook contains 4.4 relevant concepts per page.

4.2 Relevance measure

Generally, the number of raw extracted concepts is very high and results in significant manual work. Text2Onto provides a relevance rating, which can be used for sorting concepts by relevance. Assuming the rating is meaningful in the described setting, can it help removing concepts with low relevance?

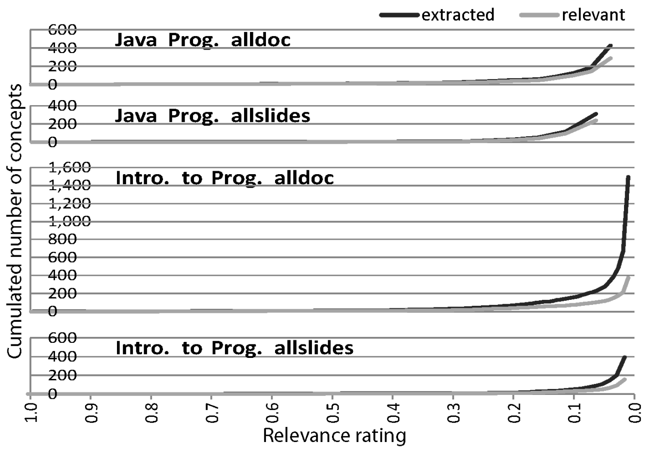


Figure 3 Cumulated concepts (extracted, relevant) per relevance value

To answer this question the first step is to analyze the distribution of cumulated extracted and relevant concepts for each present relevance value. Figure 3 shows that the cumulated number of extracted concepts per relevance value grows dramatically as the relevance gets close to zero: most concepts have a very low relevance rating. The power function $y=\alpha+r^{\beta}$ (with r the relevance rating, and α, β constants) provides a good experimental fitting of all the curves produced from the material of this study. A power function is also adequate to approximate the cumulated relevant concepts (see the grey curves in Figure 3), but grows slower than the cumulated extracted concepts.

It is possible to compute the parameters α and β without user interaction for cumulated extracted concepts, but not for cumulated relevant concepts since this requires sampling.

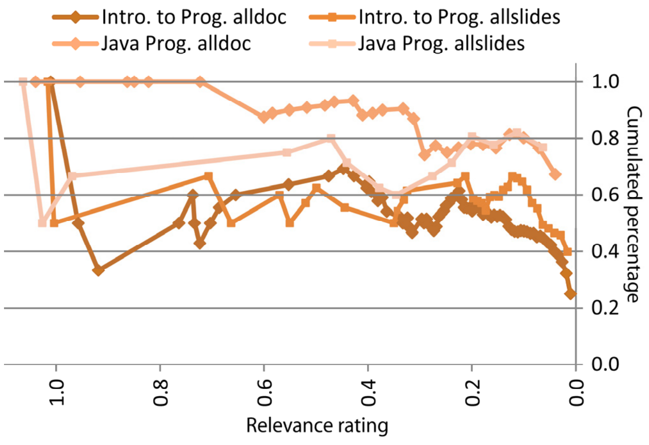


Figure 4 Cumulated percentage of relevant concepts

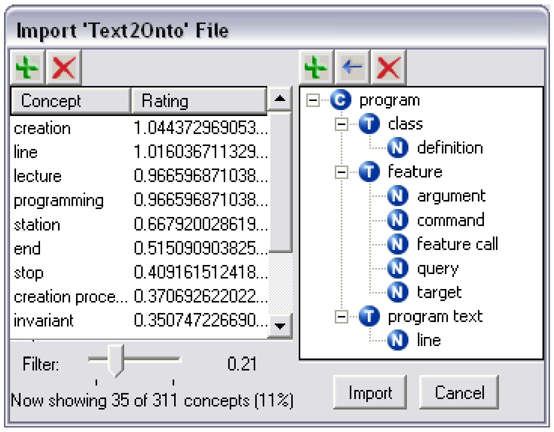
Figure 4 shows the percentage of cumulated relevant concepts for all the relevance values. This measure is interesting for the removal of concepts with a relevance rating below a certain limit. While the curves are generally very unsteady at high relevance values (due to the low number of extracted concepts at those values, see Figure 3), they exhibit a clear decrease pattern at lower values. This also applies to the curves for documents not included in Figure 4. This systematic pattern confirms that the relevance rating provided by Text2Onto is a valuable prediction measure, especially at low rating levels.

It seems impossible to find a fixed value that is suitable in all cases for optimizing the percentage of relevant concepts by removing concepts below this value.

4.3 Summary

The extraction of concepts from slides results in higher percentages of relevant concepts than their extraction from reading material. It is thus desirable to use slides for extracting concepts. The analysis shows that the rather dry style of slides and reading material from the Java Programming course is better suited for concept extraction than the illustrative and verbose style of the material from Introduction to Programming.

The relevance ratings provided by Text2Onto come out, in our setting, as a good predictor of relevance. Estimating the curves for cumulated extracted and cumulated relevant concepts predicts the percentage of relevant concepts for a certain relevance value, but requires user interaction to determine parameters for the curve fit of cumulated relevant concepts. The current user interface of TrucStudio does not implement this strategy, but provides a slider (see Figure 5) allowing users to select a value for the relevance cut point showing the number of concepts to investigate.



Import 'Text2Onto' File

Concept	Rating
creation	1.044372969053...
line	1.016036711329...
lecture	0.966596871038...
programming	0.966596871038...
station	0.667920028619...
end	0.515090903825...
stop	0.409161512418...
creation proce...	0.370692622022...
invariant	0.350747226690...

```
program
  class
    definition
  feature
    argument
    command
    feature call
    query
    target
  program text
    line
```

Filter: 0.21

Now showing 35 of 311 concepts (11%)

Figure 5 Import dialog in TrucStudio

5. CONCLUSIONS

TrucStudio is designed to help educators create and manage courses and curricula. The automatic extraction of concepts from course material is useful to support the creation of Trucs from scratch. Current ontology learning techniques cannot extract the Trucs automatically but can help with extracting notion and Truc names tagged with an estimated relevance value. Depending on how the material has been designed, the extraction can lead to quite accurate results (80% for slides with no code examples). The reading extracts generate more concepts, but with a lower accuracy. To increase the accuracy of presented concepts it is possible to show concepts with a higher relevance value only.

In the future, we would like to develop the techniques further and devise a tool that extracts the Trucs themselves out of textbooks and slides. This could be achieved by using finer-grained language processing techniques, more adapted to teaching material.

6. REFERENCES

[1] L. B. Cassel, A. McGettrick, and R. H. Sloan. A comprehensive representation of the computing and information disciplines. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, pages 199-200, Houston, Texas, USA, 2006.

[2] P. Cimiano. Ontology Learning and Population from Text - Algorithms, Evaluation and Applications. Springer US, 2006.

[3] P. Cimiano and J. Völker. Text2Onto - a framework for ontology learning and data-driven change discovery. In *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, volume 3513 of *Lecture Notes in Computer Science*, pages 227-238, Alicante, Spain, 2005.

[4] E. Duval and W. Hodgins. A LOM research agenda. In *Proceedings of the Twelfth International World Wide Web Conference*, *WWW2003,* Budapest, Hungary, May 2003.

[5] D. Fensel. Ontologies: a silver bullet for knowledge management and electronic commerce. Springer-Verlag, 2004.

[6] D. Gasevic, J. Jovanovic, and M. Boskovic. Ontologies for reusing learning object content. In *ICALT '05: Proceedings of the Fifth IEEE international Conference on Advanced Learning Technologies*, pages 944-945, Washington, DC, USA, 2005.

[7] T. R. Gruber. A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2):199-220, 1993.

[8] J. Mead, S. Gray, J. Hamer, R. James, J. Sorva, C. St. Clair, and L. Thomas. A cognitive approach to identifying measurable milestones for programming skill acquisition. In *Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education*, pages 182-194, Bologna, Italy, 2006.

[9] B. Meyer. Testable, reusable units of cognition. IEEE Computer, 39(4):20–24, 2006.

[10] B. Meyer. Touch of class - learning to program well with object technology and design by contract. Available online under: http://se.inf.ethz.ch/touch.

[11] G. Paquette. Meta-knowledge representation for learning scenarios engineering. In S. Lajoie and M. Vivet, editors, *Proceedings of AI-Ed99 AI and Education, open learning environments*. Amsterdam, June 1999. IOS Press.

[12] M. Pedroni, M. Oriol, and B. Meyer. A framework for describing and comparing courses and curricula. In *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, pages 131-135, Dundee, Scotland, 2007.

[13] M. Pedroni and B. Meyer. The inverted curriculum in practice. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, pages 481-485, Texas, USA, 2006.