Modeling Software Engineering Education with *i**

Michael Koch, Dieter Landes

Faculty of Electrical Engineering and Informatics
University of Applied Sciences and Arts
96450 Coburg, Germany
{michael.koch, dieter.landes}@hs-coburg.de

Abstract. Complex software systems play an increasingly important role in business and everyday life. Software engineering education needs to cover a wide range of technical and non-technical competencies and skills, while technologies and best practices in this area are evolving rapidly, enabling new applications for the industry. Therefore a crucial task in teaching software engineering is the incremental improvement and enhancement of courses. To make the complexity and evolution of software engineering education more transparent and traceable than through an unstructured documentation, we are searching for a way to model teaching goals. One promising candidate for modeling teaching goals is *i**. We applied *i** in a realistic setting and present a significant portion of the resulting model. We share some empirical experiences on using *i** in this setting, highlight possible barriers and limitations, and outline potential next steps.

Keywords. Software Engineering Education, Goal Modeling, *i**

1 Introduction

The influence of software in business and everyday life has become more and more important during the last years. Due to the rapid evolution of this domain and changing needs from industry, it is important to continuously evaluate and improve software engineering related courses. Since the boundaries between technical and software engineers become increasingly blurred, this issue is not only relevant for informatics students, but also in related disciplines such as electrical engineering or technical physics. Thus, the research project EVELIN (Experimental improVEment of Learning software engINeering) is devoted to incrementally improve the quality of software engineering education in various university programs.

To make the evolution of a course more transparent, comparable, and traceable, it is necessary to create and maintain a course profile including competencies, teaching goals, as well as tasks and resources to reach these goals from the perspective of an instructor.

Due to the high complexity of these profiles, a visual notation seems to be a better choice than an informal textual documentation which usually lacks adequate structure since interdependencies between relevant issues remains hidden somewhere in the text. Furthermore, graphical notations usually allow focusing on particular elements in the model while temporarily hiding others. Nevertheless, textual representations also have their merits to complement graphical notation – graphical notations are valuable for the big picture, while textual notations describe details precisely. Since competencies and teaching goals can be viewed as goals, goal modeling notations from Requirements Engineering are obvious candidates for a suitable visual notation. This contribution aims at getting some insights on how far one can get when course profiles are modelled using an existing notation such as *i**.

2 Theoretical Background

The *i** notation [1] is a popular approach for visually describing goals. This notation allows describing the relationships between goals, actors (in various roles), resources, and tasks. *i** is an agent-oriented approach for goal modeling [2]. Therefore it pays special attention to how goals may be defined and satisfied by various actors. It also offers possibilities to describe the relationships between various actors. Actors can be split up into agents (concrete persons) and roles. In our context, we focus primarily on students and lecturers as actors.

Competencies may be seen as (abstract or soft) goals of education and qualification [3]. They cannot be taught directly, but have to be acquired individually by each student. Lecturers can only provide knowledge and define tasks for students to exercise given competencies. A competency may be defined as a combination of skills, attitudes, and knowledge that enable a group or person to fulfill a role in an organization or society [4]. Referring to [5], we use competencies as a basis for deducing teaching goals. In our nomenclature, we distinguish between teaching goals (goals defined by the lecturer) and learning goals (goals set individually by each student), even though it is common to treat both terms as synonyms.

Various taxonomies defined for categorizing competencies and teaching goals have been proposed over time. One of the most popular taxonomies is Bloom's Taxonomy of Educational Objectives [6]. In spite of its shortcomings and various revisions and enhancements, Bloom's taxonomy is still fairly wide-spread. One well known revision of Bloom's taxonomy is the multi-dimensional taxonomy by Anderson and Krathwohl [7]. A domain specific taxonomy for software engineering education goals is introduced in [8], categorizing goals at six levels, namely "remember", "understand", "explain", "use", "apply" and "develop".

Course profiles are typically interrelated since basic courses establish the foundation for more advanced ones. Therefore it is common that lower-grade course profiles contain teaching goals on the understand-level, such as "be able to understand the concept of use cases". In contrast, teaching goals of higher grade course tend to be situated on the apply-level like "be able to apply the use case concept for given scenario".

3 Related Work

Monsalve and Leite [9] describe the potential of using *i** as an enabler for pedagogy transparency in the context of game-based learning for software engineering education. Their prime concern is to raise students' awareness of the teaching process and make learning more effective by telling students how they are being taught.

The description of course profiles and their evolution may also be an important source of domain knowledge for our ongoing work. In [10] we describe current work on CORE (Competency Repository), an intelligent knowledge base for software engineering education. In the future, this tool should assist lecturers and educational scientist in organizing and planning software engineering courses.

4 Modeling a Software Engineering Course

Software engineering education at Coburg University basically consists of three courses which build upon each other. In particular, a mandatory software engineering course (SE) establishes the basics, an elective course elaborates software modeling and software architectures (SMA) in more detail, and an elective capstone project synthesizes the acquired knowledge and competencies.

The SMA course contains a "Software Modeling" and an "Architecture and Testing" part. In this article we restrict ourselves to the software modeling section to limit complexity. Furthermore, the model expresses the perspective of instructors exclusively since they are in charge of improving their courses.

The software modeling section of SMA mainly focuses on advanced requirements engineering and has recently been reviewed and enhanced [11]. The most important contents are use cases and the treatment of functional and non-functional requirements, including categorization and prioritization. Students also learn to estimate the complexity and costs of a software project based on the requirements by using Function Points [12] and COCOMO II [13] and need to collaboratively write a requirements document for a fictional scenario. This article concentrates on these topics (see figure 1), even though SMA also covers business process modeling with Event Driven Process Chains (EPC), Petri Nets and Business Process Model and Notation (BPMN).

How can relevant aspects of this course be mapped appropriately to *i**? Competencies and teaching goals (on a higher level) may be fuzzy and intangible. Therefore we express them as soft goals. Abstract high level teaching goals as defined in [11] mainly represent competencies to be fostered and are represented as soft goals with a bold border (see figure 1). Competencies and teaching goals are connected with each other forming a course profile. Positive influences can be stated by using the contribution links "help" and "some+". Yet, we currently have no indication whether negative influences exist in this context or not. Exercises operationalize at least one teaching goal and can be expressed by the task element in *i**. Related artifacts or used software and hardware can be expressed by the resource element.

For brevity, the following SR diagram is only a small excerpt of the entire model, omitting several teaching goals, e.g. those about process modeling. In some cases, we model teaching goals or their representation as soft goals, occurring as leaf nodes, putting their decomposition into tasks in the background – at least for the time being, even though this is not recommended. For brevity, we also omit dependums in dependency links and only mention them inside the boundaries of actors.

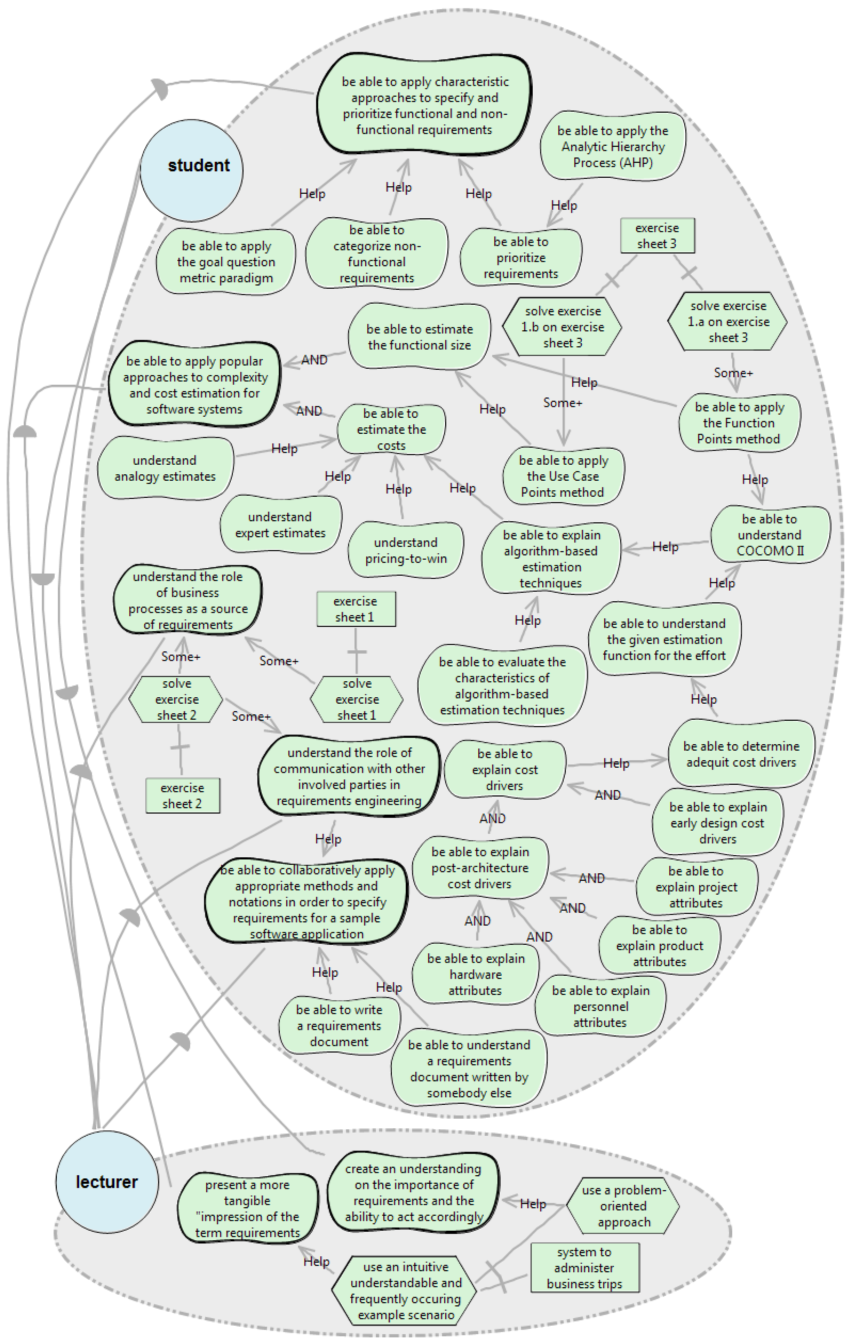
student
be able to apply characteristic approaches to specify and prioritize functional and non-functional requirements
be able to apply the Analytic Hierarchy Process (AHP)
Help
Help
Help
Help
be able to apply the goal question metric paradigm
be able to categorize non-functional requirements
be able to prioritize requirements
exercise sheet 3
solve exercise 1.b on exercise sheet 3
solve exercise 1.a on exercise sheet 3
be able to estimate the functional size
be able to apply popular approaches to complexity and cost estimation for software systems
AND
AND
Help
Some+
Some+
be able to estimate the costs
be able to apply the Function Points method
understand analogy estimates
Help
Help
Help
Help
be able to apply the Use Case Points method
Help
understand expert estimates
understand pricing-to-win
Help
be able to explain algorithm-based estimation techniques
Help
be able to understand COCOMO II
understand the role of business processes as a source of requirements
exercise sheet 1
Help
Help
be able to understand the given estimation function for the effort
Some+
Some+
be able to evaluate the characteristics of algorithm-based estimation techniques
solve exercise sheet 2
solve exercise sheet 1
Some+
Help
be able to determine adequit cost drivers
understand the role of communication with other involved parties in requirements engineering
be able to explain cost drivers
Help
exercise sheet 2
AND
be able to explain early design cost drivers
AND
Help
be able to explain post-architecture cost drivers
be able to collaboratively apply appropriate methods and notations in order to specify requirements for a sample software application
AND
be able to explain project attributes
AND
AND
AND
be able to explain product attributes
Help
Help
be able to explain hardware attributes
be able to explain personnel attributes
be able to write a requirements document
be able to understand a requirements document written by somebody else
lecturer
create an understanding on the importance of requirements and the ability to act accordingly
present a more tangible "impression of the term requirements
Help
use a problem-oriented approach
system to administer business trips
Help
use an intuitive understandable and frequently occuring example scenario

Fig. 1. Simplified SR model of teaching goals of SMA from a lecturer's perspective

5 Experiences

Even though we chose a relative small and clearly confined course, the resulting SR diagram is quite complex. Yet, we reached our goal to describe teaching goals and their relationships more clearly and intuitively than by a non-structured text only. The modeling process itself was also helpful to reflect the design of a course, e.g. the detection of isolated topics.

An issue that impacts diagram complexity is the question of whether a taxonomy with implicit relationships is used or not. For instance, the ability to apply a specific method presupposes an understanding of the respective method – thus, there is an implicit relationship between the two competencies (or teaching goals). Using a taxonomy with implicit relationships for goals that build on each other may significantly reduce the volume of a course profile and increase readability.

Our initial hypothesis was that all teaching goals defined by the lecturer give rise to corresponding expected learning objectives of an average student. Therefore, they can be positioned inside the boundaries of the actor "student". Yet, we found that teaching goals that are strongly dependent on teaching methods are mainly in the responsibility of the lecturer and therefore well placed inside his boundaries.

Since there is usually no standard way to achieve an educational goal, we can only describe aspects that help to achieve (or maybe inhabit) it, but it is hard to break the process down to hard goals and means-end links. In our context, this may be only possible on the task-decomposition level – in some cases.

Additional aspects that we would like to include in our model are the lecturers' considerations, hypotheses, or rationales which give rise to a teaching goal. This might be a use case for the belief element of *i**, but this is still an open question.

Due to the expected high complexity of those diagrams, we recommend a multi-layered view or at least a mechanism to hide or highlight relevant aspects.

6 Summary and Future Work

To make the complexity and evolution of software engineering courses more transparent and traceable than an unstructured textual documentation, we are searching for a way to model teaching goals. As teaching goals exhibit significant similarities to requirements, we analyzed the suitability of *i** for this purpose.

In a first step, we intentionally applied our theoretical concept to a course on advanced software engineering topics. Even though we have chosen a relatively compact software engineering course, we were confronted with a couple of limitations, leading to some open questions.

However, our overall experiences were positive and we see a high potential for using *i** in this context – at least in a tailored version. In addition to *i**, we will also analyze other goal-oriented approaches like KAOS [14] with respect to their suitability to model competencies and teaching goals.

In the near future, we plan to extend our software engineering education model by creating a course profile for the three mentioned software engineering courses at Coburg University, which build on each other.

Acknowledgements

The research project EVELIN is funded by the German Ministry of Education and Research (Bundesministerium für Bildung und Forschung) under grant no. 01PL12022A.

References

1. Yu, E.: Modeling Strategic Relationships for Process Reengineering. PhD thesis, Department of Computer Science, University of Toronto (1995)
2. Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J.: Social Modeling for Requirements Engineering: An Introduction. In: Social Modeling for Requirements Engineering, pp. 3-10, MIT Press, Cambridge (2011)
3. Flöttmann, H. et al.: Kompetenzen als Ziele von Bildung und Qualifikation - Bericht der Expertengruppe des Forum Bildung. In: Expertenberichte des Forum Bildung. Bonn, Germany (2002)
4. Paquette, G.: An Ontology and a Software Framework for Competency Modeling and Management. In: Journal of Educational Technology & Society 10(3), pp. 1-21 (2007)
5. Möller, C.: Technik der Lernplanung: Methoden und Probleme der Lernzielerstellung (3rd ed.), Beltz, Weinheim (1971)
6. Bloom, B.S.: Taxonomy of Educational Objectives Handbook I: Cognitive Domain, Longman, New York (1956)
7. Anderson, L., Krathwohl, D.A. (eds.).: A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Addison Wesley Longman, New York (2001)
8. Claren S., Sedelmaier, Y.: Ein Kompetenzrahmenmodell für Software Engineering. In: Proc. Embedded Software Engineering 2012, pp. 647-652, Sindelfingen, Germany (2012)
9. Monsalve, E.S., Leite, J.C.S.D.P: Using i* for Transparent Pedagogy. In: Proc. 6th International i* Workshop (iStar 2013), pp. 25-30, Valencia, Spain (2013)
10. Koch, M., Landes, D.: Design and Implementation of a Competency Repository. In: New Perspectives in Information Systems and Technologies, Volume 1, pp. 249-255, Springer, Heidelberg (2014)
11. Sedelmaier, Y., Landes, D.: Using Business Process Models to Foster Competencies in Requirements Engineering. In: Proc. 27th Conference on Software Engineering Education and Training (CSEE&T 2014), pp. 13-22, Klagenfurt, Austria (2014)
12. Garmus, D., Herron, D.: Function point analysis: Measurement practices for successful software projects, Addison-Wesley, Boston (2001)
13. Boehm, B.W.: Software cost estimation with Cocomo II, Prentice Hall, Upper Saddle River (2009)
14. Dardenne, A., Fickas, S., Van Lamsweerde, A.: Goal-directed Concept Acquisition in Requirements Elicitation. In: Proc. 6th International Workshop on Software Specification and Design (IWSSD'91), pp. 14-21 (1991)