

Feature Diagrams: A Tool for Specification of Learning Content for Microelectronics Education

Robertas Damaševičius, Vytautas Štuikys

Software Engineering Department, Kaunas University of Technology, Studentų 50-415, LT-51368, Kaunas, Lithuania

Email: robertas.damasevicius@ktu.lt, vystu@if.ktu.lt

ABSTRACT

Microelectronics teaching content is extremely wide and diverse, rich in different types of devices, circuits and systems, layers of abstraction, and design tradeoffs. We propose to use Feature Diagrams for specification and structuring of microelectronics teaching content at different layers of abstraction starting from organization of teaching material in a lecture down to specification and demonstration of particular hardware components with variability in mind.

1. INTRODUCTION

Microelectronics education must provide electrical and computer engineers with a solid understanding of how to analyze and design digital circuits. Microelectronics teaching content is extremely wide and diverse, rich in different types of devices, circuits and systems, layers of abstraction, and design tradeoffs. Following the Bologna Declaration [1], comparable teaching methodologies must be developed for microelectronics education as well.

Currently, Learning Objects (LOs) [2] are leading as a technology of choice for e-Learning support due to its potential generativity, adaptability, and scalability [3]. LOs are computer-based teaching components that are influenced by the object-oriented (OO) paradigm in computer science. However, the development of LOs remains a vague issue, because there is still no clearly defined and widely adopted LO specification and development methodology as, e.g., in software engineering, where classes and objects are modeled using UML [4]. There have been efforts to adopt UML in e-Learning domain, e.g., for modeling the interaction between LOs and a specific Learning Management System (LMS) [5], or to describe the content and process within "units of learning" in order to support reuse and interoperability [6], however, they have not been entirely successful, because of inherent UML's orientation towards a very specific technological paradigm with specialized concepts such as "concurrency" or "polymorphism" [7].

The main idea of LO is to break educational content down into small chunks that can be reused in various learning environments [8]. When reused, such small chunks are combined in various ways leading to a great variability of the learning content. Such variability cannot be modeled using UML, which has not adequate means for expressing different variants of configurations of a system.

We propose to use Feature Diagrams (FDs) for modeling learning content. Originally, FDs were introduced in FODA [9]. FDs first were applied in the context of industrial manufacturing product lines (PLs), e.g. for modeling car assembly lines. Later, it was extended to modeling software PLs [10]. The concept, if applied systematically, allows for increase of software design quality, productivity, provides a capability for mass customization and leads to the "industrial" software design [11].

Based on the success of feature modeling and PL approach in software engineering domain and intention to

introduce product families in System-on-Chip (SoC) design [12], we introduce FDs in e-Learning domain for specification, representation and structuring of learning content with variability in mind. We hope that feature modeling of LOs would ease maintenance of learning content, reduce its redundancy and duplication, allow for easy customization when applying for different teaching aims, student groups, and e-learning environments, and provide a global framework for reuse of LOs.

The novelty of this paper is systematic application of FDs for specification of the learning content at different layers of abstraction starting from organization of teaching material in a lecture down to specification and demonstration of particular hardware components.

2. FEATURE MODELING

There are several definitions of what a *feature* is. FODA defines a feature as a prominent and distinctive user visible characteristic of a system [9]. When comparing to other conceptual abstractions (such as function, object, and aspect), features are externally visible characteristics, whereas functions, objects, and aspects have been mainly used to specify the internal details of a system. Therefore, feature modeling focuses on identifying external visible characteristics of products in terms of commonality and variability, rather than describing all details of a system.

The result of feature modeling is a *feature model*. Features are identified and classified in terms of capabilities, domain technologies, implementation techniques, and operating environments. Capabilities are user visible characteristics that can be identified as distinct services, operations, and non-functional characteristics. Domain technologies represent the way of implementing services or operations. Implementation techniques are generic functions or techniques that are used to implement services, operations, and domain functions. Operating environments represents environments in which applications are used.

Features are primarily used in order to discriminate between product instances. Common features among different products are modeled as *mandatory* features, while different features among them may be optional or alternative. *Optional* features represent selectable features for products and *alternative* features indicate that no more than one feature can be selected for a product.

A FD is a graphical AND/OR hierarchy of features that captures structural or conceptual relationships among features. A FD consists of a set of nodes, a set of directed edges and a set of edge decorations. The nodes and edges form a tree. The edge decorations are drawn as arcs connecting subsets or all of the edges originating from the same node. Effectively edge decorations define a partitioning of the sub-nodes of a node (divide sub-nodes into a number of disjoint subsets). The root of a FD represents a concept. Features can be mandatory (and-features), optional, alternative or or-features (see Table 1).

And-features denoted with a filled circle. Or-features are denoted with an empty circle. Extension points are features that has at least one optional sub-feature, an edge ending in an empty circle, or at least one set of direct or-(sub)features. Extension points with optional features are simply denoted as edges ending in an empty circle. Extension points with or-features use a filled decorated edge arc, the edges ending in a filled circle to denote the mandatory features.

Three types of relationships are represented in a FD. The composed-of relationship is used if there is a whole-part relationship between a feature and its sub-features. In cases where features are generalization of sub-features, they are organized using the generalization/specialization relationship. The implemented-by relationship is used when a feature (i.e., a feature of an implementation technique) is necessary to implement the other feature. Moreover, features have dependencies: the selection of one feature may rule out (mutual exclusion) or assume (requires, includes) the inclusion of another feature.

Table 1. Types of features in a feature model

Feature type	Definition and formal description	Graphical notation
<i>Mandatory</i>	Features C and D are included if their parent A is included if A then B and C;	
<i>Optional</i>	Features C or D may be included if their parent A is included if A then B or C or no feature	
<i>Alternative</i>	At least one feature from (B, C, D) has to be selected if their parent A is selected if A then any of (B, C, D)	

Aggregation of features represents all variability within the domain and is thus the domain feature model. The derivation of a product consists of traversing the feature tree in an orderly manner and selecting the optional features. The result is a product description containing all features in the product (a *feature configuration*). The product specification is complete when all features have been selected.

FDs provide a concise and explicit representation of variability. They guide the choices to be made for determining the features of specific products and facilitate the reuse of software components implementing these features. We apply FDs as a tool for representing learning content and connecting between learning aims, teaching materials and hardware components.

3. CONCLUSION

From the methodological perspective, there is the need for specification of LO development requirements at a higher abstraction level. Feature Diagrams (FDs) is a domain, application, and implementation technology independent graphical language serving as a tool for:

1) *teachers and learners*, as a means for graphical representation of knowledge of the entire family of related learning objects (LOs) by providing domain ontologies using feature concepts, their types, values and relationships;

2) *designers and programmers*, to specify and express the variability-communality relationships of LOs at a higher abstraction level in order to develop and implement the generative LOs (GLOs) systematically;

3) *researchers and other actors*, as a vehicle for analysis and better understanding of the LO domain itself because FDs enable to express granularity, compositionality, and context explicitly to support reusability.

Since LOs are entities that 1) contain a variety of attributes with complex relationships, 2) have to be considered from different contexts (design, retrieval, learning, etc.) and perspectives (designer's, teacher's, learner's), LOs can be represented and modeled using FDs. Using FDs we can identify and model various kinds of ontologies related to LOs. In contrast to the UML notation, the syntax and semantics of FDs are simpler and thus can be easily learnt by different stakeholders (course designers, experts, teachers and learners, etc.).

FDs also have some limitations (lack of maturity, expressiveness). The LO domain requires extension of FDs (contextualization and relationships to express various ontologies may be useful). Further research is needed in order to exploit benefits and to overcome limitations of FDs. Our further work will be directed to connect the high-level specification of GLOs with the implementation technology that allow generating on demand LO instances derived from the generative specification of a LO.

REFERENCES

- [1] Bologna Declaration (1999). <http://ec.europa.eu/education/policies/educ/bologna/bologna.pdf>
- [2] Learning Technology Standards Committee. IEEE Standard for Learning Object Metadata. IEEE Standard 1484.12.1, Institute of Electrical and Electronics Engineers, New York, 2002.
- [3] D.A. Wiley. Learning Object Design and Sequencing Theory. PhD Thesis, Department of Instructional Psychology and Technology, Brigham Young University, June 2000.
- [4] M. Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley, 2003.
- [5] J. Gao, E. Marchetti, and A. Polini: Applying advanced UML based testing methodology to e-learning. Proc. of the IADIS Int. Conf. on Applied Computing, Algarve, Portugal, February 22-25, 2005, IADIS AC 2005: 74-79.
- [6] P. Laforcade. Towards a UML-based educational modeling language. Fifth IEEE Int. Conf. on Advanced Learning Technologies, ICALT 2005, 5-8 July 2005, 855 – 859.
- [7] N. Friesen. Three Objections to Learning Objects and E-learning Standards. In R. McGreal (Ed.). 2004. Online Education Using Learning Objects. London: Routledge, pp. 59-70.
- [8] D.A. Wiley. Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy, in D.A. Wiley, ed., *The Instructional Use of Learning Objects*, 2000.
- [9] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. TR CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, November 1990.
- [10] P. Clements, and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2002.
- [11] J. MacGregor. Requirements Engineering in Industrial Product Lines. Proc. Int. Workshop on Requirements Engineering for Product Lines REPL'02, Essen, Germany, 2002, 5-11.
- [12] B. Bailey, G. Martin, T. Anderson (eds.). *Taxonomies for the Development and Verification of Digital Systems*. Springer, 2005.