Model-Based Software Design Practice

Barbara Bernal Thomas[1]

Abstract – Current software engineering practices involve modeling as a key activity in the building of efficient and usable software systems. Software development's initial phases include the creation of a complete model of the user group, followed by comprehensive formulation of the task models which comprise the system. These are integrated into the conceptual model, which is evaluated and validated with usability inspection methods. The relationship between building conceptual models and developing systems from these solutions provides a sound basis for industrial-strength software engineering called model-based software engineering (MBSE). This paper illustrates the process of two fundamental, complementary sets of activities from domain engineering and application engineering. Discussions regarding application of traditional engineering principles to known solutions will be compared to the framework of model-based software engineering. A global view of the duality of domain and application engineering is provided with the relevant aspects of the modeling activities.

Keywords: Software Engineering, Software Design, Software Development, Model Based, MBSE.

INTRODUCTION

The life of a software system from its conception to its implementation is governed by well defined processes which impose consistency, structure, and control. Process standards provide guidelines for teams to develop software and may be adapted and tailored to fit an organization's particular needs. Using various process standards allows an organization to establish its own procedures, methods, tools, and environment for the software development project. The purpose of this paper is to describe a well-defined methodology for the development of software using a model-based software engineering (MBSE) practice. The pertinent industry standards for modeling activities, modeling languages, development methods, and deliverables from each phase of the software life cycle are focused with modeling principles. Building the efficient and usable software products is partitioned into two parts:

1. Domain Model
2. Application Model

The domain model is decomposed into the conceptual activities related to the user and the task. The user related activities explore and define all human, machine, and/or system interaction with the proposed software system. The task related activities comprise the complete, unambiguous, and validated description of what the software system will achieve. Analyzing these task related activities yield multiple models which together form the domain model for the proposed software system. Finally, the synthesis of solutions from the domain model are proposed, investigated, and validated to create the application model.

This practice allows software engineers to model the behavior of the users and system, design the software and model its behavior, and then simulate the entire system model to accurately predict and optimize performance. The system model becomes a specification from which you can automatically generate real-time software for testing, prototyping, and embedded implementation, thus avoiding manual effort and reducing the potential for errors.

[1] School of Computing & Software Engineering, Southern Polytechnic State University, Marietta, Georgia 30060, bthomas@spsu.edu

Teams trained in MBSE benefit from the visual, interactive environment in which to define, build, manage, and simulate the model. The graphical, hierarchical nature of this practice lets software design teams functionally model, accurately document, and effectively communicate their designs, reducing the risk of misinterpretation or misunderstanding. Changes or corrections to the system requirements and specifications are easily incorporated into the model. Figure 1 shows an overview of the approach to define models in the domain engineering which then translates to the solutions in the application engineering. The building of the solutions has multiple testing checkpoints as do the domain activities. Interactive feedback from users is present in both. Iteration is present between the domain and the application activities. [Rozenblit 2]

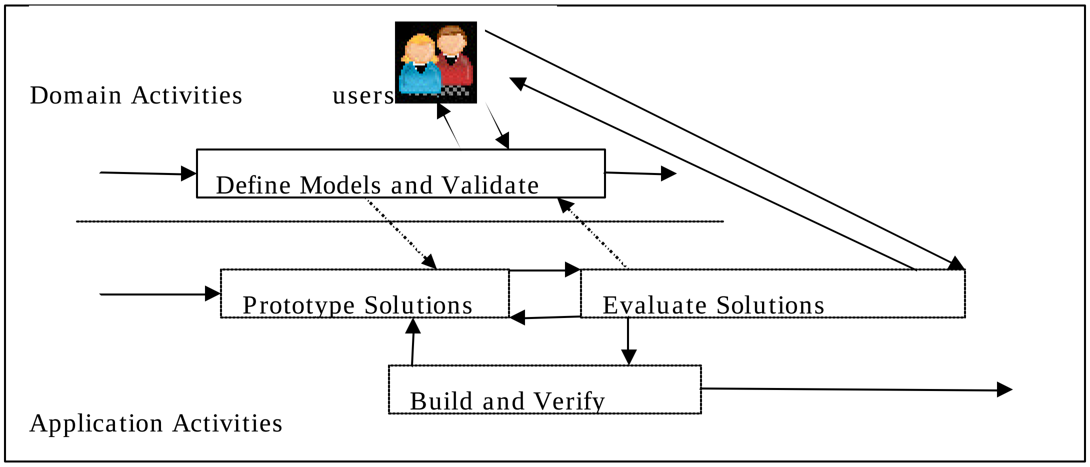

Figure 1. Project and Engineering Activities

DOMAIN MODEL

The Domain model is first initiated with the user's model which is a conceptual representation of the user and their tasks. Through the creation of a user profile, the personal and educational characteristics of the user are incorporated into the user's model. The task characteristics of frequency, complexity, structure, decomposition, etc. are described to the client and validated. A requirement mo deling language describes the ways in which the application is to be used, thus developing a final conceptual representation of 1) the User's model.[Jo 1] Several additional models complete the Domain model activities: 2) the Analysis model describes the basic classes for the system; 3) the Design Model portrays functional and non-functional representations 4) the Implementation model describes the organization of the software system; and 5) the Test model consists of test components, test procedures and test cases. These multiple models with their definitions and descriptions comprise the domain activities and are represented in figure 2 below.

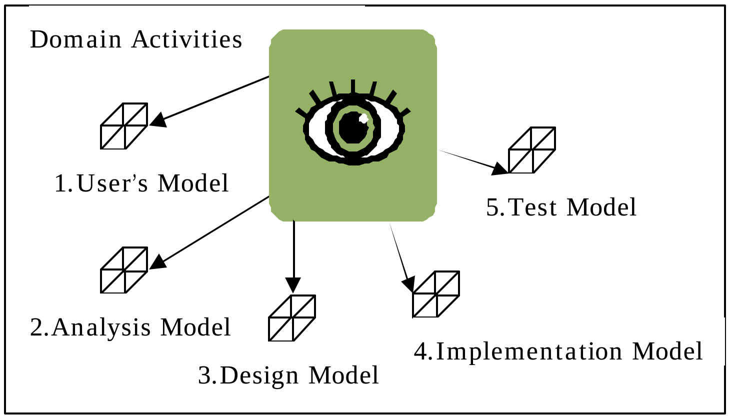

Figure 2. Domain Activities

APPLICATION MODEL

The final construction of efficient and usable software systems occurs within the application models. These models are iteratively derived from the multiple domain models created previously. The user's models contribute to the requirements phase, the analysis and design models to the design phase, implementation models to implementation and testing models to testing. Figure 3 shows this approach beginning in the center with the domain models used as blueprints for the development into the final solution application.

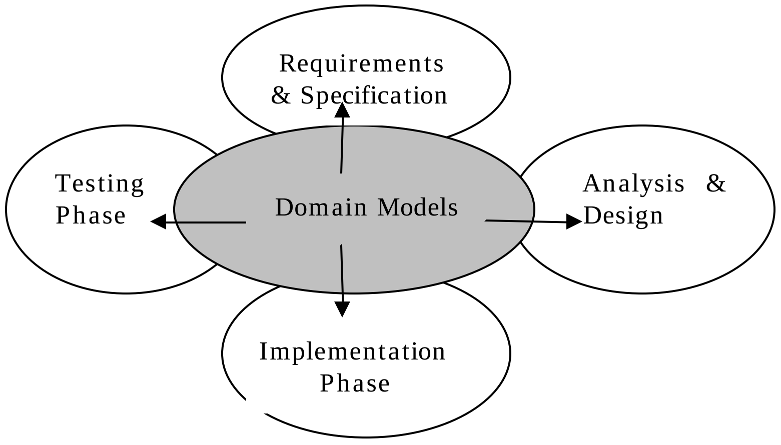

Figure 3. System Development Process

Development traditionally employs formal, analytic, top-down engineering approaches or informal, synthetic, bottom-up designer approaches. In the formal approach, individual blocks are designed and tested within the context of the top-level, thus, block level interfaces and functionality are both verified at the top-level. In this development flow, there is only one test bench and the number of synthesis and simulation scripts is minimized. In the second approach, the blocks are designed and tested individually before they are integrated together to create the top level system. Both are based on specific development representations tailored to specific qualities and needs. Providing opportunity for both traditions within MBSE enables dual reinforcement strategies to reach the final solution.

To better understand the use of model representation to engineer solutions a framework is developed for the application model. The model representation is classified along three dimensions:

- the perspective (problem- or solution-oriented) of the representation,
- the granularity of the objects described, and
- the degree of formality of the representation and its language.

The model representation classification framework has guided the development of the application model for the three main perspectives of user, tasks, and system. The framework has also been used for evaluating the proposed application models (prototypes) to reach the final application model. Specific views are examined from problem to solution; from high-level to low-level; from low-level to high-level; from formal to informal; and from informal to formal.

Set-based conceptual modeling is used for domain modeling within all these perspectives. The task modeling language is designed as a hybrid of flow-oriented process languages and traditional hierarchical sequence-oriented task languages. Key features are tight integration with the domain modeling language, expressive and flexible mutation and support for classifications of task structures.

An advantage of model-based software engineering is the close relation of the design to the user and task requirements of the application. Software engineers improve efficiency with the opportunity to refine the solution algorithms through model simulation and client evaluation of the simulations in the prototyping phase.

CONCLUSIONS

Model-based software engineering was discussed along two main divisions, the domain model with conceptual activities, and the application model with development activities. The domain model focuses on articulating knowledge, problems and solutions. The application depends upon how organizations choose to: structure the development, engineer proposed solutions to the final solution, and evaluate and verify the software system. Although model-based software engineering is a relatively new practice in the software development industry, it is an important process which mimics the natural style of human thinking. People have goals that govern their production of work. All work is processed into human thinking as models which are analyzed for the best possible solution.

MBSE will become important in the development of software systems by teams due to the easy model representational development and easy test simulation. The fundamental key activities of building domains and applications are processed and modified iteratively and evolutionally. This strategy emphasizes the design and test phases, creating opportunities for multiple development approaches. The use of modeling and simulation unifies the engineering design activities. The practice of model-based software engineering, with its methods of creating efficient and usable software systems, will play a key role in the future of industrial software design.

REFERENCES

[1] Jo, Chang-Hyun, Chen, Guobin, and Choi, James, "*A New Approach to the BDI Agent-Based Modeling*," SAC'04, Cyprus, 2004, pg. 1541-1542.

[2] Rozenblit, Jerzy, "*Integrative, Model-Based Engineering Design*," 1997 Workshop on Engineering of Computer-Based Systems (ECBS '97), 1997, pg. 476-477.

[3] Sharma, Sudhir K., "*Hardware Development and Model-Based Design,*" Dr. Dobb's Portal, November 24, 2006.

[4] Traetteberg, Hallvard, *"Model-based User Interface Design," Doctoral Thesis*, NTNU, Norway, 2002.

Barbara Bernal Thomas

Barbara Bernal Thomas is a full professor in the School of Computing and Software Engineering at Southern Polytechnic State University (SPSU) for the last 20 years. Her primary teaching areas are Software Engineering, User-Centered Design, and Computer Graphics & Multimedia. Barbara is directly involved in the Usability Center at SPSU which accepts projects and partnerships with industry (web site: usability.spsu.edu). She is involved with computer educational support for local businesses in the Atlanta area through Software Education & Support (her consulting company). She does specialized software development and evaluation as a consultant. Professor Thomas has given numerous papers, tutorials and presentations locally and internationally on Human-Computer Interaction and Software Engineering. She is currently the ASEE Southeastern Section Proceedings Editor.