A novel modeling domain application

Dr. Buthainah F. AL-Dulaimi,
College of Education for Women/University of Baghdad
Buthynna@yahoo.com

Abstract:

This paper presents a novel to model application domain. Application domain description precedes requirements engineering, and is the basis for the development of a software or information system that satisfies all expectations of its users. The domain model is used to generate project specific process models. Our aim is to develop a model description for processes which permits to create comprehensive scenarios. Modeling can be divided into a structural, and behavioral. This paper projects that an important future direction in software engineering is domain-specific software engineering. From requirements specification to design, and then implementation, a tighter coupling between the descriptions of a software system with its application domain has the potential to improve both the correctness and reliability of the software system. The greatest challenge in this area is the evolution of the application domain itself. We show how the application domain description can be mapped to requirements and discuss engineering of application domain descriptions.

Key words: Software Modeling, Domain Modeling, Domain-specific modeling, Software process models, Design modeling, Process models.

1 Introduction

This research deals with an important issue in computer world. It is concerned with the software management processes as in figure 1 that examine the area of software development through the development models.

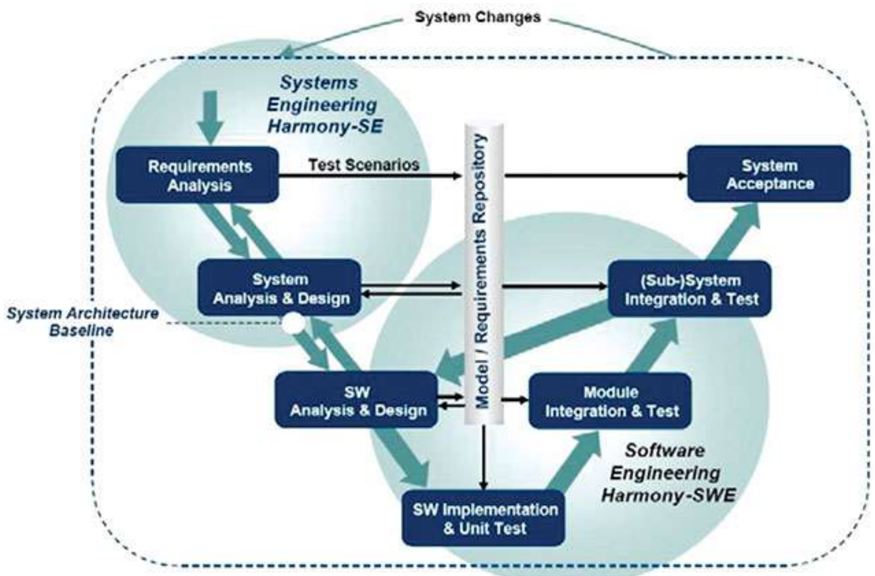
System Changes
Systems Engineering Harmony-SE
Requirements Analysis
Test Scenarios
System Acceptance
Model / Requirements Repository
System Analysis & Design
(Sub-)System Integration & Test
System Architecture Baseline
SW Analysis & Design
Module Integration & Test
Software Engineering Harmony-SWE
SW Implementation & Unit Test

Fig. 1 the software process

A software process model as in figure 2 is an abstract representation of a process. It presents a description of a process from some particular perspective as: specification, design, validation, evolution [1]. Software development is hard, requiring both domain knowledge and expertise. Some future directions are: mining domain concepts from existing application code written in general-purpose languages, using other artifacts where domain analysis has been performed already and presented in different forms [2]. Generic process models: Waterfall; Iterative development; Component-based software engineering [1].

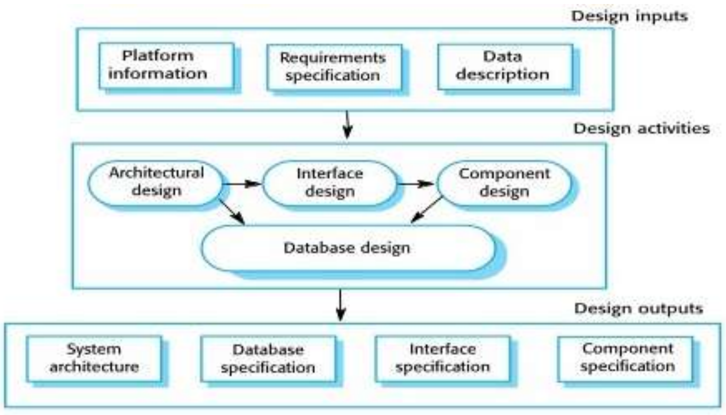

Fig. 2 Software process model

Moreover, results from domain analysis must be well-integrated with the design process. To build a large software system from a collection of components as in figure 3, and that components do not inherently carry enough information in their deployment to facilitate their composition [3]. Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems. Process stages are: Component analysis; Requirements modification; System design with reuse; Development and integration. This approach is becoming increasingly used as component standards have emerged [1].

Therefore successful composition relies on two cross-cutting domains: application domain and technology domain. Application domain knowledge imparts what components would naturally compose with other components to build the application system. Technology domain knowledge provides the technical infrastructure on how the components should be composed, including generation of code. The impact of domain-specific software engineering on building such systems is that domain knowledge is an inherent part of software systems, including components. This domain knowledge may be used to facilitate composition as well as reasoning about the composition with respect to correctness, reliability and various other quality measures (e.g., security) [3].

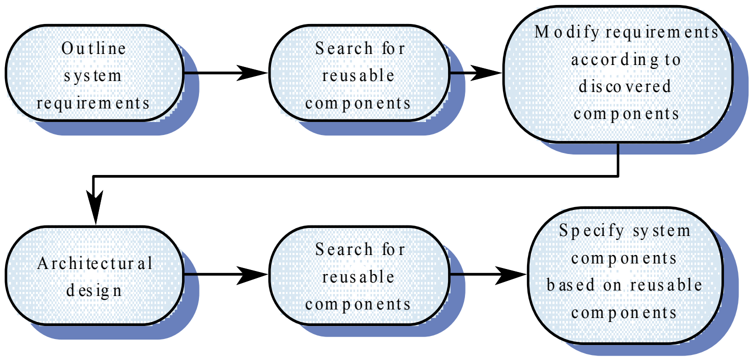
Outline system requirements
Search for reusable components
Modify requirements according to discovered components
Architectural design
Search for reusable components
Specify system components based on reusable components

Fig. 3. Component based system model

2. Domain Model

A Domain Model is an analysis model for the application domain that is needed to implement the application. An application domain is represented by means of multiple views, such that each view presents a different aspect of the domain [4]. It is well-known that requirements engineering cannot be conducted effectively without domain engineering. Domain-specific requirements specification requires that there be a framework for expressing domain entities at the specification level [5]. Providing requirements specification in terms of domain abstractions will also make such specifications easier for domain experts who are not software engineers to validate, because the specification will be expressed in terms of concepts which they understand. Software engineers may then concentrate on the formal specifications needed to model the appropriate domain behavior. DSM (Domain-specific modeling) has enabled both end-users and software developers in describing the key characteristics of a system from the perspective of the problem space, without getting overwhelmed by the accidental complexities of the solution space. By providing a notation that is often visual and graphical in nature, while also matching the abstractions of the domain, the essence of the problem can be captured in a way that removes the coupling with implementation concerns. Model transformations are used to translate a source model into some other form [6]. It is clear that we need a paradigm shift in software development to manage the complexity of development and maintenance. The same system functionality must be achieved with less code, which is also often easier to validate and maintain. Modifications to domain-specific programs are easier to create and can be understood and validated by domain experts who do not know how to program in a general-purpose language [7] [8].

3. Domain engineering

Domain engineering is designed to improve the quality of developed software products through reuse of software artifacts. Domain engineering shows that most developed software systems are not new systems but rather variants of other systems within the same field. Domain engineering focuses on capturing knowledge gathered during the software engineering process. By developing reusable artifacts, components can be reused in new software systems at low cost and high quality. Because this applies to all phases of the software development cycle, domain engineering also focuses on the three primary phases: analysis, design, and implementation, paralleling application engineering. This produces not only a set of software implementation components relevant to the domain, but also reusable and configurable requirements and designs.

3.1 Domain analysis

Domain analysis is used to define the domain, collect information about the domain, and produce a domain model. Domain analysis aims to identify the common points in a domain and the varying points in the domain. Domain analysis is derived primarily from artifacts produced past experience in the domain. Existing systems, their artifacts and customers are all potential sources of domain analysis input. During the domain analysis process, engineers aim to extend knowledge of the domain beyond what is already known and to categorize the domain into similarities and differences to enhance reconfigurability. Domain analysis primarily produces a domain model, representing the common and varying properties of systems within the domain. The domain model assists with the creation of architectures and components in a configurable manner by acting as a foundation upon which to design these components. An effective domain model not only includes the varying and consistent features in a domain, but also defines the vocabulary used in the domain and defines concepts, ideas and phenomena, within the system. Feature models decompose concepts into their required and optional features to produce a fully formalized set of configurable requirements.

3.2 Domain design

Domain design takes the domain model produced during the domain analysis phase and aims to produce a generic architecture to which all systems within the domain can conform. In the same way that application engineering uses the functional and non-functional requirements to produce a design; the domain design phase of domain engineering takes the configurable requirements developed during the domain analysis phase and produces a configurable, standardized solution for the family of systems.

Domain design aims to produce architectural patterns which solve a problem common across the systems within the domain, despite differing requirement configurations. In addition to the development of patterns during domain design, engineers must also take care to identify the scope of the pattern and the level to which context is relevant to the pattern. Limitation of context is crucial: too much context results in the pattern not being applicable to many systems, and too little context results in the pattern being insufficiently powerful to be useful. A useful pattern must be both frequently recurring and of high quality. The objective of domain design is to satisfy as many domain requirements as possible while retaining the flexibility offered by the developed feature model. The architecture should be sufficiently flexible to satisfy all of the systems within the domain while rigid enough to provide a solid framework upon which to base the solution.

3.3 Domain implementation

Domain implementation is the creation of a process and tools for efficiently generating a customized program in the domain [9] [10].

4. The proposed model

The complete proposed model can be summarized as illustrated in figure 4. For the purpose of understandings, the model might be simply divided into several stages, namely analysis stage, design stage and implementation stage, as described below.

a. ***The analysis stage*** comprises of three phases; problem phase, conceptual phase and requirement phase. This stage is concerned with examination of the problem domain and usage domain producing an object oriented model that determines functional and non-functional system requirements including hardware and software components requirements. Besides, it specifies a behavior model and a framework for the object oriented design stage. During analysis stage, the user and the developer closely cooperate in order to construct the model. The process is controlled by a system definition, delimitation, modeling and evaluation with the customer verification. It is iterative process and only stops when the user and the developer agree that the descriptions are usable and express a common understanding.

b. ***The design stage*** is concerned with specifying the overall structure of the system, resulting mainly into object oriented system model. Besides, the developer refines the object model and introduces an architecture model in order to understand the system model. Functional and non-functional requirements are supported at this stage and system model is mapped onto logical platform. At the design level, software domain is considered including either partial or complete software descriptions. The system model is constructed from the object model. The behavior model, together with the functional requirements, supplies the information for performing the functionality to the object model. Constructive solution for how the non-functional requirements and the organization of the logical platform combined with object model and system model to produce the architecture model which starts with identification then performs classification of components into classes**.**

c. ***Implementation stage*** is concerned with the realization of the system model. The structure captured by the design must be implemented in certain programming language. In the implementation stage, the developer integrates the architectural model into this program during the process and transforms the system model into a refined model in the form of programs and configurations. Hence, the perspective on the platform becomes a physical perspective during the implementation. This stage involves software domain, physical platform and object-oriented programming language. The model of the implementation process includes an iterative cycle, where the program is constructed from the system and architecture models under the influence of relevant non-functional requirements and the physical platform. During implementation stage, programming techniques are available for the developer in order to build the programs. Hence, the structure and the interactions described by the architectural design must be implemented in a programming language, therefore it allows for system execution which is the target platform.

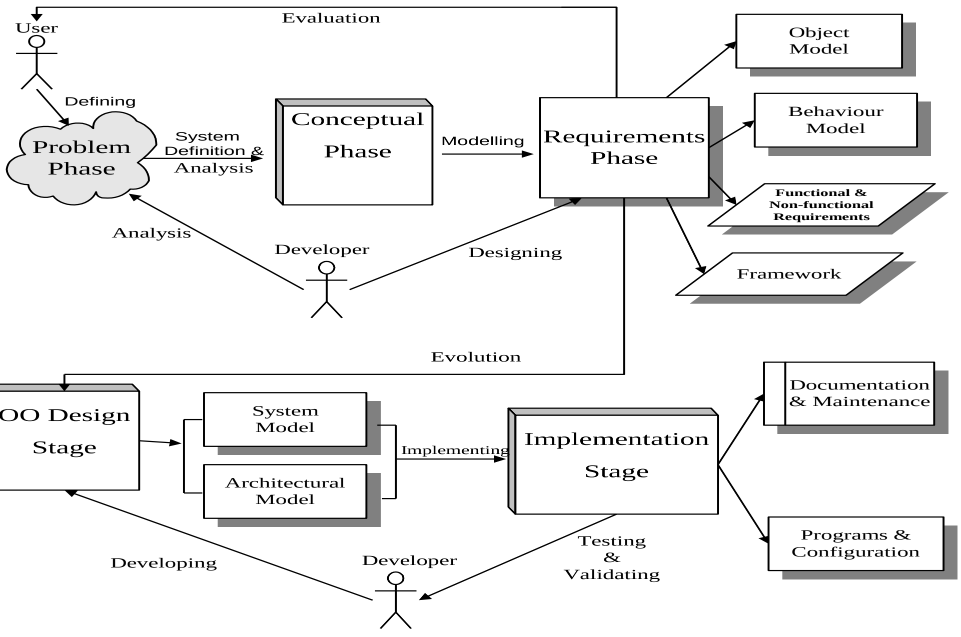

Fig. 4 The proposed domain model

5. Conclusions

The aim of software engineering is to create a suitable work that constructs programs of high quality. Our position for this paper is focused on the role that domain specific software engineering plays with respect to requirements specification, modeling and implementation. The proposed model is based on "real world" entities or objects. To reduce cost, the model is iterative and influenced by task analysis, user interface design. It is planned to evolve into system model and architecture model. An important part of our model is the inclusion of analysis and evaluation activities as part of architecture design that meets stakeholder's goals or concerns. A good domain model serves as a reference to resolve ambiguities later in the software process, a repository of knowledge about the domain characteristics and definition, and a specification to developers of products which are part of the domain.

References:

1. Ian Sommerville, "Software Engineering", Addison Wesley, 7th edition, 2004
2. Mernik, M., Hrnčič, D., Bryant, B. R., and Javed, F. 2010. Applications of GI in software engineering: DSL development. In Mathematics, Computing, Language, and Life: Frontiers in Mathematical Linguistics and Language Theory, C. Martín-Vide, Ed. Imperial College Press, London.
3. Cao, F., Gray, J., and Bryant, B. R. 2009. Component-based software engineering. In Wiley Encyclopedia of Computer Science and Computer Engineering, B. Wah, Ed. John Wiley & Sons, Inc., Hoboken, NJ.
4. Ehrig, H., K . Ehrig, C. Ermel, F. Hermann and G. Taentzer. 2007. "Information Preserving Bidirectional Model Transformations." In: Fundamental Approaches to Software Engineering, edited by M. Dwyer and A. Lopes
5. Bjørner, D. 2010. Domain engineering. In Formal Methods; State of the Art and New Directions, P. Boca, J. P. Bowen, and J. I. Siddiqi, Eds. Springer-Verlag, London, 1-41. DOI=http://dx.doi.org/10.1007/978-1-84882-736-3_1.
6. Kelly, S. and Tolvanen, J.-P. 2008. Domain-Specific Modeling: Enabling Full-Code Generation. Wiley-IEEE Computer Society Press, Hoboken, NJ.
7. Harrison W. 2004. The dangers of end-user programming, IEEE Software 21, 4 (July/Aug. 2004), 5-7. DOI= http://dx.doi.org/10.1109/MS.2004.13.
8. Sutcliffe, A. and Mehandjiev, N. 2004. End-User Development: Tools that Empower Users to Create their Own Software Solutions, Commun. ACM 47, 9 (Sept. 2004), 31-32. DOI= http://doi.acm.org/10.1145/1015864.1015883.
9. Harsu, Maarit (December 2002). A Survey on Domain Engineering (Report) (Report 31). Institute of Software Systems, Tampere University of Technology. p. 26. ISBN 9789521509322.
10. Reinhartz-Berger, Iris; Sturm, Arnon; Clark, Tony; Cohen, Sholom; Bettin, Jorn (2013). *Domain Engineering: Product Lines, Languages, and Conceptual Models*. Springer Science+Business Media.ISBN 978-3-642-36654-3.

نموذج لتمثيل مجال التطبيق

أ.م.د. بثينة فهران الدليمي
جامعة بغداد/ كلية التربية للبنات
buthynna@yahoo.com

الملخص:

وتعرض هذه الورقة نموذج يمثل مجال التطبيق. وصف مجال التطبيق يسبق هندسة المتطلبات، وهو الأساس لتطوير برنامج أو نظام المعلومات وتكون الاساس للعمل وتوافق كل التوقعات لمستخدميها. يتم استخدام نموذج المجال لتوليد نماذج عملية محددة للمشروع. هدفنا هو تطوير وصفا نموذجيا للعمليات التي تسمح لتكوين سيناريوهات شاملة. ويمكن تقسيم النمذجة إلى الهيكلي، والسلوكية. هذه الورقة يمكن أن تمثل الاتجاه المستقبلي المهم في هندسة البرمجيات وهو تخصيص التمثيل لمجال البرمجيات من وصف المتطلبات ثم التصميم، ومن ثم التنفيذ والذي يؤدي الى تحسين كل من صحة وموثوقية البرمجيات. التحدي الأكبر في هذا المجال هو تطور مجال التطبيق نفسه. وتبين لنا كيف يمكن تعيين وصف وهندسة متطلبات وصفات مجال التطبيق.