From Model to Internetware

A Unified Approach to Generate Internetware

Junhui Liu(✉), Qing Duan, Yun Liao, Lei Su, and Zhenli He

School of Software, Yunnan University, Kunming, Yunnan 650091, China
Key Laboratory for Software Engineering of Yunnan Province, Republic of China
HanksLau@gmail.com,
{qduan,YunLiao,LeiSu,ZhenliHe}@ynu.edu.cn

Abstract. Model driven development has been considered to be the hope of improving software productivity significantly. However, it has not been achieved even after many years of research and application. Models are only and still used at the analysis and design stage, furthermore, models gradually deviate from system implementation. This paper integrates domain-specific modelling and web service techniques with model driven development and proposes a unified approach, SODSMI (Service Oriented executable Domain-Specific Modelling and Implementation), to build the executable domain-specific model so as to achieve the target of model driven development. In this work, Domain-specific modelling is the key to construct xDSM (the eXecutable Domain-Specific Model). Web services are used as the implementation entities of the core functions of xDSM with the support of DSMEI (the Domain-Specific Model Execution Infrastructure). Finally, xDSM is transformed into the form of internetware to achieve system implementation.

Keywords: Model driven development · Domain-specific modelling · Executable model · Model execution infrastructure · Internetware

1 Introduction

Software is the spirit of a computer system. It has substantial impacts on success in business today. However, faced with increasing demands and more challenging market pressures, software systems become more and more large and complex. The traditional software development technologies are insufficient for ensuring a successful outcome that fulfills requirements and quality goals set out [1]. The complexity, variety and changeability make the large software projects have staggering failure rates: difficult to maintain, low dependability, high cost and the longer time-to-market. The Standish Report [2] states that nearly a third of projects are cancelled before completion and more than half suffer from serious cost overruns.

Developing and maintaining complex, large-scale, product line of highly customized software systems is difficult and costly. Part of the difficulty is due to the need to communicate business knowledge between domain experts and application programmers. Domain specific model driven development (MDD) addresses this

V.C.M. Leung et al. (Eds.): TridentCom 2014, LNICST 137, pp. 338–347, 2014.
DOI: 10.1007/978-3-319-13326-3_32

difficulty by providing domain experts and developers with domain specific abstractions for communicating designs [3]. Model describes system and its environment from a given view. It is an abstract representation of system and its environment. For a specific aim, model extracts a set of concepts relevant to the subject in order to make developers focusing on the whole system and ignoring irrelevant details [4].

In this paper, we discusses how to build the executable domain-specific model to achieve the target of MDD. This paper proposes an approach to the executable domain-specific modelling based on web services. Domain-specific modelling is the key to construct the eXecutable Domain-Specific Model (xDSM). Web services are used as the implementation entities of the core functions of xDSM with the support of the domain-specific model execution infrastructure which named DSMEI. Finally, xDSM is transformed into the form of internetware to achieve system implementation.

2 Proposed Approach

The role of model for software analysis and design is irreplaceable. Developers establish software analysis and design models in accordance with a variety of software standards, and communicate with each other by models. Model is expected to bring an essential leap of software development, and drive the whole software development process. It means that modelling is not only related to the requirement analysis, software design and software implementation, but also able to support unit testing, system testing, long-term system maintenance and software reuse, etc. The above all require the executability of model. Only executable models can strictly ensure that model validation, system-generation and system maintenance are based on the models.

The key elements of the executability of model lies in whether there are a well-defined models and whether there is a code generator which can automatically and completely generate code. Both of them are mutually constraining and complementary. Code generator can be simple and easy to implement while the model is complete and accurate. On the contrary, code generator must be difficult to achieve with complex structure and required adaptability and flexibility while the model is imprecise. In order to build the executable model, and achieve the automatic transformation from models to system implementation, there are two aspects both need to be concerned. On one hand, models ought to be refined and the degree of abstract ought to be reduced so that models can gradually approach system implementation; on the other hand, code generator ought to have strong adaptability and flexibility to reflect the model description.

This paper is based on domain-specific modelling to construct the executable model. During the process, the key is behaviour modelling. Based on the complete, consistent, detailed and accurate model description by XDML, model parsing and executing mechanism are used to replace code generator, and combine with Domain Framework as the infrastructure of the domain-specific model implementation. Different from other domain specific modelling approach, the abstract level of code implementation is enhanced by the standardised, self-contained, self-describing, modular web services. Encapsulating the details of code implementation, the related

domain-specific software functional entities are provided to DSMEI (Domain-Specific Model Execution Infrastructure) by the way of web services cluster. The system running is driven by parsing and executing the behaviour models. The above is the core idea of This paper. The framework of SODSMI (Service Oriented executable Domain-Specific Modelling and Implementation) is shown in Figure 1.

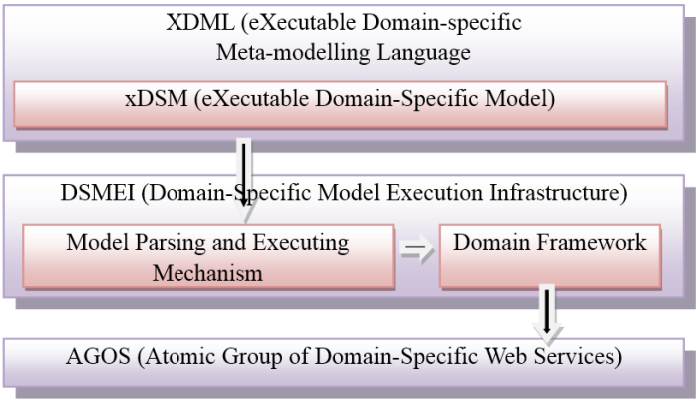

Fig. 1. Framework of SODSMI

SODSMI constructs executable models and their execution infrastructure based on domain-specific modelling through the model refinement and the enhancement of code implement.

From the perspective of functionalities, SODSMI is divided into three levels, corresponding to four core elements:

- xDSM -- Executable Domain-Specific Model
- XDML -- Executable Domain-specific Meta-modelling Language
- DSMEI -- Domain-Specific Model Execution Infrastructure
- AGOS -- Atomic Group of dOmain-specific web Services

XDML is used to describe xDSM. xDSM is parsed and executed in DSMEI. Its execution depends on the corresponding interfaces provided by Domain Framework. Domain Framework provides the core software functional entities through domain-related services of AGOS, and supports the xDSM execution upwards. xDSM, XDML, DSMEI and AGOS constitute the framework of SODSMI together.

3 xDSM – Executable Domain-Specific Model

The primary task of SODSMI is to build executable models, while the executability of model is always an underbelly of MDD for a long time. Software itself is dynamic. Static models can describe some profiles of software, for examples, the subordinate structure and the system hierarchy. But it can describe neither the entire software, nor the running process of software. At the same time, the abstract of models restricts the

accuracy of models, which makes models lack of many of the key elements that are used to construct entire software. In MDA system, UML can be used to build models of the system from different perspectives and aspects. Model views represent a part or a profile of the system. However, there are neither positive connections nor constraints among those model views. Model views can be more or less, be concrete or abstract. The process of building a model can be ceased at any phase. It is very difficult for modellers to construct a complete software model unless they understand all the details of code generator. That makes the executable models difficult to achieve in UML system.

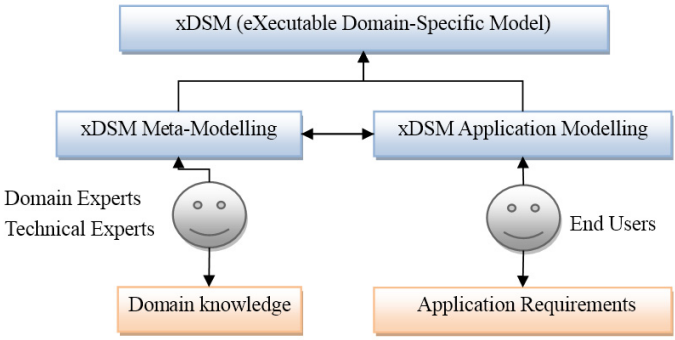

Fig. 2. xDSM Meta-Modelling and Application Modelling

xDSM is constructed based on the domain-specific model, and is technically applied to solve the software development problems existing in a certain application domain. xDSM represents the concepts and rules of the domain. The model is targeted, that narrows the scope of the description effectively and is helpful to define the model accurately. xDSM modelling process is divided into two phases: the xDSM meta-modelling phase and the xDSM application modelling phase. The former is carried out by domain experts and technical experts, and the latter is carried out by end users. The duty and the role of modellers in each modelling phase are different, as shown in Figure 2.

xDSM is required to meet MMLs standards 5 (Modelling Maturity Levels) [5]. It requires the model definition is sufficiently precise. The accuracy here is to describe the details relevant to the modelling objectives accurately rather than to describe all aspects of modelling. The core of xDSM is behaviour modelling. It is required to describe domain concepts and system behaviours unambiguously. In the meta-modelling phase, domain concepts are described unambiguously, including domain objects, relationships, constraints and any operations embodied in the domain concept. In the application modelling phase, the target is to meet all the requirements to software systems. The accurate software behaviour modelling is carried out by using meta-model. The model does not care about the implementation of local software

functions, but it does not ignore the necessary details of the behaviour execution yet -- the data flow, the control flow and the related constraints of behaviours must be described in detail.

On one hand, the measurement of the accuracy of models is determined by domain experts and technical experts through xDSM meta-modelling and DSMEI. Namely, if the application model which is built according to the definition of the meta-model can be accurately and completely executed by DSMEI, the models can be regarded accurate enough. On the other hand, the application model which is built in accordance with end users' requirements can ensure the integrity of the model. Namely, if the results of the application model execution meet the system requirements completely, or the generation system realises the functional requirements completely, the models can be regarded complete enough. Moreover, application modelling also facilitates the improvement of meta-modelling and the execution environment, to meet the requirements to application modelling better.

Furthermore, the description of the behaviour details in xDSM also increases the complexity of modelling. It requires to adjust the complexity of modelling through meta-modelling and application modelling. That is guided by domain experts and developers mainly in the meta-modelling phase. On one hand, the behaviour complexity is encapsulated in the meta-model while the behaviour details are hidden in domain objects and relationships with the different granularity; on the other hand, the complex behaviour descriptions are hidden by the implementation convention of the meta-model and the execution environment. So end users can do the application modelling simply and flexibly. So it is easier for end users to build the executable model with high-quality.

4 XDML – Executable Domain-Specific Meta-modelling Language

Following the guide of MMLs5, XDML is defined to describe xDSM meta-model and its application model. XDML extends the semantic basis of XMML language -- a visual meta-modelling language [6], and integrates the well-defined behaviour semantics to support the domain-specific behaviour modelling. XDML defines the concrete syntax of AS&MC which provides accurate definition for dynamic behaviours of models.

XDML improves the description accuracy of the specific domain problem and its solutions, and reduces the complexity of the language itself. XDML is simpler and more accurate in syntax and semantics than the universal modelling languages. That reduces the difficulty of XDML compiler, interpreter and the supporting environment development.

XDML is at a higher abstract level. Generally, the main domain concepts are mapping to the objects in XDML, while other concepts are mapping to the attributes, relationships, sub-model of the object or model links of other languages. Therefore, XDML makes developers use domain concepts directly to construct the domain models. It is able to describe domain concepts, the relationships between domain concepts and domain rules with larger granularity morpheme. Developers can use the domain knowledge elements in XDML directly to develop the application system, rather than

develop program code or components that are corresponded to domain concepts from the most basic classes or objects from the scratch. So the system development efficiency is improved effectively.

For enhancing the accuracy of models and the ability of the behaviour modelling in MDA system, OMG issued UML 2.0 which integrates action semantics [7] to improve the ability of the behaviour modelling, and uses OCL to enhance the ability of the accurate model description in MDA system. And ASL (Action Specification Language) is also introduced into xUML to define the system actions in detail. The ultimate goal of above all is to make the behaviour modelling more accurately. UML, OCL and ASL are overlapped in semantics. A part of the abstract syntax of OCL is introduced from the abstract syntax of UML 2.0, especially the introduction of action semantics [8]. ASL is consistent with the action semantics of UML [9]. The coexistence of several sets of abstract syntax of several languages makes it needs a lot of correspondence and references among those languages, and depends on the cohesion of the model reflection interfaces, so as to make the whole syntax architecture huge and complex.

The core of xDSM is the complete and accurate behaviour modelling, with the well-defined behaviour semantics, the accurate model constraints and action specifications as its necessary conditions. XDML is extended based on the semantics of the visual meta-modelling language – XMML. It integrates the well-defined behaviour semantics, supports the domain-specific behaviour modelling adequately, and constructs the concrete syntax of XDML based on XML meta-language. It constructs the textual concrete syntax of AS&MC (Action Specifications and Model Constraints) based on the behaviour semantics of XDML to provide the accurate definition for the dynamic behaviour of models, as shown in Figure 3.

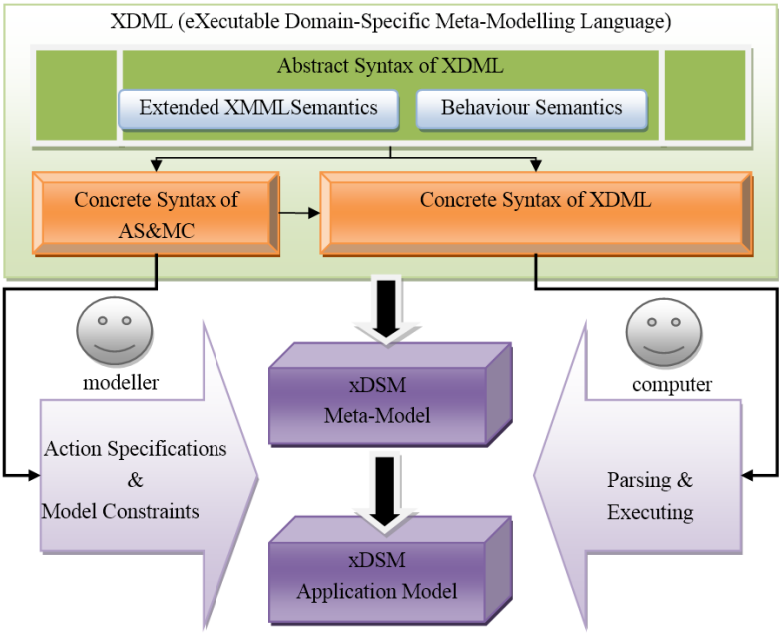

Fig. 3. XDML Architecture and Work Process

5 DSMEI – Domain-Specific Model Execution Infrastructure

Today, the scales of software systems are increasing, and the number of people who are involved in software applications is also increasing, so as to make software architecture more and more complex. The software is no longer limited to a stand-alone desktop system, but gradually evolved into the networked and complex systems which are integrated with each other. In this case, the functionalities of code generator are limited because the generated code may be only a part of the complex software system. Moreover, code generator is also a software product. It is more complex than the generated system, and it is also needed to face the changes of the generated system itself, that requires code generator to be strongly adaptable and flexible [10].

DSMEI is combined with Domain Framework, and employs the model parsing and executing mechanism substituting the code generator to execute xDSM models directly. Domain Framework is used to provide the interface of the underlying platform to the generated code. DSMEI encapsulates the architectures, platforms and concrete implementation of the domain-specific application system into Domain Framework, which reduces the complexity of the generated code significantly, as shown in Figure 4.

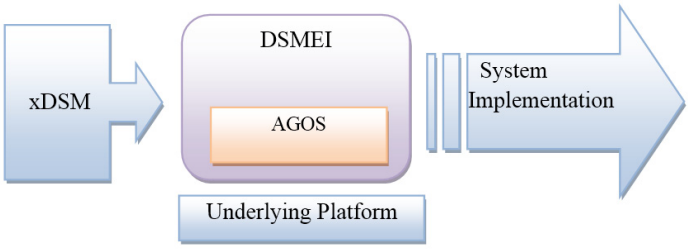

Fig. 4. DSMEI Functional Structure

The system behaviours are able to be described by xDSM completely and accurately. Based on that, the model parsing and executing mechanism is used by DSMEI to replace the code generation process. xDSM is parsed into the operations with precise semantic, and the operations are corresponded to the interfaces provided by Domain Framework. Here the model itself is an executable software product. As the evolution of Domain Framework is independent of the parsing and executing of the model, the model can be transform into the system implementation on DSMEI dynamically and flexibly. Furthermore, DSMEI is combined with Domain Framework, and encapsulates the parts of domain-related implementation into the modular web services through AGOS. So that it can focus more on the parsing and executing of the model, as well as the combination with web services which are related to the specific domain. That makes the architecture of DSMEI general, while the dynamic characteristics and the virtualisation techniques of web services make DSMEI more flexible, so that a common and flexible supporting environment is provided for the model execution by this way.

6 AGOS

To a certain extent, the code is also a model. It is the most refined model, and a language description defined precisely. It can be used to describe a system, but it is also platform-dependent. But such an iterative refinement is not necessary. On one hand, over-refinement makes the scale of model so large that the model loses its abstract nature. On the other hand, to deal with the ever-changing system requirements, even if the advanced language also needs to be added SDK (Software Development Kit) continuously, it must be much harder to the model which only have a weaker descriptive ability. Consequently, a better software functional entity must be found to realise the executable model.

The software functional entity has undergone several evolutions: from functions to objects, from objects to components, then from components to web services. Web services architecture adds and standardises a new layer, named "Service Layer" between the logistic layer and technical implement layer. The standardisation and dynamic characteristics make web services be able to provide the abundant and flexible software functional entities. AGOS adopts web services that is standardised, self-contained, self-described and modulised to enhance the abstract level of the code implementation, encapsulates the details of the code implementation, and provides the related domain-specific software functional entities to DSMEI by the way of web services cluster. Web services are not stand-alone. They depend on the domain-specific application systems and their processes. The development and reuse of web services have already been determined when the xDSM meta-model is constructed. It is a top-down design process. Based on the domain concepts, it describes the domain behaviour process dynamically according to the model, and drives the definition and functionalities of web services according to the realisation requirements of the model. The design principles of web services are as follows: the common parts of the specific domain are encapsulated into web services. The changeable parts are divided into two kinds: one kind that is easy to deal with by xDSM is defined directly by model; the other kind that it is not easy to deal with by xDSM will be transformed into service parameters, and use the parameterised means to handle the change-point. Web services provide the minimal software functional entities in the entire system. It is also the implementation foundation of the entire executable model.

Various web services at the different levels are required to support the problem space involved in the domain-specific modelling. AGOS regards a group related web services of a specific domain as a service cluster. On one hand, it requires a lot of web services entities to provide different functions; on the other hand, there may be several corresponding web services entities to the same functional requirement. So DSMEI is able to not only support the protocol of the service itself, but also deploy web services cluster dynamically in the software life cycle, for examples, querying services, matching services, assembling services, replacement services, load balancing of the service group of the same functional node, and adjustment of the coordinated services, etc. The flexible architecture of DSMEI is the foundation of the above all. It is able to provide Domain Framework dynamically based on web services, and adjusts the existing web service cluster to adapt software changes quickly.

7 Features of SODSMI

SODSMI is aimed at modelling for system implementation, which reduces the model complexity and improves the model accuracy. This method has a holistic and sustainable system to support the transformation from models to system implementation. Compared to other modelling methods, such as MDA system, the proposed approach is more suitable for the establishment and support of executable models, mainly shown as follows:

1. SODSMI is customised for solving software development problems in a certain application areas. It is dedicated and problem-oriented. Although it is at the expense of commonality, it improves the accuracy of the description on domain specific problems and its solutions, and reduces the complexity of modelling.
2. SODSMI improves the abstract level of models, and XDML provides an abstract mechanism to deal with the complexity of specific domains. It provides concepts and rules of the corresponding application domain, rather than those of a certain given programming language. Modellers face the domain concepts with different granularity directly, rather than construct the implementation details in the light of classes and objects, etc.
3. SODSMI pays attention to the integrity of MDD. Its goal is to achieve the system implementation, rather than to simply use models as a means of analysis and design. SODSMI completes the whole process from model establishment to code generation.
4. SODSMI emphasises on the capacity of meta-modelling, and adopts the separation of meta-modelling and domain application modelling to establish models that adapts better to specific domain. At the same time, it is able to separate users' application modelling from domain experts' meta-modelling as well as developers' creating support tools.
5. In SODSMI, the establishment of meta-model and code generator are developed within the organisation. They are mutually complementary: the model establishment is adapted completely to code generator; the generated code is practical, readable, and efficient as same as the code is written by experts who define the code generator. Meanwhile, the establishment of meta-model and code generators implicates a lot of implicit implementation convention that need not be expressed at the model layer, which observably reduces the complexity of models.
6. SODSMI is based on domain engineering, which provides a well support in essence for software reuse; on the contrary, the software reuse techniques also provides a well support for the DSM method.

Acknowledgment. This work is funded by the Open Foundation of Key Laboratory of Software Engineering of Yunnan Province under Grant No. 2011SE13.

References

1. Georgas, J.C., Dashofy, E.M., Taylor, R.N.: Architecture-Centric Development: A Different Approach to Software Engineering. ACM Crossroads **12**(4), 6–23 (2006)
2. Johnson, J.H.: The CHAOS Report. The Standish Group International, Inc. (1994)
3. Hen-Tov, A., Lorenz, D.H., Schachter, L.: ModelTalk: A Framework for Developing Domain-Specific Executable Models. In: Proceedings of the 8th Ann. OOPSLA Workshop Domain-Specific Modeling (DSM 2008), Nashville, TN, USA, pp. 19–20. ACM Press (October 2008)
4. Kuhne, T.: What is Model? Language Engineering for Model Driven Software Development. In: Dagstuhl Seminar Proceedings (2005)
5. Davis, M.D., Sigal, R., Weyuker, E.J.: Computability, Complexity, and Languages. Fundamentals of Theoretical Computer Science. Academic Press, Inc. (2008)
6. Zhou, H., Sun, X.P., Duan, Q., et al.: XMML: A Visual Metamodelling Language for Domain Specific Modelling and its Application in Distributed Systems. In: Proceedings of 12th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS), Kunming, China, pp. 133–139 (October 21-23, 2008)
7. Frankel, D.S.: Model Driven Architecture:Applying MDA to Enterprise Computing. John Wiley & Sons (January 2003)
8. OMG. UML 2.0 OCL Specification. Object Management Group. Framingham, Massachusetts (2003)
9. Mellor, S.J., Balcer, S.J.: Executable UML: A Foundation for Model Driven Architecture. Addison Wesley, Massachusetts (2002)
10. Yang, F., Mei, H., Lu, J., Jin, Z.: Some Discussion on the Development of Software Technology. Acta Electronica Sinica **26**(9), 1104–1115 (2003)