Using visualisation to elicit domain information as part of the Model Driven Architecture approach

John Mathenge Kanyaru, Melanie Coles, Sheridan Jeary, and Keith Phalp

Software Systems Research Centre
Bournemouth University
{jkanyaru, mcoles, sjeary, kphalp @bournemouth.ac.uk}

Abstract

Model Driven Architecture adopts a visual approach to software development. The main development activities are the construction of visual (typically Unified Modelling Language (UML)) models and the transformation of source models into target models, including application code generation. The use of visual models to produce application code often starts at the design (Platform Independent Model) level, and whereas business processes (Computation Independent Models (CIM)) have lately been considered, they are not used in MDA to either derive design models or application code. This paper enhances the MDA process by considering the early stages of software development that pertain to problem domain analysis. We argue that problem domain analysis and modelling can form valuable input to the more formal MDA phases at the CIM and PIM levels. We propose the use of a visual notation that allows informal modelling of domain-based concepts. Modelling at this stage using the proposed notation is geared to support involvement of non-technical business stakeholders whilst feeding into business process modelling at the CIM phase.

Keywords: domain analysis, elicitation, MDA, traceability.

1 Introduction

The Model Driven Architecture (MDA) approach emphasises the development of software systems based on visual models as the primary software artefacts [1]). For example, business processes, termed CIM models are typically constructed using the Business Process Modelling Notation (BPMN) [2]. On the other hand PIM models may be constructed using the Unified Modelling Language class diagram notation. The key development activity is to derive application code by applying transformation technology on the PIM model [3]. One benefit attributed to use of visual notations in development of software systems is the ease of comprehension [4] of graphical notations as compared to textual code. Hence, the use of visual (typically UML) notations in MDA can be seen to have twofold benefits. The main benefit cited by the OMG [5] is the potential to derive application code for different platforms from one model (typically PIM model). From the visual development perspective, there is the benefit of the visual models being amenable to understanding by non-technical development participants [6].

This paper considers a visual-oriented development approach for the early phases of development that are concerned with analysis of the problem. Such development activities, unlike those of CIM and PIM development, are largely informal, and often involve interactions with business stakeholders [7]. Jeary et. al [8] argue that the use use of a pre-CIM level in MDA is necessary to create the means for MDA to capture the richness of the business domain. In this paper we agree that the MDA approach does not consider such pre-CIM issues. Consequently current MDA tools (e.g., [9], [10], [11]) do not provide support for pre-CIM development activities, nor a means to construct typical pre-CIM artefacts. Furthermore, progression from CIM models to PIM is not supported by mainstream tools.

A key strength of MDA is the emphasis on software development by way of building visual models of the software system [12]. This paper discusses how software support for the business user is possible at the pre-CIM level. Section 2.1 looks at how the user may store and organise domain information demonstrated using a Scrapbook concept. Section 2.2 discusses how an informed analysis of the domain is made and details how an informal model may be constructed based on the scrapbook information. Section 3 discusses the benefits to be gained by using visual notations for pre-CIM development and Section 4 outlines possible mappings between analysis and CIM models.

2 Visual development at pre-CIM level

The Object Management Group has attempted to address concerns about business support by incorporating the CIM [2] phase into the MDA process model. CIM modelling however entails the semi-formal modelling of a business process using clear and unambiguous elements of the BPMN notation [13]. Analysts (or developers) initially attempt to comprehend the problem domain by eliciting information from domain experts [14]. Therefore the production of business models is often not the first step of the software process [15]. We argue that the analysis of such elicitation information should be used to build business process models that constitute the MDA's CIM model. The MDA approach misses out the domain analysis phase.

2.1 Organising problem domain information

A key challenge to software development is the elicitation of domain information [16] and the organisation of that information in a way that is accessible for successive phases of development. The use of metaphors is recognised [17] as one way of enhancing comprehensibility of either problem domain information or even software artefacts. We use the scrapbook metaphor as a means of organising information elicited from the problem domain. Such information maybe textual, or could be folders that may in turn contain subfolders. The storage and organisation of domain information is useful for subsequent stages of development because it acts as a basis for validating subsequent artefacts. Customers, end-users, or other stakeholders with knowledge regarding the problem domain can populate the scrapbook.

Consider a scenario where an organisation pursues business opportunities with prospective (or existing) clients. Such an opportunity elicitation process may require identifying possible clients, visiting the client and obtaining a lead. The organisation might want to store documents relating to previous successful opportunities, or unsuccessful ones with reasons to their success or lack of it. The MDA process does not provide a means to record such informal information. We propose the scrapbook concept to record and inter-relate artefacts that are built or elicited during problem domain analysis. Each item in the scrapbook model is a scrap item, which can be refined or expanded when further information comes to light. Figure 1 demonstrates an example of the usage of a scrapbook.

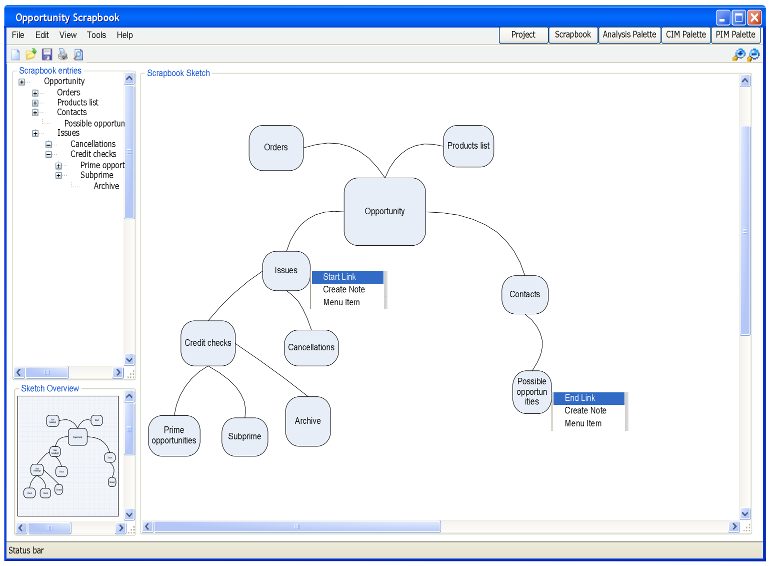

Figure 1: Scrapbook model editor

It shows scraps organised and linked in the scrap editor, with the left side of the screen showing a tree structure of the scrap items, including a preview of the scrap model. The elements in the scrapbook model are associated based on the way in which a business user understands their domain. The links may be annotated to indicate the relationship between any two items. The main contribution of the scrapbook to the MDA process is recording and organising domain information, and affording non-technical users flexibility in creating very informal models of the storage and organisation of their information.

2.2 Domain Analysis

Elicitation of problem domain information and the organisation of such information using the scrapbook concept provides a record of such information for use in the MDA process. There are a variety of issues and concepts that domain experts (or business users) identify from the scrapbook that may interest business and system analysts during the elicitation of problem domain information. For example, many business stakeholders will be familiar with their organisational hierarchy and with the roles and responsibilities of various stakeholders. In addition they will have knowledge of which of these stakeholders produce or consume data, and indeed who has ownership of that data. All this information is of interest to the business and system analysts.

Analysis has been defined as an effort in trying to understand a problem [12], and problem domain analysis, like requirements analysis [18], is bound to be challenging in various ways. For example, the business user might be unclear about aspects of their business concerns which are of interest to the business or systems analyst, whilst the complexity of the problem domain might pose significant learning overheads on the part of the analyst. There are therefore likely to be concepts that aren't clearly articulated or understood by both business users and analysts; one might want to record them with a view to elicit further clarity in the future. We have given the term "Bloop" to such unidentified concepts. A cloud figure is used to depict these Bloops on an informal analysis model in the Analysis Palette. Hence a prevalence of Bloops in an analysis model may be an indication that further analysis of the problem is needed. Identification of this issue so early in the development process highlights the value of this concept. Further analysis might mean that Bloops are broken down into the more clear concepts such as activities, roles or data objects.

Consider a business situation where an enterprise seeks to manage arising business opportunities, the contacts made for enabling pursuit of the opportunities, and production of quotations where the opportunity has progressed to a business transaction. One might envisage a business user constructing their own informal model where these items regarding opportunites are shown along with their inter-relatedness, including any items that may not be clear.

We have developed tool support for enabling business users to build these informal models that depict their understanding of the problem domain. The Analysis Palette (see Figure 2) provides a means for creating models of the domain using notational elements such as roles, activities, data objects and bloops. We use a visual notation to depict these concepts in order to construct such models as part a MDA development process. Activities are shown as rounded rectangles with a letter **A** at the top left corner of the rectangle. Roles and data objects are depicted using a similar shape, with the indicative letters **R** and **D** at the top left corner of the rectangle. Figure 2 shows a number of activities (e.g., Make Order), bloops (e.g., Sales) and roles (e.g., Customer).

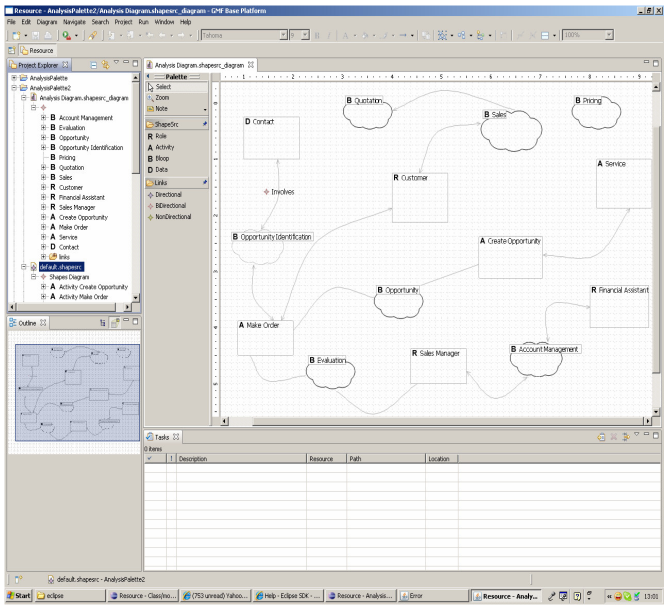

Figure 2: Analysis model of the domain in VIDE Analysis Palette

A typical challenge for development of software systems is the traceability [19,20] of information across phases. This is particularly relevant in the MDA process. For example, existing MDA tools have no means to indicate to a developer where any elements of a PIM model are derived from within an associated CIM model. The richness associated with models created at the CIM level is lost in subsequent successive stages. We provide tool support that provides traceability between the scrapbook and the domain model, and the domain model and the subsequent CIM.

The use of a visual notation, rather than textual description in this setting has a number of advantages. Firstly, an analysis model of the problem domain that shows activities, the roles that perform those activities and the produced or used data is one that non-technical business stakeholders can identify with and therefore validate. Secondly, the use of informal notations such as Bloops, or annotated rounded rectangles means that modelling is simple because there are no strict rules on using such elements. Thirdly, whereas the notational elements that depict the concepts of activities, roles, and data are informal, similar concepts are used in the MDA's CIM development phase. This suggests the possibility of one-to-one mapping between similar concepts between both pre-CIM and CIM.

3 Value of information visualisation at pre-CIM level

The traditional approach of producing software implementations is by describing using precise syntax of a language (e.g., Java, C++) to specify the program. Whereas many programming languages have graphical environments in which to specify the program, such programs cannot themselves be specified using graphical elements. A visual programming language is one that is seen to provide graphical elements for program construction, with no obvious textual counterpart [6]. The concept of MDA is based on such a visual language (mainly, the UML). One of the main development phases is the production of PIM models using UML class diagrams. The OMG does not specify a counterpart textual language for PIM development since application code is to be generated from PIM models directly.

The OMG outline support for business process modelling at the CIM level, but there is no support for the direct use of CIM models to build PIM models. We note however that, emphasis on visual development in MDA is beneficial for the reason that, generally, a diagram is easier to build and comprehend than textual descriptions [21].

In section 2, we described three advantages of using a visual notation. The business user is able to validate any models, that informal notations are easy to understand, and if models are simple they are easy to map to formal notations. These advantages are all based on the increased communication level between the business user and the analysts. It is also a much more reasoned communication because the notation is based on vocabulary that is used in the business domain. However, the underlying significance of our contribution (with domain analysis and modelling) to MDA is that by taking a 'step back' from the MDA process and considering the pre-CIM we are contributing to the initial analysis of the problem. MDA ignores this phase and always assumes that the problem is well understood, hence the emphasis on CIM and PIM modelling. We argue that producing domain models during MDA development facilitates early communication between analysts on the one hand, and business stakeholders on the other without forcing the immediate consideration of formal notations for CIM or PIM modelling. Moreover, it has been argued by others [22] [23] the use of visual notations during analysis can help tease out tacit knowledge from domain experts.

4 Mapping between analysis models and CIM models

MDA development environments are maturing, and depending on the MDA tool that one is using, there are several intermediate models (e.g., [24][25]) to be built in order to move from a PIM model to application code. Regardless of these tool-specific models, there are two main software development activities within the MDA approach. First is the construction of a model as a primary software artefact. Second is the application of a transformation technology to derive a target model from a source model. Most MDA support tools only apply transformation technologies to generate application code from PIM models. There is no support for generating PIM models from CIM models.

This paper proposes a means to derive CIM models from analysis models by direct use of domain model elements to build subsequent CIM model elements. For example, activities, roles, and data objects within a domain model would be used in a similar way within a CIM model. Rather than proposing a radical approach of transforming domain models into CIM models using formal transformations, we argue that, both models are representations of different world views and that human intervention is necessary for moving from one to the other. Therefore a set of guiding heuristics will be of more value than trying to create a model-to-model transformation standard.

5 Conclusion

This paper outlines the significance of eliciting and organising information about the business domain and details undertaking further analysis including informal modelling as part of the MDA development process. In particular, the paper outlines the need to use visual notation that is informal and accessible to business stakeholders whilst considering desirable mappings to the CIM phase.

The benefits for an informal, visual development approach seem well recognised [26]. The main benefit is the comprehensibility of visual artefacts as opposed to their textual counterparts. The benefit of visual development at pre-CIM level is the intention to involve non-technical domain experts in developing the models, thereby adding the benefit of model validation prior to CIM development.

Given the MDA approach suggests transition from a source model to a target model, we demonstrate the plausibility for developers to derive parts of a CIM model from the informal model of the domain. This paper has therefore described one way of enhancing the MDA approach to provide seamless development from domain models to CIM models. Additionally, where model elements are derived from a given source model, traceability among such elements is supported. The current set of MDA tools have largely ignored the domain analysis phase (in favour of design and code generation), and have not considered traceability among different models either.

References

1. OMG 2003. MDA Guide version 1.0.1; document no. omg/2003-06-01.
2. OMG.*Business Process Modeling Notation Specification*. 2006, dtc/06-02-01
3. Mellor, S., Kendall Scott, Axel Uhl and Weise, D. *MDA Distilled*. Addison Wesley, 2004.
4. Petre, M. "Why Looking Isn't Always Seeing: Readership Skills and Graphical Programming." in *Communcations of the ACM*, 1995, 38(6).

5. OMG 2006. Meta Object Facility (MOF) Core Specification; document no. formal/06-01-01; www.omg.org.
6. Green, G. and Petre, M. "Usability Analysis of Visual Programming Environments:a ‘cognitive dimensions’ framework." in *Visual Languages and Computing*, 2001.
7. Olsson, E. "What active users and designers contribute in the design process." in *Interacting with Computers*, 2004, 16.
8. Jeary, S. F., A. Phalp, K. Extending the Model Driven Architecture with a pre-CIM level. in 1st International Workshop on Business Support for MDA, 2008. Zurich, Switzerland.
9. PathFinderSolutions. "PathMate MDA transformation environment (http://www.pathfindermda.com/products/index.php)." Retrieved 07/2006 2006, from http://www.pathfindermda.com/products/index.php.
10. Objects, I. "ArcStyler MDA tool (http://www.arcstyler.com/)." Retrieved 08/2006 2006, from http://www.arcstyler.com/.
11. Codagen. "Codagen Architect for MDA (http://www.codagen.com/)." Retrieved 07/2006 2006, from http://www.codagen.com/.
12. Heckel, R. and Lohmann, M. "Model-driven development of reactive information systems." in *International Journal on Software Tools for Technology Transfer*, 2006.
13. White, S.*Introduction to BPMN*. 2006, IBM
14. Jackson, M. *Problem Frames: Analyzing and structuring software development problems*. Addison-Wesley, 2001.
15. Pressman, R. *Software engineering: a practitioner's approach*. McGraw-Hill, 2000.
16. Sánchez, P., P. Letelier and Ramos, I. Validation of Conceptual Models by Animation in an Scenario-based Approach. in ACM Conference on Object-Oriented Programming, Systems,Languages, and Applications: Workshop on scenario-based round-trip engineering., 2000. Minneapolis, Minnesota, USA.
17. Petre, M. and Quincey, E.*A gentle overview of software visualisation*. 2006, Open University
18. Jørgensen, J. B. and Lasen, K. B. Aligning Work Processes and the Adviser Portal Bank System. in 1st International Workshop on Requirements Engineering for Business Need and IT Alignment, in conjunction with RE'05, 2005. Paris, France.
19. Alexander, I. SemiAutomatic Tracing of Requirement Versions to Use Cases: Experiences & Challenges. in 2nd International Workshop on Traceability in Emerging Forms of Software Engineering 2003.
20. Susanne Sherba, Kenneth Anderson and Faisal, M. A Framework for Mapping Traceability Relationships. in 2nd International Workshop on Traceability in Emerging Forms of Software Engineering 2003.
21. Jungpil Hahn and Kim, J. "Why Are Some Diagrams Easier to Work with? Effects of Diagrammatic Representation on Cognitive Interation Process of systems Analysis and Design." in *ACM Transactions on Computer-Human Interaction*, 2000, 6(3).

22. Harel, D. "Statecharts: A Visual Formalism for Complex Systems." in *Science of Computer Programming*, 1987, 8.

23. J. Magee, N. Pryce, D.Giannakopoulou and Kramer, J. Graphical Animation of Behavior Models. in Proceedings of the 22nd International Conference on Software Engineering, 2000.

24. Budinsky, F., Steinberg, D., Merks, E., Ellersick, R. and Grose, T. 2004. Eclipse Modelling Framework. E. G. L. N. J. Wiegand, Addison-wesley.

25. Compuware. "OptimalJ MDA tool (http://www.compuware.com/products/optimalj/)." Retrieved 08/2006 2006, from http://www.compuware.com/products/optimalj/.

26. Howse J. and Schuman, S. " Precise Visual Modelling: a case study." in *Software Systems Modelling*, 2005, 4(3).