

METAMODEL OF A SYSTEM DEVELOPMENT METHOD.

By Zygmunt Jackowski

Introduction

The term ‘method’ is so commonly used in IT community in the context of software development that hardly any effort was ever made to define its semantic. With hundreds of methods with which to develop IT systems anything that looks like some kind of a software development process or architecture framework or even notation for a system modelling is called a ‘method’ (and sold as such) no matter how close it resembles one. With no strict definition or blueprint of a software development method itself, it’s difficult to assess a particular method in a systematic way.

With established and emerging open standards such as UML notation for OO modelling [9] and OMG’s Model Driven Architecture (MDA) [6, 10] the next logical step should be to define an open methodology metamodel that can provide a common framework to define any existing and new system development methodology. The benefits of such a framework will be significant and will include:

- Common way of defining a methodology, which is widely understood in the IT community.
- Common pattern with which vendors of methodology related products can show the value of their product to the client and the later one has a template for a particular methodology evaluation.
- Easier mapping between different methodologies for the purpose of comparison, adopting or defining a new methodology or applying agile modelling approach [1].

The metamodel in question should be agile enough to accommodate the oldest, structured methods as well as object oriented development processes. The emerging breed of agile software development approaches should be able to find its way to apply the metamodel as easily, especially when it itself is aiming at becoming a kind of meta-method approach by combining various aspects and elements of the method spectrum. No matter how agile the process is meant to be, we need to define it first and having a consistent framework for that purpose will aid the task of both describing the process and maintaining it throughout an agile project life cycle.

In this paper we introduce a methodology metamodel aimed to provide an open standard for defining system development methodology compliant with MDA. Defining high-level concept with the use of metamodels has been adopted by OMG and IT community and is accurately described by Metamodel Group [5]:

“In this case, a metamodel allows a language designer or methodologist to better capture, analyse and understand what they are actually writing about in all those methodology books.”

Methodology metamodel.

A metamodel defines the language for expressing the model. In other words, a metamodel describes concepts and their relationships for the purpose of building and interpreting models that themselves are an abstraction of reality: “A metamodel is the collection of "concepts" (aka things, terms, ...) that are the vocabulary with which you are

talking about a certain domain.” [5]. UML itself is an example of such metamodel defining concepts with which to model IT and other systems.

The metamodel of methodology shown as an UML class diagram in Fig. 1 defines two high-level, interdependent methodology concepts. The first one is System Architecture consisting of all system model Artifacts, which are created in a Process (which is the second high-level methodology concept) of the system development described by the methodology. System model Artifact (shown as an abstract class) can be produced in the form of Model Diagram (graphical model of any kind) or File (e.g. text file, source code, executable file). System Architecture is defined by several different Views (examples of system architecture defined through its views can be found in [2, 3, 4, 7]) that can be described by system model Artifacts at several Levels (such as: Organisation, Process, Job level etc. [8]). In some architecture frameworks, Levels are not explicitly defined but instead included into appropriate Views like Use Case View, Process View etc.

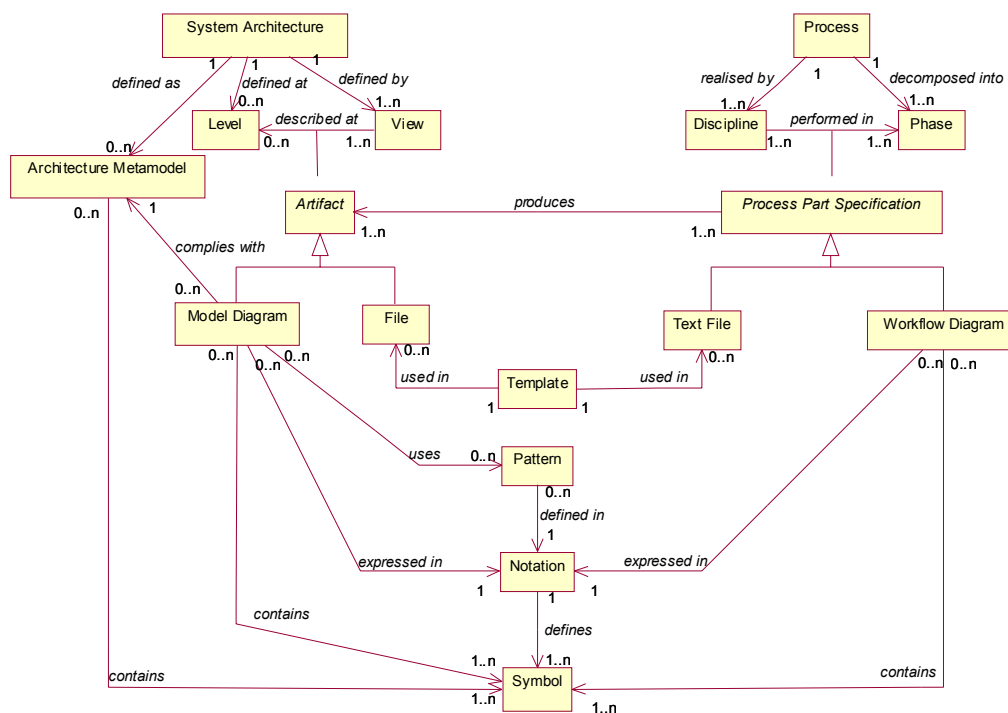


Fig. 1. Methodology Metamodel.

The methodology Process describes ‘how’ and ‘when’ in the project live cycle to produce system model Artifacts. Process cycle is decomposed over time into several sequential Phases. For example, in RUP the Phases of the project are: Inception, Elaboration, Construction and Transition. A Discipline such as Business Modelling or Analysis and Design shows workflow of activities one can go through to produce a particular set of Artifacts. Even though in many methodologies Disciplines (e.g. Analysis, Design and Implementation) are equivalent to project Phases, in general – the same Discipline can be performed at several project Phases (e.g. in RUP) and therefore it makes sense to differentiate the two. Workflow details of how to perform a particular Discipline at a particular Phase to produce a desired set of Artifacts makes up the Process Part

Specification, which can be provided in a form of textual description (Text File) or/and Workflow Diagram.

Model and Workflow Diagrams are expressed in a Notation(s) defining syntax and semantics of a particular modelling language. Notation defines a number of graphical Symbols. Pragmatic rules on how to build models in a defined System Architecture can be expressed in a form of Metamodel using the same Notation. Model Diagrams use the same symbols according to the rules defined by the Metamodel. Examples of how to define System Architecture with Metamodels can be found in [2, 3, 4]. To reuse modelling structures addressing similar problems, a number of generalised solutions called Patterns can be created as prototypes to be applied in a particular problem situation. Patterns can be used to generate parts of the Model Diagrams. Document Templates can be used to create Files of any type.

Conclusion

The methodology metamodel presented in the paper is an open standard template for specifying system development methodology. It is technology/tool independent and can be applied to structure and validate matured methodology specification of any kind (OO & structured methods). With a number of methodologies for system development to choose from, there is an obvious need to standardise the methodology specification itself. Adopting such a standard would benefit both IT and business communities in a significant way.

References

1. S.W. Ambler, The Object Primer: Introduction to Techniques for Agile Modeling, Ronin International White Paper, downloaded from <http://www.ronin-intl.com/>
2. H.E. Eriksson, M. Penker, Business Modeling with UML, Business Patterns at Work, J. Wiley & Sons, 2000.
3. Z. Jackowski, Business Modelling with UML: Process Centred Architecture, <http://www.agilealliance.org/articles/articles/BPM.pdf>
4. C. Marshal, Enterprise Modeling with UML, Designing Successful Software through Business Analysis, Addison-Wesley, 2000.
5. Metamodel Info, downloaded from <http://www.metamodel.com/>
6. Model Driven Architecture (MDA), Document - ormsc/01-07-01, downloaded from <http://www.omg.org/>
7. P. Kruchten, "The 4+1 View Model of Architecture," IEEE Software, 12 (6), November 1995, IEEE, pp. 42-50.
8. G.A. Rummler, A.P. Brache, Improving Performance, How to Manage White Space on the Organization Chart, J. Wiley & Sons, 1995 (second edition).
9. Unified Modeling Language (UML), version 1.4, downloaded from <http://www.omg.org/>
10. J. Siegel, Making the Case: OMG's Model Driven Architecture, SD Times, Oct. 15, 2002, <http://www.sdtimes.com/news/064/special1.htm>