Risk, Reliability and Safety: Innovating Theory and Practice – Walls, Revie & Bedford (Eds)

AMDesigner: An architecture modeling tool for the use in multi-disciplinary design projects

M. Rajabalinejad & G.M. Bonnema
Laboratory of Design, Production and Management, Faculty of Engineering Technology, University of Twente, Enschede, The Netherlands

ABSTRACT: This study resembles the use of AMDesigner (AMD) in several multi-stakeholder projects to explore the commonly used elements of this framework. For this purpose, AMD was used in several projects (technical and none-technical). The users had no prior experience on the use of architecture modelling tools. To stimulate the usage, a 30 minutes description of the tool was offered. This paper reviews the commonly practiced frameworks for the architecture modeling, briefly reviews the elements of AMD, describes the set-up for the experiments and draws conclusions form its implementations.

1 INTRODUCTION

The design and engineering of complex systems demand effective communication among members of the design team or the people who have interest in the product, system or services: the so-called system stakeholders. Good communication preserves many resources and positively influence the consistency of available information and so the integration phase (G. M. Bonnema, 2011; Woestenenk, 2014). This demands for an architecture model which can be easily communicated with the stakeholders. Furthermore, an architecture model provides the basics for capturing the design information to enable its reuse. These aspects have clear influence on the elements of performance triangle: cost, time-to-market and quality (Rajabalinejad et al., 2015; Theisens, 2014). Here we present the commonly practiced languages for the system modelling.

Unified Modelling Language (UML) was developed as a general-purpose modeling language developed for software engineers extended for the use of system architects with tools for design and implementation of software-based systems, modelling business or similar processes (ISO/IEC, 2012).

To engineer complex products or systems, the International Community of Systems Engineers (INCOSE) considers the Model Based Systems Engineering as an essential enabler for systems engineers (INCOSE, 2015). This community uses the OMG SysML language for describing the system. SysML reuses a subset of UML and provides extensions for covering the requirements for system engineers (OMG, 2015). The Architecture Analysis and Design Language (AADL) was developed by SAE to design and analyze the software and hardware architecture and performance-critical real-time systems. It focuses on special needs of these systems and enables analysis of their properties. The Zachman framework provides ontology for describing the enterprise (Zachman, 1987). Thus it provides basis for the system architecture. The Department of Defense Architecture Framework (DoDAF) is a comprehensive framework and conceptual model for architectural description within DoD. It provides structure for organizing architecture, assumptions, and terminology about the operational solutions (DoD, 2015). The Open Group Architecture Framework (TOGAF) is a supporting tool for developing enterprise architecture (GROUP, 2011). Object-Process Methodology (OPM) is a compact conceptual approach capable of establishing basis for system architects, designers and system developers (ISO, 2013). The British Ministry of Defense Architecture Framework (MODAF) to support defense planning and change management (see modaf.com). The Nato Architecture Framework (NAF) is developed for developing Enterprise Architectures and creating a model of current and future state of an enterprise (see nafdocs.org). The Business Process Model and Notation (BPMN) provides an understandable notation easily understandable by business users for modeling and analysis of business processes (OMG, 2011). The Federal Enterprise Architecture Framework (FEAF) supports planning and decision-making through documentation and information that provides an abstract view at different levels of scope and detail (www.egov.gov, 2013).

This research builds on the tested hypothesis that the use of a common language that describes the system architecture and information flow improves the communication in the course of design (G. Maarten Bonnema, 2014). It adapts the theoretical architecture model (AM) (see (Woestenenk, 2014)) and presents the benchmarks for developing a software tool on its basis. Some specific aspects of this tool have been

highlighted here. Furthermore, the observations on the use of this tool have been summarized through different multidisciplinary projects. Conclusions have been drawn and considerations have been highlighted for the future development of this tool. Next section provides on the specifics of the architecture model developed through the IOP project.

2 THE HIGHLIGHTS

As a part of the national program for innovation (IOP), the research aims to develop a prototype tool to automate the generation of control software for mechatronic machines. To achieve this, earlier studies concluded the importance of the architecture modelling. Among others, K. Woestenenk and A. Cabarera concluded through their PhD dissertations (Cabrera, 2011; Woestenenk, 2014) the essential influence of architecture modelling for developing mechatronic machines. To represent an architecture model, there has been a need for a software tool that is user-friendly, robust and able to support companies for their project in real-time. The earlier mentioned PhD theses provided a solid framework for such an architecture modeler.

University of Twente (UT) and Embedded System Institute (ESI) conducted research projects about architecture modelling tools. UT kept its focus around creating a pragmatic tool for high-tech companies in order to capture the Architecture Model. This architecture model is used for improving communication among the design team-members resulting in consistency of views and data. This accelerates the integration process in the course of developing complex products (Woestenenk et al., 2012) given the diversity of stakeholders' opinions (Rajabalinejad et al., 2014). Furthermore, it facilitates access to the information for future use.

The results of discussions with industrial partners and information exchange with software developers highlighted the items below for development of the AMDesigner.

- requirements for the tool
- capturing specialised design processes
- testing the results in real-world projects
- document the findings for future development

2.1 *The user interface*

The tool has to be robust. That was concluded from previous research, explicitly indicated through the proposal, and clearly requested by industrial partners. To serve this need, the AMDesigner was developed under the *Model-View-View-Model (MVVM)* design pattern. Developed by Microsoft, this pattern was created for decoupling the user-interface and none-user-interface codes. This pattern enables a layered data structure for binding the data and user interface. It organizes the code in such a way that it facilitates future changes/ revisions as it structures the code on basis of individual

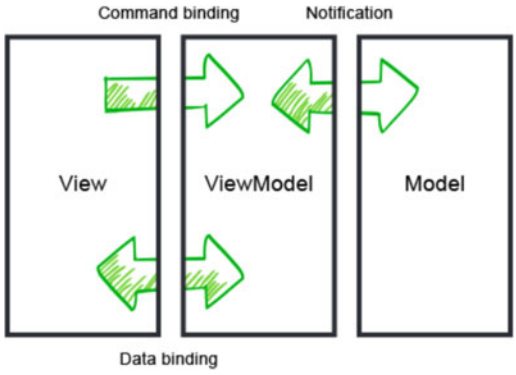

Figure 1. This picture represents the MVVM architecture implemented in AMDesigner (source: Telerik.com).

parts. Adapting this pattern is of primary importance for this tool as it leads to a robust tool and facilitates software automation.

In other words, the MVVM pattern creates a loose software-architecture which is less error-prone than the earlier used tightly-coupled software architecture. More information about the MVVM is available on the Microsoft website. The picture below represents the concept of loose architecture through different modules.

3 WORKBENCH

3.1 *Architecture language*

The PhD research of K. Woestenek (Woestenenk, 2014) resulted in defining an architecture language composed of concepts and relations. AMD is built on this principal definition for concepts while it adapts the definition for relations. As these definitions are prerequisite for its use, the definition of concepts and relationships are included through the text below.

3.2 *Reference model*

It is common practice to create a complex and multi-disciplinary product based on a reference model. Such a model contains design information to improve understanding of the essential design components. The AMDesigner focuses on the essential system components required for a functional system. The implemented reference model for the AMD includes the following components.

- Abstract which represents the abstract view of things; therefore, details are not parts of the model.
- Entities and Relationships which are both required to create a reference model.
- Environment helps to show a limited number of entities and relationships encapsulated by an environment.

	BO	AM	V	P	F	R	E	Fx	A	DE	DT	FR	ER	DR	SF
BO		inh	inh	inh	inh	inh	inh	inh	inh	inh	inh	inh	inh	inh	inh
BO		con	con	ass											
AM			inh												
V			con												
P				dep											
F					com	ass	ass		ass			dep			ass
R						com			ass			ass			
E							com		ass				dep		
Fx													ass		
A									com	ass	ass				
DE										com	ass		ass		
DT													dep		

Figure 2. The allowable relations in the Architecture model adapted from (Woestenenk, 2014).

3.3 *Concept*

- Basic Object (BO) is the abstract information unit required for all of the concepts.
- Architecture Model (AM) is the container of the model.
- View (V) is a stakeholder's perspective.
- Parameter (P) is the most basic information unit which defines the information content for other concepts.
- Function (F) describes what a system does or should do.
- Requirement (R) is the border of allowable design space.
- Entity (E) is an structural unit or an object of a system.
- Formula (F) defines how parameter values change and are related.
- Aspect (A) is a quality or characteristic of a system that is of interest to a stakeholder.
- Domain Entity (DE) is a mono-disciplinary entity defined in a certain design domain.
- Design Task (DT) used to model a design task.
- Function Relation (FR) defines the possible interface between two functions.
- Entity Relation (ER) defines the possible interface between two entities.
- Design Task Relation (DTR) defines the possible interface between two design tasks.

3.4 *Relationships*

The Architecture Model communicates to different stakeholders/ experts and integrates information to serve the design process primarily from the architecture perspective. This requires a well-defined relationship between those defined concepts (information unit). There are three sorts of relationships available through this tool

- Composition (com) relation which describe a parent/child relationship.
- Dependency (dep) relation that defines and input/output or sequence relation.
- Association (ass) which relates different concept types. This relation is called mapping through (Woestenenk, 2014).

Table below presents the permitted relations between different concepts. Further information is available through pages 201–207 of (Woestenenk, 2014). Next section provides some snap-shots on the use of this tool.

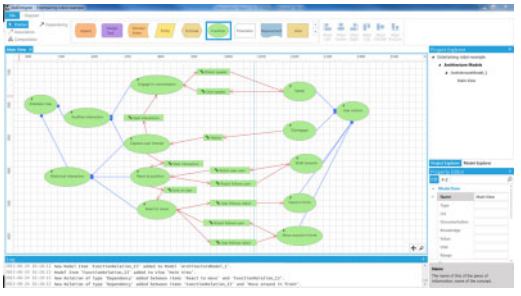

Figure 3. A functional view generated by AMDesigner.

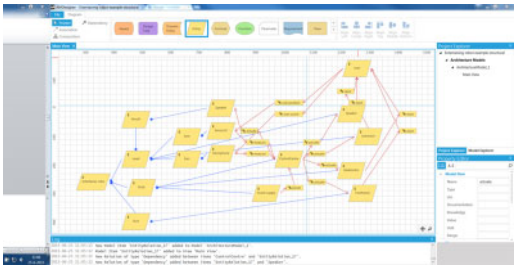

Figure 4. A structural view generated by AMDesigner.

4 PRESENTATION

This section presents functional, structural and requirement views for an example robot. These are customized views generated by AMD which are available for a system architecture.

4.1 *Functional view*

Through functional view, the architecture or expert can create a network of main system functions and define the functional interfaces. Figure 3 shows an example snapshot of the functional relationship. The figure is presented in low scale on purpose. Readers who are interested in further details are advised to see (Woestenenk, 2014).

4.2 *Structural view*

Through the structural view, the physical components and their interfaces are shown. This view shows how different components or electronic elements are located or related in order to perform the required tasks.

4.3 *Requirement view*

A well designed system must fulfill its requirements. A separate requirement view helps a designer to pay attention to the key/derived requirements for the components or performances. Figure 5 shows an example requirement view.

4.4 *AMDesigner*

The software package can be shared for non-commercial and research related uses. To obtain an installable version of the AMDesigner, please contact the authors.

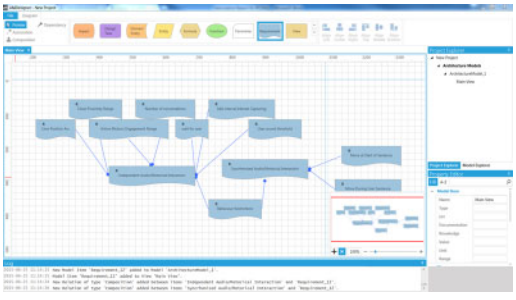

Figure 5. A requirement view generated by AMDesigner.

5 TEST/OUTCOME

AMDesigner was used in 10 different projects by engineers, specialists, or designers (here we call them designers) owning at least a Master of Science and often with several years of experience. The projects were real-world application of different systems. The category of each project is shown in the first column of Table 1. The projects range from technical (control systems) to none-technical (educational science) and include different industries. All the projects include stakeholders from business.

To facilitate the use of AMDesigner, a 30 minutes instruction was provided to the designers over the structure, elements and usage of AMDesigner. Then, the designers were asked to use AMDesigner for the following purposes:

1. Describing the project architecture for communication with the stakeholders and obtaining feedback.
2. Modeling (architecting) safety (or another desired performance indicator) in the architectural model.

The designers had the possibility to deliver a first draft for feedback and had to deliver their model. Examination of the delivered models has resulted in Table 1 concluding the following.

1. The entity, function and aspect were used in all of the models. These seem to be the most common ingredients for developing an architectural model.
2. A couple of designers did not include requirements in their architectural model. These focus on the solution domain and do not explicitly present the relationship between the needs and the solutions.
3. Only two designers used the concept of views. This apparently shows that most of the designers had preferred one integrated view that includes all the elements and their relationship in one single view.

Table 1. This table presents the factual observations on the use of AMDesigner.

Project subject	Requirements	Entity	Entity relationship	Function	Functional relationship	aspect	dependency	association	composition	Views	parameter
BIM		o	o	o	o	o	o	o	o		
Instrument		o	o	o	o	o	o	o	o		
Automotive	o	o	o	o	o	o	o	o	o	o	
Intensive human interaction	o	o	o	o		o	o	o	o		
Education science	o	o		o		o	o	o			
Numerical analysis	o	o	o	o	o	o	o	o	o	o	
Control system	o	o		o		o		o	o		o
Education science	o	o		o		o	o	o			
Process engineering*	o	o		o		o		o			
Automation	o	o		o		o	o	o	o		

*The designer had used some domain-specific notations for describing the architecture.

This has been observed by other researchers (see e.g. (Rozanski et al., 2012)).

4. It seems that the designers who would like to show the architecture model in one single view have to comprise for the use of available
5. elements. In other words, they have not used all the available elements.
6. The designers who did use the concept of views created a more elaborated model and used almost all of the elements presented in the table.
7. The outcome shows that in order to model a key performance indicator through one integrated view (single view), the designers had focused on either the structural or functional view of the system. We did not observe a good balance for modeling a KPI (key performance indicator) in one single view that connects to both functional and structural elements of the system.
8. In general, the designers find the use of parameters less important for developing/communication of their system architecture.

6 CONCLUSIONS

Having clear objectives and the theory of an architecture model, a very practical test was performed to see the use of AMDesigner in a very wide variety of applied projects from product-related, numerical, educational, process related, automotive and control systems. Though the outcomes, one may conclude that:

- the mostly used ingredients of the tool are Entity, function, and aspects.
- a very limited use of parameters was observed for presenting the architectural model.
- the use of views can promise a more detailed representation of the architecture model.

ACKNOWLEDGEMENT

The authors would like to acknowledge IPCR130065 for providing the means for conducting this research.

REFERENCES

Bonnema, G. M. (2011). Insight, Innovation, and the Big Picture in System Design Application of FunKey Architecting. *Systems Engineering, 14*(3), 223–238. doi:Doi 10.1002/Sys.20174

Bonnema, G. M. (2014). Communication in Multidisciplinary Systems Architecting. *Procedia CIRP, 21*, 27–33. doi: http://dx.doi.org/10.1016/j.procir.2014.03.188

Cabrera, A. A. A. (2011). *Architecture-Centric Design: Modeling and Applications to Control Architecture Generation.* (PhD), TUDelft.

DoD. (2015). DoD Architecture Framework, V2.0, Volume I (Vol. I: Overview and Concepts).

GROUP, O. (2011). Open Group Standard TOGAF Version 9.1. U.S.: The Open Group.

INCOSE. (2015). *A World in Motion, Systems Engineering Vision 2025*.

ISO. (2013). ISO TC 184/SC 5\WG 1 N 516 ISO/PDPAS 19450 ISO TC 184/SC 5/WG 1 Automation systems and integration – Object-Process Methodology.

ISO/IEC. (2012). ISO/IEC 19505-1 First edition 2012-04-15 Information technology – Object Management Group Unified Modeling Language (OMG UML) – Part 1: Infrastructure.

OMG. (2011). Business Process Model and Notation (BPMN).

OMG. (2015). OMG Systems Modeling Language (OMG SysML^{TM}).

Rajabalinejad, M., & Bonnema, G. M. (2014). *Determination of stakeholders' consensus over values of system of systems.* Paper presented at the Proceedings of the 9th International Conference on System of Systems Engineering: The Socio-Technical Perspective, SoSE 2014.

Rajabalinejad, M., Bonnema, G. M., & Houten, F. J. A. M. v. (2015). *An integral safety approach for design of high risk products and systems*. Paper presented at the Safety and Reliability of Complex Engineered Systems Zurich, Switzerland.

Rozanski, N., & Woods, E. (2012). *Software systems architecture: working with stakeholders using viewpoints and perspectives*: Addison-Wesley.

Theisens, H. C. (2014). *Climbing the Mountain*. Amsterdam: Lean Six Sigma Academy.

Woestenenk, K. (2014). *Consistency, Integration, and Reuse in Multi-disciplinary Design Processes.* (PhD), University of Twente, Enschede.

Woestenenk, K., Bonnema, G. M., Cabrera, A. A. A., & Tomiyama, T. (2012). Capturing Design Process Information in Complex Product Development. *Proceedings of the Asme International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 2011, Vol 2, Pts a and B*, 1351–1360. Retrieved from <Go to ISI>://WOS:000324221200139

www.egov.gov. (2013). Federal Enterprise Architecture Framework.

Zachman, J. A. (1987). A Framework for Information-Systems Architecture. *Ibm Systems Journal, 26*(3), 276–292. Retrieved from <Go to ISI>://WOS:A1987M250000003