Approaches to Design Complex Software Systems

Chaimae Ouali-Alami[a], Abdelali El Bdouri, Nisrine Elmarzouki, Ayoub Korchi and Younes Lakhrissi[b]
SIGER Laboratory, Sidi Mohamed Ben Abdellah University, Fez, Morocco

Keywords: Complex software system, Decomposition, Multi-modelling, Design.

Abstract: Despite the evolution of design techniques in the field of software engineering, building complex computer systems remains a very difficult task for the modelling, design and analysis team due to the complexity, richness of information and greatest scope. The modelling of complex software systems is a very rich and immense field of research. Decomposition of complex systems into smaller pieces or components is the best available tool to reduce complexity and dimension also to facilitate construction by humans. System decomposition is a fundamental component of the systems modelling process. In this paper, we will present some methods and techniques of model decomposition. We will explain the importance of decomposition, itemize decomposition methods as we will focus on a set of approaches.

1 INTRODUCTION

The ubiquity of technology in all fields; computer science, mechanics, industry, economics, and commerce, requires a lot of effort in terms of analysis, modelling, and design. Conceptual models have been created: (1) to understand a problem and its context, (2) to provide a basis for analysis and development. This task facilitates the choice of the most suitable solution. With the increased need of users and more deep requirements, the complexity of the studied problem increases more and more. Solve such complex and difficult systems become an urgent need. The system decomposition method is the best way to manage and analyse a complex system by decomposing it into elementary problems to pick out an adequate solution.

To attack a complex system, it is necessary to determine the needs of the actors although the technical requirements (Lakhrissi, 2010) whose construction of the global model remains a difficult task, despite the evolution of analysis and design techniques in the field of software engineering. Multi-modelling approaches are model-oriented approaches, using model development separately. To discuss these so-called model-oriented approaches, it is important to return to object-oriented approaches (Abiteboul, 1991) (Kriouile, 1995) (Harrison, 1993), that is limited when faced with a multidimensional, complex, and highly parallel requirements.

To cope with this complexity, we use, more and more, so-called multi-model modelling approaches. This method offers good practices of decomposition.

This paper focuses exclusively on the design of the complex system. In the second section, we present the decomposition of systems and some good practices. Then in the third section, we present four multi-modelling approaches: (1) point of view modelling, (2) role modelling, (3) aspect modelling, (4) subject modelling. Finally, the last section presents the conclusion and possible future works.

2 DECOMPOSITION METHODS

When we are facing a complex system, we need to break them down into sub-systems. For this, we must: (1) estimate the complexity and the size of the sub-systems, (2) scale the system into pieces, main functions or main sub-systems, (3) split each system into sub-functions or components.

[a] https://orcid.org/0000-0003-3862-9149
[b] https://orcid.org/0000-0003-2718-7090

Ouali-Alami, C., El Bdouri, A., Elmarzouki, N., Korchi, A. and Lakhrissi, Y.
Approaches to Design Complex Software Systems.
DOI: 10.5220/0010740200003101
In *Proceedings of the 2nd International Conference on Big Data, Modelling and Machine Learning (BML 2021)*, pages 533-537
ISBN: 978-989-758-559-3

In his paper (Chiriac, 2011) Chiriac proposes three approaches. The first one refers to the Disassembly/Assembly method following a set of instructions: (1) visually inspect the system, (2) identify sub-assemblies that can be easily removed, (3) remove sub-assemblies that are probably represented by large pieces, (4) record them in a DSM (Design Structure Matrix). The second proposed approach is functional decomposition that implements a decision by disciplines up to components to fill a DSM. The last approach is a service-based decomposition.

The Design Structure Matrix (DSM) is a compact representation of a system or project. It consists of a list of subsystems, activities and their associated information exchange patterns. The information pieces that are required to start an activity are referred to as parameterizations. The output information is also used by other tasks within the matrix. There are three types of DSM configuration:

- Parallel: fully independent design elements,
- Sequential: the second parameter depends on the output of the first,
- Coupled: all parameters have a dependency between them.

The figure 1 bellow illustrates those types with their graph representation and DSM representation (Guenov, 2004).

Stephen proposes a modelling process (Topper, 2013) through the creation of a set of artefacts:

- Description of the system and its environment "Domain Model".
- Preparation of a diagram of "Use Case".
- Division of the use cases in detail by showing the flow of activities and transitions state between components.
- Broaden the domain model by adding attributes and operations. It is an iterative process that tires to create the final result and to complete the solution within its problem domain.

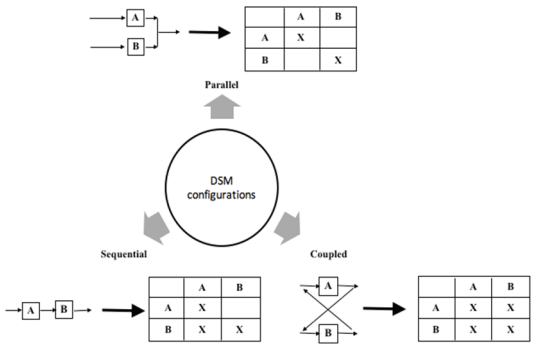

Figure 1: DMS configurations

The notion of decomposition presented in (Guenov, 2004) has two objectives: (1) basics for the complex systems analysis, (2) basics for the complex systems design. According to Yair Wand, he quoted in his paper (Wand, 1990) a very good point in the principles of good decomposition concerning the subsystems «weakly coupled». Before the definition of the sub-systems developed by the designers, he defined a set of transformation rules before the decomposition to check the complete system design afterwards.

Good design practice is axiomatic design that insists on (1) design decision in domain and hierarchy, (2) maintenance of functional requirements, (3) minimization of design information content.

Quite a lot of research supports multi-agent systems (MAS), MAS is recognized as abstraction technologies for the modelling and construction of complex, and autonomous systems called useful and efficient. This poses a challenge due to the distribution and openness of these systems, the autonomy of ownership, as well as the complex nature of the internal functionalities of agents and interactions between agents(Alaca, 2021).

To develop MAS using model-led engineering (MDE) techniques, the most popular way is Domain-Specific Modelling Language (DSML) (Sredojević, 2018) with an Integrated Development Environment (IDE).

3 MULTI- MODELLING APPROACHES

When tackling the complexity of huge software systems, separation of concerns is important for keeping the event process, the produced models and therefore the code manageable. The separation of concerns are often wiped out alternative ways, but the objective is always the same: having the ability to spot relatively independent "parts".

In this section, we present the four major multi-modelling approaches: Views modelling, Aspects modelling, Subject modelling and Role modelling.

3.1 Point of View Modelling

VBOOM (View-based Object-Oriented Method) (Kriouile, 1995) is an object-oriented programming method that use the point of view concept and the view-oriented approaches based on UML(Anwar, 2009).

The VUML language (View-based Unified Modelling Language) (Nassar, 2005) is a UML profile based on the point-of-view modelling approach to deal with the limitations of object-oriented programming. Primarily the implementation of views by multiple instances. This approach is based on the concept of point of view to analyse and design a software system, in particular, the establishment of several partial models. It is applied especially to an information system which is often characterized by a strong interaction with users; whose actor is defined (Lakhrissi, 2010) as a human being or an entity that interacts with the system. A point of view can be defined as a vision of an actor on a part of the whole of the system. A View is a modelling entity on which the point of view of the actor is applied. Generally, the patterns associated with software developments could be several types of actors:

- development actors when designing and implementing the pattern :Analyst, designer, programmer, tester, and maintainer.
- final-users when instantiating the pattern in a given application domain (figure 2).

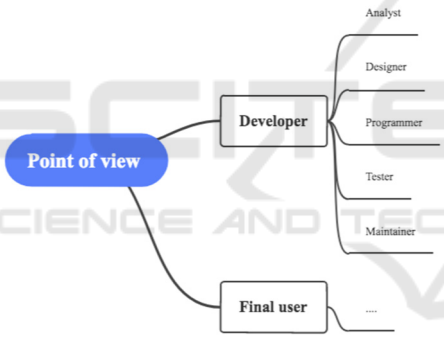

Figure 2: Hierarchy of viewpoints

In view approaches, the decomposition level is different from that adopted by the aspect approaches. This is often decomposition consistent with the views of actors of the system. The views are developed independently of any other sub-system and without making any distinction between the essential functionalities and the cross-functionalities. The results of this process of decomposition may be a set of entities described by the subjective views of the actors of the system.

3.2 Aspect Modelling

After an advanced search for aspect-oriented programming, this approach has reached maturity (Chavez, 2002). There is a need to address a fair amount of interdependent design problems to have software. The purpose of modelling by aspect (France, 2004) is to create software that takes into account safety, fault tolerance (Sutton, 1999) and to treat reliability problems. In a particular way, this problem can make difficult or impossible to solve other problems. That pushes the developers to adopt aspect-oriented modelling (AOM) approach that facilitates the task of exploring other ways to address concerns during software modelling. Then this approach focused mainly on the balance between the reliability of the keys and other concerns during the early stages of development. This can help developers better manage product risks through the identification and early resolution of conflicts and behaviours that address different concerns. In this context, the authors define a Concern as a problem associated with the desired objective. The Concern Solution Model is a model that describes how the concern is addressed (Chavez, 2002) (France, 2004).

We distinguish between two types of concerns:

- A concrete concern: includes solutions that can be expressed in functional and structural terms in a model (Access control and error recovery).
- A qualitative concern: based on the qualities or attributes of a system (related to system performance and memory usage).

In general, traditional design has difficulties in creating and above all in having complex and reliable systems evaluated due to several factors such as the difficulty of changes related to these concerns, the dissemination of information has become more difficult, especially with the multiplicity of design teams (Noda, 1999). The mechanisms provided by multidimensional concern separation techniques (MDSoC: Multi-Dimensional Separation of Concern) can reduce the cognitive burden of creating and evolving reliable software (Tarr, 1999).

The aspect method decomposes the system into functional units and non-functional units (Andersen 1992).

3.3 Role Modelling

In object-oriented programming, a class is a type of data that is complemented by a formal or informal OCL description of their behavioural effect, whereas when faced with a method that is part of an activity involving several communicating objects. It is difficult to understand this description and difficult too to add a description for each relevant class. That lead to the emergence of role-based modelling, it allows the designer to focus on several aspects or a single aspect with several independent levels of detail. The designer has to build a role model for each

activity or task or to build several role models for the same activity at different levels of detail. This shows that role modelling is more adjacent to the functional approach than the objects approach.

A role model can be defined as a design unit that identifies a structure that includes two or more entities in external role interaction. The latter is considered as a requirement or liability (Kendall, 1999) (Kristensen, 1996). Saying that, an object can be considered as a particular role, which explains the definition of role as a temporary point of view (Riehle, 2000). Moreover it is a type that describes the view that an object vis-à-vis another which allows an object to play different roles at a given moment (Ossher, 1995).

In the role approach, it is to represent an entity of the model through multiple objects. Each object models an exact role played by an entity. Unlike modelling by views, roles are objects resulting from local entities subjective views without being linked to actors of the system. The various subjective views of the system represent all of the appliance concerns, and every concern is represented by the various roles and their interactions.

3.4 Subject Modelling

Harrison introduced another technique of separation of concern (Harrison, 1993): Subject-based programming based on multidimensional separation of concerns. It is defined as a visibility of the world as a whole by a particular application or tool, and so a subject and a generalized perception, whose characteristics of an object can be applied to any other object. An object can activate several subjects and a subject can be activated for several objects. This approach present a group of advantages like unscheduled extension and composition, decentralized class development, also a code that implements a feature that will be programmed as a subject [23].

The subject approach extended by the MDSoC approach (Multidimensional Separation of Concerns) offers a decomposition of the system into more arbitrary dimensions, where each dimension may be a collection of particular concerns. We mention concern within the broad sense, without differentiating between basic concerns or crosscutting concerns.

4 SYNTHESIS

The concept of decomposition methodology is to interrupt a posh classification task into simpler and more manageable steps, which can be solved by using existing induction methods.

The main interest of the modular approach is to facilitate the construction of systems by allowing 1) to write the module with little knowledge about the other code modules, and 2) to replace one or more modules without reassembling all the system.

The construction of the system is more understandable, manageable, and maintainable. One limitation of the approach is we do not have the possibility to personalize a module, it means to allow make many variations of an existing module.

The Architectural Description approach focuses on designing systems by assembling architectural bricks. Three concepts of architectural bricks are popularly accepted components, connectors and configurations. Architectural bricks are modelling portions of an abstract system without defining their implementations. So we can use these portions when the system is in the analysis phase to allow create the architecture description. This description can help us later to make simulations on the system or to allow other people to analyse the system. Therefore, the variety of components created compatibility problems, portability, multiplicity, complexity and lack of standardization.

5 CONCLUSION

This paper has described a set of approaches to support system analysis and system modelling and system design using the decomposition process. Current approaches provide good support for modularizing systems along a few dimensions. They can significantly enhance support for separation of concerns, and also can help developers to manage the complexity of building complex software. The process of decomposition involves a step-by-step process where a global representation of a system is proposed (Models Composition). This step is necessary to get a complete global representation of the system under construction.

The concept of the composition (Elmarzouki, 2016) model is a challenging topic to discuss as it concerns to the definition of new approaches.

REFERENCES

Lakhrissi, Y., 2010. Intégration de la modélisation comportementale dans la conception par points de vue (Doctoral dissertation, Universite Toulouse le Mirail-Toulouse II).

Abiteboul, S. and Bonner, A., 1991. Objects and views. ACM SIGMOD Record, 20(2), pp.238-247.

Kriouile, A., 1995. VBOOM, une méthode orientée objet d'analyse et de conception par points de vue. Unpublished doctoral dissertation, University Mohammed V, Rabat, Morocco.

Harrison, W. and Ossher, H., 1993, October. Subject-oriented programming: a critique of pure objects. In Proceedings of the eighth annual conference on Object-oriented programming systems, languages, and applications (pp. 411-428).

Chiriac, N., Hölttä-Otto, K., Lysy, D. and Suh, E.S., 2011. Three approaches to complex system decomposition. In DSM 2011: Proceedings of the 13th International DSM Conference.

Guenov, M.D. and Barker, S., 2004, January. Requirements-driven design decomposition: A method for exploring complex system architecture. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (Vol. 46962, pp. 145-151).

Topper, J.S. and Horner, N.C., 2013. Model-based systems engineering in support of complex systems development. Johns Hopkins APL technical digest, 32(1).

Wand, Y. and Weber, R., 1990. An ontological model of an information system. IEEE transactions on software engineering, 16(11), pp.1282-1292.

Alaca, O.F., Tezel, B.T., Challenger, M., Goulão, M., Amaral, V. and Kardas, G., 2021. AgentDSM-Eval: A framework for the evaluation of domain-specific modeling languages for multi-agent systems. Computer Standards & Interfaces, 76, p.103513.

Sredojević, D., Vidaković, M. and Ivanović, M., 2018. ALAS: agent-oriented domain-specific language for the development of intelligent distributed non-axiomatic reasoning agents. Enterprise Information Systems, 12(8-9), pp.1058-1082.

Anwar, A., 2009. Formalisation par une approche IDM de la composition de modèles dans le profil VUML (Doctoral dissertation, Thèse de doctorat, Université de Toulouse).

Nassar, M., 2005. Analyse/conception par points de vue: le profil VUML (Doctoral dissertation).

Chavez, C. and Lucena, C., 2002, April. A metamodel for aspect-oriented modeling. In Workshop on Aspect-Oriented Modeling with UML (AOSD-2002).

France, R., Ray, I., Georg, G. and Ghosh, S., 2004. Aspect-oriented approach to early design modelling. IEE Proceedings-Software, 151(4), pp.173-185.

Sutton Jr, S.M., 1999. Stanley, J., & Sutton, M. (1999, November). Multiple Dimensions of Concern in Software Testing. In First Workshop on Multi-Dimensional Separation of Concerns in Object-oriented Systems (OOPSLA'99).

Noda, N. and Kishi, T., 1999, November. On Aspect-Oriented Design-Applying Multi-Dimensional Separation of Concerns on Designing Quality Attributes. In First Workshop on Multi-Dimensional Separation of Concerns in Object-oriented Systems (OOPSLA'99).

Tarr, P., Ossher, H., Harrison, W. and Sutton, S.M., 1999, May. N degrees of separation: Multi-dimensional separation of concerns. In Proceedings of the International Conference on Software Engineering (IEEE Cat. No. 99CB37002) (pp. 107-119). IEEE.

Andersen, E.P. and Reenskaug, T., 1992, June. System design by composing structures of interacting objects. In European Conference on Object-Oriented Programming (pp. 133-152). Springer, Berlin, Heidelberg.

Kendall, E.A., 1999, October. Role modelling for agent system analysis, design, and implementation. In Proceedings. First and Third International Symposium on Agent Systems Applications, and Mobile Agents (pp. 204-218). IEEE.

Kristensen, B.B., 1996. Object-oriented modeling with roles. In OOIS'95 (pp. 57-71). Springer, London.

Riehle, D., 2000. Framework design: A role modeling approach (Doctoral dissertation, ETH Zurich).

Ossher, H., Kaplan, M., Harrison, W., Katz, A. and Kruskal, V., 1995, October. Subject-oriented composition rules. In Proceedings of the tenth annual conference on Object-oriented programming systems, languages, and applications (pp. 235-250).

Elmarzouki, N., Lakhrissi, Y. and Elmohajir, M., 2016, March. A study of behavioral and structural composition methods and techniques. In 2016 International Conference on Information Technology for Organizations Development (IT4OD) (pp. 1-6). IEEE.