European Journal of Computer Science and Information Technology
Vol.4, No.1, pp.42-55, January 2016
Published by European Centre for Research Training and Development UK (www.eajournals.org)

INTRODUCING A MULTIVIEW SOFTWARE ARCHITECTURE PROCESS BY EXAMPLE

Ahmad K heir[1], Hala Naja[1] and Mourad Oussalah[2]

[1]Faculty of Sciences, Lebanese University
[2]LINA Laboratory, University of Nantes

ABSTRACT: *Although every software system has an architecture, not every system has an architecture that is effectively communicated via an architecture description. A good architecture description is a Road map for engineering that guides designers, engineering teams and managers to understand what the system is expected to do and how it will do it. It is ultimately the visible representation of the hidden system architecture. In this paper a walkthrough an architecture definition process, called MoVAL-ADP, is presented. This process is associated to an architecture description approach called MoVAL that is based on multiviews/multi-hierarchy concepts and is consistent with the ISO/IEC/IEEE standard 42010-2011.*

KEYWORDS: Software Architecture, Viewpoint, Abstraction, Architecture Definition Process

INTRODUCTION

An Architecture Description (AD) intends to document a software architecture by containing all the information needed to communicate this architecture effectively with different categories of stakeholders who need to understand the system from different perspectives and cover different levels and aspects in different circumstances [Oussalah1].

In order to address this disparity of different stakeholders' perspectives and level of understanding, the concept of viewpoints was introduced in many previous important solutions which contributed effectively in software engineering and were the basis of other works. Among those solutions we can mention the "4+1" View Model [Kruchten], the Views & Beyond approach [Clements], the software systems architecture approach that was presented by N. Rozanski and E. Woods [Rozanski], Siemens model [Soni], Zachman framework [Zachman], and the Reference Model of Open Distributed Processing (RM-ODP) [Raymond].

However, these solutions had many limitations that hampered the optimal benefit of them as we presented extensively in another paper [K heir].

Hence, we have designed our approach MoVAL (Model, View and Abstraction Level based software architecture). By this approach, we intended to provide concepts describing different aspects (called views) of the system at multiple levels of details (abstraction levels) according to the stakeholders needs. Then, in order to guide the software architect to produce effectively and incrementally the AD using those concepts, we have suggested an architectural description process (ADP) that we called

MoVAL -ADP.

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

In the next section of this paper, we will introduce very briefly the basic concepts of our approach MoVAL and its associated ADP. Then we will apply it to develop a university management system in the latter section.

MoVAL approach

Despite the important advantages conducted by the viewpoint based software architecture approaches in general, these approaches still suffer some limitations like the treatment of complexity that resides inside a single view, the lack of guidelines to apply these approaches or the lack of architecture description processes, and the resolution of inter-views inconsistencies.

In order to cure these limitations, we have designed a new approach called MoVAL that reduces the complexity inside an AD by applying the concept of decomposition in three different but complementary aspects. Thus MoVAL :

- decomposes an AD into Views;
- decomposes modeling details in a View according to several abstraction levels;
- decomposes the architecture definition process (ADP) into phases. The proposed ADP is incremental. It guides the architect to produce an AD of the target system simultaneously while others (analysts/designers/developers) build this system. The ADP is decomposed into phases.

MoVAL is a software architecture approach that is based on the construction of multiviews architectures, and the decomposition of each architectural view to several abstraction levels conceptualized in two different dimensions: the achievement levels, and the description levels. It is based on the IEEE 42010 standard [IEEE], which was designed by the IEEE architecture planning group (APG) in order to formalize the definition of multi-views software architectures and their main elements. Thus, MoVAL inherits the definition of its main concepts from this standard and extends those definitions to handle the multi-hierarchy aspect introduced in this approach.

Also, MoVAL adopts the Model-driven Architecture (MDA) strategy which was launched by the Object Management Group (OMG). The MDA provides a set of guidelines for the structuring of specifications, which are expressed as models. The strategy of this approach is to define system functionality using a platformindependent model (PIM), then to translate the PIM to one or more platform-specific models. In fact, MoVAL adopts this strategy by allowing the architect to define several achievement levels and represent each of those three levels defined in the MDA standard by a number of achievement levels.

MoVAL basics

One of the most important concepts of our approach is the architectural view or simply view. Based on the IEEE definitions, a view is a representation of the system considering, from one side, a set of the development process' aspects, and from another side some problems associated to a specific category of stakeholders or a group of categories of stakeholders. A view represents the whole system from the perspective of a related set of concerns and it conforms to multiple viewpoints, which describe the rules and conventions used to create a view based on this viewpoint. In MoVAL, a viewpoint identifies the semantics and syntax and their

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

associated tools, called formalisms, which shall be used afterwards to model the inherent views. Hence a view in MoVAL is the entity that makes up the software architecture description. In fact it is common that the construction of a software architecture is too complex to be performed easily. Hence, in order to control this complexity in a gradual way, we have defined in MoVAL a hierarchy that makes appear different levels of understanding, or different abstraction levels, in a view. The application of such concept in a software architecture approach helps the architect to move forward step by step in the software architecture construction leading thereby to treat the systems complexity. Accordingly, abstraction in MoVAL is a term that refers to the definition of a set of models of a software architecture more concretely. So, when the software architect, in MoVAL, decides that he needs some additional details to satisfy the inherent requirements or to concretize their realization, he creates a new abstraction level containing the needed details. Actually, two types or dimensions of abstraction were defined in MoVAL, which are: (1) the achievement abstraction and (2) description abstraction (cf. achievement level and description level).

Actually, we have defined an achievement level as a set of artifacts defining an architecture at a specific phase of the architecture definition process that conforms to a specific viewpoint. For example, one achievement level could have the business study artifacts like use case diagrams, business flow diagrams, etc., and another more concrete achievement level could contain other artifacts showing a way to realize the previous business study like the components or class diagrams and therefore those achievement levels will be related by a link called is_{+A}. Each achievement level contains a set of description levels holding the models and that allow the architect to provide multiple descriptions of different granularity levels starting with the coarsegrained and ending with the Fine-grained models. For example, in the achievement level holding the business study, the software architect could represent the inherent requirements by a high level use case diagram containing high level use cases without going deeper into those use cases' details, and then he could make a zoom-in in a further description level to represent more details about those use cases and therefore those description levels will be related by a link called is_{+D}.

Also, one of the most important features in this approach is that all architectural views could be incorporated in one single rough and consistent architecture in a very simple way. Hence, the architect could link different achievement levels of different views of an architecture, one to the other in order to create a kind of achievement order among the entire set of architecture's achievement levels; those links are called correspondence links. The correspondence between two achievement levels indicates that they hold models and artifacts of the same phase or concurrent phases of the development lifecycle.

MoVAL -ADP

MoVAL-ADP is an architecture definition process that we designed and offered to the architect in order to apply MoVAL approach in a simple way. It complies with the Unified Process (UP)[Booch], which is an iterative software development process.

Thus, MoVAL-ADP's activities and milestones are distributed on the four different phases that was defined in the UP, which are the inception, elaboration, construction, and transition phases.

In the Inception phase the software architect must form a global idea about the scope and context in which the system shall be running. Also, the main groups of stakeholders should be

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

identified in order to gather their expectations about system's offered functionalities in light of the defined context, and their outlook of the heaviest risks that could be encountered.

The second phase of the architecture definition process is the Elaboration phase, where a candidate architecture must be proposed. In this phase, the software architect shall identify different viewpoints that could be applied in the architecture, and the groups of concerned stakeholders for each viewpoint. Then, he must review the important architectural styles of the domain and identify the ones that could be relevant in this case, in order to build and propose upon these information a candidate architecture that he foresees meeting the stakeholders expectations.

The construction phase begins by identifying an overall plan for the construction iteration; so an examination of existing views is required to find out which one should be removed, added or modified. Then, the software architect selects the set of views that will be used during this iteration and picks one to start the construction with by capturing the architectural significant functional and non-functional requirements of the inherent view, eventually by performing some 3-tiers meetings joining the architect, analysts, and the concerned stakeholders. Then, the architect must develop the architectural view in light of the captured requirements and the critical risks identified in the inception phase. So, he will need to identify the considered concerns, and to refine or build upon them a model. In order to achieve this goal, the architect chooses the repository wherein he needs to create a new model or refine an existing one. In case he needs to refine a model, the architect adds further description levels to detail his model, or other achievement levels to concretize his model. After that, the software architect could either define other models for the same set of concerns, or define another set of concerns, or re-organize models into hierarchy levels through the review of the models belonging to the current view. During the next activity, the architect must build the correspondence between different achievement levels of the lately developed view and the achievement levels of other views of the architecture. The architect must then expand his architecture by adding the lately developed view and identify its potential relations with other views in order to solve the inconsistencies that might be existing. Finally, in this phase the architect must evaluate the architecture by his own, and then evaluate it with the concerned stakeholders.

The last phase of the architecture definition process is the transition phase or the fine tuning phase. In this phase, the effort spent on architecture definition is in its lower levels. Actually, this phase of the architecture definition process occurs simultaneously with the construction and transition phases of the development lifecycle, hence the architect must wait for constructive feedbacks from the development team in order to refine his architecture and keep it up-to-date.

Case study

In this section, an example of the application of MoVAL approach will be presented. This case study represents a global system dedicated for the management of different parts and departments of a university like the library, students' affairs, employees' affairs, academic content, etc., so here only those four parts will be considered. Normally, an interesting number of stakeholders have different interests in such a system. In some cases, those stakeholders could have interests in more than one part of the system, and in other cases they could have interests in a single part of it. Sometimes different stakeholders could have interests in the same part of the system but at different levels of details due to a variant level of understanding or interest towards a specific part of the system. Indeed, this will lead to some important

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

difficulties when trying to represent the software architecture of such a system including the organization and interaction of each sub-system with the other subsystems and the coordination of the interests of each stakeholder with the interests of the other stakeholders considering their different levels of details. Also, one of the most important challenges while developing such a system is the ability to organize all the artifacts and models of every part of the system in a good and controlled way. Further in this section, MoVAL approach will be applied to build some parts of an architecture for this system, in order to confirm the contribution of this approach and to highlight the novel ideas and concepts introduced earlier. Thus, a walkthrough MoVAL-ADP will be presented.

Walkthrough MoVAL-ADP

The walkthrough MoVAL-ADP starts with the Inception phase. Table 1 presents a briefing of the results of each activity of the Inception phase.

Table 1. Inception phase activities results

Activity	Results
1- Define initial scope and context	Here a textual document defining the scope of the university management system must be provided and defining also the context in which this system must be running.
2- Identify main stakeholders	Librarian, Employees, Professors, Students, software architect, system analyst, Web developer, Desktop application developer, Database admin, Network engineer, etc. ...
3- Capture first-cut concerns	Here the first-cut concerns of each identified stakeholder must be captured. For example, here are some first-cut concerns for the librarian: Librarian shall be able to manage users (Add/Edit/Remove) Librarian shall be able to manage books Librarian shall be able to reserve books for students Etc.
4- Identify critical risks	Students confidential data stolen Concurrent reservations for the same books to different users Etc.

Also, Table 2 presents a briefing of the results of the Elaboration phase activities.

Table 2. Elaboration phase activities results

Activity	Results
1- Identify different views of the architecture	L ibrary-Functional Student-Functional Employee-Functional Information Physical Etc.
2- Identify relevant architectural styles	N/A
Outline a candidate architecture	Only part of the candidate architecture will be considered in this case study, and this part includes the L ibrary-Functional view, Student-Functional view, Information view, Physical view, etc. this candidate architecture must be somewhat similar to the architecture illustrated in Figure 1.

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

The most important phase of the architecture definition process is the construction phase, in which the software architect must build upon the candidate architecture.

Thus, he must consider each view of the candidate architecture, and develop it. Note that in this application of MoVAL-ADP, only one view, the Library-Functional view, was considered and briefly developed in the remainder of this section. However, the view construction process must be applied normally to develop every view needed to cover the entire system. In addition, while developing a view, every aspect of this view must be covered by creating as much achievement levels, description levels and models as needed. Figure 1 illustrates a part of the potential resulting software architecture built through MoVAL-ADP, obtained at the end of some construction iterations. This figure shows how models and artifacts of the architecture are well organized in a clear, well controlled and coherent hierarchy.

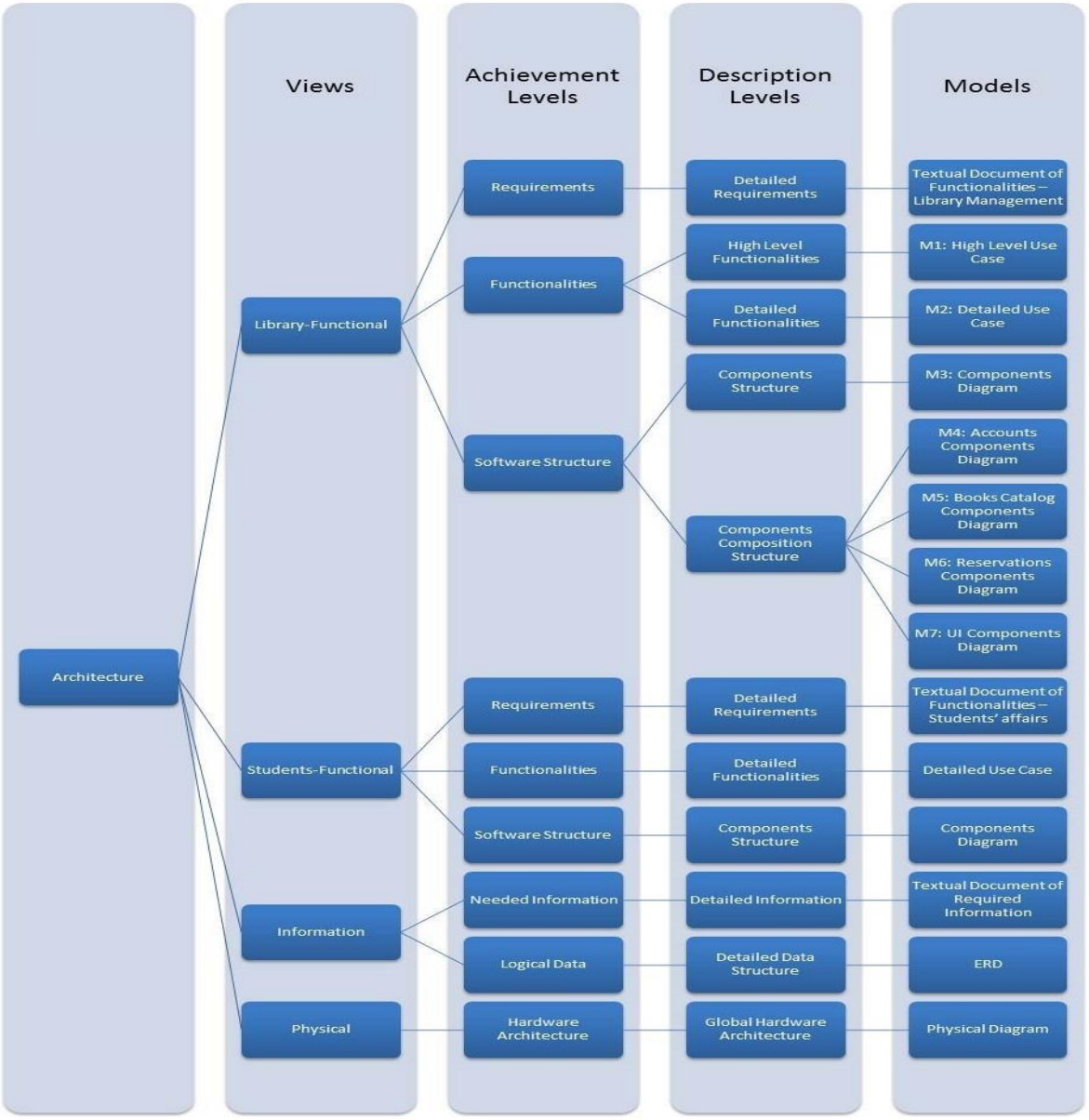

Figure 1. Overview of the university management system's architecture.

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

Library-Functional view achievement and description levels

In the Library-Functional view, three achievement levels have been defined. Obviously, those achievement levels are linked one to the other via "is_{+A}" links and will be presented here starting from the most abstract to the less abstract levels:

- Requirements achievement level: this is the first achievement level of the Library-Functional view which must hold the informal definition of this view's requirements. For this purpose, one single description level is defined, the Detailed Requirements description level, in which one model or artifact will be included, which is a complete textual document of the detailed requirements of this view.

- Functionalities achievement level: the aim of this achievement level is to hold models and artifacts representing the analysis of the detailed requirements that were defined in the first achievement level. Hence, the Functionalities achievement level includes two description levels. The first one, the High Level Functionalities description level, contains one use case model defining the top level use cases of this system. The second level, the Detailed Functionalities description level, contains another use case model that is much more described or detailed in order to develop and clarify the top use cases of the previous model.

- Software Structure achievement level: this is the third achievement level for this view that aims to include models defining how to implement the functionalities extracted and showed in the Functionalities achievement level. Also this level contains two description levels. The first one, the Components Structure description level, contains one high level components diagram defining the global structure of components implementing this part of the system (Library management system). The second description level, the Components Composition Structure description level, contains four components diagrams defining each the composition of a specific subset of the components of the first description levels.

Library-Functional view models

During multiple construction iterations the architecture evolves and develops by refining the existing models and consequently creating new models.

- After some meetings between the analyst, the software architect, and the client, all needed requirements were captured and transformed into a textual document considered as the draft model M0 of the Library-Functional view. This model is the one of the Requirements achievement level of this view;

- M0 will be developed into M1 in the next achievement level to construct the high level use case diagram presented in Figure 2. This figure represents model M1 which is a global vision of the functionalities that must be considered for the library management part of the university management system;

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

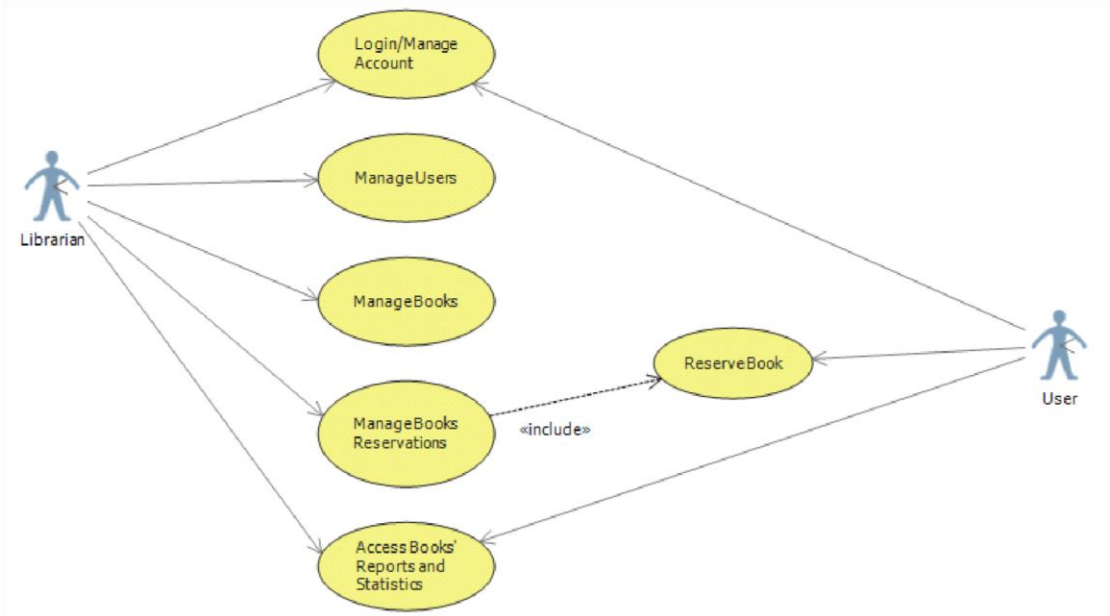

Figure 2. M1: High level use case diagram of the Library-Functional view.

- After that, M1 must be transformed into other less abstract models. Thus, Figure 3 presents another model M2 at the same achievement level but at a higher description level. It presents another use case diagram for this view in which the use cases of the model M1 were expanded to better clarify all the features that must be offered in this part of the system;

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

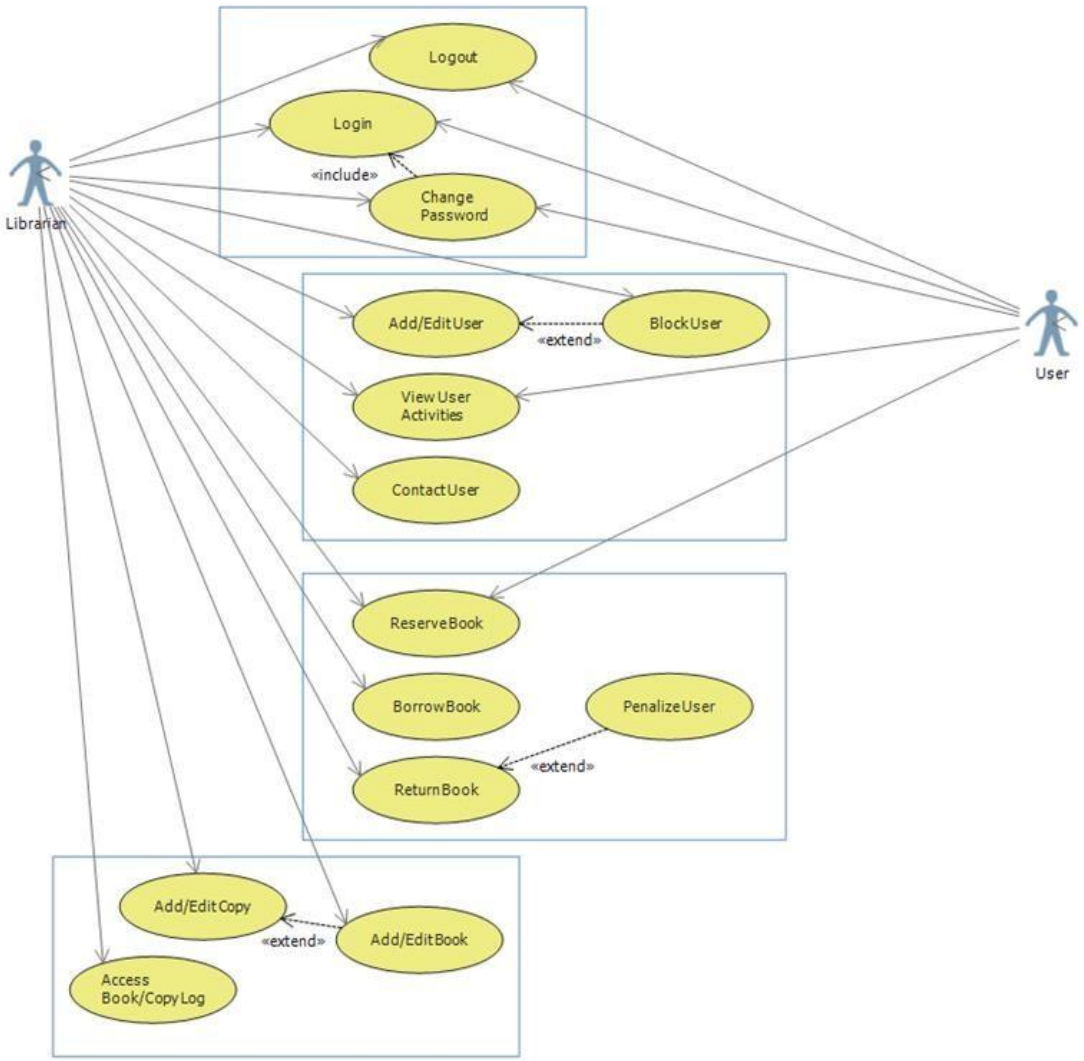

Figure 3. M2: Detailed use case diagram of the Library-Functional view.

- Figure 4 illustrates another model M3 for the Library-Functional view but this time at a higher achievement level. It presents a components diagram showing a way to realize what was intended in the previous use case diagrams M1 and M2;

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

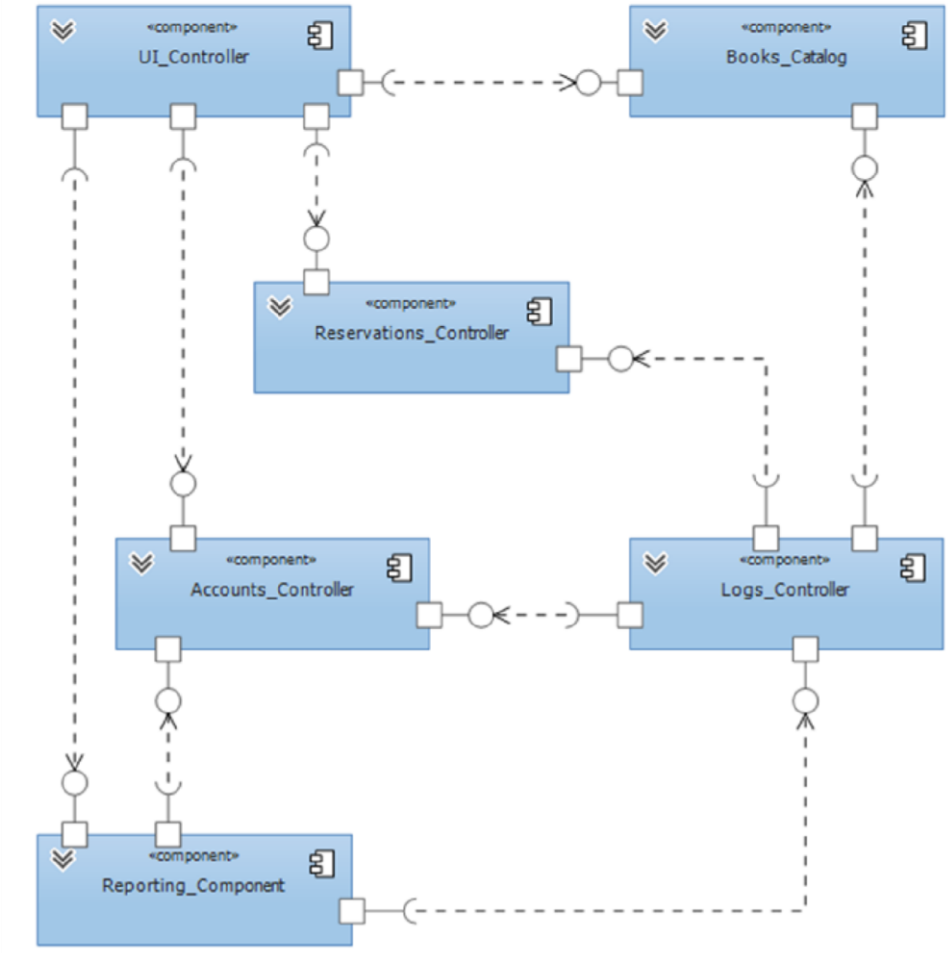

Figure 4. M3: Components diagram of the Library-Functional view.

- In addition, figures 5, 6, 7 and 8 show four models representing different expanded parts of the components diagram of Figure 4:
    - Model M4 of Figure 5 shows the components structure that allows management of all system's accounts;

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

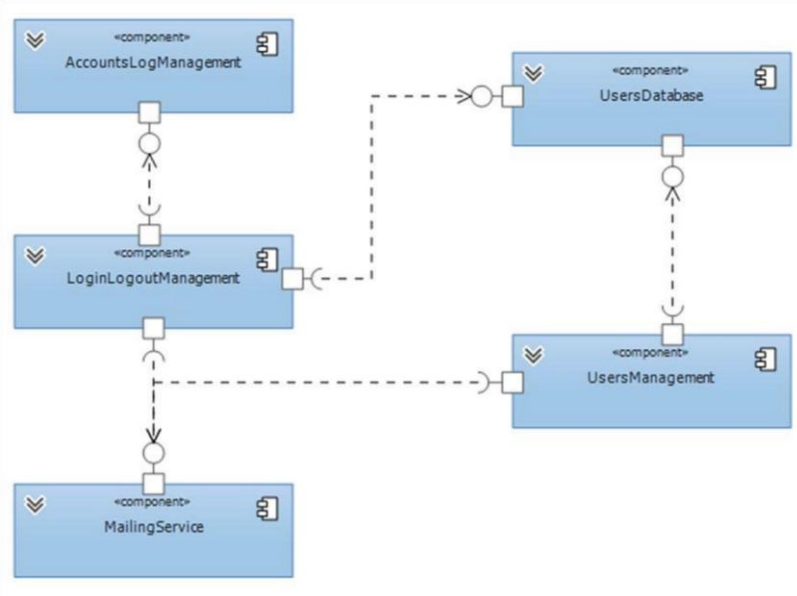

Figure 5. M4: Components diagram of the accounts part of the Library-Functional view.

o Model *M5* of Figure 6 presents the components structure of the books catalog;

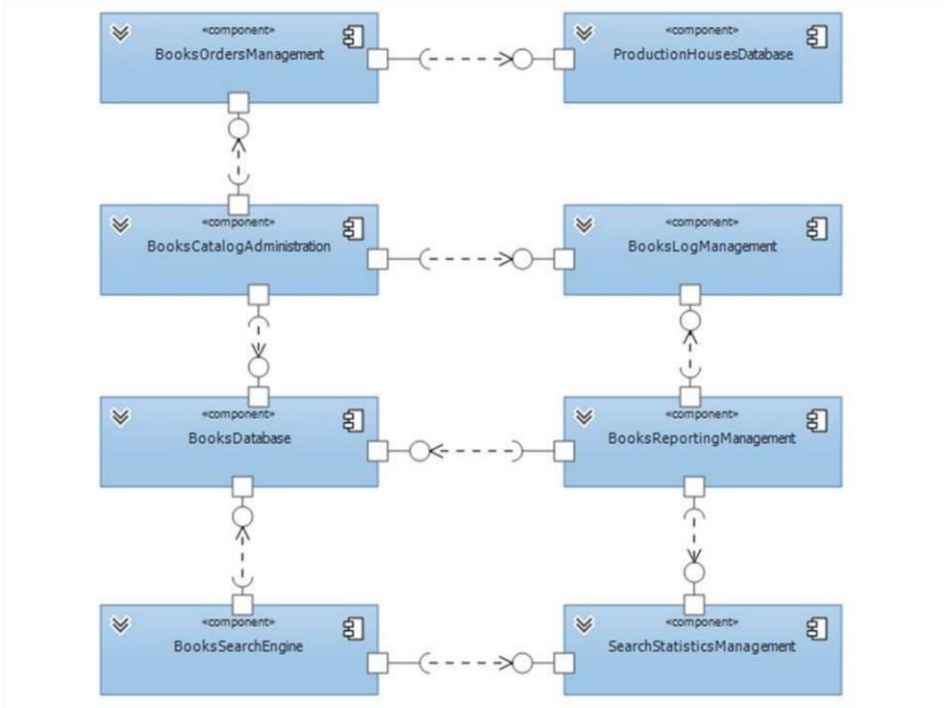

Figure 6. M5: Components diagram of the books catalog part of the Library - Functional view.

o Model M6 of Figure 7 presents the components structure of the reservations part of the system. In this model the component BooksDatabase is shown with different color to indicate that its defined externally in another model and reused in this model;

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

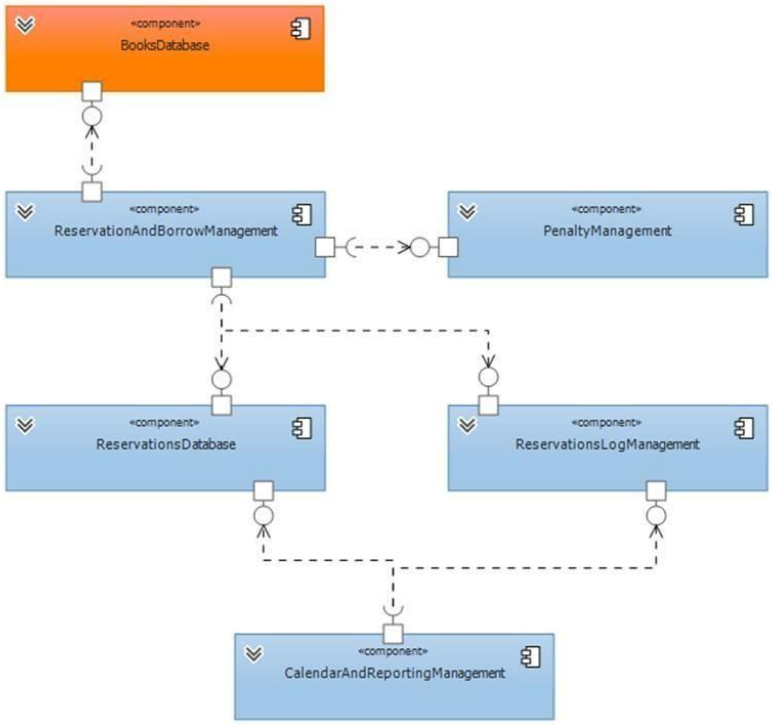

Figure 7. M6: Components diagram of the reservations part of the Library-Functional view.

- Model M7 of Figure 8 presents the structure of the user interface components. Hence, in this model some components are shown with different color indicating that they are defined externally other models and reused in this model.

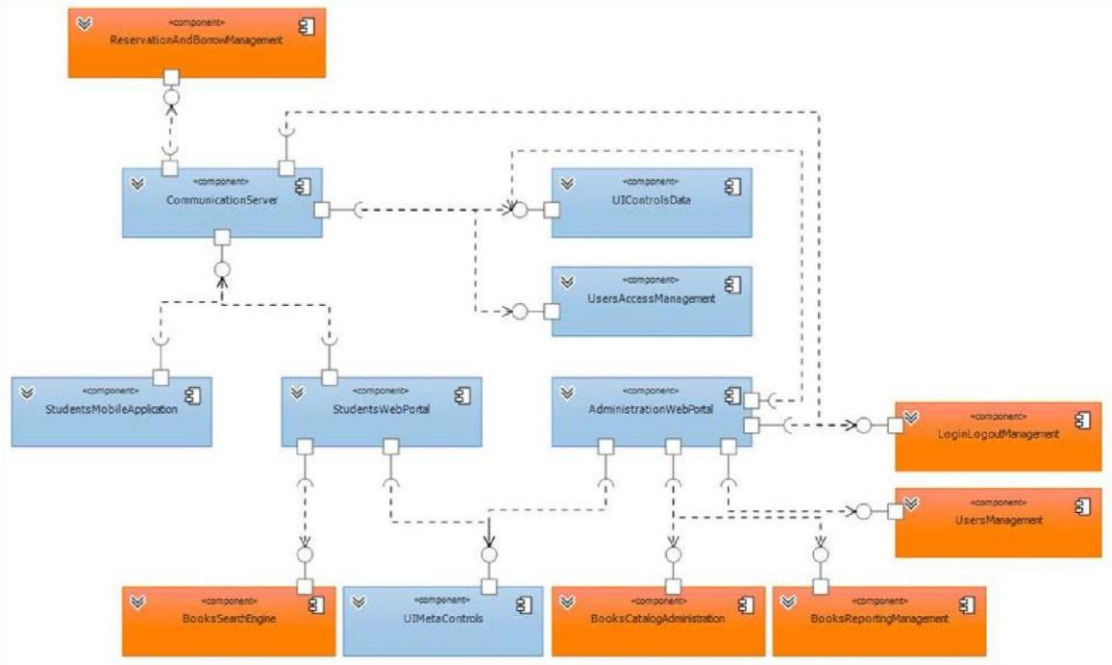

Figure 8. M7: Components diagram of the interface part of the Library-Functional view.

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

Structural links between models definition

The models of the Components Composition Structure description level of the Software Structure achievement level have some kind of relations between them. Actually, some components defined in the models representing books catalog, reservations, and accounts, are referenced in the model representing the user interface components. Thus, three relations is_R were defined between these models. In addition, one component of the model representing the books catalog was referenced in the model representing the reservations. Thus here also a relation is_R was defined between the models representing the books catalog and the reservations.

Figure 9 illustrates two achievement levels of the Library-Functional view which are the Functionalities achievement level and the Software Structure achievement level, their models, and all the relations or links that was defined.

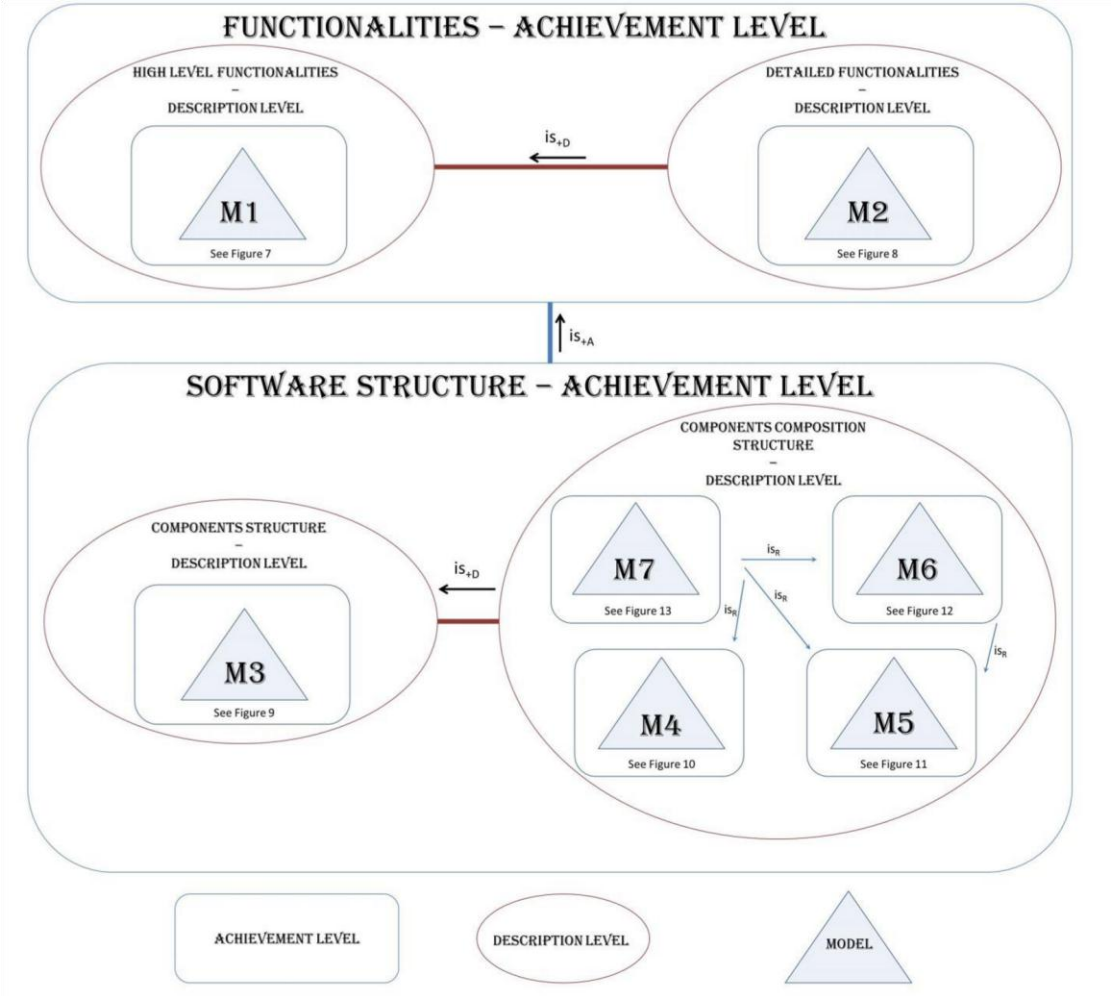

Figure 9. Overview of some elements of the Library-Functional view and relations between them.

Normally, when the architect finishes the construction of a view, he must check for consistency problems between this view and other views of the architecture, and resolve them by adding some architectural links. This part of the approach is out of this paper's focus and will not be detailed.

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

Finally, the architect must evaluate the newly constructed view and the entire architecture, and re-evaluate it with the concerned stakeholders, in order to generate an evaluation report that must be taken into account in the next iteration.

CONCLUSION

MoVAL is a software architecture approach that is based on the IEEE 42010 standard and that complies with the Model Driven Architecture standard (MDA). This approach is based on the definition of multiple views and multiple hierarchy levels for each view in order to construct a complete architecture using a specific architecture definition process called MoVAL-ADP.

In this paper, we present a walkthrough for the application of MoVAL approach for the development of a university management system in order to confirm its feasibility, contribution, and importance.

REFERENCES

[Booch] Booch, Grady, J. I. and Rumbaugh, J. (1999). The unified software development process. Addison-Wesley.

[Clements] Clements, P., Bachmann, F., Bass, L ., Garlan, D., Ivers, J., L ittle, R., Nord, R., and Stafford, J. A practical method for documenting software architectures.

[IEEE] ISO/IEC/IEEE (2011). Systems and software engineering – architecture description. ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000).

[K heir] K heir, A., Naja, H., and Oussalah, M. (2014). MoVAL , a new approach to software architecture and its comparison with existing views based approaches in software engineering. INFOCOMP Journal of Computer Science, 13(1):26– 37.

[K ruchten] K ruchten, P. (1995). The 4+ 1 view model of architecture. Software, IEEE, 12(6):42 – 50.

[Oussalah1] Oussalah, M. (2014a). Software Architecture 1. Wiley, Paris.

[Raymond] Raymond, K . (1995). Reference model of open distributed processing (RM-ODP): introduction. In Open Distributed Processing, pages 3–14. Springer.

[Rozanski] Rozanski, N. and Woods, E. (2011). Software systems architecture: working with stakeholders using viewpoints and perspectives. AddisonWesley.

[Soni] Soni, D., Nord, R. L ., and Hofmeister, C. (1995). Software architecture in industrial applications. In Software Engineering, 1995. ICSE 1995. 17th International Conference on, pages 196–196. IEEE.

[Zachman] Technology, M. (2008). Mdg technology for zachman framework user guide. Technical report, Zachman Entreprise

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)