Effect Detection of Software Architecture Changes in xADL

Artit Udomsomruedee and Wiwat Vatanawood

***Abstract*—Software architecture is the process of defining solution that meets all of the technical and operational requirements. It involves a set of related software elements and their relationships to be constructed afterwards in the rest of development phases. A good software architecture design obviously ensures the quality of software product. Typically, the changes of software architecture model during the development phases may effect the expected design rationales, its performance and the complexity of the software product.**

In this paper, we propose a tool called "xADL: Software Architecture Changes Effect Detection Tools – xSACEDT" in order to detect the effects of the software architecture changes. The original software architecture model, written in xADL, will be compared with the new model. All of the modification issues will be detected and reported. Moreover, the effects on requirement checklists and its design quality attributes are also traced and alerted.

***Index Terms*—Effect detection, Architecture model, quality attribute, software architecture, design rationale**

I. Introduction

SOFTWARE architecture [1] has become crucial and mandatory for the large scale software design and development. It is a key to control the complexity and performance of the software product. Therefore, the software architectural design method is commonly conducted to ensure the expected quality of the software product even before writing the source codes and during the maintenance period.

In general, as shown in figure 1, the software architectural design method considers the given software requirements specification as an input which provides a list of mandatory functional and non-functional requirements. A preliminary software architectural design model should be outlining the overall functional checklists.

Right after the modeling of software architectural model, the process of assessment takes place in order to estimate the expected quality attributes. In case of the model does not meet with the user needs and system constraints, it should be reconsidered for the modification of its software elements and their relations to solve the predicted problems.

The software architectural model will be analysed and transformed, known as "Software Architecture Evolution."

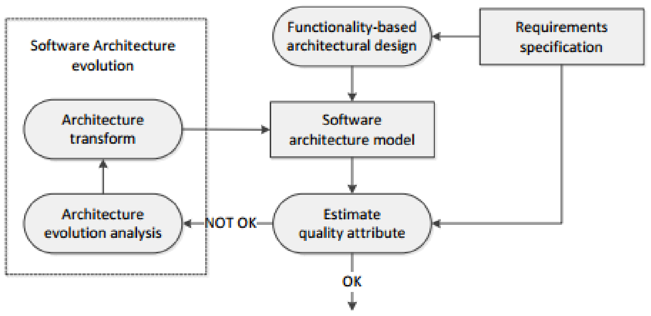
Software Architecture evolution
Architecture transform
Architecture evolution analysis
Functionality-based architectural design
Requirements specification
Software architecture model
Estimate quality attribute
NOT OK
OK

Figure 1. Software architecture design method [2]

In the recent days, the UML diagrams is the most popular representation of the design model. However, it is still difficult to manipulate and analyze the UML diagrams by machine. Therefore, it should be transformed into the meta design model using any meta-language. In our approach, we select xADL meta-language, one of the famous architectural description language in which we can provide our software architectural models in XML format. The XML format is standard and easier to be manipulated by machine.

The more software systems become large scale and highly complicated, the more people get involved in the software architectural design process. The need of automated tool is supportive. According to the related research works [3, 4, 5] of the software architecture evolution, the following problems or limitations may occur:

- Complex software architectural design is difficult to be considered by hand.
- Software architect or designer difficultly identifies the related requirements, quality attributes and the implementing classes that will be effected by the changes of the software architectural model.
- The effect detection of the software architecture changes by hand is often mistaken.
- No supporting tools are available to classify the differences between two versions of the design model.

In this paper, the related works are shown in section 2. The proposed detection of software architecture changes is described in section 3. In section 4, our supporting tool is demonstrated and the section 5 is the conclusion.

Manuscript received January 08, 2013; revised January 30, 2013.

Artit Udomsomruedee is with Department of Software Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok Thailand (email: artit.u@student.chula.ac.th).

Wiwat Vatanawood is with Department of Software Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok Thailand (email: wiwat@chula.ac.th).

II. RELATED WORK

A. *Architecture Description Language*

The architecture description language [5, 6] is designed to describe software elements and their relationships in formal manner. However, the xADL [6] is one of the architecture description languages which provide the extensible structure of the software architectural model using XML format. The software architectural model is defined as the following elements:

Component: an element which represents the interconnected components that construct software architecture. Each component is referred by a unique identifier and its description. A component communicates in and out of its boundary via the connection port called interface.

Interface: an element which represents the connection port of the interconnection between components. An interface is attached to a component. The directions of the interconnection are specified as input or output direction.

Connector: an element which represents the bridge of the interconnection between components.

Link: an element which represents the path between the interfaces.

Basic elements of xADL are shown in figure 2. The Agent and Environment components are connected via ActionCon connector. The interfaces are assigned and the links are drawn.

The xADL modeling tools are available to support and encourage the development of the software architectural model in xADL. These tools include Apigen [8], xArch [9], ArchStudio4 [10], and ArchStudio5 [11], which are architecture-centric development environment.

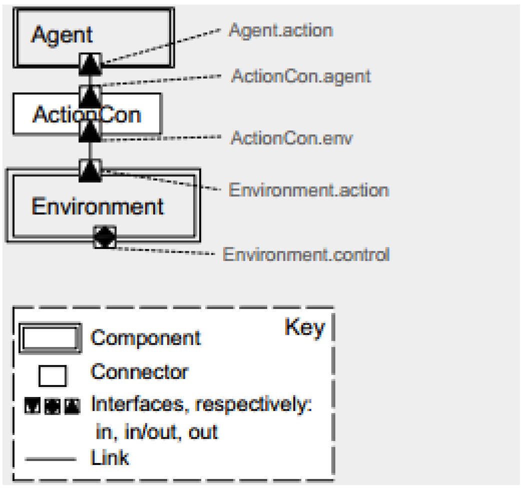

Figure 2. Example xADL, showing the Agent and Environment component, connected with the ActionCon connector [4]

B. *Rationale-Based architecture model*

In [12], the rationale-based architecture model for design traceability and reasoning is proposed to describe design rationale issues and theirs implementation along with the supporting tool. The main advantages of the design rationale are listed as follows for architects and designers:

- To understand the reason of the designed elements.
- To analyse the impact of changes in software architecture.
- To trace and analyse the root cause of the design.
- To encourage the check of the completeness of the design and maintenance.

III. PROPOSED APPROACH

In our approach, the software architectural model is attached with the related requirements description checklist and the design quality attributes. As we mentioned earlier, a software architectural model in xADL consists of a set of model elements - components, interfaces, connectors, and links. Each element is attached with the additional information to ease the traceability.

We propose three additional activities into the typical software architecture design method shown earlier in figure 1. These activities are "the project requirements description management", "the design description management", and "the software architecture change detection", shown in the shaded activities of figure 3.

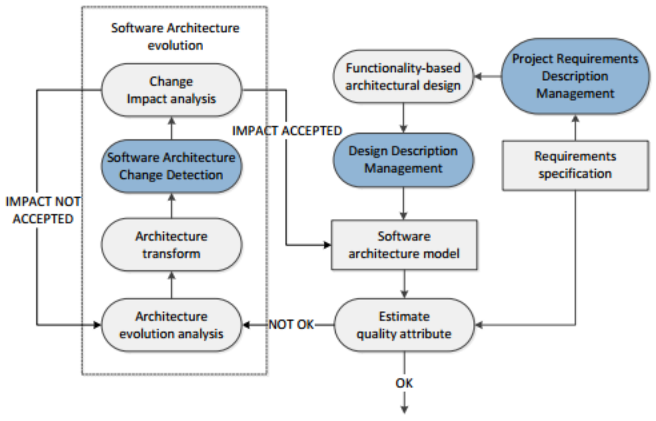
Software Architecture evolution
Change Impact analysis
Software Architecture Change Detection
Architecture transform
Architecture evolution analysis
IMPACT NOT ACCEPTED
IMPACT ACCEPTED
NOT OK
Functionality-based architectural design
Design Description Management
Software architecture model
Estimate quality attribute
OK
Project Requirements Description Management
Requirements specification

Figure 3. Software architecture design method based on design description.

The project requirements description management will provide the designer to key in the expected requirements checklist as a baseline to be traced. While, the design description management will provide the designer to key in the design rationale issues and quality attributes and they are then assigned to any element of software architectural model if needed. Whenever the design model is changed, the new design model will be detected against the previous design model and its design rationale. That is what our third tool called, the software architecture change detection will come to help the designer locate the changes and report the possibility of the ripple effects to the expected requirements and related design quality.

In order to clarify our activities mentioned, the following steps of the implementation are described along with the figure 4-6.

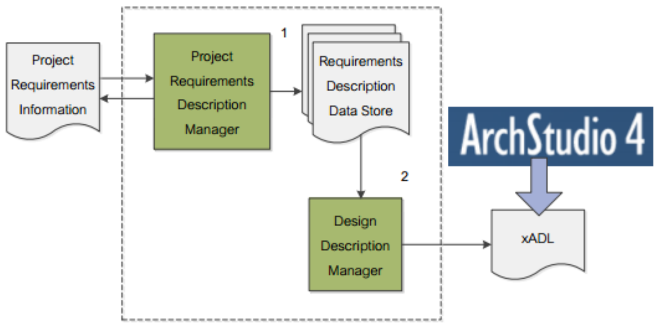
Project Requirements Information
Project Requirements Description Manager
1
Requirements Description Data Store
2
ArchStudio 4
Design Description Manager
xADL

Figure 4. Project requirements description manager and design description manager method.

A. Project Requirements Description Manager

The basic information regarding software project is essential for us in this step. The project name, description and the list of mandatory requirements are stored and expected to be traceable. In general, both functional and non-functional requirements checklists are recommended to be identified and stored in this step. A file of basic information, in our specific format, is allowed to be imported. Vice versa, the stored basic information is allowed to be exported as well.

B. Design Description Manager

In addition to the requirements checklist, the quality attributes regarding functionality, reliability, usability, efficiency, maintainability, and portability will be assigned to any software architectural model element in this step. The xADL architectural model will be read using DOM parser [13]. Each architectural element in term of component, or connector, or interface and link will be parsed and listed to be ready. Then, the designer will be provided with a GUI window to do the assignment easily. We intend to attach these requirements and design rationale items from the assignment, called "Design description tag", as an annotation or comment tag in the target xADL architectural model. That makes the xADL architectural design model still valid to the original xADL schema standard, as shown in figure 5.

```
...
        <implementation:implementation
xsi:type="lookupimplementation:LookupImplementation">
            <lookupimplementation:name
xsi:type="lookupimplementation:LookupName">com.arcad
e.interfaces.Interface</lookupimplementation:name>
         </implementation:implementation>
      </types:interfaceType>
   </types:archTypes>
<rationale:archRationale
xsi:type="rationale:ArchRationale"/>
</instance:xArch>
<!-- Arcade Game|componentfffffea-fada54ec-ec6aedd4-
6d180005,1,QA1|componentfffffea-fadac02b-fec6e352-
6d180029,3,QA1,QA2,QA3,QA4|componentfffffea-
fadb3253-b2a0d0d8-
6d180080,3,QA1,QA2,QA3,QA4|componentfffffea-
fadbddfb-c0198eec-6d1800f0,5,QA1,QA2,QA3,QA4 -->
```

Figure 5. Sample design description tag stored with xADL files.

The design description tag is defined as follow:

<! -- [ProjectName] | [[ComponentID], [RequirementsID]*, [QualityAttributesID]*]* -->

where [ProjectName] represents the project name, [ComponentID] represents the unique identifier of any component, [RequirementsID] represents the unique identifier of any requirements item, [QualityAttributesID] represents the unique identifier of any quality attribute. The symbol * represents the repeatable item, so that the [RequirementsID] and [QualityAttributesID] are repeatable for each [ComponentID].

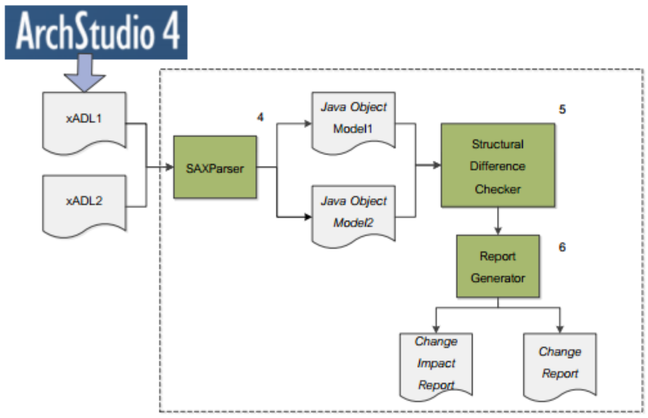

Figure 6. Structural difference checker and report generator method.

C. Sax Parser

During the evolution of the software architectural design, the previous architectural model is changed into a new one. We use a Sax parser [14] to read the xADL definition of both design models and generate the equivalent Java objects to both design models. Practically, the java object representing the specific design model will be efficiently handled in the programming language.

D. Structural difference checker

The structural difference between the previous design model and the corresponding new design model will be detected in this step. A hash table [15] is exploited in our detecting algorithm to locate the changes and categorize them into three categories, Added, Removed, and Modified Category. We are capable of identifying the difference between the renamed component and the added component.

E. Report Generator

The report is divided into two parts. The first part shows the effects of the software architecture changes in terms of the added, removed, modified elements - components, connectors, interfaces, and links. While, the second part shows the possibility of the impacts on the corresponding software requirements, quality attributes, and even the implementing classes of the functions.

IV. DEMONSTRATE

In order to demonstrate the resulting report of our "xSACEDT-Tools", we select an architectural model of an arcade game from [16] and revise into an on-line arcade game. The original arcade game's architectural model is drawn in figure 7, using ArchStudio editor. Then, the on-line arcade game's architectural model is redrawn in figure 8 as to be a new on-line system. Both design models are drawn and defined in xADL. The software architectural design method in figure 3, is conducted.

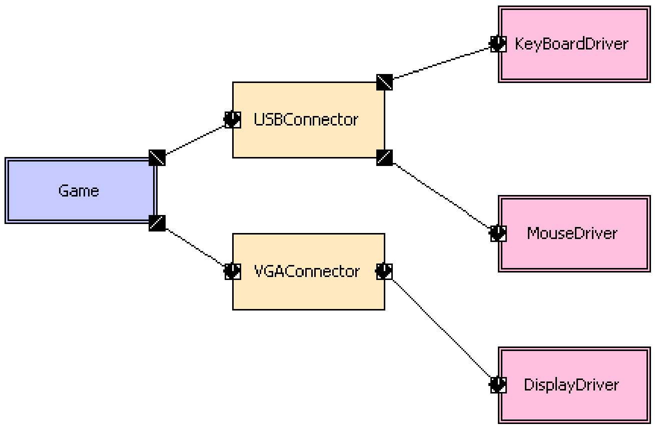

Figure 7. Arcade game architecture design.

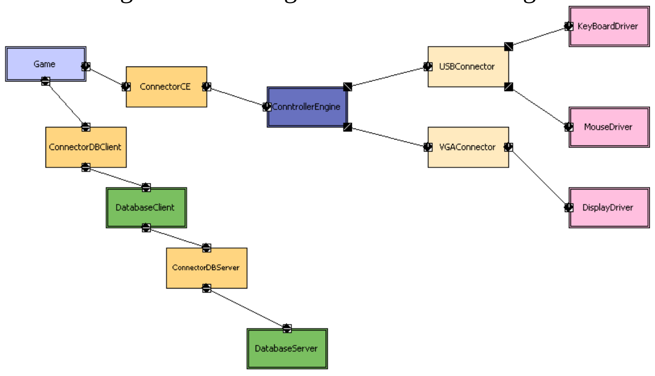

Figure 8. On-line game architecture design.

The software architecture changes are listed and shown in Table 1.

TABLE I
CHANGE RESULT DATA

No	Element Name	Type	Difference Classification
1.	ConntrollerEngine	Component	Add-in
2.	DatabaseClient	Component	Add-in
3.	DatabaseServer	Component	Add-in
4.	ConnectorCE	Connector	Add-in
5.	ConnectorDBClient	Connector	Add-in
6.	ConnectorDBServer	Connector	Add-in
7.	CCEInf1	Interface	Add-in
8.	CCEInf2	Interface	Add-in
9.	CDBCInf1	Interface	Add-in
10.	CDBCInf2	Interface	Add-in
11.	CDBSInf1	Interface	Add-in
12.	CDBSInf2	Interface	Add-in
13.	CEInf1	Interface	Add-in
14.	CEInf2	Interface	Add-in
15.	CEInf3	Interface	Add-in
16.	DBCInf1	Interface	Add-in
17.	DBCInf2	Interface	Add-in
18.	DBSInf1	Interface	Add-in
19.	Link1-1	Link	Add-in
20.	Link1-2	Link	Add-in
21.	Link1-3	Link	Add-in
22.	Link1-4	Link	Add-in
23.	Link1-5	Link	Add-in
24.	Link6	Link	Add-in
25.	Link7	Link	Add-in
26.	Link8	Link	Add-in
27.	USBInf1	Interface	Modify (Description)
28.	Link1	Link	Remove
29.	Link2	Link	Remove

It says that three additional components are found, named ConntrollerEngine, DatabaseClient, and DatabaseServer. Moreover, three additional connectors, twelve interfaces and eight links are found in the new model. Only one interface is modified and two links are removed.

TABLE II
CHANGE IMPACT RESULT DATA
(REQUIREMENTS EFFECT)

No	Requirement Name	Relate to components
1.	1 - Game engine development	Game
2.	3 - Connector Input Testing	KeyBoardDriver, MouseDriver
3.	5 - Display output testing	DisplayDriver
4.	6 - Online Game Implementation	ConntrollerEngine, DatabaseClient, DatabaseServer

In table 2, there are four requirements items will possibly be impacted according to the listed components. It says that the connector input testing will possibly impacted by the changes of KeyBoardDriver, and MouseDriver components.

TABLE III
CHANGE IMPACT RESULT DATA
(QUALITY ATTRIBUTES EFFECT)

No	Quality Attributes	Relate to components
1.	Efficiency	ConntrollerEngine, DatabaseClient, DatabaseServer, DisplayDriver, KeyBoardDriver, MouseDriver
2.	Functionality	ConntrollerEngine, DatabaseClient, DatabaseServer, DisplayDriver, Game, KeyBoardDriver, MouseDriver
3.	Maintainability	ConntrollerEngine, DatabaseClient, DatabaseServer
4.	Portability	ConntrollerEngine, DatabaseClient, DatabaseServer
5.	Reliability	DisplayDriver, KeyBoardDriver, MouseDriver
6.	Usability	DisplayDriver, KeyBoardDriver, MouseDriver, DatabaseServer

In table 3, the quality attributes listed as Efficiency, Functionality, Maintainability, Portability, Reliability, and Usability, are possibly impacted according to the listed components. For example, the usability of the software may be impacted by the changes of DisplayDriver, KeyBoardDriver, MouseDriver, and DatabaseServer components.

TABLE V
CHANGE IMPACT RESULT DATA
(IMPLEMENTATIONS EFFECT)

No	Class Name	Relate to components
1.	com.arcade.core.Engine	Game
2.	com.arcade.Input	KeyBoardDriver, MouseDriver
3.	com.arcade.Output	DisplayDriver

Finally, table 4 also shows the implementing classes which will be effected by the listed components.

V. Conclusion

In this paper, we propose an alternative software architecture design method for the large scale software product. The changes of the consecutive architectural design model will be detected and located using our proposed supporting tool, called xSACEDT tool. In our approach, the essential information on project description, the related requirements, and design quality attributes is attached into the architectural model as an annotation or comment tag in xADL. The resulting xADL is still valid and conform to the schema standard. We demonstrate the final report of the impacts and effects found after the detection. It is potentially useful during the evolution of the software architectural design model.

References

[1] SEI, What Is Software Architecture? *http://www.sei.cmu.edu/architecture/*.

[2] M. Mei Rong, Changlin Liu and Guangquan Zhang, “Modeling Aspect-oriented Software Architecture Based on ACME,” The 6th International Conference on Computer Science & Education (ICCSE 2011) August 3-5, 2011.

[3] D. Garlan and B. Schmerl, “Ævol: A tool for defining and planning architecture evolution,” in Proc. ICSE’09, Vancouver, BC, May 16–24, 2009.

[4] D. Perry and A. Wolf, “Foundations for the study of software architecture,” ACM SIGSOFT Software Eng. Notes, vol. 17, no. 4, pp. 40–42, 1992.

[5] B. Schmerl and D. Garlan. AcmeStudio: Supporting style-centered architecture development. Proc. ICSE’04, Edinburgh, Scotland, May 23–28, 2004, pp. 704–05.

[6] D. Garlan, R. Monroe, and D, wile. ACME: Architecture Description Interchange Language. In Proceedings of CASCON'97, November 1997

[7] Nelis Boucke, Alessandro Garcia and Tom Holvoet, “Composing architectural crosscutting structures in xADL,” Early Aspects 2007 Workshop, LNCS 4765, pp. 115–138, 2007.

[8] ISR, Institute for Software Reteach. Apigen tool set for the xadl language, *http://www.isr.uci.edu/projects/xarchuci/tools-apigen.html*.

[9] ISR, Institute for Software Reteach. xArch tool t set for the xadl language, *http://www.isr.uci.edu/architecture/xarch/*.

[10] ISR, Institute for Software Reteach. Archstudio 4.0 tool set for the xadl language, *http://www.isr.uci.edu/projects/archstudio-4/www/archstudio/*.

[11] ISR, Institute for Software Reteach. Archstudio 5.0 tool set for the xadl language, *http://www.isr.uci.edu/projects/archstudio/*.

[12] Antony Tang, Yan Jin and Jun Han, “A rationale-based architecture model for design traceability and reasoning,” The Journal of Systems and Software 80 (2007) 918–934.

[13] Document Object Model. *http://www.w3.org/DOM/*. January 19, 2005.

[14] Simple API for XML. *http://www.saxproject.org/*. April 27, 2004.

[15] Hash Table. *http://en.wikipedia.org/wiki/Hash_table*.

[16] Arcade Game Maker Pedagogical Product Line: Architecture Documentation, Volume 2 - Software Architecture Views, *http://www.sei.cmu.edu/productlines/ppl/software_architecture_views.html*.