5th SASTech
Iran |Mashhad
May 12 - 17 | 2011
5th Symposium on Advances in Science & Technology

Analysis and Evaluation of Software Architecture by Using Ontology

Sadrollah Abbasi [1], Reza Javidan [2], Mashaallah Abbasi Dezfouli[3]

[1] Islamic Azad university-Khoosestan Science and Technology Branch, Iran;
[2]Assisstant professor, Islamic Azad University–Beyza Branch, Iran
[3]Assisstant professor, Islamic Azad University–Khoosestan Science and Technology Branch, Iran

Abstract

Software architecture has a significant role in the lifecycle of any application software in which it must be appraised effectively. Quality characteristics, risks and non-risks, tactics or architectural decisions are elements that must be noticed during the software architecture assessment and analysis.

Software architecture assessment includes evaluating of architectural decisions attributes and combination possibility of these attributes to access the expected quality features. In the complex architectures usually by using of different architectural methods which are the results of best experiments, is trying to access to these attributes. In most of scenario-based architecture evaluation methods such as reconciliation-based architecture analysis, the way of doing the architecture decisions analysis and combination possibility to access to quality attributes, has not been noticed but only the steps and sequences of processes, input/outputs is emphasized.

In this paper, a new method for supporting the software architecture evaluation is proposed and has been tried to more clarify the circumstance of the analysis steps during the evaluation depend on the experiences and knowledge. In this method, reusing and sharing of architectural knowledge is enforced and therefore two types of anthologies have been offered that the attribute-based architectural styles have significant roles in construction of them. Finally, to show the effectiveness of the proposed method in different conditions, three sample surveys have been considered and analyzed.

Key words: software architecture, architecture evaluation, ontology, architectural decisions, architectural styles.

1. Introduction

The role of the software architecture in the life cycle of the software systems is undeniable; the documenting software architecture is prepared to serve different purposes. Therefore, such a documenting has to be so tangible that new people entering the project can understand it and so detailed that it can act like a road map and informative enough to make evaluation operation [1]. The documenting software architecture is both prescriptive and descriptive, meaning that it prescribes some rules and limitations for interested people's decisions and it talks about decisions made about system architecture for the interested people [2].

In the architectural product line phases, the conflict between requirements has to be determined and incomplete descriptions have been clarified from the point of view of different interested people. There are different models for the analysis and evaluation of software architecture which are different in terms of the explicit goals and quality attributes, techniques and activities and the amount of conflict between the interested people. The aim of the software evaluation architecture in a system is to predict the system quality for its production, to identify potential risks and to reconsider meeting the quality requirements in the design method.

As it was mentioned in [3], the software architecture is a set of different concepts which is performed according to the software system goals; so the quality attributes of risks and non-risks, architectural decisions or tactics and so on are elements which should be taken into consideration at the time of the analysis and evaluation of software architecture.

The accuracy of the results obtained from the scenario-based analysis of software architecture depends to a large extent on the quality of the applied scenarios in the software architecture evaluation since all of these methods are scenario based.

2. Terminologies

- **Software architecture**

  By increasing the complexity of software systems, the problem of architecture goes beyond the algorithms and data base. Here, the design and the specification of the general construction are put forward as a new problem.

  This is the software architecture in design. So, the complexity is one of the main concerns that software architecture has to be responsible for them [4]. It has to be disintegrated for the management of system complexity which this in itself brings about new difficulties. This also has to be considered in the software architecture. Some of the problems are as follow:

  - How are the disintegration operations performed?
  - Do we have all the essential components?
  - Are the components compatible with each other?

  Another concern in architecture is whether the solution is homogenous with the environment or not? This is not confined in interface or the relationship between external systems. Coordination and compatibility between them are very important.

The homogeneity between the advancement of profession and goal are also very important in software architecture which should be taken into consideration. [5, 6]

- **The Analysis and Evaluation of software architecture**

The sooner we find the problems in a software system the better we can remove them. The cost correcting an error which is incurred at the time of requirements or the early stages of designing is much less than the cost of correcting the same error at the time of system trial.

Software architecture is one of the first products of the architecture phase and its influence on the system and the project is very deep. An in proper architecture, it may lead to a disaster on the software project. Architectural evaluation is a low-cost method to prevent such disasters. Furthermore, it also determines the project structure architecture: configuration, timeliness and budgeting, useability, team structure, the pattern of documenting structure, maintenance and trial activities. If the architecture undergoes adaptation and changes in the middle of the operation due to the late discovered faults, the whole project ends up in a chaos. So it is better to evaluate the architecture in the first stages [7, 8].

- **Ontology**

Generally speaking, people use their own language for communicating and making models out of the universe. Natural languages are not appropriate to make models in computer science because they are ambiguous.

Therefore, official languages are used to demonstrate the universe models. A finite set of signs and a set of finite generating rules produce an infinite set of phrases and sentences which define the language. Despite this, producing a correct syntactic language dose not mean that one can understand the meaning of the sentences in a language.

Ontologies are the means to fill the semantic gaps between the syntactic display of the information and the concepts they convey, sharing knowledge and their reusability among the systems, which by increasing the use of different phrases, describing information will be more complex and difficult [5,6,9.10,11].

3. Software architecture analysis method

3.1: Scenario-based analysis method

Scenario-based analysis method was developed in 1993 to better understand the general concepts of architecture. It was a basis to demonstrate that a software system meets more than one requirement [13], [12].

This method is the first method for documenting architecture analysis and is widely used. Architecture designers have made lots of claim over their own designed architecture, without having compelling and systematic reasons for their own claims. As a result the scenario-based analysis method was developed to practically test the validity of their claims and to elucidate their claims about the quality attributes.

The scenario-based architecture analysis method was first developed to analyze the architecture to meet the quality attributes of adaptability, but this method not only meets the task setting requirements but also in practice it is used successfully to analyze many quality features such as adaptability , portability , expandability , integrability [14]. Architecture analysis method is performed as follows:

1 – Developing a scenario

2 – Describing the architecture (s)

3 – Classifying or prioritizing scenarios

4 – evaluating the scenarios indirectly

5 – identifying and recognizing scenarios interaction

6 – doing the general evaluation

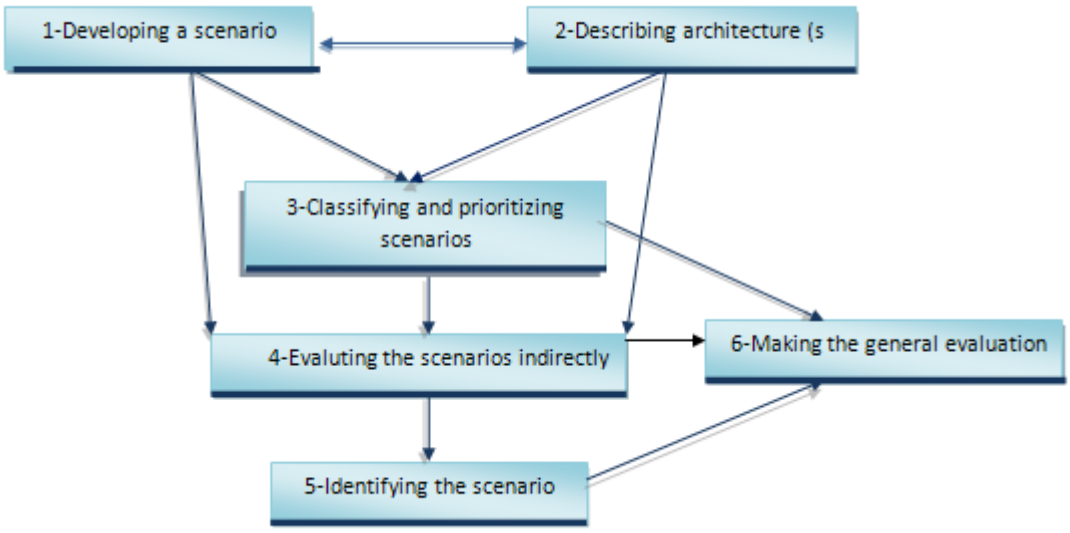

Figure 1: the activities and interdependencies between different stages in scenario – based architecture analysis

3.2: The architecture trade-off analysis method

The architecture trade-off analysis method [15] was developed in 1998 on the basis of scenario-based architecture model to evaluate the architecture for meeting the different quality attributes in architecture.

In this method as the name suggests, one can understand not only how much architecture can meet the particular quality attributes but also gain the insight on how the quality goals interact in the from of a trade off [14,2].

The architecture trade-off analysis method aims to provide a basic method to understand the capability of software architecture and to make the system access the ideal quality attributes which are sometimes in conflict with each other.

in the architecture trade-off analysis method before developing the system it becomes clear whether we need a trade-off between the different quality attributes or not.

The different steps in this method are as follow:

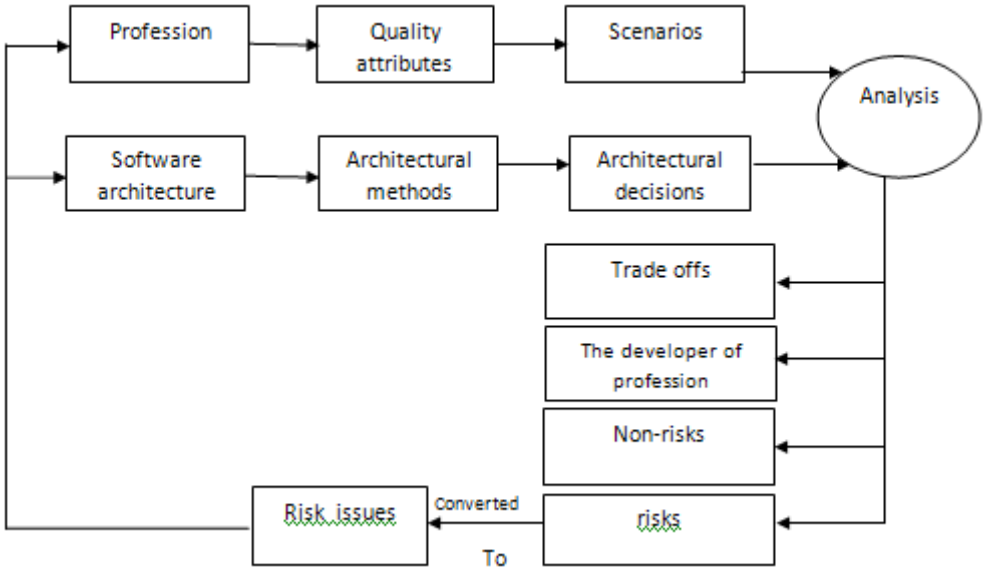

Figure 2: the process of the architecture trade-off analysis method

3.3: Active reconsideration method for middle design

Both scenario-based architecture evaluation method and the architecture trade-off analysis method are comprehensive methods to evaluate architecture. Architectures are usually developed gradually.

As a result, each architecture consists of several stages which should be followed to achieve a complete architecture. Active reconsideration method for middle design is a combination of these two methods: the first, scenario-based architecture analysis method or the architecture trade-off analysis method and the second: Active reconsideration method, this method responds to many questions on whether the primary designs performed are compatible with the final requirements or not. This is a low-cost and useful method.

3.4: Architectural-level modifiability Analysis method

Architectural-level modifiability analysis method was developed to evaluate architecture in meeting the quality attributes of adaptability. This method uses a structure like the one in the scenario-based architecture analysis method and the architecture trade-off analysis method: therefore, this method is also scenario-based. The difference between this method and the methods discussed previously is that the evaluation analysis is different in this method. This method defines an integrated process which has the following attributes:

- Focus on adaptability
- The distinction between different analysis goals
- The conversion of implicit presuppositions into explicit ones.
- Providing the repeatable techniques to perform the necessary steps

3.5: the scenario-based architecture analysis method for complex scenarios

In the scenario-based architecture analysis method for complex scenarios, it is assumed that the most important factor for the evaluation is the recognition of the risks. As the name suggests, this method is also scenario-based.

Architecture evaluation is to find the effect of scenarios on each other. The quality attribute which is emphasized in this method is flexibility. This method is used in the last version of the architecture and requires that the architecture be described in details. In this method, it is assumed that the system dose not work in isolation but rather considers all the dimensions of the environment in which it is placed. So the description of software architecture is divided into two parts: micro-architecture and macro-architecture.

The scenario-based evaluation method for complex scenario uses measuring tools to determine the effect of scenarios and to analyze them.

The defined tools include factors which influence the complexity of the scenarios. In this method three different factors are identified:

- Four levels of scenario effect (no effect, effect on one item, effect on several items, and effect on architecture)
- the number of owners involved in information system.
- four levels with regard to the presence of conflicts between versions (the likelihood of the presence of different versions with out fault , the presence of different version is not ideal but it is not avoidable either , it creates complexities which are not due to the management of configuration , it creates conflicts).

3.6: the comparison of different architecture evaluation methods and offering suggestion

The society of software engineering has provided various methods for the evaluation of software architectures with regard to ideal quality attributes such as maintainability, efficiency and … . However few works have been done on the comparison of these methods in a systematic way and to elucidate the similarities and differences between the methods. For example if we consider the quality attributes of the different architecture one gets the following points:

- The architecture trade-off analysis method, this method is not offered to evaluate the quality attributes but its history is related to its use in the quality attributes of safety , reliability , and efficiency.
- The scenario-based evaluation method, this method was originally developed for the quality attributes, variability and the operational capability.
- Active reconsideration method for middle design. This method is qualified enough in this field.

As another example, if different objects are compared in different methods, the followings are to be expected:

- In the architecture trade-off evaluation method, the architecture methods or styles, documents displaying the prospect of data flow, process, applications, physical or modular.
- In scenario-based evaluation method, architecture documents, especially logical or modular perspective.
- In active reconsideration method for middle design: the description of item interfaces.

As is evident from the comparison made by [18, 19] reusability of the available knowledge, and experiences, backing up the process, evaluation method and tools are not taken into account in most of the methods discussed. The researchers are trying to overcome these shortcomings by suggesting two ontologies and offering methods on how to use them in software architecture evaluation.

4. The recommended ontology

Interested people and their concerns, architectural decisions, architectural styles, and architectural attributes play important roles in the evaluation of software architecture and especially in the architecture trade-off analysis method since these factors indicate whether an architecture meets the quality attributes of high priority or not.

As a result these factors and their semantic relationships are focused on to create ontology.

To increase the amount of such ontologies, their semantic relationships should be constructed with high precision. The main classes in this ontology are as follow:

- interested people
- concerns
- architectural decision
- scenario
- architectural style
- architectural attribute
- quality attributes

This ontology is based on the work done by Akerman and Tyree to support the software architecture development [5] and also on their work on architecture decisions [20]. The suggested ontology can be used even under the circumstances in which there isn' a complete set of architectural documents available and in the early and final stages. Also by using this ontology, an architecture is developed to make decision for the continuation or discontinuation of evaluation.

This is done by using architectural styles and their semantic relationships with architecture attributes and architectural decisions.

An architectural style includes the kind of items and their interdependence, a description of data patterns, and interaction between items, along with a description of advantages and disadvantage [21].

Each style installs several architectural decisions each of which attends to different attributes [2]. Architectural decisions can be classified in different groups for demystification [20,22].

The accuracy of an architectural decision can be displayed by rationality [22]. Each architectural decision can have different substitutes. Each architectural decision can be in different states such as accepted or suspended. It entails some limitations. On the other hand, each architectural style can play a role in software architecture by the architectural attributes. This semantic relationship is evident in the suggested ontology in figure 3.

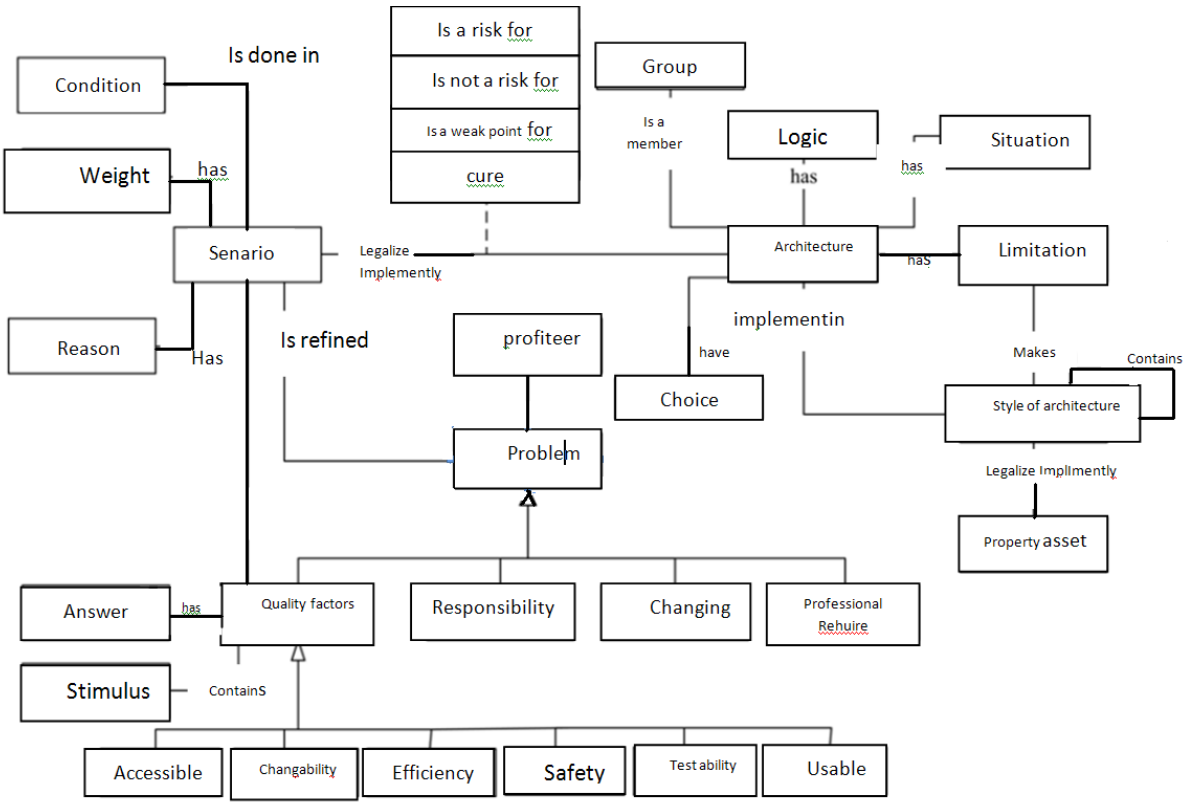

Figure 3: Ontology Of Suggestion Method

As it is clear in figure 3, each architectural decision supports some scenarios. Each scenario is weighed at the time of the evaluation for the trade-off. Each scenario covers a particular quality attribute with a specific rationale under certain conditions. Each quality attribute discussed in the project is one of the concerns of the interested people. The quality attribute can be divided into six main categories. Other interested people's concerns are task setting requirements, variability in requirements, and professional requirements.

In addition to the ontology discussed, another ontology is suggested as a support in making tradeoff in different stages of scenario-based architecture analysis method. This ontology is developed on the basis of the architectural styles discussed in [23]. This ontology is discussed in figure 4.

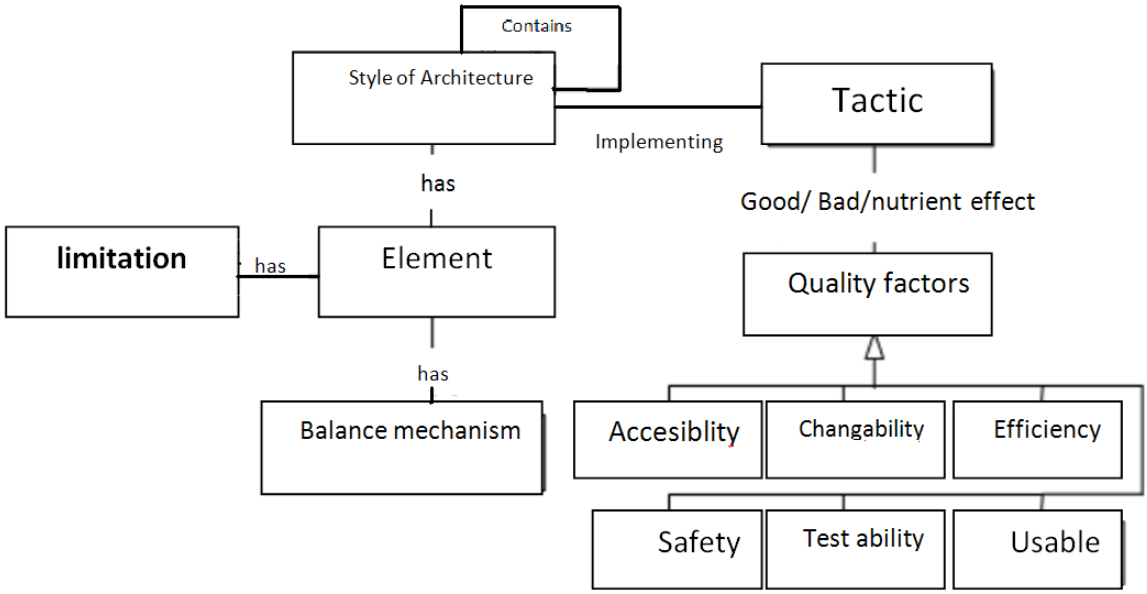

Figure 4: Ontology Of Architectural Styles Discussed

As one can see in figure 4, each architectural styles consists of different elements. For example, the pipe and filter style consists of different kinds of pipe and filter. None of the filters control the upper elements and lower elements (semantic constraints) and the data are transferred through the pipes (interaction) [2]. Each architectural style installs several tactics (architectural decisions). Each tactics can have several effects (good, bad, neutral) on one or more quality attributes.

To show the relationships between architectural styles, tactics and quality attributes, pattern-oriented software architecture can be used as an example. the pattern-oriented software architecture installs the information encrypted tactic which has a good effect on variability. To make the necessary evaluation one needs to model the attribute based architectural styles using architectural styles based on attributes and evaluation experiences, so that, it can be used for evaluation all the time. After some evaluation, the ontology of the attribute based architecture can be upgraded in terms of the added knowledge and even semantic relationship and also it can be updated. The main purpose of the ontology of the attribute –based architectural styles is to reuse the architectural knowledge and to share that knowledge. The suggested ontologies by Protezh ontology instrument can be installed. But before making the architectural ontology and the ontology of attribute-based architectural styles compatible and before creating a relationship between them, several principles (rules) should be considered:

- Principle 1: the ontology of architecture should consider the ontology of attribute-based architectural style.

- Principle 2: the members of the architectural style class should be chosen from the available members in the class of the ontology of attribute architectural styles.

Principle 3: the members of the architectural decision class should be chosen from the available members in the tactic class belonging to the ontology of attribute-based architectural styles.

Principle 4: the members of the class of quality attributes should be chosen from the available members in the class of the quality attributes belonging to the ontology of attribute-based architectural style.

Principle 5 : the semantic relationship between architectural styles and architectural decisions is deduced at the time of the selection of each of the members belonging to tactic class in the ontology of attribute-based architectural styles . This is done by using the first compatibility principle which is used in the descriptive logic language by using Protez instruments. This principle is as follows:

OABAS : tactic (?t) ^OABAS : Architectural style (? As)

^OABAS : implements (? as , ? t) ^ Architectural decision (? T) → Architectural style ? (? As) ^ OABAS : Implements (? As , ? t)

In which OABAS is the name of the space of the ontology of attribute-based architectural styles in which the architectural ontology is inserted.

Principle 6: in the architectural ontology there is another class beside the one which is shown in figure 4. This class consists of the members which are in conflict with the other available members in the architecture. The members of this class are put in this class with regard to the second compatibility rule.

The second compatibility rule:

OABAS: tactic (? T) OABAS : Architectural style (? as)

^ (? As)^ OABAS: implements (? as , t) ^ OABAS : quality Attribute (? qa) ^ OABAS : has bad effect on (? t , ? qa) ^ Architectural Decision (? t) ^ implements (? as , ? t) inconsistent set (? qa)

The suggested ontologies as a meta model for software architecture and attribute-based architecture styles are developed by applying the rules of the meta model in the meta model ontology (meta syntactic), the meta model ontology of the software architecture and meta model ontology of attribute-based architecture styles. By modeling the created ontologies and by applying the rules of the issues dominating the architecture and the attribute-based styles, the ideal ontology of software architecture and the ideal attribute-based architecture styles are obtained. Announcing the meta-syntax and the environment rule in ontology insures that the obtained ideal ontologies observe rules which is in itself a guarantee of adaptability and compatibility with the setting of the ontologies .

This is portrayed in figure 5.

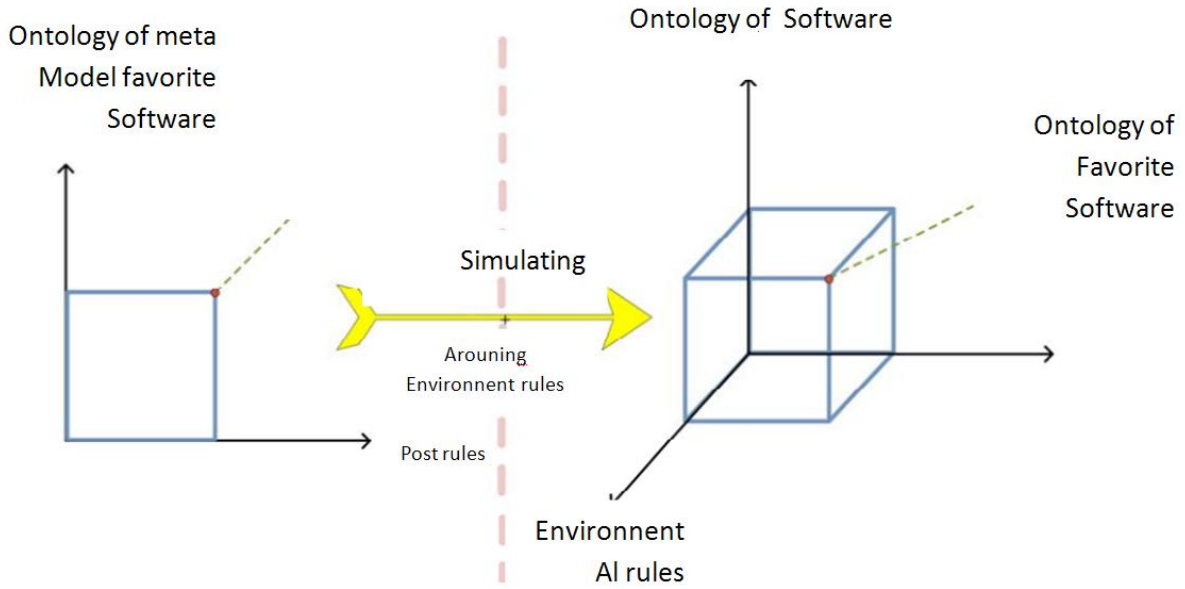

Figure 5: applying the meta-syntax in the meta-model ontology of the software architecture

The modeling shown in figure 5 about the architecture ontology is done during the software evaluation in the ontology of attribute- based architecture style the modeling is done at first by using attribute-based styles and best experiences and then with the passage of time and making more evaluation is developed by evaluation team.

5.Conclusion

With the suggested ontologies and the procedures discussed on using them, the researchers have attempted to make a framework to elucidate how to do the analysis on the basis of reusability of the experiences. This is done by involving the architectural styles and their semantic relationships with architectural attributes and architectural decisions. It also has made the analysis of different combination of decisions to achieve the quality attributes (they are very important in the evaluation of complex architectures) possible along with a capability for tracking. It is noteworthy that the reusability of the architectural evaluation knowledge even when the architect and the evaluation team are experienced and have the implicit knowledge about the evaluation is very important. Also the suggested ontology can be used when there is not a complete set of documenting architecture and in the primary and final stages.

6.References

[1].Clements P.,et al., Documenting Software Architectures, Addison-Wesley,2001.
[2]. Bass Len,Clements Paul, Kazman Risk, Spftware Architecture in Practice, Addison-Wesley,2003.
[3].Babu,T., L., Seetha Ramaiah,M., Prabhakar, T. V. , and Rambabu, D., " ArchVoc Towards an Ontology for Software Architecture" , In Processing of the Second Workshop on Sharing and Reusing Architectureal

Knowledge Architecture, Rationale, and Design intent, International Conferenceon Software Engineering. IEEE Computer Society, Washington, DC,5,2007.
[4].Garlan, D., Perry, D., E., Guest editorial to the IEEE Transaction on Software Engineering, April 1995.
[5].Akerman, A., and Tyree, J., "Using Ontology to Support Development of software architectures" IBM System Jornal,45,4,813-825,2006.
[6].Maedche, A., Ontology Learning for the Semantic Web,Springer,2002.
[7].Clements P.,Kazman R.,klein M.,Evaluating Software Architectures : Methods and Case studies, Addison-Wesley,2002.
[8].Mattsson, M., Grahn, H., Martensson, F., " Softwre Architecture Evaluation Methods for Performance, maintainability, Testability, and Portability". Proceedings of second International Conference on the Quality of software Architectures(QoSA 2006),Vasteras,Sweden,36,2006.
[9].Daconta, M., C.,Smith, K., T., and Obrst,L., J., The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management, John Wiley & Sons, Inc., New York,USA,2003.
[10].Coral.C,. Francisco, R., and Mario, P. Ontologies for software Engineering and Software Tecnology. Springer-Verlag,2006.
[11].Gasevic,D.,Djuric, D., Devedzic, V., and Selic, B.Model Driven Architecture and Ontology Development. Springer-Verlag New York,Inc.,2006.
[12].Kazman R., Abowd G., Bass L., Webb M., Analyzing the Properties os User Interface Software Architectures, Technical Report, CMU-CS-93-201, Carnegic Mellon University, School of Computer Science,1993.
[13].Kazman R., Abwod G., Bass L., Clements P., " Scenario-Based Analysis of Software Architecture" , IEEE Software, pp.47-55,Nov.1996.
[14].Shaw, M., and Garlan, D., software Architecture : Perspectives on Emerging Discipline, Upper Saddle River, NJ : Prentice Hall,1996.
[15].Kazman R., Klein M., Barbacci M., Longstaff T., Lipson H., Carriere J., The Architecture Tradeoff Analysis Method, Proceedings of IEEE,ICECCS,1998.
[16].Bengtsson P., Lassing N., Bosch J., "Architectural-Level Modifiability Analysis",Journal of Systems and Software, vol. 69, 129–147, 2004.
[17].Lassing N., Rijesenbrij D., Vliet H.V., "On Software Architecture Analysis ofFlexibility, Com[0.56]p[0.56]lexity of[0.56] Changes: Size isn't Everything", Proceedings of 2nd Nordic Software Architecture Workshop, 1999.
[18].Babar, M. A., Zhu, L., and Jeffery, R. , "A Framework for Classifying and Comparing Software Architecture Evaluation Methods" , In Proceedings of the 2004 Australian Software Engineering Conference(Aswec'04), IEEE Computer Society, Washington , DC,309,2004.
[19].Dobrica, L., and Niemel<E4>, E.,"A survey on software architecture analysis methods",IEEE Transactions on Software Engineering 28, 7, 638-653, Jul. 2002.
[20].Tyree, J. and Akerman, A., "Architecture Decisions: Demystifying Architecture"IEEE Software, 19-27, Mar. 2005.
[21].Zhu, H., Software Design Methodology - From Principles to Architectural Styles, Elsevier Science and Technology, Elsevier Science, 2005.
[22].Krutchen, P., "An Ontology of Architectural Design Decisions in Software Intensive systems," In 2nd Croningen Workshop on Software Variability, Rijksuniversiteit Groningen, pp. 54–61, 2004.
[23].Klein, M., H., Kazman, R., Bass, L., Carriere, J., Barbacci, M., Lipson, H., "Attribute-Based Architecture Styles", Software Architecture, Proceedings of the First Working IFIP Conference on Software Architecture (WICSA1), 225-244, 1999.
[24].Schmidt, Douglas, M. Stal, H. Rohnert and F. Buschmann, Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects. Wiley, 2000.