A METRIC-BASED APPROACH FOR MEASURING THE CONCEPTUAL INTEGRITY OF SOFTWARE ARCHITECTURES

Sayed Mehran Sharafi[1] and Zahra Sabet[2]

[1,2]Faculty of Computer Engineering, Islamic Azad University, Najafabad Branch, Esfahan, Iran

ABSTRACT

Software architectures evaluation has an important role in the life cycle of software systems. The conceptual integrity is one of the quality attributes which could be closely related to software architectural design. It is the underlying theme or vision that unifies all levels of the system's design. In this paper, a method for measuring the conceptual integrity of software architecture is provided. Conceptual integrity measurement is done in several steps by extracting a graph structure which its nodes are architectural concepts and its edges are relationship between them. The constructed graph is then weighted according to the type of relationship among the architectural concepts. Finally, a metric for evaluating the conceptual integrity from the refined graph is provided.

KEYWORDS

Software Architecture Evaluation, Conceptual Integrity

1. INTRODUCTION

Software architecture is a very important step in the software life cycle [1]. There are various definitions of software architecture [2]. It defines the structure or structures of the system, including software elements, the properties of the elements that are externally visible and the relationship among them [1].

Conceptual integrity is one of the quality attributes associated with system design or system architecture which defines the consistency across the system architecture. It means that in the all parts, from designing modules, coding them, naming of variables and components must be emerged consistently [3]. Brooks observes that the conceptual integrity is one of the most important considerations in system design. He argues: “It is better to have a system omit certain anomalous features and improvements, but to reflect one set of design ideas, than to have one that contains many good but independent and uncoordinated ideas”[4]. According to Booch's argument [5], the conceptual integrity of the system design, makes the systems which are more maintainable. From his viewpoint, the conceptual integrity of a system design originates from combining simple systems without unorganized complexity. Software system with conceptual

DOI : 10.5121/ijsea.2015.6109

integrity has software architecture, user interface and functionality that result in easy understanding, maintenance and use of the system.

Evaluation of quality attributes at the architectural level with the goal of re-designing architecture to provide the desired attribute can be very important. In the systems development process, the software architectural design is one of the early and major stages. Therefore, if the problems are addressed at this stage, they could not leak through next steps such as detailed design and coding. A wide variety of researches have been conducted in the field of quality attribute assessment from software architectures [7], [8], whereas, there are few efforts on the conceptual integrity evaluation.

In this paper, we use an ontology-based approach to extract the concepts or ideas from software architecture. In Section 2, a short overview of related works on conceptual integrity will be presented. In Section 3, an ontology to capture the architectural decisions will be recommended and expanded for measuring the conceptual integrity. The relationship among architectural decisions will be divided into several types. In Section 4, a weighted graph is introduces to specify architectural decision and communication among them. Finally a metric for measuring the conceptual integrity will be provided.

2. RESEARCHES IN THE FIELD OF CONCEPTUAL INTEGRITY

In [9], a research for measuring the conceptual integrity of application programs has been introduced. In this research, an ontological exploitation by communication with interface of application has been used for identifying the concepts of the application. Ontology of computing system is its theory in real world. It means that the concepts which form the computing system in the real world. For identifying the ontological concepts, the external surface of program has been extracted. For identifying the morphological elements, with external surface of application has been communicated. Then the morphological map has been drawn. The graph's nodes can be every recognizable element in system. The edges represent that how elements are available. Then, the relationship among concepts and conceptual network has been constructed. Conceptual coherence is the degree to which an application's concepts are tighly related. In this research, measuring the conceptual coherence is the first approximation for measuring the conceptual integrity.

In [10], conceptual integrity has been defined differently from previous research. Since the conceptual integrity including the existence of harmony in design concepts, this harmony must be considered between the UML diagrams that describes the system .System's diagrams should not include the conflicting concepts. The author of this study recalls another aspect of conceptual integrity in [11]. Requirement traceability is a method for tracing the existence of one requirement in design process. In [12], one of the ways to identify the semantic integrity is expressed as coordination between different views of behavioral and structural modeling.

3. ONTOLOGY EXTRACTION OF DESIGNED ARCHITECTURE

In this paper, at first we generalize the approach introduced in [9]. In our work, instead of an application program, the software architecture is considered. We extract the concepts from the

software architecture, whereas in [9], the concepts have been extracted from the application's interface.

Gruber [13] defined ontology as:"a formal specification of conceptualization". Through the conceptualization, it can be recognized the real world concepts and relationship among them. For measuring the conceptual integrity, we extract the concepts that form the software architecture. For identifying the concepts of software architecture, we extract the ontology of software architecture.

3.1 Research in the Field of Software Architecture Ontology

Many of researches proposed methods for creating the ontology in specific domain [14], [15]. Some of the researchers have combined the ontology with existing methods for software architecture or software development to find the better result [16], [17], [18], [19] and [20]. Many studies on ontology have used the reusability property of ontology ([16], [21] and [22]).

Reference [23], introduces methods for representing the architectural design, decisions properties and relationship among them. Some of the important elements of software architecture such as non-functional requirement, design patterns and tactics are not considered in the proposed ontology. In [16], ontology has been suggested for supporting the evaluation of software architecture with ATAM. In [17] an ontology-based method for software architecture documentation is proposed which is a design rationale with the goals of reusability and evaluation of software architecture.

4. STRUCTURE OF THE PROPOSED ONTOLOGY

The proposed ontology is a structure for documenting the design decisions; it is suitable for measuring the conceptual integrity of design decisions as well. . It is developed by the inspiration of some previous efforts such as: [16], [24], [14], [25], [23], and [26]. The concepts of design elements and relationship among them that emerges in the form of architectural decisions are important parts in the proposed ontology. Figure 1 represents this ontology. In this paper, for analyzing the conceptual integrity, only the specific part of the ontology related to architectural decisions, their relationship and architectural patterns are expanded. The decisions and their relationship can be represented by a graph.

According to [23] types of the relationships between the architectural decisions A and B include:

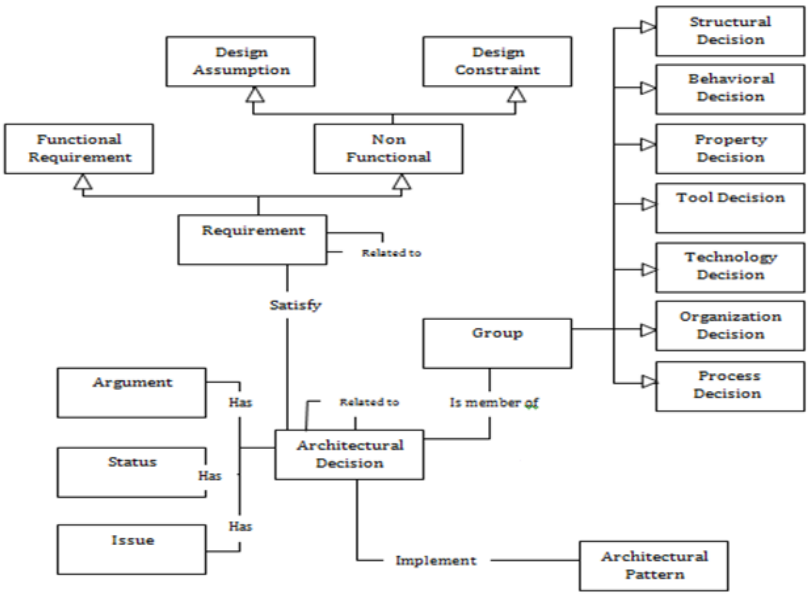


Fig 1.The proposed ontology of software architecture

- Constraints: if B for decision making needs A. with the loss of A, B is destroyed.
- Subsumes: B is a generalization for A. For example, all database tables must be created by SQL-server subsumes that table-X must be created by SQL-server.
- Is made of: A can be divided to B which is a more detailed decision.
- Is an alternative to: A and B have same issues and every of them can be used.
- Conflicts with: If A and B conflict together, this relationship is created. For example differ binding time is a tactic for modifiability. The reduction of number processed event is a tactic for performance. These two tactics conflict together.
- Depends: A and B relates to each other and the type of relationship between them did not either of mentioned relationship, this relationship is considered.

5. THE METHOD FOR CONSTRUCTING THE GRAPH OF ARCHITECTURAL DECISIONS

In the proposed ontology, architectural decisions are concepts. Thus, edges of the graph are connections among the architectural decisions or architectural concepts. After the documentation of architectural decisions based on proposed ontology, a graph is constructed according to the following steps.

1- For each architectural decision, a node is created in the graph. One edge is added in to the graph for each relationship between the decisions. The edge's label represents the type of relationship. In Figure 2, different types of transformations are shown

2-Since the selection of the appropriate architectural patterns is an architectural decision, for every selected pattern or style, a node is added in to the graph with an edge labeled "implements", connecting the node to the corresponding architectural decision.

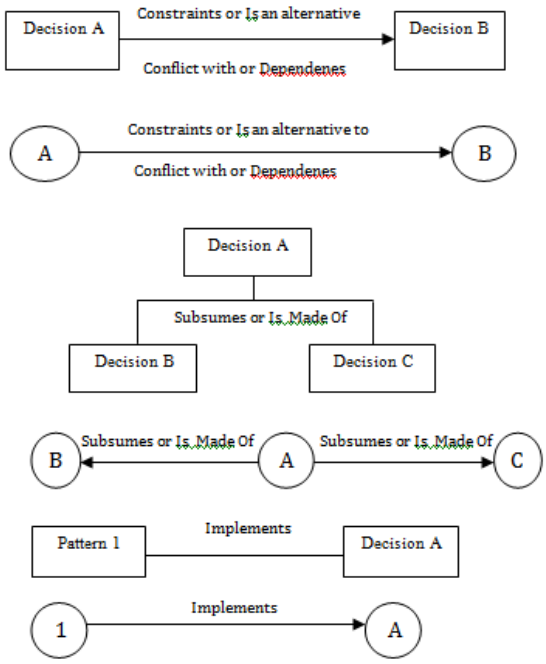


Fig2.Transformation the architectural concepts and their relationship to the graph

3- According to the type of the relationship, the weights are assigned to the edges. These weights are related to the effect of each of the relationships on the conceptual integrity. Table 1 shows the weights that are suggested for each type of the relationship. The highest weight is assigned to the "Implements" relationship connecting a pattern and an architectural decisions or architectural tactics. The architectural patterns have been used and evaluated several times and therefore we can be relatively sure that the architectural decisions could be satisfied by them. Furthermore, in [27] it is shown that the use of architectural patterns could enhance the conceptual integrity. Thus, we assign a high weight (actually the highest one) to this relationship. Assigning the other weights and the order considered is described as follow:

"Depends" relationship is a positive dependency between the architectural concepts which do not present limitation or incompatibility; Furthermore, regarding to its simplicity, low complexity and high understandability, the high level of conceptual integrity in this type of relationship, can be realized.

In the "Is made of" or composition relationship, one architectural concept is divided into smaller architectural concepts. If the main concept destroyed, the detailed concepts cannot be considered, because these smaller concepts without the main concept does not mean. But in "subsumes" relationship, one concept can be generalization of its subset concepts. With losing the general

concept, the inherited concepts can exist. For this reason the conceptual integrity of "Is made of" relationship is higher than the conceptual integrity of "Subsumes"relationship, thus we consider w5>w3. On the other hand, the composition relationship is less complicated than generalization relationship [28], furthermore, because the conceptual integrity and complexity are conflicting attributes, one can conclude that the weight of conceptual integrity of "Is made of" relationship must be higher than the conceptual integrity of "Subsumes"relationship. In comparison between the "Is made of "relationship and "Constraints" relationship, one can point out that the relationship between the subset concepts of composition relationship and its general concept is stronger than the relationship between the concepts of "constraints" relationship thus we consider w4<w5.

"Constraints" relationship exists between two concepts which the first one causes the second one to be existed and with losing the first concept, the second concept also will be loosed, thus it could be concluded that w4>w3.

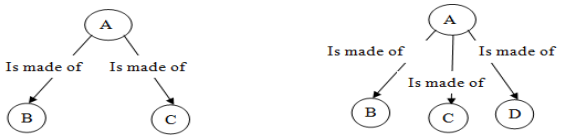
Conceptual integrity of The "Is an alternative to" relationship is a weak relationship in terms of conceptual integrity .But, this type of relationship has no negative effect on the connected concepts and its conceptual integrity is greater than "Conflicts with" relationship. From the above discussion one can conclude: 0<w1<w2<w3<w4<w5<w6<w7.

TABLE1

Different Types of Relationships and Their Weight

Weight	Type
W1<0	Conflicts with
W2	Is an alternative to
W3	Subsumes
W4	Constraints
W5	Is made of
W6	Dependenes
W7	Implements

4- In this step, some of the weights are refined. In figure 3 two graphs are seen. In these graphs, the "Is made of" relationship has been established between the main node and its subsets. The question is that in the same relationship such as "Is made of", whether the number of subsets of the main concept affects on the conceptual integrity or not? To answer this question, we perform the following calculations on each graph:



(a) (b)

(b)

Fig3.Two graphs with "Is made of" relashionship

Betweenness Centrality(BC): This parameter measures the number of shortest path between the all pairs of nodes in the graph that contain a particular node. In [29], this parameter has been used for identifying the central concepts in graphs that have no weight or have the edges with equal weights. By calculating this parameter for concept A in two graphs in Figure 3, we obtained the same value for BC. It means that the value of centrality in "Is made of" is not related to the number of subset nodes.

Conceptual coherence: This parameter measures the degree to which an onology's concepts are tightly related [9]; it can be considered as a metric for estimating the conceptual integrity as well. In the following formula "geodesic" is the shortest path between a pair of nodes, "dist" is the shortest path distance between a pair of nodes and N is the number of graph nodes.

$$CCM = \left(\frac{\sum_{i,j} dist(geodesic(i,j))}{N}\right)^{-1} * 100 \quad (1)$$

Conceptual Coherence Metric (CCM) of the graph (a) in Figure 3 equals to 75% and CCM of the graph (b) equals to 44.4% .Thus the conceptual coherence of the graph (a) in Figure 2 is greater than graph (b).

Structural complexity: This parameter measures the structural complexity of one concept. In [28], this parameter has been used for measuring the structural complexity of the classes. Since class could be generalized to concept [30], we can use this parameter to calculate the structural complexity of node A in two graphs. In the following formula, W is the weight of "Is made of" relationship and n is the number of destinations from A.

$$\text{Complexity of A} = W*\left(2- \frac{1}{n}\right) \quad (2)$$

The structural complexity of concept A in the first graph of Figure 3 equals to 1.5*w and it equals to 1.6*w in second graph. Thus, the structural complexity of A in the first graph in Figure 3 is less than the structural complexity of A in the second graph.

The value of the CCM and Complexity parameters in two graphs can give this result that in the graph with "Is made of" relationship, if the number of main node's subset are increased, then the conceptual coherence of graph is decreased and the structural complexity of the graph would be increased. Thus, the conceptual integrity which have positive relation with conceptual coherence and negative relation with structural complexity, is decreased. Thus, the assigned weight to the "Is made of" relationship in Table 1 must be multiplied in a factor. This factor should be such that with adding the number of subsets of main concept or number of destinations from main concept, the weight is reduced. This factor can be $(1+\frac{1}{n})$ and the weight of "Is made of" relationship is changed to $W5*(1+\frac{1}{n})$.

The related weight to "subsumes" relationship with the same reason that mentioned, must be changed to $W3*(1+\frac{1}{n})$ where n is the number of destinations from node A.

5-In this step, a weight is calculated for each node. In this way, the weight of the external edges from the node is summed. If the node had some external edges with type "Is made if" or "subsumes" , only one of the weights are considered. In formula (3), $W(n_i)$ is the weight of $node_i$ and $w(e_{i,j})$ is the weight of the edge from node i to node j.

$$W(n_i)=\sum_j \text{distinct}(w(e_{i,j})) \qquad (3)$$

6-Finally, sum of the weight of nodes is calculated and this value is divided by the number of nodes. The obtained value is a metric for measuring the conceptual integrity of software architectural design. In formula (4), G(D) is the graph of the architectural decisions.

$$\text{Conceptual Integrity}=\frac{\sum_{ni\in G(D)} W(n_i)}{N} \qquad (4)$$

This formula is a metric for calculating the conceptual integrity in the software architectural design.

Formula (4) is better than the introduced formula in [9] for measuring the conceptual integrity. Because, our concepts' graph is a weighted graph and these weights make the calculated conceptual integrity more accurate by considering different types of relationships among architectural concepts. In [9] the graph has no weight and different type of concepts have the same effect on the conceptual integrity.

6. CONCLUSION

The evaluation of conceptual integrity is very hard [26], because there is no explicit definition for it .The first definition of conceptual integrity is seem simple at first. But this definition included everything and was very complicated. For this reason, researches in this field are limited.

Brook's definition from conceptual integrity gives an idea to measure the conceptual integrity. This idea is excavation the concepts from software architecture in the first step. This has been identified in computer literature as ontology. Thus we found an ontology for software architecture. This ontology is suitable structure for documenting of software architecture. The different relationships among architectural decisions are defined. Each of the relationships has an effect on conceptual integrity .The weight is assigned to the relationship. Finally the graph of architectural decisions with weighted edges is constructed and a metric for measuring the conceptual integrity is introduced. This metric can compare the conceptual integrity in different architecture.

References

[1] B. Len, C. Paul, K. Rich , Software Architecture in practice(A. Wesley,2003).
[2] IEEE 1471, IEEE Recommended Practice For Architectural Description Of Software-Intensive systems-Ansi/IEEE-std-1471, 2000.
[3] HTTP://www.msdn.com visited at 1/15/2011.
[4] B. Fredrick,The Mythical Man-Month (A.Wesley ,1995).

[5] G.Booch ,The Defenestration Of Superfluous Architectural Accountrements,IEEE SOFTWARE,pp.7-8,2009.
[6] B.Cox,"Communicating Conceptual Integrity in Distributed Systems Through Intelligent Assistance" ,IEEE,2000
[7] L.Jun,T.Jiang X.Yuan , An Approach to Performance Evaluation Of Software Architecture ,First International Workshop on Education Technology and Computer Science(Page: 853 Year of Publication: 2009 ISBN:978-0-7695.3557-9/09).
[8] S.A.Tona,A.Ashkan, L.Tahvildari, Evaluting Architectural Stability Using a Metric-Based Approach,Proceedings of the conference on Software Maintenance and Reengineering(CSMR'06) (Year of Publication: 2006 ISBN:0-7695-2536-9/06)
[9] I.Hsi ,Analysing The Conceptual Integrity Of Computing Applications Through Ontological Excavation And Analysis ,Ph.D. Thesis, Dept. College of Computing, Georgia Institue of Technology, Georgia, 2005.
[10] Egyed Alexander ,A system For Defining And Analyzing The Conceptual Integrity Of UMLModels,Software Engineering Notes, Vol. 25, no. 1,pp.108,2000
[11] A.Egyed,P.Grunbacher,Automating Reqirements Traceability: Beyond the Record & Replay Paradigm ,Proceedings of the 17th IEEE International Conference on Automated Software Engineering(ASE) , (Page: 163-174 Year of Publication: 2002)
[12] M.Snoeck,G.Dedene,Existence Dependency: The Key to Semantic Integrity Between Structural and Behavioral Aspects of Object Types,IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Vol. 24, no. 4,pp.233-251, 1998.
[13] T.R.Gruber, Toward Principles for the Design of Ontologies Used for Knowledge Sharing,Technical Report,ksl 93-04,Knowledge System Labroratory Standford University,Palo Alto,CA,USA,1993.
[14] L.Babu,M.S.Ramaiah,T.Prabhakar, ArchVoc-Toward an Ontology for Software Architecture,in Second Workshop on SHAring and Reusing architectural Knowledge Architecture ,Rationale,and Design Intent (SHARK-ADI 07), (Year of Publication: 2007 ISBN:0-7695-2951-8/07).
[15] S.H.Hsieh,H.T.Lin,N.W.Chi,K.W.Chou,K.Y.Lin, Enabling the development of base domain ontology through extraction of knowledge from engineering domain handbooks,Advanced Enginnering Informatics 25,pp.228-296,2011.
[16] A.Erfanian,F.S.Aliee,An Ontology-Driven Software Architecture Evaluation Method , SHARK'08 ACM,pp.79-86, 2008
[17] C.Lopez,V.Codocedo,H.Astudillo,L.M.Cysneiros, Bridging the gap between software architecture rationale formalism and actual architecture documents:An ontology-driven approach,Science of Computer Programing 77,pp.66-80,2012.
[18] C.Chun,Z.Jianxun,WU.Wenna, Acquirement of Class Relations for Ontology Based on Language Expression and Knowledge Structure,pdf file,Institue of Scientific and Technical Information of China,Beijing,2008.
[19] C.M.Bogdan, Concern-oriented and Ontology-Based Design Approach of Software Architectures,in 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, (Page: 249-252 Year of Publication: 2008 ISBN:978-0-7695.3523-4/08).
[20] K.jaroslaw, Passing from Requirements Specification To Class Model Using Application Domain Ontology,Proceedings of the 2nd International Conference on Information Technology , (Page: 28-30 Year of Publication: 2010).
[21] C.Lopez,L.M.Cysneir,H.Astudillo, NDR Ontology:Sharing and Reusing NFR and Design Rationale Knowledge ,First International Workshop on Managing Requirements Knowledge(Mark 08), (Year of Publication: 2008 ISBN:978-0-7695.3627-9/08).
[22] J.Lee,H.Chae,C.H.Kim,K.Kim, Design of product ontology architecture for collaborative enterprises, Expert Systems with Applications 36 ,pp.2300-2309,2009.
[23] P.Kruchten, An Ontology of Architectural design decisions in software intensive systems,Second Groningen Workshop on Software Variability, (Page: 54-61 Year of Publication: 2004).

[24] J.Tyree,A.Akerman, Architectural Decisions :Demystifying Architecture, IEEE SOFTWARE,pp.19-27,2005.
[25] L.Chung,B.A.Nixon,E.Yu,J.Mylopoulos, Non-functional Requirements in Software Enginneering ,Springer,1999.
[26] R.R.A.Issa and I.Mutis, Ontology Based Framework Using a Semantic Web for Addressing Semantic Recognition in Construction , Lect ,Notes Artif,Int,LNAI 4200,pp.348-367,2006
[27] S. Douglas, S. Micael , R. Hans, B. Frank , Pattern Oriented Software Architecture,(J.Weley,Sons,1996).
[28] D.Kang,B.Xu,J.Lu,W.C.Chu, A Complexity Measure for Ontology Based on UML ,Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems(FTDCS'S 04), (Page: 853 Year of Publication: 2004 ISBN:0-7695-2118-5/04).
[29] I.Hsi,C.Potts,M.Moore, Ontological Excavation :Unearthing the core concept of the application ,Proceedings of WCRE, (Page: 345-352 Year of Publication: 2003).
[30] S.A.Sloman,B.C.Love,W.K.AHN,Feature Centrality and Conceptual Coherence,COGNITIVE SCIENCE, vol.22(2),pp.189-228,1998