Software Architecture modeling framework using UML

R. Aroul canessane
Research Scholoar, Sathyabama University
aroul_308@yahoo.co.in

Dr. S. Srinivasan
Professor & Head, Dept. Of Computer Science and Engineering,
Anna University , Madurai
sriniss@yahoo.com

Abstract

The software architecture's are built using some specific languages while developing a project. Architecture design languages are used in research and industrial projects that are used represented using Unified Modeling Languages. However UML is not a completely qualified Architecture design language. But it has the ability to use as a substitute for all other languages that has been used earlier. UML can be used for creating Architectural viewpoint, this paper shows how UML can be extended for creating Software Architectures with architectural abstractions. The example that is used here for extended framework of UML is the CTScan device.

***Keywords*: Architectural view; Enterprise Architecture; Streaming; abstraction; ADL..**

1. Introduction

The IT organization to run their business in the society requires long running and large software intensive systems. To support those business oriented systems which needs an evolution, the tools must be represented for building effective software architecture. Software architecture provides an ability to easily understand, communicate and states the need of software intensive system from different views of perception.

The current notations that are used for modeling the software architectures are based on architectural description language; they are informal and are adhoc. The ADL's, they take only a single perception view, from which the architect has to identify all its key aspects which are used in software systems.

The UML uses some advanced techinques and notations which supports from requirement phase to implementation phase. Views of Multiple perception can be created using UML. But UML doesn't provide all the constructs which are related to software architecture modeling, as configurations, styles and connectors etc. We don't have any notations for connectors in UML, it has to separate the concerns regarding communication. The UML component diagram describes the software system as components and interconnections at the level of specification.

Extending the UML with some ADL's has been already done [7]. In this paper we propose an approach for extending UML with its mechanism which can improvise the description of software architecture, though a detailed research is going under object management group.

This paper is organized as section 2 Research in Software Architecture, section 3 UML as Architectural definition Language with an example, section 4 Modeling Framework for Software Architecture and section 5 conclusion.

2. Research in Software Architecture

So far no standard definition is there for software architecture, no such a single framework has been developed for this architectural concept. There is a lack of standards for defining a common standard for the software architectures. An architectural definition language is required for modeling a detailed description about the components interconnection [7]. Architectural definition language has improved the analysis and language standards [6,3]. A single ADL does not give a complete multiple views for checking the completeness of the Software Architecture.

Some of the researcher's use UML as the industrial trend for Architectural design [1]. UML does not have enough notations to make a complete model, such as abstract notations are not present for components, configurations, connectors and architectural styles.

3. UML as ADL

In order to define an UML as an Architectural definition language, we have to identify all the key concepts which are required for the description of the software architecture and know how those concepts have a relation. We have explained using CTScan as an example in this paper.

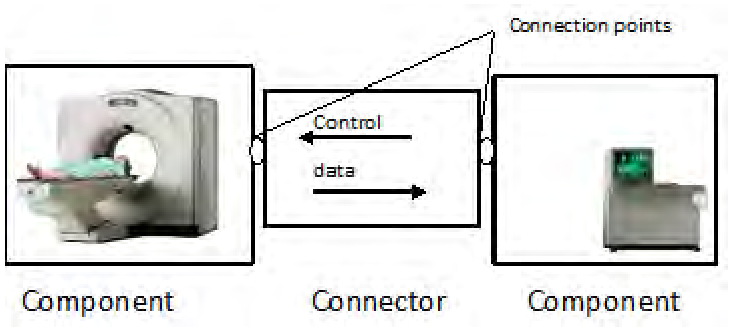

Fig.1. Architectural representation of CTScan system

The fig.1 shows the CTScan device the information receivers are shown as the components and a connector is used between these components. The fig.1. Gives a realization view of the CTScan device. The CTScan device component encapsulates a set of medical images of a patient whereas the control station controls the device remotely by streaming the images. The connector acts as a communication platform, it consists the way of interaction between the connection points.

The connections between the components are deployed using the connectors, whereas the CTScan device is the hardware port and the control component is the software port. The architectural styles that are used in the CTScan are using the client server styles and the pipe’s and filters styles. The elements that have been used for making the software architectures provided by the current ADL’s but there is a drawback for the above architectural style for CTScan, the models cannot integrate properly. The UML supports all sought of integration compared to other artifacts which are under research. The ADL can be extended with the help of UML which can give a different architect view point [5]. The different artifact terms are explained in the forth coming sections.

4. Modeling Framework for Software Architecture

This section defines some extension to the UML notations to define the software architecture. Two mechanisms have been already proposed by the OMG [8] that are heavyweight and lightweight extension mechanisms. A lightweight mechanism adapts the UML notations and the semantics without modifying the UML metamodel with the built-in extension mechanisms, whereas heavyweight mechanism allows using the UML semantics with the help of the extending standard metamodel.

In the forth coming section we have tried to implement an ADL with the help of the above two mechanism. The following sections are classified as first view points of an architecture implemented using heavy weight mechanism which uses the UML metamodel concepts. Second lightweight mechanism for components, connectors, patterns[2] and software architecture configuration

4.1. Architectural viewpoint

Software architectures based on the ADL’s are focused on the structural constraints at system level [8 ,7] focusing only on the structural constraint does not cover all the constraints of software architecture, we cannot cover all the concerns of different stakeholders view. The different views of the stakeholders may affect the quality of architecture such as scalability, security, performance, persistence and reliability.

Software architectures must hold the above properties or any required combination of the properties which are required for the stakeholder. To achieve the properties one can define architectural constraints with the perception of different views. The different viewpoints are helpful in grouping the different stakeholders. The notations of the architectural views and the viewpoints have been referred [4] and the ISO reference Model. Both the models have a limitation i.e. Limited views and viewpoints which doesn’t allow you to create a new one. It is not sufficient to cover all the aspects of software architectures. A view point is a pattern [IEEE P1471] which has been developed using individual audience views would be helpful for architecture creation and analysis.

The above definition is interpreted by us and we provide a UML based conceptual framework for describing the architecture. This interpreted model does not have any limit for creating the views and the viewpoints. As shown in the fig2. A view point is defined as a model created using the single perception representation of software architecture, which encompasses of multiple architectural views. The rules that are determined is based on the Stakeholders specific viewpoint and are represented using notations of architectural elements, these

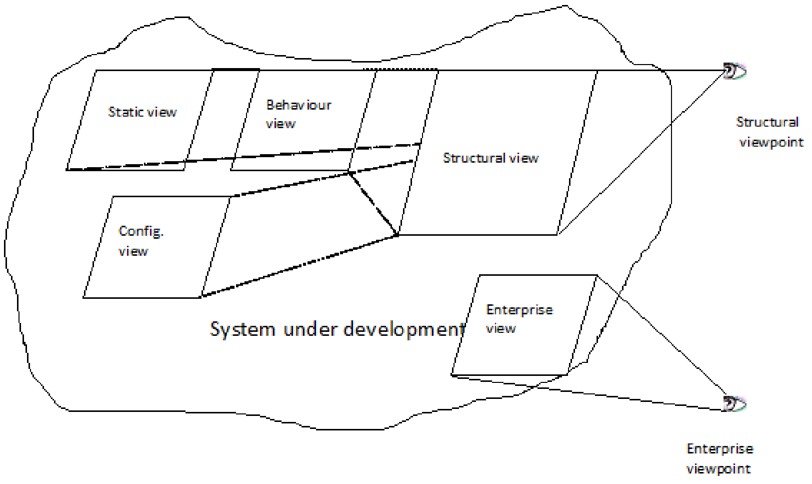

Fig. 2. Viewpoints and View representation

notations are used for creating the language elements with respect to the architect perception. The structural view point has multiple view and focus on specific concern. Fig.2 shows two different viewpoints 1. Structural viewpoint and 2. Enterprise view point, both defines the main view which illustrates the basic focus of the architect view point. The main view point must be projected into different views which elaborate distinct aspects in modeling the software architecture. The structural view point is further projected to static, behavioral and configuration view detailed in section 4.2.

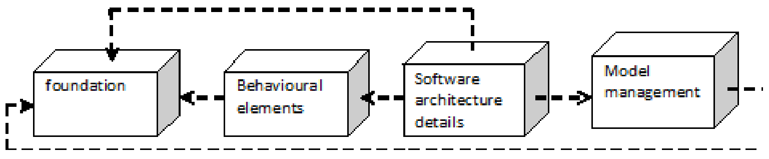

Fig. 3. UML metamodel and Software architecture dependency

The heavy weight extensions have been used to create an architectural framework. The fig3. Shows the relation between the architecture details and the UML. We have used only the UML package for creating the framework. The model that has been created is an extension of IEEE P1471 which has been used by main view. This architectural framework has been created based on the multiple views in software architecture. In this framework we lay on the view and viewpoints. The different viewpoints that we have come across are structural viewpoint, ODP viewpoint, quality management view point [1] etc.

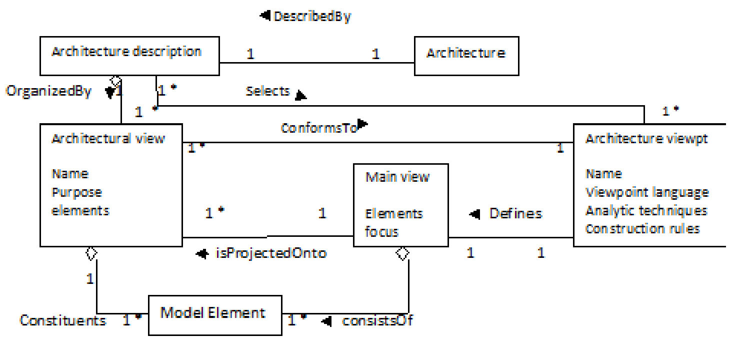

Fig. 4. Extended conceptual model

The fig5. Shows the structural viewpoint of CTScan device. The structural view in the fig 5. Are identified in terms of components, connectors and also the conditions for deployment. The different structural views are created for CTScan device and CTcontrol station which are connected using CTconnector. The CTconnector is complex and it has been described in the section 4.2. The architectural views that we have considered are static, dynamic and configuration view.

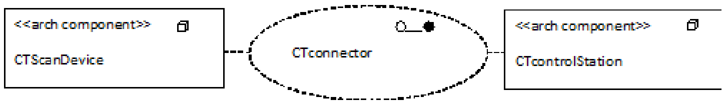

Fig. 5. Structural view of CTScan system

4.2. Static view

The structural view of the software architecture is projected as a static view by using the modeling elements in terms of component and connector types. In static view the components are encapsulated as combination of data and the corresponding computations are represented by the computational component <<computational>>. The computational components static structure is represented as a component type. It is defined as the combination of interface element and interface component.

The notation capsule [8] is used for operational, stream and signals which as interface elements. The computational component defines some added class compartments which are labeled by the keywords streams, signals and operation. They are used for the interface element declaration, computational component. These interface elements which are declared are not equivalent to interfaces of UML.

To represent the computational components we have used some icons for a better understanding. The computational components static structure represented using the ctcapturedevice. The interfaces for the component are start, pane, zoom and stop that allow making the control from the ctcontrol. The label ctcontrol on association end represents the classifier role with an instance of ctcontrol with the configuration view is explained in the session 4.3. A *connector type* acts as an interface between the components. This connector is referred to the collaboration notation in UML. The collaboration is the abstraction of classifier and association role. [5]. A stereotype is used for the <<*connector*>> which is implements a connection role and points of connection. The advantage of this approach is, it is easy to design a single connector type and also a complex type. A simple connector type has two connection points and one connection role. The connection role is similar to the pipes as in pipes and filter style.

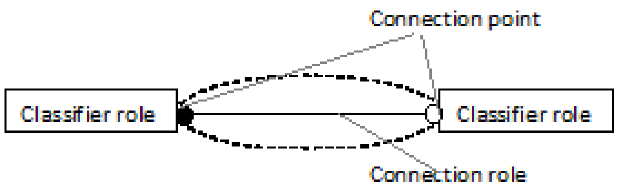

Fig. 6. Static structure of a simple connector

We have shown two connection points shown in the fig.6. For simple connection. The complex type is a combination of simpler connector as shown in the fig.7. ctconnector has the collection of simple connectors *imagestream*, streamcontroller, virtualdevice, streamendpointsignal and streamendpoint. The streamcontroller

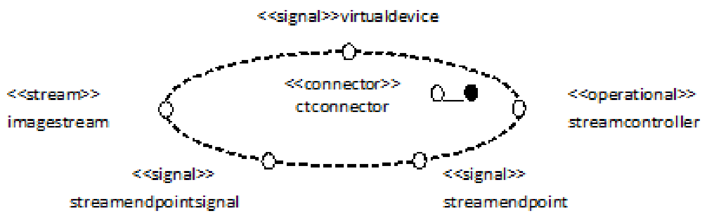

Fig. 7. Static structure of ctconnector

has the information about the starting and the ending of the signal, the virtualdevice alters the signals with respect to the with respect to the interaction components, streamendpoint describes the management of the virtual device. These signal connectors are elaborated shown in the fig.8.

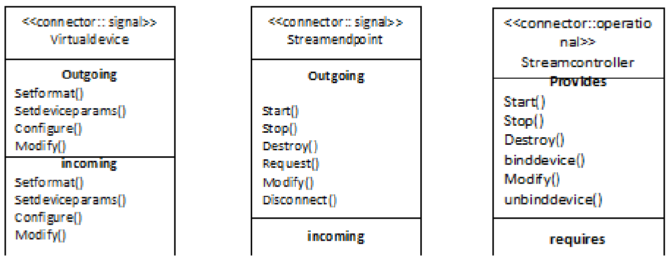

Fig. 8. Eloboration of simple connector to ctconnector

4.3. Behavioral view

The dynamic properties are shown by the behavioral view. The constraints of the elements can be described by their behavior. The behavior of the computational component is defined by using the interface protocol which shows the data flow and the different events that occurs when a component is engaged. A state machine has been show in for the component interface. The behavior is also defined using the state machine using the UML state protocol.

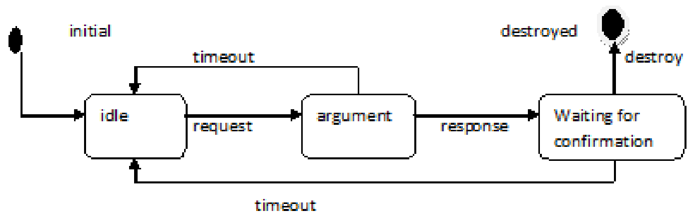

Fig. 9. Behavioural an d static view of streamendpoint signal

In fig.9 we have represented the streamendpoint signal and the sequence of events occurred. The state machine we have shown the initialization of the idle state in turn requests the *argument* state, the signal is passed and it is updated in the confirmation state. The exceptions are handled using the signal as timeout, after completion the state is destroyed.

4.4. Configuration view

It described the earlier stage of the software architecture system which is under development, has the view of the component and connector types that are defined in the session 4.2. The configuration elements are the instance of connector and computational types. We have two types of connectors defined as simple connectors for two components and higher level connector for more number of components.

The simple connector instances are classified as dynamic ports and link between these ports. The fig.10. Is an example of the configuration view of CTscan, it has the *ctcontrolstation* , *ctdevice* which are connected using simple connector type instances as signaling and streaming.

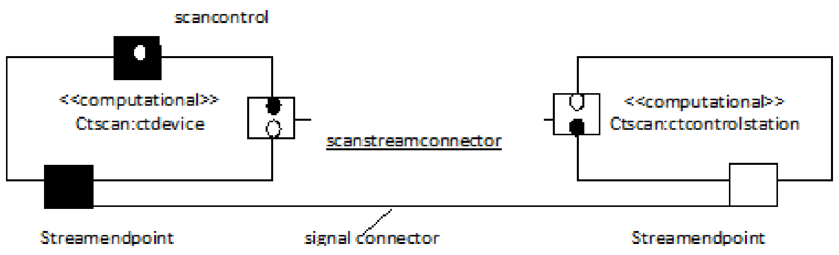

Fig.10. simple configuration of CTScan system

5. Summary and Conclusion

In this paper we have extended the standard UML with the help of the key constructs in ADL, to form a modeling framework to create software architecture. We have given importance on the certain concepts such as views, viewpoints, components, configurations and components. We have used some extended notations to represent the software architecture in UML.

We have also explained in detail about simple and complex connectors using some specific notation, the streamendpoint protocols are also defined here which would be helpful in designing a complex software architecture. The model that has been presented in this paper has a base of P147. We have used the CTscan device as an example. In future we can enhance the framework for other examples and say how it is comfortable in multidimensional issues.

References:

[1] Aldrich. J, "Using Types to Enforce Architectural Structure," Proc. Working IEEE/Int'l Federation for Information Processing (IFIP) Conf.Software Architecture, pp. 211-220, 2008.

[2] Buschmann. F, Jäkel. C, Meunier. R, Rohnert. H, Stahl. M, Pattern-Oriented Software Architecture – A System of Patterns, John Wiley & Sons, 1996.

[3] Harel. D and Rumpe. B, "Meaningful Modeling: What's the Semantics of "Semantics?" Computer, vol. 37, no. 10, pp. 64-72, Oct.2004.

[4] Hofmeister. C, Kruchten. P, Nord. R. L, Obbink. H, Ran. A, and America. P, "A General Model of Software Architecture Design Derived from Five Industrial Approaches," J. Systems and Software, vol. 80, no. 1, pp. 106-126, 2007.

[5] Krogstie. J, Sindre. G, and Jørgensen. H, "Process Models Representing Knowledge for Action: A Revised Quality Framework," European J. Information Systems, vol. 15, no. 1, pp. 91-102, 2006.

[6] Romina Ermo, Vittorio Cortellessa, Alfonso Pierantonio, Michele Tucci., Performance Driven architectural refactoring through bidirectional model Transformation., ACM SIGSOFT Conference on Quality of Software Architectures, QoSA2012.

[7] Roh. S, Kim. K, and Jeon. T, "Architecture Modeling Language Based on UML 2.0," Proc. Asia-Pacific Software Eng. Conf., 2004.

[8] The SAE Architecture Analysis and Design Language (AADL),http://www.aadl. info/, 2009.

Authors Profile

R. Aroul canessane received the M.E.in Computer Science and Engineering from Sathyabama University, Chennai, Masters of Computer Applications from St. Joseph's College of Engineering, Chennai, University of Madras. He is presently pursuing the Ph.D degree in the Department of Computer Science and Engineering, Sathyabama University, Chennai, India. He has 13 years of experience in Teaching. He has also held various responsibilities as a part of his research. He has published around 6 research papers in journals and conferences. He has written books for some of the universities. His research area is Software Engineering and also interested in Data Base Management Systems and Data Warehousing and Mining.

Dr. S. Srinivasan received the Ph.D degree in Computer Science & Engineering from the Sathyabama University, Chennai, and M.Tech. in Computer Science & Engineering from the Indian Institute of Technology, Chennai, and M.B.A. in Systems & Finance from Sathyabama Engineering College, University of Madras, Chennai, and the M.Sc. in Mathematics from the Gobi Arts College, Gobichettipalayam, Bharathiar University, Coimbatore. He is presently working as Professor and Head, Department of Computer Science and Engineering , Anna University, Regional Centre, Madurai. He has 20 Years of experience in Teaching and Research. He has also held various positions and responsibilities in Technical Institutions. He is acting as expert member at various universities in various capacities.. He has published more than 40 research papers in journals, books, conferences, and workshops. His research interest include text mining, data mining & data warehouse, and Software Engineering.