2016 3rd Asia-Pacific World Congress on Computer Science and Engineering

Comparative analysis of Software Architecture Documentation and Architecture Languages

Mateen Ahmed Abbasi
UIIT-PMAS
Rawalpindi, Pakistan
mateenabbasi@msn.com

Dur –e-Benish Batool
UIIT-PMAS
Rawalpindi, Pakistan
benimalik@rocketmail.com

Rahil Butt
UIIT-PMAS
Rawalpindi, Pakistan
rahilbutt82@hotmail.com

Tanveer Mehmood
Anjum
UIIT-PMAS
tanveer2382@gmail.com

Abstract:

Research on software Architecture is vigorous from the early 90's and its lead to a number of different architecture description languages (ADLs). These languages are increasing in a huge amount and are different in term of analysis produced by these languages and abstraction supported by them. Moreover, a lot of other languages not intended as ADL serve reasonably fine on analyzing and representing software architecture like Unified Modeling Language 2.0. UML accomplish approximately all necessities of Architecture Description Language. This research work focuses on comparative analysis of Architecture description languages and software architecture documentation on the bases of usability, connector support, formal semantics, behavioral specification, Language quality, views and dynamic architecture support.

Key words:
ADLs, Software Architecture, SAD, UML

I INTRODUCTION

The concept that software architecture is a subdivision of software engineering in near-about twenty years old. In the last twenty years many software description languages came into view and gone, excluding only a few architecture description languages no architecture language is admired by the professionals but software architecture documentation which uses UML which even not acknowledged as Architecture description language or considered with vacillation is became a regular notation in the industry for documenting software architecture.

The software architecture system explains standard structure of interrelate component of software architecture. For describing the interrelationship linking the different components the architects uses informal boxes and arrow diagrams. Many different Architecture Description languages were build up in the starting phase of research on the software architecture, with the help of developers who perform experiments on structures needed to support relations and architecture description.

ADLs formally represent the system's architecture. In the last twenty years it is has been seen that the appearance of many Architecture Description Languages is proposed by researchers in academics and software industry. [1][2] [3]. To analyze and represent architectural design the architectural description languages provide details. [4][5][6]. these type of software details provide mutually conceptual frameworks and tangible syntax for describe architecture of the system. A few newly well-known Architecture description languages are Wright, Unicon, Adage, Drawin, C2, Aesop, Mata-H and Rapide. This type of Architectural Description Languages apprehensive with architecture's design and provide distinct properties. SAD serves for several intentions. Software architecture documentation can be effortlessly and rapidly understandable by new-fangled developers. SAD acts as a prototype for creation of architecture and has sufficient information that SAD be able to serve for analysis. SAD represents the architecture in a three dimensional way and is descriptive. This research focuses on recently most popular Architecture Description languages like Unicon, Wright and the most new one Architecture Description language Software Architecture Documentation (SAD) and Architecture Analysis and Design Language (AADL). Many surveys were conducted for Architecture Description Languages [7] [8] but they haven't mention SAD in any survey that SAD also provide descriptive analysis in the type of documentation. In this research evaluated these architectural languages and Software Architecture Documentation against a number of important parameters: (i) Formal Semantics (ii) Dynamic Architecture Support (iii Language Definition Quality (iv) High level Connector (v) Formally Analyzable (vi) capturing design views (text , graphic, both) (vii) High level components.

978-1-5090-5753-5/16 $31.00 © 2016 IEEE
DOI 10.1109/APWC.on.CSE.2016.39

II. COMPARISON OF LANGUAGES

Before doing anything we will discussed the Architecture Description Languages (Unicon, Wright, Architecture Analysis and Design Language) and then Software Architecture Documentation on the given constraints

- Component and Connector Support
- Semantics
- Dynamic Architecture Support
- Behavior Specification
- Language Quality Definition

a) Wright

The Architecture Description Language Wright is renowned for its unambiguous and formal behavior of connectors in architecture design. [1]

Component and Connector support:

The Architectural language Wright's focal point is on the formal analysis of components which are based on connectors and embrace supports for detaining software architecture features. Connectors in Wright are started with the instance of the connectors which allowed reprocess of the similar interaction pattern on dissimilar frameworks and moreover the study of connectors in seclusion. The Type of the Connectors is defined by the role which represents the participating components and a fixative which coordinates with the behavior of the role. [8]

Semantics:

The Components of Semantics are describing by writing and designing the ports CSP process by the specification process which synchronizes the ports. [8] The structural language wright's semantics are also described in CSP.

Dynamic Architecture support

Almost all the Architecture Description Languages are unambiguous in nature. Wright provides incredibly modest support towards dynamic architecture. [9]

Behavior Specification:

In communicating sequential process the behavior specification in Wright is done. The behavior specification of component forms is controlled in two fractions: Specification process and port process, where the specification corresponds to the internal behavior for multifaceted, complex types and ports represent their outer behavior of the components. [8]

Language Quality Definition

Wright provides the reliable and comprehensive architecture specification and it provides the high level of language quality. [9]

b) UNICON

Unicon is one more premature Architecture Description Language which permits the designers to identify the connectors and components. Unicon supports evolution of system design to implement the code flatter and real life applications.

Component and Connector support:

The software architecture illustrated in Unicon consists of a quantity of connectors and components. The Components represents 868 data or computational units of the system. The units of data of a system are characterized by components and the connectors in unicon act as negotiator in the communication between unicon's components. Every component is linked with an implimentaion and an interface. The connectors in the unicon intercede the communication among components. [10] But not as Wright, unicon limits protocols to be definite types for example Data access, Procedure Call, Pipe, thus avoiding designers from freely specifying their types.[8]

Semantics:

Unicon's focus is on early generation of code from architecture specification and it doesn't formally describe semantics. Unicon present a set of tools for mapping architecture in C source code. Whereas it permits system replication, and it's challenging for formal verification. [8]

Dynamic Architecture support:

Unicon can handle the dynamic arc hitecture because it is specific in nature [9].

Language Quality Definition:

Unicon provides dreadfully modest support to constancy of architecture specification and doesn't give complete architecture specification because it does not support formal semantics.[9]

Behavior specification:

Unicon doesn't permits formal behavioral specification of architectural components where as Wright, Unicon and Darwin allows it. However, UniCon recommends a set of integral attributes for connector and components templates and as well players/roles.[8]

c) Architecture Analysis and Design Language (AADL)

Architecture Analysis and design language is an Architecture Description Language that intended to support hardware, software and mixed system architecture. [11] [12] AADL has the highest amount of users among other architecture languages because of its specialization. To design and analyze the software and hardware architecture of real time system AADL use graphics and text. AADL illustrate the important features, performance and functional interface of components. [11] [12]

Component and Connector support:

Unlike the above mentioned architecture description languages AADL has low level built-in component support. Architecture Analysis and design language provides no support for connectors. Components interrelates using ports or by suing subprogram-calls and the connections are limited to the subsequent mechanisms: parameter connections, component access connections, port, subprogram calls and connections. But there is no support in favor of specifying innovative connector types that can characterize complex interaction protocols. [8][10] [12]

Semantics:

The semantics of Architecture Analysis and design language are described in natural language because it is not formerly developed with the specific semantic. But many attempts were made in this sense afterward. [8] [10]

Dynamic Architecture support:

Architecture Analysis and design language doesn't fully support the dynamic architecture but at someplace it supports dynamic architecture and variability. [8][10][12]

Language Quality Definition:

AADL not utterly sustain the language class characterization because ADL is not developed with accurate semantics. Natural language is used in AADL that's why to complete the architecture specifications AADL make available intermediate support and a little hold to the stability of architecture requirement. [10]

Behavior specification:

To perform behavior specification in AADL behaviors seize is attached to module specification. [13]

d) Software Architecture Documentation (SAD)

Similar to Architecture Description Language, Software Architecture Documentation illustrate element interfaces, Test scenarios, Subsystems limitation, third-party module buying choices, exterior services, Behavioral specification, Team structure and schedule dependencies. [12] Software Architecture Documentation consisted on natural languages and Unified Modeling Language diagrams and SAD must encompass element relation properties rules. [14]

Component and Connector support:

A few of UML 2.0 notations are used by Software Architecture Documentation because UML supports mutually model based and object oriented perception. UML 2.0 improved support towards the modeling architectural problems of the software system. Among the most important features add up enriched interfaces, ports, superior components and connectors.[14][15]

Semantics:

The theory Software architecture Documentation is based on well defined prescribed rules and regulations and can be processed and checked by machine because SAD uses Unified Modeling Language. 14] [15]

Dynamic Architecture support:

Software architecture Documentation entirely holds up to grip dynamic architecture as there are a number of tools available for Unified Modeling Language. UML have different diagrams like, Object diagram, Use case diagram, activity diagram etc. [16]

Language Quality Definition:

Due to uncertainty in Software architecture Documentation SAD provide low support to consistent architecture and provide intermediate support in the completeness of architecture specification. Sometimes SAD uses mutually graphic and natural language that's why here be dilemma of uncertainty. And in a few cases it only utilize natural language or graphic (UML). [14][15]

Behavior specification:

Software behavior Document (SBD) describes the performance of software by setting and requirements. [15] In Software Architecture Documentation Unified Modeling Language 2.0 covers behavioral as well as structural characteristics of software system. [17]

III. OBSERVATIONS

In table 1.0 comparison of ADLs (Wright, Unicon, AADL) and SAD are given on different parameters. Following are the meanings of the symbols used in table 1.0:

Hs: High capability: language gives comprehensive and unambiguous support

Ms: Medium: capability may be achieved in a roundabout way. Language offers standard features

Ls: Low: modest support granted

Ns: No Support

Table 1.0: **RESULT**

Attributes	**Unicon**	**SAD**	**Wright**	**AADL**
Consistency of architecture Specification	Ls	Ms	Hs	Ls
Completeness of architecture Specifications	Ms	Ms	Hs	Ms
Behavior Specifications	Ns	Hs	Hs	Hs
Textual	Hs	Hs	Hs	Hs
High level Component	Hs	Hs	Ls	Ns
Connector Support	Ns	Ls	Hs	Ns
Formally defined Semantics	Ns	Ms	Hs	Ns
Formally Analyzable	Ns	Ls	Hs	Hs
Graphical	Hs	Hs	Ns	Hs
Dynamic Architecture Support	Ns	Hs	Ns	Ms

IV. DISCUSSION

From the reviews presented in earlier section we can assume that there is increasing attention in Architecture Description Languages as they offer precise support in the development of software architecture. Architecture Description languages are mainly popular in safety critical applications such as process control, infrastructure, medicine, spaceflight and various others. Almost every hardware or software architecture gets benefit from the rigidity brought by Architecture Description languages. ADLs has some limitation as well as advantages, Software architecture documentation sustain equally Model based and object oriented concepts as Software architecture documentation mostly uses unified modeling language in architecture description, with the help of SAD the limitations of Architecture Description Languages can be overcome. In The Table No 2.0 we describe some strengths and weakness of the ADLs and SAD. Most of the properties of SAD is because of Unified Modeling Language (SAD use some of its diagrams)

Table 2.0. Strengths and Weakness of ADLs and SAD

	Strengths	**Weakness**
ADLs	• ADLs signify software architecture in a clear and error free manner. • ADLs hold recitation of system at advanced stage of notion. • As the mainstream of ADLs is textual as a result machine understandable and appropriate for automation. • Due to proper representation of ADLs they permit analysis of architecture's exactness, completeness, vagueness and performance. • ADLs sustain automatic production of systems which run hardware and application programs.[8] • They present graphical language rules and a textual form also. • Properly described logics and rules. • The help for manufacturing and verification is available by Every ADL. • The modification of architecture is done by ADLs. • ADL is handy and user friendly. • ADLs overpass the space among research and the real world, provide the requirements of practitioner. • Similarly to Unified Modeling Language ADL provide multiple visions. • Only reliable real work no visualizing. • ADLs are mutually extendable in tools and language support [17] [18] [19].	• Most of ADLs are domain depended like avionics etc and are only fit for that type of domains. • Mostly ADLs are text based and are less attractive for other domain's software architects [16] [17] [8]. • The main weakness of ADLs is that they be short of sustaining tools with the exception of few .[8][18]
SAD	• Stakeholders can easily understand the system using SAD. • It gives clear behavior specification as SAD use unified modeling language. • SAD offers greatest connector support. • Provide graphical illustration to software architecture. • For the stakeholders it provides multiple view. • Several tools are there for unified modeling language. • SAD can grip distributed problems. [15] [16] [17]	• SAD is not appropriate in favor of computerized analysis of verification and validation of system architecture. • Due to be deficient in prescribed semantics SAD becomes basis of haziness and discrepancy in a few cases. • Lacking of stability is caused due to not up-to-date documentation of software architecture. [17][18] • SAD documentation is repeatedly conflicted. Conflicting move toward in special figures, resembling to conflicted structure inside and across documents or ambiguous information. [18] [19]

In year 2013 a research study was performed in which forty-eight practitioners from forty different information technology organizations of fifteen countries take part to study that what requirements industry need from Architecture languages.

The value of ADLs features in precedent and upcoming projects was studied. Worth of ADLs features in precedent projects are :

i. Support for iterative architecting,
ii. Versioning,
iii. Well-defined semantics,
iv. Support for multiple architectural views
v. Tool support ,
vi. Analysis ,
vii. Graphical syntax.

In Table 3.0 we provide the summary of our research work, we evaluated ADL and SAD on the extent of little, average and high on four factors which we studied.

Table 3.0

Factors/	ADLs	SAD
Language Quality Definition	High	High
Behavior specification	High	average
Semantic	average	little
Dynamic Architecture support	average	High
Component and Connector support	average	average

V. CONCLUSION

Ever since the early on nineteen's a number of ADLs) have been projected that allows the developer to specifically design their system architectures in a proper, specific and presentable way. Architecture description languages are normally recognized with their ample support for the system architecture specification and their premature prescribed analysis. Still, regardless of the strength provided by architecture description languages. These languages still have not come into the mainstream. In this research work we acquired two early prominent Architecture Description Languages, one recently and mostly utilized language and also acquired Software architecture Documentation But the professionals are still not capable to develop an Architecture Description language which make possible the specification of multifarious systems in a method that allows premature formal analysis and at the similar moment promises that e architecture is analyzable, practicable, consistent and absolute . However in real time application systems and in small scale applications architecture as compare to other Architecture Description Languages, Software architecture Documentation is much consistent, since SAD uses unified modeling language and due to its properties and many professionals are moving towards Unified Modeling Language. UML is cost-effective and simply understandable. This research provides information to three communities, 1st is architect who decide an Architecture Description Language, 2nd is technology sponsor who fund for development of architecture language and the 3rd is language creator. This research is an effort to to increase the motivation towards software documentation language. The scope of this paper was limited. In future work additional effort could have been expended in identifying a more parameters of ADLs and SAD.

ACKNOWLEDGEMENT:

This research has been conducted with the partial support from i-Lab Australia

References

[1] Shaw, DeLine, Klein, Ross, Young, Zelesnik "Abstractions for Software Architectures and Tools to Support Them". 1994

[2] N. Medvidovic, P. Oreizy, J. E. Robbins and R. N. Taylor.(1996) Using object-oriented typing to support architectural design in the C2 style.

[3] M. Moriconi, X. Qian and R. Riemenschneider.(1995) Correct architecture refinement. IEEE Transactions on Software Engineering, Special Issue on Software Architecture

[4] P. Binns and S. Vestal. (1993) Formal real-time architecturespecification and analysis.

[5] L. Coglianese and R. szymanski,(1993) DSSA-ADAGE:An Environment for Architecture–based Avionics development.In Proceedings of AGARD'93, May 1993.

[6] D. C. Luckham, et all.1995 Specifications and analysis of system architecture using Rapide.

[7] R. K. Pandey2010. Architecture Description Languages (ADLs) vs. UML: A Review

[8] Mert Ozkaya and Christos Kloukinas. (2013) "Are We There Yet? Analyzing Architecture Description Languages for Formal Analysis, Usability, and Realizability".

[9] Paul C. Clements.1996. "A Survey of Architecture Description Languages".

[10] George A. Papadopoulos .2008. “Evaluating the Use of ADLs in Component-Based Development”.

[11] Shenglin Gui and Lei Luo, .2008. “UCaS: A Schedulability Analysis Tool for AADL Models”.

[12] Peter H. Feiler, Bruce A. Lewis, Steve Vestal, 2006. “The SAE Architecture Analysis & Design Language (AADL) A Standard for Engineering Performance Critical Systems”.

[13] ROBERT ALLEN and DAVID GARLAN .1997. “A Formal Basis for Architectural Connection”.

[14] Book: Documenting Software Architectures: Views and Beyond. 2003 Paul Clements.

[15]Klaas Andries de Graaf ,Antony Tang,Peng Liang *And* Hans van Vliet. 2012” Ontology-based Software Architecture Documentation”.

[16]Antony Tang et all.2011. “Software Architecture Documentation: The Road Ahead”.

[17] B.Bharathi, Dr.D.Sridharan. 2009 “UML as an Architecture Description Language”.

[18] Matúš NAVARČÍK .2005 “Using UML with OCL as ADL”

[19] Ivano Malavolta, Patricia Lago, Senior Member, IEEE, Henry Muccini,Patrizio Pelliccione, and Antony Tang, “What Industry Needs from Architectural Languages: A Survey”, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 39, NO. 6, JUNE 2013

[20] Werner Heijstek et all .2011 “Experimental Analysis of Textual and Graphical Representations for Software Architecture