WEB OF SCIENTIST: INTERNATIONAL SCIENTIFIC RESEARCH JOURNAL
ISSN: 2776-0979, Volume 3, Issue 8, Aug., 2022

DESCRIPTION OF THE ARCHITECTURAL SURVEY LANGUAGES

Fryal Jassim Abd Al-Razaq
University of Babylon , IT College , Hillah, Iraq

Safa Saad AL-Murieb
University of Babylon , IT College , Hillah, Iraq

Abstract

The architecture is a broad concept, and the idea of each project / field what the architecture and what types of data, which should contain. As a result, he introduced many of the exact scope and general purpose architecture description languages (ADLS) and use them in various fields, with particular emphasis on architectural issues. Nevertheless, the ADL and nobody ever appropriate for all projects or fields. This diversity creates a problem for users of these languages, including: the implementation of a code division architecture, allowing contradictions, which leads to confusion, violating the architectural features that prevent the development of software. To overcome these problems begin to describe the basic elements used to build some of the recent ADLS including AADL, MetaH, SADL, ArchJava and ACME. And we also show how these elements interact with each other, and how to use the restriction to control their behavior. We compared the classification depends on the support of the attributes of each of these languages (for example, fields, views, architectural design, construction, interfaces and types, semantics, and restrictions, volatility, updates, maintenance, easy to use and so on ...) into three main categories Features-oriented system, linguist and features, as well as in processes intended function. And finally, we hope that this will result in a building designed structure is well thought out and very well knows that gives engineers the opportunity to talk about the system characteristics (such as resistance) at a high level of abstraction, and is essential to Success of a complex system based on software layout.

Keywords: Software Architecture, ADLs.

1. Introduction

Architecture in the past were mainly representatives of pairs of box and line drawing. During the past decade has seen considerable progress in research, Development of concepts, tools, methods and supports for software architects. Many formal description languages of Architecture (ADLS) -I- have been developed and implemented in real word applications. Specialized for the architects of the

instrument of analysis can reveal hidden problems, Such as blockages, race conditions do not match the compatibility of the interface system platform of category. Many of these languages support the complex architecture of logical analysis, focused on development, support and gold. Table (1.1) shows a list of some recent ADLs. There is no clear line between ADLS not ADLS. In principle, programming languages and the development of language models and material description languages, the general aspects to ADLS. Languages, which are born as ADLS net advantage over architecture on the basis of representation in other languages are emerging. The purpose, then represent ADAPTE architecture. Table (1.2) includes a list of ADLs and their purpose.

Table 1.1 A List of Some Recent ADLs, Organization, And Team Leader

ADL	Organization	Leader
AADL	Society for Automotive Engineers (SAE)	Bruce Lewis
ACME	Carnegie Mellon University	David Garlan
AESOP	Carnegie Mellon University	David Garlan
CODE	University of Texas at Austin	J. C. Browne
ControlH & MetaH	Honeywell Technology Center	Steve Vestal
Demeter	Northeastern University	Karl Lieberherr
Gestalt	Siemens Corporate Research, Inc.	Bob Schwanke
Modechart	University of Texas at Austin	Al Mok
Rapide	Stanford University	David Luckham
UML	Rational Software Corporation	
UniCon	Carnegie Mellon University	Mary Shaw
Wright	Carnegie Mellon University	David Garlan
ArchJava	University of Washington	Jonathan Aldrich

Table 1.2 A List of ADLs and Their Purpose

ADL	Purpose
AADL	Provides textual and graphic notation with precise semantics for embedded and real-time systems, standard modeling notation for applications, open source tool solution.
ACME	Simple, generic software architecture description language (ADL) that can be used as a common interchange format for architecture design tools and/or as a foundation for developing new architectural design and analysis tools.
AESOP	Provides a generic toolkit and communication infrastructure that users can customize with architectural style descriptions and a set of tools that they would like to use for architectural analysis.
CODE	The CODE system can produce parallel programs for shared-memory and distributed-memory architectures, including networks of workstations.
MetaH	Specifies how software modules developed in a variety of styles are composed together with hardware objects to form complete system arch.
Demeter	Split software into at least two parts: The first part defines the objects. The second part defines the operations. The goal of Demeter is to maintain a loose coupling between the objects and the operations.
Rapide	language effort focuses on developing a new technology for building large-scale distributed multi-language systems.
UniCon	Is an architectural description language whose focus is on supporting the variety of architectural parts and styles found in the real world and on constructing systems from their architecture descriptions.
Wright	allows architects to specify temporal communication protocols and check properties such as deadlock freedom.
C2	Is a general component-base and message-based architectural style that is well- suited for large-scale, heterogeneous, and distributed applications. The C2 SADL is an ADL for expressing architectures in the C2 style.
Darwin	Is to specify system architectures in terms of components and their interconnections.
xArch	Is a standard extensible XML-based representation for software architect
ArchJava	Extends the Java language to include explicit architectural modeling constructs, and use type system techniques to ensure that the implementation conform to the architecture.

2. Related Work

In recent years a number of international conferences and seminars have been organized and numerous research papers have been published to address various issues in the architecture of description languages. In AD Fuxman [2, 2000], several ADLS ranked and compared, including the Rapide, Darwin, Aseop, Unicon, Wright and ACME. Comparisons supports the basic architectural elements: components, connectors, configurations and styles, focusing on focusing on their functionality for formal modeling and analysis, such as formal semantic models and automatic property verification Architectural. N. Medvidovic and RN Taylor [16, 2000], presented the framework for the definition and classification of ADLs. The structure used to classify, compare and identify the key properties of several existing AVQs. A simple test document for AVQ is provided by the definition, which largely reflects the consensus of the community on what is important in the modeling architecture: architecture is different from other signs of its architectural configurations and connectors. They also showed that the accompaniment scope and definition can be used for deciding if the destination is ADL, some estimations were rejected as potential ADLs. Components, configurations, and connectors should be clearly modeled by the ADL. P. Mishra; N. Dutt [25, 2005], this paper studies existing ADLS terms (a) the internal characteristics of languages and (b) supports the methodology for simulation, synthesis, synthesis, testing and verification of Integrated system. It concludes with a discussion of the advantages and disadvantages of the existing relative and future ADLs of expected performance.

J. Aldrich, C. and D. Chambers Notkin [3, 2002], shows a case study showing that ArchJava can be applied to a moderate size of Java program with relatively little effort, resulting in program structure Which more closely corresponds to the designer of the conceptual architecture. Thus, Java Arche facilitates effective architecture based on the development, understanding and implementation of the evolution program. P. H. Feiler, B. Lewis, C. Vestal [9, 2003] have demonstrated AADL over complex safety critical systems constructed in the field of avionics. An increasing number of tools, as well as supporting the ALDA transition become available, including current AADL graphics. This paper is organized as follows: Section 3 describes the basic elements for the last ADL, including: AADL, MetaH, SADL, ArchJava and ACME. Section 4 classifies and compares the ADLS according to the features supported by each ADLS. Finally, in Section 5, Results of our survey results are tabulated.

3. ADLS Support Architectural Elements

Components 3.1: Provide the basic computational elements, and to store the data system.

All languages cannot distinguish between a component and a component, for example an interface that displays the interface. All languages provide syntax and semantics for the component interface specification. All languages see interface specification component, as defining the type of component that can be multiple instances of components that have the same interface [15]. The architecture of the components can be as a singular procedure (for example, procedures of Metah) or it can be as whole application (for example, macros of MetaH). This may need the owned datafor it or / and a space for execution, or it can apply the concept of sharing with other parts [16]. The components arevAADL type which is a component of the description, as well as one or more implementations. Generalization of the components supported by types of component and embodiments which can be expressed as an extension of further types of implementation and components. AADL is not suitable for the design and installation of interior components. Application modeling system is supported by two groups of component categories. The execution behavior, which consists of fibers and processes the source code system that consists of a unit packet in the source text block and the component of data as passive application data. The internal development and implementation of these components can be determined by linking the original text written in the language of a software program, such as Ada95 or C, and so, or domain-specific modeling languages such as Matlab / Simulink [8, 9].MetaH allows specifications for multiple implementations that present the same abstract interface and allow the selection or instantiation of different implementations for components of the same type (same abstract interface) within an architecture, the MetaH interface specifications are Considered as behavioral abstractions Comprehensive behavioral specifications. MetaH allows static declarations of multiple instances that present the same interface. For MetaH, the interface and implementation can be specified separately. In MetaH, components must also be categorized according to a set of language-specific classes: events, ports, subroutines, packages, processes, macros, and modes. The user-specified components must belong to one of these categories and each category imposes certain language-specific restrictions (for example, authorized classes of subcomponents, component attributes). In MetaH, the elements of an interface can themselves be components (interfaces with subcomponents). It provides basic interface element types (ports and events in MetaH). MetaH also allows complex interface components that themselves have specifications (component declarations in MetaH) [15,16] . New constructs of

language are added by ArchJava in order to support components, ports, and connectors. T-he component is a special type of object that communicates with other components in a structured way. Components are instances of component classes. The classes can contain ad component ports and connections, in addition to the statement statements that can be contained in ordinary classes. Components can be created dynamically using the same new syntax used to create ordinary objects. At the time of writing, each component stores the component instance that created it as a parent component. For components that are instantiated for example, the chassis component, the parent component is invalid. The component will eventually be collected in the absence of references to components or compounds are not available from the roots of garbage collection. Components connected to other components via explicitly declared ports. The port is the logical links between the components, which are interconnected. Ports set to announce the three methods provided for the application, supply and dissemination of keywords. The purpose of this project is to determine the port as a service, implemented by a component should do the job. The interfaces that were required make explicit dependencies, by reducing linkage between components and separation in components understanding. Ports also facilitate reflection on communication component models [3, 7]. ACME components conform to the descriptions of the boxes and lines of software architectures. Typical examples of components: a client, a server, a filter, and database objects. Component Interface defines a set of ports. The component can provide multiple interfaces using different types of ports. The interface port can be as simple as a single procedure or a more complex interface, such as a set of procedure calls, or only once the interface of multi-player events [10]. SADL is a semantic unit, in which the components of the language specification represent one of the simulation components that are either hardware or software simulation device. SADL interface component is a port and interface points marked in MetaH, SADL and other languages. SADL makes explicit use of the parameters of the signature interface for components. Typically, this is done in such programming languages as C ++ and Ada [16, 26]. All languages are supported ADLS interface component specifications. They be different only in the information type they have provided and the terminology, such as ACME, AADL calls the port.

3.2 Connectors: represents the interaction between the components

AADL modeling support three types of interactions between components of control and exchange of data between components, including: transferring messages, events, passing and remote calls [9]. MetaH has certain linguistic categories of compounds, as they do for components: connection events, port connections, joints and

equivalence connections for MetaH access. Each category has its communication semantics and restrictions based on class attributes and components are connected. MetaH has only a fixed set of defined compound language courses; The user can not specify the new classes of compounds or abstraction possible for the components (although MetaH provides a number of connection attributes for the user can set the fine tuning of the communication) [15,16,19] . In ArchJava symmetric connect primitive connects two or more ports together, linking each method required to provide a method with the same name and signature. Arguments for the connection may have its own ports one of the components or subcomponents are in finished domains. Verification of the integrity of the connections are made to guarantee that every method which required is associated with a single means provided. It provides methods which can be performed by transferring calls to necessary methods or components from other ports. Semantic support language for sending and disseminating methods, a semantic alternative of compounds (eg instantiated, in ArchJava asynchronous communication can be performed when a custom intelligent connector is written. The integrity of the connection limits the methods that can be used in component interfaces Because only the parent component can call its Methods directly, it is important to get links to the sub-has not escaped the scope of their parent component. Dynamically created components can be linked to each other at runtime by using the join expression Considering integrity requires that each component is clearly documented the types of architectural interactions are allowed between the roll up. For describing a set of compounds the transparent connection is used, whereas at running time it can be instantiated by the connected expressions. Port interfaces describe the orifice, which can be instantiated repeatedly to communicate with the help of various compounds (such as, in the Web server communicates the router component with multiple pieces of work, each through a different connection.) Ports Interfaces instantiated using CONNECT statements, a connection object is returned by CONNECT operator expression, it implements all connected ports ArchJava port interfaces support two types of methods: calls Direct method and method calls by Java ports enforces the integrity of communication, providing the invariant that all components -typed expression scope reaches the C component refer to itself or to the sub-immediate C in ArchJava C-type system, only the instance of the named component in the interface port type is allowed to make calls via the connection object that implements this type of interface port. Type arc. Java prohibits calls to methods that violate architectural constraints [3, 4]. ACME connectors correspond to box-office and online lines. Simple interaction forms include: message passing, pipes, broadcast events, and calling of remote procedure. Sophisticated

forms of interaction include: client-server protocol and a SQL link. The systems are configured with components and connectors. The interfaces of connection are points of interaction between the latter and the components which are attached to it. This makes it possible to reason on well-educated configurations. Connectors Interfaces define the roles. There are two roles of binary connectors, like the RPC connector that is called and a caller, read and write the role of the conduit, or the sender and the role of the connector receiver [10]. SADL connector is a part of the design that used for a specific style of architecture. In the architecture, the specification of the connector is specifying the brackets type of data connector. in the definition of SADL connector style, arity and restrictions of uses of a connector, etc. are given. SADL bases connector types on the communication protocol, it also provides parameterization tools, which allow the specification of flexible signatures and restrictions on the semantics of the socket [16, 26]. Only the connectors of ADLS assist the connection interfaces specification, explicit port components and then the necessary connector roles in architectural configurations.

3.3 Configuration:

Architectural configuration or topology bound graphics components of the architectural structure and connectors which describe the structure. Such information is necessary to decide whether the respective connected components and their interfaces are the same, the connectors for good communication and the semantics in combination lead to the desirable and demand behavior. Configuration descriptions assess the two aspects of a distributed architecture, for example, reliability, possibility of deadlock (or failure) and starvation, security, performance, etc. .also Configurations allow you to analyze respect for designning constraints and heuristics type [16]. Model of architecture AADL according to the hierarchy of the elements, whose interaction is represented by the compounds [9]. Configuration descriptions in online configuration ADLs, for example, MetaH connection parts, usually overloaded, while the explicit configuration example Adis, ACME, has the best potential to promote clarity architectural structure. Both textual and graphical notations are provided by some languages. A Graphical type achieves. However, it is only in the case that there is a clear link between the graphs and the description of the basic model, MetaH support is "semantically its" graphic symbol, while ACME does not [16]. ACME is built from seven types of objects to represent the architecture, including: components, connectors, systems, ports, roles, views, and repmaps. ACME supports a hierarchical description of the architecture. Any component or connector may be represented by one or more detail. Each description is a representation in

ACME. Component or connector, which has an architectural representation of previous display representations indicating the correspondence between the internal systems and the external representation of the component interface or a connector presented [10].

Developers can be assured that ArchJava architecture accurately reflects the relationship between components, such as language semantics to ensure the integrity of communication. The integrity of the Java connection means that the components of the architecture can not call methods on each other between the ports of the claimed compounds. In the architecture, every component can use its ports for communicating with the other components that is connected to them [3]. SADL assists connector sub-types and their updates through abstraction's styles and levels. In the configuration section defines the SADL restrictions configuration of the components and connectors described above. These restrictions may argue that, for example, the function or a variable component access / write component, a component sends a signal that the component obtains, the direction of the data flow between the two components, as well as on that Shutter of the read operation is called [16, 26].

4. ADLS Specifications

To compare and contrast existing ADLS, The study used a range of important characteristics that certain ADL may or may not be.These functions can be structured into three categories: feature oriented systems features language-oriented, and process-oriented features.

4.1 system-oriented features

System-oriented functions are functions related to the application of the system derivative of the description of the architecture. For example, some may not express ADLS limit in real time, while others have this function [1, 2].

4.1.1 Architectural styles

The architecture in the computer system is a configuration of constituent and connectors that occur repeatedly. Typically, the architecture is similar to the type of components and connectors. They have a particular way of interaction between components. They are not a complete software / subsystem system. Architectural styles are deliberately ambiguous by the number present in its components and connectors. Architectural styles put restrictions on components and connectors. Most ADLs provide a mechanism for determining architectural styles, such as pipes and filters; Programs and subprogram’s; Multi-level; Object oriented; Data exchange

processes; Event system; Transactional database systems; Blackboard; Translator; On the basis of the rules; Heterogeneous styles; And other styles [15,16,21].

4.1.2 Areas: What are ADL application areas, specifically designed for support, if any, and how and to what extent?
Development models AADL embedded systems with complex resource (size, weight, power) the limitations that are difficult to requirements and a strict reaction in real time, the system should endure disadvantages of these devices and use of I / O specialized equipment. Including avionics system, flight control, power management engine and medical devices, control equipment for industrial processes, robots, space technology, and can be extended to support other essential applications [8, 9]. MetaH was designed to create avionics and flight control systems, missiles and aircraft. It was developed to integrate several software applications in the field of avionics, generated architectural background based on the planning and implementation of formal methods. Save MetaH provides developers with a simple but precise language to indicate the architectural requirements, from which it extracts the formal parameters for the simulation of several analyzes. It includes both hardware and software components. It will generate the architectural integration of hardware and software components in the system with the modeled behavior of complaints [22, 23]. ArchJava for any Java application designed to run on the same Java Virtual Machine. For applications written in different languages and covering a few cars, some of them are written in Java and runs on the local level are still able to get ArchJava. Note also that the methods used in ArchJava can be applied to other typed languages (eg, C #), although our current system only supports Java [3, 4]. SADL is used for the formal specification of the level of detail of architecture and a graphic area, ADL-specific, making it easier to spec a high-level hardware and software aspects of driving simulation system, Avionics real time, is designed to work in a variety of hardware configurations, including multiprocessor systems. [26]

4.2 oriented features Language
Oriented linguistic characteristics are the characteristics of ADL. An example of formalized syntax and formal semantics and ADL are the architectural abstraction that embodies the ADL [1, 2]. Quality Score 4.2.1 Language as syntax and formal semantics are determined by language? Is there a loudspeaker (or user-defined) rules for completeness and consistency with the architectural description is made on the language? The concept of sequence is determined between two different descriptions

of the architecture? Suffer if the language (to provide useful operations on incomplete information) description?

ALDA and processed to provide accurate information and vehicles to describe the concept and implementation of the architecture provides the basis for modeling and analysis of systems facilitates the automation of a number of development measures, And significantly reduces the development and implementation of loopholes. Although the core language provides a number of concepts with accurate semantic modeling and mapping in the framework of the platform and performance [8, 9]. ACME does not embody the semantics of a specific architecture calculation. Most likely, ACME is based on an open semantic structure, which provides basic structural semantics, allowing specific computers or the productivity behavior associated with the ADLS architecture, using the properties of the structure. The structure of open semantic supplies a direct mapping of the language structural aspects in a logical form basing on constraints and relationships. In such context, the ACME specification is derived predicate, called its appointment. The names of the predicate properties that take the object to which the property is used as an argument and returns the name of the properties of the object. Property values that are considered primitive atoms without their own semantics [10]. Views 4.2.2: How well does ADL support a variety of perspectives that highlight different aspects / perspectives of architecture? What types of syntax are supported? Graphic, textual, ... etc .. What is the opinion of semantic care? The flow of data, flow control, in terms of processes, etc. Do ADL translate between species? In order to determine the architecture, variant stakeholders may demand variant architecture views. Customers may agree a "box" Description; Developers may require detailed models of components and connectors; Managers may require the presentation of an appropriate development process. AADL modeling language support in several forms of address and architecture analysis and performance of applications using the modeling and execution system explicit elements and reference platforms clearly defined semantics simultaneity and Interaction time properties / performance. AADL can be considered as the notation of modeling which can be supplemented by estimates of the specific objectives of a particular analysis and report modeling. These additional characters can be entered using value chain properties. Alternatively, other types of semantic modeling and processing of some of the tests can be demonstrated in UML models terms of sub-loudspeakers [8, 9]. MetaH supports two basic types of architecture: text and graphics. MetaH further distinguishes the different types of components and connectors iconically. This not only allows for the high level and detailed understanding of the building blocks [16]. The appearance of ArchJava system shows

an overview of the system. This point becomes more precise in terms of what would be the starting point for a sequential control unit operation that makes it possible to get the desired product. The systems we develop, but provided by other systems. ArchJava When a component or system is determined by interacting with external systems to connect the external components of the system are already installed adapter. The adapter will be responsible for covering the specific application. Inside view of the system logic in the box style and the line can be represented. With ArchJava possible at an early stage to describe the composition of the system only by referring to the various components used. [10].

4.2.1 Characteristics of targeted users: Domain analyst engineer, applications, systems, software manager? Required skills: domain expertise and experience of software development, experience in a programming language? AADL enables the developer to perform an analysis of compiled systems and components like security analysis, schedulability of the system, and analysis. For these analyzes, the structural changes and compromises can be evaluated by the designer [8, 9]. Software developers who are responsible for embedding architecture and embedded real-time systems and software on your target platforms. Responsible for the successful development and maintenance of critical performance systems, large-scale programs. Commercial producers of tools and engineering companies interested in providing design, analysis and generation solutions for the embedded computing community. Researchers in the universities, industry and government are looking for the architectural research platform with a direct link to the community of practitioners. [13] MetaH support for analysis, verification and manufacturing in real-time, fault-tolerant, reliable, multi-processor firmware. Language does not allow functional specifications, in addition to facilitating the entry of common names, objects and outputs. Instead, MetaH captures behavior information and connectivity information associated with real-time scheduling, fault tolerance, security and scalability of multiprocessor systems. MetaH for use in combination with other specialized tools and language library services that define the functionality of component characteristics (eg, ControlH) [15, 22]. By providing a common format for the exchange of architectural projects, the ACME architecture allows developers to easily integrate their tools with other additional tools. Tools In addition, Acme-compatible architects have a wide range of analysis and design tools available to them that architects trapped in an ADL [10].

4.2.2 modifiability of the software architecture description: How well ADL facilitates the evolution of architecture and its representation? The extent to which the ADL can be large and / or complex systems, hierarchies? AADL communication port between the wires of the partition is used to implement a variable and adjustable system design, while maintaining an efficient and predictable implementation and communication [9]. Metah source modules can be more independent of hardware and software applications on the context in which they are used. The MetaH architecture allows the system to be quickly reconfigured to adapt to changing equipment and functional requirements, without changing the modules of the source code application [18, 22]. ArchJava hierarchical software is expressed in constituent elements, which consist of several support elements connected to each other. Subcomponent is an instance of the component being invested in another component. Sub-Singleton, usually declared as a component type of finite fields. The described structures in above, represented static examples of the hierarchal architecture where the interaction was among its components. Nevertheless, some system architectures and require the installation of connections with a number of components dynamically. X25BillingCenter genuine experience that uses the approach of designing a complex system by assembling components. This experiment showed how the software architecture approach can lead to the rapid implementation of complex software systems using ArchJava [3, 4].

4.2.3 The expressive power ADL

Primitive powerful signs: Is the primitive powerful architecture at the level of the language? AADL data type is used for input ports to specify the types of conventional parameters, and to provide copies of these components. Extension component type mechanism can simulate inheritance. In particular, the types of components of the procedure may have processes and provide the data component as claimed in the proportion of said protocol class competition command. The necessary access to the copies of the existing data components provided in the desired cell type signatures [9]. Direct relaxation eliminats ACME and styles, this allows tools of ACME to translate any ACME description into a language kernel that is more primitive for direct exchange. This illustrates the simplicity of the language. As a result, ACME satisfies its secondary objectives of readability and abstraction, are not incompatible with the main purpose of exchanging descriptions of the software architecture between heterogeneous AVQs [10].

Scalability: Scalable if, in the sense of adding new applications in the language?

AADL was expandable in three respects. First, developers can define an extensible set of components in the form of standard component specifications and their implementation through the use of the extension mechanism. Second, the language can be extended with the ability to introduce new functions and widen the range of acceptable property values for existing properties. Third, AADL draft standard includes specification as a UML profile. AADL proposes the concept of organization to describe the type of component and the implementation of the library. New properties are supportd by language via extending the allowed values set, and they are connected with the existing ports, connections, and components categories by created extension sets. Restrictions can be entered via properties with row values, which means that tools for analyzing internalized constraints [9]. ACME provides a framework and a reliable and scalable infrastructure that allows builders a tool to avoid without the need to restore a standard set of infrastructure tools. In addition, the origin of Acme as a universal exchange language allows the tools developed using Acme as a native architectural presentation, to be compatible with a wide range of architecture description languages and existing tools with little or no further development effort. [10]

Readability: The extent to which ADL supports incorporate comments? The extent to which the architect's control over the presentation (eg, layout), information architecture? Adding ACME templates and styles greatly enhances readability and abstract linguistic features [10].

Variability: How well ADL are variations in application systems, which can be obtained from the architecture?

AADL components may have multiple implementations that are defined for a component type. This variability can reflect in the implementation of the various components of the sub-contents, the different modes of interaction of the components, a variety of dynamic configuration and the execution of the modal characteristics by AADL modes, as well as the realization of the property values Specific. We will examine each of them for their role in the modeling of family components and systems [13].

4.3 process-oriented characteristics

Process-oriented characteristics are the characteristics of the process associated with the use of ADL to build, test, analyze and refine the architecture and build a system of its application [1, 2].

4.3.1 Architecture Support

Creation of an architecture: editor of Justification, edition, Import tool? AADL developed by the Society of Automotive Engineers (SAE) under the auspices of the Committee of Experts, Architecture and Analysis of Design Language (AADL) was approved and published in November 2004. The ALDA is the language of text and graphics which is used to describe the hardware and software components of avionics systems (sometimes called Avionics Architecture Description Language) and the interfaces between these components [12]. Language MetaH has both textual and graphical syntax and tools for specification for viewing and editing as interchangeable in all formats. It specification Metah graphic can be automatically translated into MetaH text version or vice versa. Architecture Features of MetaH can be developed using either graphical or textual syntax, where tools can convert one form to another. In addition to editing tools, there are tools for performing various checks and tests and tools for automatically generating load images [22, 23]. SADL is a graphical language, that uses a graph of oriented type which is as nodes connected by arcs for representing the architecture of simulation, whereas nodes act the component of simulation while arc acts the interactions between them. A custom drawing is introduced for the design, it was maked with the help of the field simulation environment (DOME), which was improved in the technology center of Honeywell company to aid tools of improving and prototyping and graphic specifications. DOME assists the alter methods implementation, which are used for applying rules of the design, and creating artifacts like code and documentation. Means for describing the nonfunctional attributes are also provided by the SADL, that are associated with components of simulation, like time and frequency of execution, which can be used for planning purposes [26]. Acme project began in early 1995 to provide a common language that could be used to support the sharing of architectural descriptions between varieties of architectural design tools [12]. Specification: Support for browser search tool, refinement step tool, version control, architecture comparison? Some languages allow the generation of the system directly from the architectural specifications; This is usually one of the implementation languages of limitation. AADL supports the evolution of components in order of succession, allowing more specific components to be cleaned from the more abstract component. AADL listed design specifications or identified risks. Larger models can be generated automatically from the design database systems and organized into separate packets for the development of a version control command, and [8, 9]. MetaH allows you to specify source files, which are compatible with elements of architectural. Several problems exist with this approach, where as the assumption of a relationship between the

architectural attends and elements as a result of an I-to maybe unfounded, with no guarantee that the desired behavior will be correctly implemented by the particular source modules, or any changes in these modules in future will return to the architecture, and inversely. [16]

In the first step in determining the ArchJava architecture, all methods of each port are displayed in a single component, as all ports provide only the services provided. All the methods of each orifice is one of the components. Thus, the mapping is direct and does not require multiplexing functional characteristics. Thus, each component that is attached to the external port has a similar port. In the next step, each component must be found if the component that is causing the same problem [3, 4]. SADL is very strict in its refinement as much as architecture. SADL differs from other ADLs because it supports the distribution structure on several levels. This is called a refinement of the terminology of the system structure of a high level SADL. However, SADL can provide structural support to extend along a limited number of measurements (eg, components, connectors, configuration). SADL support for behavioral modeling is very limited [16, 26].

4.3.2 Architecture Analysis: What support is available for analyzing ADL-level information architecture to predict or project the quality of the final system?

Maintainability: capacity for correction, scalability, management? Since the model design and architecture and implementation specifications of the AADL reader, they can be stored to accommodate changes based on model-driven architecture throughout the system lifecycle. AADL also clearly identifies objects, messaging, and decomposition properties [8, 9]. MetaH was designed to ensure the rapid development and evolution of the system. One aspect of evolution is the ability to rapidly reconfigure these complex systems in real time on the new mode of hardware environment. MetaH process development system offers a low risk of rapid changes to execution. This radically changes the high risk associated with the concept of hardware development at present we use. Now, with MetaH, from the point of view of the execution environments of the material / program software can develop repeatedly during a system area and more power development using modern processors [18].

Portability: Hardware independence, software independence? AADL is very compact because the functional and non-functional requirements are isolated from the hardware. Equipments and Fasteners Simulation are fully supported and AADL of the device described in SW API [8, 9].

Reliability: tolerance errors, degradation of performance, availability? AADL use error model support as a standardized extension to support AADL simulation / reliability and fault risk analysis [9]. Metah determine the nature of the error, which describes the types of defects that can occur (eg, continuous transition), and then an error condition that the objects can be (for example, incorrigible errors did not) , And the collection of state machines, describe the state of the error object can change that events and error error propagations. Reliability modeling and analysis allows the system architect to determine the probability of failure of a fault-tolerant system that is prone to random error of future events. The set of errors, defects, and various objects reaction in the event of an error are defined by the architecture of the system. Two types of data models, simulation of user input errors can be derived by the reliability analysis, they are:

1. The model of Stochastic parallel process. It is well-defined applications among the model objects, specification and implementation of a code modules set and Metah entities.
2. The model of Markov chain. It can be represented by solving tools of the reliability model of Markov chain (such as, the file can be introduced directly into toolbars, PAVE, NASA, and SURE).

Ergonomics: comprehensibility, ease of learning, ease? Graphical support and AADL hierarchical views. The attributes of the approach and formalism of practical, easy uses, and regular, lead to learn and interact with extra approaches [9]. Language specification MetaH architecture includes as part of its review of the definition of principles used for the first coding module. These principles, along with the MetaH features language defines a set of programming tools / hardware / software and software common interface. Thus, source units may be more independent of the software context and application hardware in which they are used. MetaH supports increasing the reuse of source modules [22].

4.3.3 Use of the building

The system includes: composing integration or replacing authorities to create an executable software and a compiled system: a homogeneous distribution of distributed system processing systems of heterogeneous components of the system, written in more than one language? AADL composition of model systems and productivity software components of the hierarchical platform. The system may include a data line, a group of threads, processes, memory, processor, buses, peripherals, and subcomponents of the system. This system can request and grant access to the bus and components [8, 9] Data. MetaH allows the implementation of

bean must be defined by a set of subcomponents and the links between these subcomponents, which we will call subarchitecture. It is based on a static, as opposed to dynamic reference subcomponents for instances, interfaces, and connections. In MetaH this decision stemmed from two goals of maximizing the compilability of time and testability computability and minimizing overhead performance in real-time systems. MetaH provides distinct language, built to declare components and connections, as well as declaration compounds appear only under the (sub-architecture) specifications [15].

Query Generation Support: Is the ADL code generation component there, generation of event test code for packaging production, documentation generation? The system is modeled in AADL includes the application software displayed on the platform at runtime. Data, routines, and stream group processes together constitute the application software. They are called software components. Processor, memory, bus and peripheral devices, as well as the performance of the platform. They are called the platform components at runtime. Software components of the original model, text, virtual address spaces and concurrent threads. The programmer writes the source code either using a programming language (like C or Java, Ada 95), or using modeling languages of domain-specific like ESTEREL, SDL, Glossy, UML, and Simulink, where as the executable code is generated for it. [9] MetaH provides a simple approach of wrapping for code interaction variety sources, which involves the inheritance code from the application code generator, and a new code developed manually. Other tools can be used by the application engineers which they are a specialized for application part, and they can wrapped by the system engineers to use with MetaH and they are quickly integrated on built-in hardware code generation. The code that is used to create an executable image of Metah specifications can be divided into two steps by two tools. The compiler of MetaH converts a source code, compiles, and links directives, whereas a tool for Assembly (MakeH) uses the information that are prepared in compiler of MetaH to do compiling, linking, and probably uploads images to the hardware system, one for every CPU [22].

4.3.4 Tool for maturity, availability and support:

AADL supports the decision to open a source. SEI Open Source AADL Environment Tool (OSATE) and a commercial support tool [8]. MetaH designed to mimic applications aspects planning, reliability and security multiprocessors in real time. Currently, a set of tools includes a real-time graphical analyzer, and the Millennium Development Goals operation to add security analyzers and security analyzers. The analysis is carried out using numerical tools (math) rather than modeling the analysis,

whether the results are complete or sound, in the sense that they all involve the performance of possible behaviors [15]. ArchJava AcmeStudio and can work together to help architects and engineers to model, analyze and implement the consistent architecture throughout the software lifecycle. AcmeStudio this IDE architecture, it is written as a plug-in for IBM Framework IDE Eclipse, which supports ADL Acme [5].

5. Summary

The following tables summarize our survey.

Table 5.1: System-oriented features

ADL Attributes	AADL	MetaH	SADL	ArchJava	Acme
Architectural Styles	Set of packages of AADL that specify the architecture	categorized according to a language- specific set of classes: events, ports, subprograms, packages, processes, macros, and modes	intercommunicating using synchronization mechanisms, data transfers, or both	communication integrity, through method calls between components	Set of related templates that make up the common vocabulary of a family of systems (Reusing)
Domains	Avionics, flight, , medical devices, control equipment, robotics, space app.	Architectures in the guidance, navigation, and control (GN&C) domain	Formal refinement of Architectures across levels of detail; system simulations, multiprocessor systems	applicable to any application written in Java that is intended to run on a single JVM	Common interchange format for arch.design tools, foundation for developing new architectural design and analysis tools.

Table 5.2 Language-oriented features

ADL Attributes	AADL	MetaH	SADL	ArchJava	Acme
Language definition quality	Automation of several development activities, reduce design and implementation defects			The type system prohibits method calls that would violate arch. constraints	No computational semantics, open semantic framework basic structural semantics
Views	Timing/performance views. notations tailored	Both top-level and detailed views of composite elements; textual and graphical	Textual only	External view, internal view, Adapter to hide the specifics of the foreign system	Multiple representations allow encoding multiple views , encapsulation boundaries
Characteristics of intended users	Designers, Program managers, Commercial tool vendors , academia, industry, government, and practitioner community	Provides tools for Developers, Designers, and analyzers	used to design real-time simulation software for advanced avionics systems	Designers, Java Developers,	ADL tool Developers, Designers have broader array of analysis and design tools,

WEB OF SCIENTIST: INTERNATIONAL SCIENTIFIC RESEARCH JOURNAL
ISSN: 2776-0979, Volume 3, Issue 8, Aug., 2022

Modifiability		Port between threads provide more modifiable, Hardware independent	Allow arch. Analyzer to change hardware and functional requirements		Hierarchical , composite components, subcomponents	Hierarchical, annotation of architectural structure using properties
Expressive power	Extensibility	Extensible in three respects: add set of component specification, extend the set of valid property values, library concept	Uses an accompanying language, ControlH, for modeling algorithms in the guidance, navigation, and control domain	It is flexible enough to allow designers to use other simulation models	Inherited methods of components through references to the appropriate super-class	Solid extensible foundation for tool builder, compatible with a broad variety of existing ADLs
	Abstraction	Package category, decomposition structure The data component category supports representing data types and class abstractions	High level of abstraction	Multiple levels of abstraction	Support design with abstract components and ports allow an architect to specify and type-check arch. before impl	Templates and styles. Otherwise, Architect explicitly specifies the structure for each design element
	Readability		In-line textual specification with many connectors details; provides graphical notation	Explicit, concise textual specification	The communication analysis yielded a number of refactoring opportunities	Templates and styles greatly enhanced the readability
	Variability	Different implementations imply different contained component substructures, component interactionpatterns, dynamic configurations			components can be dynamically instantiated using the same new syntax used to create ordinary objects	Open semantic framework , basic structural semantics

Table 5.3 Process-oriented features

ADL Attributes		AADL	MetaH	SADL	ArchJava	Acme
Architecture support	Creation	Textual and graphical language	Textual and graphical Syntax (Tool)	Graphical language (Tool e.g. DoME)	Constructs, Some commercial tools even provide graphical ways	Text editor and graphical shell are provided by tools
	Refinement	Inheritance (refined from more abstract component during development, risks)	Compiler; primitive components are implemented in traditional programming language	Very rigorous in its refinement. It support structural decomposition at multiple levels	Assembling components. (refinement tree and check the internal and external views)	Multiple representations allow multiple refinement levels
Architecture Analysis	Maintainability	Hardware independent, clearly define object, messaging, properties, and decomposition	offers low risk rapid changes in the execution environment		Communication integrity. At every stage of the software lifecycle, Program understanding, and evolution	Open semantic framework basic structural semantics
	Portability	Very portable since the functional and non functional requirements are isolated from hardware.			Run on a single JVM	Templates and styles can be replaced to the more primitive core language for straightforward interchange
	Reliability	Supports an Error Model as extension to support fault/reliability modeling and hazard analysis	Supports as a tool for reliability analysis		Communication integrity. At every stage of the software lifecycle	Multiple representations allow multiple refinement levls
	Usability	Graphical and hierarchical views, formalism is simple, uniform, practical.	Multiple tools ; graphical; textual	Multiple tools ; graphical;	Applications can be translated into ArchJava with a modest amount of effort, and without excessive code bloat.	Templates and styles can be eliminated, and described to the more primitive core language for straightforward interchange

Application building	System composition	Hierarchical compositions(data, thread, thread group, process, memory, processor, bus, device, and system subcomponents)	Supported via macros; Subcomponents and connections between those subcomponents	Mappings relate an architecture to a component	Components communicate through explicit connections as well as through shared objects	Templates, and rep-maps
	Application generation support	Application software mapped to an execution platform (Processor, memory, bus, and device)	DSSA approach; compiler generates Ada code; Simple wrapper approach to interfacing code from a variety of sources	None	Run on a single JVM	None
Tool maturity	Availability	OSATE and many other commercial tools.	S/H binder, real-time scheduler analyzer, reliability analyzer, and safety/security analyzer	Graphical toolset; parser; checker for adherence of arch.	ArchJava and AcmeStudio can work.	AcmeStudio, Plan 9
	Born Date	Approved and published in Nov, 2004.	1993	1995		Acme project began in early 1995.

References

1. Paul C. Clements: “A survey of architecture description languages”. In Proceedings of International Workshop on Software Specification and Design (IWSSD), pages 16-25, 1996.
2. Ariel D. Fuxman: “A survey of architecture description languages”. Technical Report CSRG-407, Department of Computer Science, University of Toronto, 2000.
3. J. Aldrich, C. Chambers, and D. Notkin: “ArchJava: Connecting Software Architecture to Implementation”. Proc. International Conference on Software Engineering, Orlando, Florida, May 2002.
4. D. Bennouar, T. Khammaci, and A. Henni: “The Design of Complex Software with ArchJava”. Journal of computer science volume 2-11, pages: 807-814, ISSN 1549-3636, 2006.
5. B. Schmerl, and D. Garlan: “AcmeStudio: Supporting Style-Centered Architecture Development”. Proc. International Conference on Software Engineering, Edinburgh, Scotland, May 2004.
6. Jonathan Aldrich, David Garlan ,Bradley Schmerl and Tony Tseng: “Modeling and implementing software architecture with acme and archJava”, ACM Press , Pages: 156 - 157 , ISBN:1-58113-833-4, 2004.

7. J. Aldrich, C. Chambers, and D. Notkin: “Architectural Reasoning in ArchJava”. Appear in European Conference on Object Oriented Programming, Malaga, Spain, June 10-14, 2002.
8. P. H. Feiler, B. A. Lewis, S. Vestal, Member, IEEE: “The SAE Architecture Analysis & Design Language (AADL) A Standard for Engineering Performance Critical Systems”. Proceedings of the 2006 IEEE Conference on Computer Aided Control Systems Design Munich, Germany, October 4-6, 2006.
9. P. H. Feiler, B. A. Lewis, S. Vestal: “The SAE Avionics Architecture Description Language (AADL) standard: A Basis for Model-Based Architecture-Driven Embedded Systems Engineering”. In RTAS 2003 Workshop on Model-Driven Embedded Systems, May 2003.
10. David Garlan, Robert Monroe, David Wile: “Acme: An Architecture Description Interchange Language”. InProceedings of the 1997 conference of the centre for advanced studies on collaborative research, page 7, 1997.
11. D. Garlan, R. T. Monroe, and D. Wile: “Acme: Architectural Description of Component-Baed Systems”. Foundations of Component-Based Systems, G.T. Leavens and M. Sitaraman, eds., Cambridge Univ. pages: 47-68, 2000.
12. http://www.sei.cmu.edu/str/descriptions/adl_body.html.
13. http://en.wikipedia.org/wiki/Architecture_description_language
14. http://www.arch.java .org
15. S. Vestal: A Cursory Overview and Comparison of Four Architecture Description Languages. Technical Report, Honeywell Technology Center, February 1993.
16. N. Medvidovic and R. N. Talor: A Classification and Comparison Framework for Software Architecture Description Languages. IEEE Transaction on SoftwareEngineering, 26(1):70{93, January 2000.
17. http://www.htc.honeywell.com/projects/dssa/dssa_tools/dssa_tools_mh.html
18. J. H. McDuffie: Using the architecture description language MetaH for designing and prototyping an embedded spacecraft attitude control system, Digital Avionics Systems, 2001. DASC. The 20th Conference, Vol.2, Iss., Oct 2001, Pages:8E3/1-8E3/9 vol.2
19. J.W. Krueger, S. Vestal, B. Lewis: Fitting the pieces together: system/software analysis and code integration using METAH, Digital Avionics Systems Conference, 1998. Proceedings, 17th DASC. The AIAA/IEEE/SAE, Volume 1, 31 Oct.-7 Nov. 1998 Page(s):C33/1 - C33/8 vol.1
20. B. Lewis: Software portability gains realized with METAH and Ada95. In Proceedings of the 11th international workshopon Real-time Ada workshop, pages 37-46. ACM Press, 2002.

21. N. Medvidovic and D.S. Rosenblum: Domains of Concern in Software Architectures and Architecture Description Languages, Proc. USENIX Conf. Domain-Specific Languages, pp.199- 212, Oct.1997.
22. http://www.htc.honeywell.com/projects/dssa/dssa_tools/dssa_tools_mh.html
23. http://www.htc.honeywell.com/projects/dssa/dssa_tools.html
24. P. Kogut and P.C. Clements: Feature Analysis of Architecture Description Languages, Proc. Software Technology Conf. (STC '95), Apr. 1995.
25. P. Mishra; N. Dutt: “Architecture description languages for programmable embedded Systems”, Computers and Digital Techniques, IEE Proceedings- Volume 152, issue 3, 6, Pages: 285 - 297, May 2005.
26. KG Ricks, MSF Center, JM Weir, BE Wells: “SADL: Simulation Architecture Description Language”, International Journal of Computers and Their Applications, 2002 - kricks.eng.ua.edu.