|  |
| --- |
| Lair MARTES JUNIOR |
| Sys-ADL Reference Manual |
|  |
|  |
| **© Lair Martes Junior - 2010** |
| **17/06/2014** |

|  |
| --- |
| Reference manual for Sys-ADL user |

Content

[1 The Sys-ADL 7](#_Toc390808320)

[1.1 Sys-ADL Elements 8](#_Toc390808321)

[1.1.1 Concept 9](#_Toc390808322)

[1.1.2 Channel 9](#_Toc390808323)

[1.1.3 Format 11](#_Toc390808324)

[1.1.4 Layer 12](#_Toc390808325)

[1.1.5 Node 14](#_Toc390808326)

[1.1.6 Objective 16](#_Toc390808327)

[1.1.7 Quality 19](#_Toc390808328)

[1.1.8 Receiver 20](#_Toc390808329)

[1.1.9 Role 21](#_Toc390808330)

[1.1.10 Sender 23](#_Toc390808331)

[1.1.11 Stakeholder 25](#_Toc390808332)

[1.1.12 System 27](#_Toc390808333)

[1.1.13 Transition 29](#_Toc390808334)

[1.2 Sys-ADL Element Data 31](#_Toc390808335)

[1.2.1 Concept 32](#_Toc390808336)

[1.2.2 Mandatory fields 32](#_Toc390808337)

[1.2.3 Additional Fields 33](#_Toc390808338)

[1.2.4 Field graphical representation 33](#_Toc390808339)

[1.2.5 Stereotype 34](#_Toc390808340)

[1.3 Sys-ADL Relations 36](#_Toc390808341)

[1.3.1 Concept 37](#_Toc390808342)

[1.3.2 Connection 37](#_Toc390808343)

[1.3.3 Composed-by 38](#_Toc390808344)

[1.3.4 Depends-on 42](#_Toc390808345)

[1.3.5 Is-a 43](#_Toc390808346)

[1.3.6 Represents 45](#_Toc390808347)

[1.4 Sys-ADL Structures 48](#_Toc390808348)

[1.4.1 Concept 49](#_Toc390808349)

[1.4.2 Concern 49](#_Toc390808350)

[1.4.3 Deposit 51](#_Toc390808351)

[1.4.4 Device 53](#_Toc390808352)

[1.4.5 Host 54](#_Toc390808353)

[1.4.6 Interface 55](#_Toc390808354)

[1.4.7 Port 56](#_Toc390808355)

[1.4.8 Protocol 57](#_Toc390808356)

[1.4.9 Requirement 59](#_Toc390808357)

[1.4.10 Responsibility 59](#_Toc390808358)

[1.5 Sys-ADL Communications 62](#_Toc390808359)

[1.5.1 Concept 63](#_Toc390808360)

[1.5.2 Conversation 63](#_Toc390808361)

[1.5.3 Installation 65](#_Toc390808362)

[1.5.4 Network 66](#_Toc390808363)

Index of figures

[1.1.2‑1 – Sys-ADL Symbol for Channel 9](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808261)

[1.1.2‑2 – Network between Server A and Server B 10](#_Toc390808262)

[1.1.2‑3 – Conversation between systems 10](#_Toc390808263)

[1.1.2‑4 – Scanner installed in a desktop 11](#_Toc390808264)

[1.1.2‑5 – System receiving information from Pen Drive 11](#_Toc390808265)

[1.1.3‑1 – Sys-ADL Format symbol 12](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808266)

[1.1.3‑2 – System publishing information 12](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808267)

[1.1.4‑1 – Sys-ADL Layer symbol 12](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808268)

[1.1.4‑2 – Three layer system architecture 13](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808269)

[1.1.4‑3 – Logical division of system components 14](#_Toc390808270)

[1.1.5‑1 – Sys-ADL Node symbol 14](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808271)

[1.1.5‑2 – Webserver has an HTTP port avaiable 15](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808272)

[1.1.5‑3 – Nodes connected through HTTP port 15](#_Toc390808273)

[1.1.5‑4 – Device example 15](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808274)

[1.1.5‑5 – Scanner installed in a desktop 16](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808275)

[1.1.5‑6 – User desktop hosts Web Browser 16](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808276)

[1.1.6‑1 – Sys-ADL Objective symbol 17](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808277)

[1.1.6‑2 – Objective sample 17](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808278)

[1.1.6‑3 – Objective map 18](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808279)

[1.1.6‑4 – Bill payment simplified example 18](#_Toc390808280)

[1.1.8‑1 – Sys-ADL Receiver symbol 20](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808281)

[1.1.8‑2 – Web Server A is a server in a HTTP connection 20](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808282)

[1.1.8‑3 – Network between Webserver and Desktop 21](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808283)

[1.1.8‑4 - Protocol sample 21](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808284)

[1.1.8‑5 – Web Server is able to send HTML messages to Browser through HTTP connection 21](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808285)

[1.1.10‑1 – Sys-ADL Sender symbol 24](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808286)

[1.1.10‑2 – Device sample 24](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808287)

[1.1.10‑3 – Installation communication sample 24](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808288)

[1.1.10‑4 – System A and B share same Server and they interchange information without network 25](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808289)

[1.1.11‑1 – Sys-ADL Stakeholder symbol 26](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808290)

[1.1.11‑2 – Acme is concerned on System A 26](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808291)

[1.1.11‑3 – System Administrator is responsible to firewall set up 27](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808292)

[1.1.11‑4 – Non-technical concern example 27](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808293)

[1.1.12‑1 – Sys-ADL System symbol 28](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808294)

[1.1.12‑2 – User changing information scenario 28](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808295)

[1.1.13‑1 – Sys-ADL Transition symbol 29](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808296)

[1.2.2‑1 – Short description of each mandatory field 32](#_Toc390808297)

[1.2.3‑1 – Data validation information required to validate data provided by Sys-ADL user 33](#_Toc390808298)

[1.2.4‑1 – Showing element data in design time 34](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808299)

[1.2.5‑1 – Different stereotypes differentiates different system types 34](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808300)

[1.2.5‑2 – Fields in no stereotype stakeholder… 35](#_Toc390808301)

[1.2.5‑3 – … are different from <<user>> stereotype 35](#_Toc390808302)

[1.3.2‑1 – Sys-ADL Connection symbol 37](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808303)

[1.3.2‑2 – Interface structure with Connection 38](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808304)

[1.3.2‑3 – Port structure with Connection 38](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808305)

[1.3.3‑1 – Sys-ADL Composed-by symbol 38](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808306)

[1.3.3‑2 – Explicit dependency between System B 39](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808307)

[1.3.3‑3 - Implicit dependency between System B 40](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808308)

[1.3.3‑4 – Composed-by relation creating Objective Maps 40](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808309)

[1.3.3‑5 – Server infrastructure using Nodes 41](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808310)

[1.3.3‑6 – Composed-by relation using Layers 41](file:///C:\Users\lj71053\Documents\sysadl\manuais\sysadl-reference-manual.docx#_Toc390808311)

[1.4.9‑1 – Partial representation of a Requirement structure 59](#_Toc390808312)

[1.4.9‑2 - Requirement use example 59](#_Toc390808313)

Stylistic conventions

|  |  |
| --- | --- |
| **Convention** | **Example** |
| Sys-ADL element, structure or communication is written using italic and bold character. | ***System*** element.  ***Stakeholder*** element.  ***Deposit*** structure  ***Requirement*** structure  ***Network*** communication  ***Conversation*** communication |
| Sys-ADL instance is written using courier new character. | ***System*** element labeled Web Application.  ***Stakeholder*** element called Operator.  ***Channel*** labeled send file to provider. |
| Element field names are written using courier new and bold character. | **Element Namespace.**  **Element Stereotype.**  **Element Id** |

# The Sys-ADL

## Sys-ADL Elements

### Concept

System architecture is composed by elements that exist separated. It has systems, files, integration tools, stakeholders, computers and processes that appear in system use and definition.

Sys-ADL Element is the smallest piece of system architecture. It is the fundamental brick that we connect to each other to create system design.

Sys-ADL forecast twelve fundamental bricks: ***Channel***, ***Format***, ***Layer***, ***Node***, ***Objective***, ***Quality***, ***Receiver***, ***Role***, ***Sender***, ***Stakeholder***, ***System*** and ***Transition***. They are fully presented in this section.

### Channel

Modern system architecture has no structures that can work alone. Many times we must to connect servers through a network or make two system change interfaces. Although it is possible to create a system that connects to anyone this would be a boring system because, from an external view, this system would do nothing. Talking about servers, in nowadays distributed architecture is almost impossible exist a computer that is not connected to another. Show such communications is the role of ***Channel*** element.

***Channel*** element represents the communication between structures of the system architecture such communications like systems to systems, servers to server and devices to computers. ***Channel*** is used in following cases:

* ***Conversation***: Show that an interface is send to another system;
* ***Installation***: Show that a device is installed in a computer;
* ***Network***: Show that a computer is connected to another computer through a network.

Each case shows a situation where we must to connect two structures.

The symbol that represents ***Channel*** in Sys-ADL is a straight line from one structure to another as shown in figure 1.1.2.1.

|  |
| --- |
|  |
| channel.png  1.1.2‑1 – Sys-ADL Symbol for Channel |
|  |

In this section are some examples that will show the use of ***Channel*** element. The goals of these examples are not to give a detailed idea of each communication type because communications in Sys-ADL are explained detailed in section 1.5.

A common scenario that is found in system architecture is to represent the network infrastructure. We have servers, which are represented with Node symbol, which we can connect to represent such network topology. Note that to connect two servers it is necessary to set up a logical server port in host server and this port is represented using Port structure.

Figure 1.1.2.2 shows the network between Server A and Server B. Communication is unidirectional because there is only one ***Port*** Structure represented by Receiver FTP connected to Server B. Server B is our server and Server A is our client.

|  |
| --- |
|  |
| ObjExemplosManualReferenciaChannelNetwork.png  1.1.2‑2 – Network between Server A and Server B |
|  |

***Channel*** has been used in this case to show a network connection between servers.

Other example of use of ***Channel*** element, figure out a system called System A that creates a file in plain text format with some information that it processes. Now, we introduce a system that will receive this interface and we will call it System B and it can receive this file using File Transfer Protocol (FTP). ***Channel*** element describes the conversation between these systems. Figure 1.1.2.3 shows this situation

|  |
| --- |
|  |
| ObjExemplosManualReferenciaConversation.png  1.1.2‑3 – Conversation between systems |
|  |

Conversation shown above is very common in a distributed environment where there are systems that process information in several servers that are connected through a network and protocols.

Many times, we have devices such pen drives, CDs, smart cards, scanners and others that we connect in a computer and such devices have important information for a process and we must represent this situation in our system architecture diagram.

Our scenario for example is Desktop A that has a scanner connected to it using Universal Serial Bus (USB) port of this desktop. This scenario is showed in Figure 1.1.2.4.

|  |
| --- |
|  |
| ObjExemplosManualReferenciaChanneInstallation.png  1.1.2‑4 – Scanner installed in a desktop |
|  |

Every time that we have a device that must be connected to a computer or other kind of ***Node*** element, ***Channel*** element is used to represent the physical installation of this device.

Our last example is a conversation that does not use a network. In this case, we have a pen drive, represented by ***System*** element Pen Drive, that has a file in plain text format with some information and this data must be transferred to System C through USB device. Diagram of this case can be seen in Figure 1.1.2.5.

|  |
| --- |
|  |
| ObjExemplosManualReferenciaConversationNoNetwork.png  1.1.2‑5 – System receiving information from Pen Drive |
|  |

This group of conversation is common, although is a bit hard to maintain it. We have situation where data is copied without network use in many cases such a user that copy files manually or several tables that are copied in night batch process to other servers to increase performance.

***Channel*** is the element that is fundamental for communication representation in Sys-ADL and aids system architect to show the dynamic of system architecture.

### Format

Every system processes information and system’s architect planned to publish the result of such process for another system. Internal information interchange, for instance, program calling, database updating and others are not, in most cases, an issue from system architecture point of view. Nevertheless, when it is necessary send information for external systems, it is required to establish a format that allow other system receive this information because they was written in different languages or are hosted in different servers.

A classic example of the need of format definition is a very common integration between COBOL applications hosted in mainframes and Java applications hosted in Windows™ or Unix® servers. Besides it is very difficult such applications share the same memory space sharing data through program calling, COBOL application has a large tradition using plain text based communication to share data between processes due an inheritance form punch cards and Java applications has a lot of standards APIs for XML parsing because this platform was developed during SOA and Internet era and Java has almost any component to read plain text formatted streams. Anyway, these platforms must talk to each other and part of this conversation is performed defining the format whereby the information will be published.

The symbol that represents ***Format*** element in Sys-ADL is a small circle as shown in figure 1.1.3.1.

|  |
| --- |
|  |
| format.png  1.1.3‑1 – Sys-ADL Format symbol |
|  |

As the same way a system that does not publish its processed information, a file that is not published is as uninteresting as such system. Interface structure is how Sys-ADL represents file publishing. For example, figure out a system called System A that publishes a plain text file as shown in figure 1.1.3.2.

|  |
| --- |
|  |
| ObjExemplosManualReferenciaInterface.png  1.1.3‑2 – System publishing information |
|  |

System A processes internal information and publishes the result of this processing in a plain text file represented by ***Format*** element TXT. Interfaces are explained in details in section 1.4.6.

Systems create a file to publish information it processed. Sys-ADL represents a file created by a system using ***Format*** element.

### Layer

The quote “Divide to conquest” assigned to René Descartes is applicable in a lot of situations. In system architecture design we have sometimes a reasonable quantity of ungrouped elements causing a feeling that we are in front of a messed draw. It is a good practice to organize our elements and structures to show the role of each one in system solution.

***Layer*** is the Sys-ADL element that divides a system diagram in smaller groups allowing system designer show how elements are organized in a solution. ***Layer*** element has another interesting feature in Sys-ADL because it has a ***Composed-by*** relation with ***System*** and ***Node*** elements that are visually completely inside of ***Layer*** element. ***Relations*** are subject of section 1.3.

The symbol that represents ***Layer*** element in Sys-ADL is a square with rounded corners as show in figure 1.1.4.1.

|  |
| --- |
| 1.1.4‑1 – Sys-ADL Layer symbol |
|  |
|  |

An example of ***Layer*** element use is representing a classical architecture with three layers. Software architect plans a presentation layer, business logic layer and data layer with different servers for each one. Layer element is used to show such division. In figure 1.1.4.2 are shown a very simplified example of architecture with three layers.

|  |
| --- |
|  |
| ObjExemplosManualReferenciaLayer.png  1.1.4‑2 – Three layer system architecture |
|  |

All elements are owned by a logical layer of this architecture and a ***Composed-by*** relation is created between layer and inside elements. ***Composed-by*** relation is section 1.3.3 subject.

Other common logical grouping found in system architecture is business role. Solution architect is divided in business group and system is organized by them, as shown in figure 1.1.4.3.

|  |
| --- |
|  |
| ObjLayerBusinessExample.png  1.1.4‑3 – Logical division of system components |
|  |

Architect decided to organize system structure in business and assign related ***System*** elements to each layer.

Some system architecture are better explained dividing elements of a diagram into related areas helping deduce much information about a Sys-ADL element depending on which layer it is. System architect elements are grouped through Layer element.

### Node

Systems need to run processes or maintain data generated by internal system process or provided by other system or user and, to reach such goals, it is required a device that is capable to process or storage information.

Nowadays system infrastructure is composed by a lot of servers from several platforms such as Unix®, OS/360™ and Windows™. User is able to access its information in different kinds of devices such as desktops, laptops, other portable devices, mobile phones. Finally, system gets used to iterate with input and output devices such as scanners, printers and others.

***Node*** is the Sys-ADL element that represents any required hardware in system architecture solution. Servers, desktops, laptops, mobile phones, flash cards, printers, scanners and any other hardware that has processing or storage capacity is represented by ***Node*** element.

The symbol that represents ***Node*** element in Sys-ADL is a tridimensional box as show in figure 1.1.5.1.

|  |
| --- |
|  |
| node.png  1.1.5‑1 – Sys-ADL Node symbol |
|  |

A ***Node*** element is part of a greater topology composed with other ***Nodes*** elements that are connected through a network and for a computer be able to reach another it is necessary to set up a logical port in destiny node.

***Port*** structure represents a logical port, such as HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), ODBC (Open Database Connectivity) or any other logical port which able two nodes be connected. Port structure is assembled connecting a Receiver element to a ***Node*** element. ***Receiver*** element is detailed section 1.1.7, ***Connection*** relation is explained in section 1.3.2.

An example is a server Web Server A which has a HTTP port configured in it. This example is shown in Figure 1.1.5.2.

|  |
| --- |
|  |
| ObjExamplesReferenceManualProtocol.png  1.1.5‑2 – Webserver has an HTTP port avaiable |
|  |

Web Server showed in figure 1.1.5.2 is accessed by a desktop through HTTP port through a ***Channel*** element creating a ***Network*** communication as shown in figure 1.1.5.3. ***Network*** communication is detailed in section 1.5.4.

|  |
| --- |
|  |
| ObjExamplesNetwork.png  1.1.5‑3 – Nodes connected through HTTP port |
|  |

Another structure that is created with ***Node*** element is ***Device*** structure that is formed when ***Node*** element is connected to a ***Sender*** element representing a ***Node*** element that is accessed using a physical port such as Universal Serial Bus (USB), floppy disk drivers, scanners, printers and other. An example of ***Device*** structure is shown in figure 1.1.5.4 and this structure is detailed section 1.4.4.

|  |
| --- |
|  |
| ObjExamplesDevice.png  1.1.5‑4 – Device example |
|  |

***Device*** structure is part of another Sys-ADL communication with ***Node*** element. ***Installation*** communication, which is represented in figure 1.1.5.5, is formed with ***Device*** structure and a ***Node*** element, representing a scanner connected to a desktop through a USB port.

|  |
| --- |
|  |
| ObjExamplesInstallation.png  1.1.5‑5 – Scanner installed in a desktop |

Other common structure where ***Node*** element is enrolled is ***Host*** structure. Systems require to be hosted in a computer or other kind of device such as mobile phone, for instance. ***Node*** element is connected to a ***System*** element representing ***Host*** structure that shows a system hosted in a node. An example of ***Host*** structure is shown in figure 1.1.5.6 where Web Browser is running in User Desktop. ***Host*** structure is detailed in section 1.4.5.

|  |
| --- |
|  |
| ObjExamplesHost.png  1.1.5‑6 – User desktop hosts Web Browser |
|  |

Sys-ADL ***Node*** element represents any computational device that participate in a system architecture and is able to be connected to ***Receivers***, ***Senders*** and ***Systems*** representing structures that are accessed by other structures in system using Sys-ADL communications.

### Objective

Create a system infrastructure requires investments in human resources, time and money and it is expected that goals planned by its sponsor be reached, i.e., servers, systems, protocols, layers, stakeholders and other system architecture elements, structures and communications will work together to reach an objective.

***Objective*** is the Sys-ADL element that represents a goal which a system infrastructure set intents to reach and binds such system infrastructure represented in a diagram to its goals. ***Objective*** is not a physical architecture element but, for the architect, it the most important element in Sys-ADL universe because it shows the objectives of a system infrastructure and all elements represented in a Sys-ADL diagram are bound to this objective.

It is very important to note that a Sys-ADL diagram is not valid until it has at least one ***Objective*** element on it.

The symbol that represents ***Objective*** element in Sys-ADL is an ellipse as shown in figure 1.1.6.1.

|  |
| --- |
|  |
| objective.png  1.1.6‑1 – Sys-ADL Objective symbol |
|  |

***Objective*** element is used in a Sys-ADL diagram putting it in any visible part of a diagram. Figure 1.1.6.2 shows an example of ***Objective*** element use.

|  |
| --- |
|  |
| ObjExamplesObjective.png  1.1.6‑2 – Objective sample |
|  |

Diagram in figure 1.1.6-2 shows a reference diagram representing application architecture. Elements HTTP, HTML, Browser, Web Server A and Customer represented in diagram are now bound to Application Reference Architecture.

***Objectives*** element can be related to others ***Objectives*** elements using ***Composed-by*** relation creating an objective map that helps system architect browse complex structures. In figure 1.1.6-3 is shown an example of an objective map. ***Composed-by*** relation is detailed in section 1.3.3.

|  |
| --- |
|  |
| ObjExamplesObjectiveMap.png  1.1.6‑3 – Objective map |
|  |

Example in figure 1.1.6.3 shows a start objective called Company Bank Inc, representing a business objective that has business segments such Cash Management, Securities and Financial Investments and each one has its own business sub segments. Now, system architect has a map to guide him through solution system and infrastructure and is able to understand the business behind systems and infrastructures, because such objectives are put in respective Sys-ADL system diagrams and elements in each diagram are related to the big picture. For example, regard the system diagram in figure 1.1.6-4 created to explain element relations for Bill Payments ***Objective*** element, represented in figure 1.1.6-3.

|  |
| --- |
|  |
| ObjBillPayments.png  1.1.6‑4 – Bill payment simplified example |
|  |

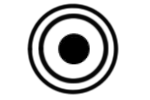
Figure 1.1.6.4 has a set of elements such as Customer ***Stakeholder*** element, Internet Banking element, messages and other relations that are related to Bill Payments. Using ***Composed-by*** relation all these elements are related to Payments, Cash Management and, finally, Company Bank Inc. ***Objective*** elements.

***Objective*** is the glue that binds all system architecture solution. ***Objective*** provides an answer for why an infrastructure exists and a system conversation happens and helps project managers, technical staff and system architects to understand its environment.

### Quality

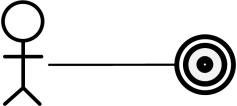
One of main roles of a software architect is to document non-functional requirements of a system. Based on business drivers, it is necessary to identify attributes that will help the system be a successful case.

***Quality*** element represents any non-functional requirement of a system. The symbol that represents ***Quality*** element is a target as shown in figure 1.1.7.1.



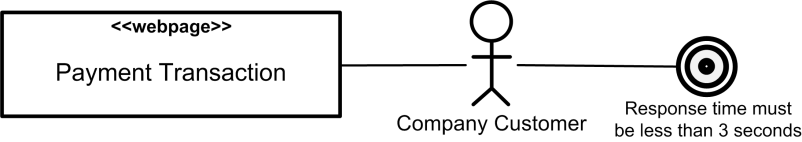
1.1.7‑1 Sys-ADL Quality symbol

A ***Quality*** element can be connected only to a ***Stakeholder*** element as shown in Figure 1.1.7.2.



1.1.7‑2 - Quality only can be connected to Stakeholder

Connection relation shown in Figure 1.1.7.2 is only valid if connected to a ***Concern*** structure and this connection creates another structure called ***Requirement***. This relation indicates which element of the architecture will be affected by a requirement and the ***Stakeholder*** element that is requiring it. Figure 1.1.7.3 shows an example of ***Requirement*** structure.



1.1.7‑3 - Requirement structure example

In figure 1.1.7-3, the webpage Payment Transaction must attend a quality attribute regarding performance. In this case, this web application must respond any request from Company Customer within 3 seconds.

***Quality*** element is used to document non-functional requirements of a system. Connection with ***Stakeholder*** element helps to know who is impacted by this quality attribute. ***Concern*** structure allows identifying which architecture element is affected by requirement.

### Receiver

When it is necessary to connect two computers a network infrastructure is created to connect them physically. In simple cases, such connections are made using a network cable that is plugged into a network board installed in each computer. But a network wire is not enough to make these computers talk each one. It is required to set up a logical port in the operational system of the computer to allow this computer receive external requests.

Systems have a similar problem. A system needs to send a message to other and a protocol is required for message reach its destiny.

***Receiver*** element represents logical port that is configured in a node to allow other computer being connected in it through a network. ***Receiver*** element also represents the protocol which a system receive an interface from other system.

The symbol that represents ***Receiver*** element in Sys-ADL is a trapezium as shown in figure 1.1.7.1.

|  |
| --- |
|  |
| receiver.png  1.1.8‑1 – Sys-ADL Receiver symbol |
|  |

By the time a ***Receiver*** element is connected to a ***Node*** element a ***Port*** structure is created and this means that such ***Node*** is a server in a connection. Figure 1.1.7.2 shows an example of ***Port*** structure, which is a subject of section 1.4.7.

|  |
| --- |
|  |
| ObjExamplesPort.png  1.1.8‑2 – Web Server A is a server in a HTTP connection |
|  |

Figure 1.1.7.2 has Sys-ADL element Web Server A, represented by ***Node*** element, which is connected to a ***Receiver*** element through a ***Connection*** relation, representing the HTTP protocol that is configured in this server. ***Connection*** relation is section 1.3.2 subject. Now, this structure is able to communicate to further ***Node*** creating a ***Network*** communication as shown in figure 1.1.7.3 where a desktop is connected to a web server. ***Network*** communication is detailed in section 1.5.4.

|  |
| --- |
|  |
| ObjExamplesCommunicationDesktop.png  1.1.8‑3 – Network between Webserver and Desktop |

Other structure created using ***Connection*** relation is ***Protocol*** structure which is formed connecting a ***System*** and a ***Receiver*** element as shown in figure 1.1.7.4.

|  |
| --- |
|  |
| ObjExamplesProtocol.png  1.1.8‑4 - Protocol sample |
|  |

***Protocol*** structure showed in figure 1.1.7.4 is assembled by ***System*** element Browser and ***Receiver*** element HTTP protocol. This means that Browser is able to receive interfaces through HTTP protocol. A ***Conversation*** communication is shown in figure 1.1.7.5 where Web Server sends an HTML file to Browser. ***Conversation*** communication is detailed in section 1.5.2.

|  |
| --- |
|  |
| ObjExamplesConversationBrowser.png  1.1.8‑5 – Web Server is able to send HTML messages to Browser through HTTP connection |

***Receiver*** element represents the gateway through ***Nodes*** or ***Systems*** elements communicate with external world.

### Role

In system development or use, there are programmers, users, external entities and others that have some activity on components of the solution.

***Role*** element represents any activity that a ***Stakeholder*** executes on system. The symbol that represents this element is a rectangle with curved sides as shown in Figure 1.1.9.1.

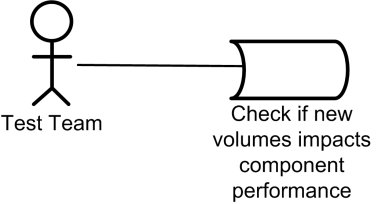


1.1.9‑1 - Sys-ADL Role symbol

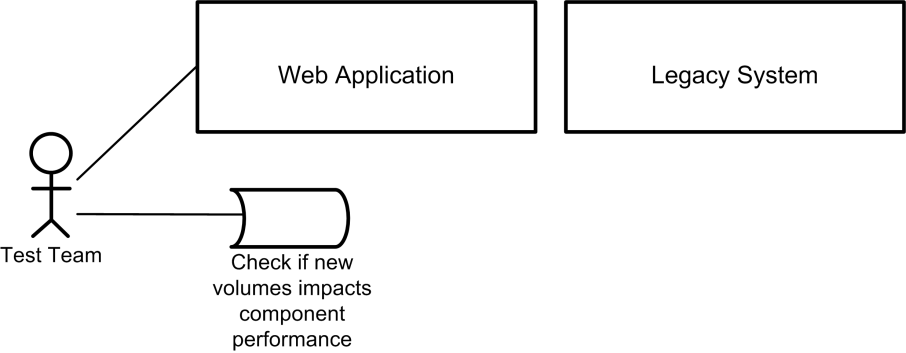
When ***Role*** is connected to a ***Stakeholder*** element, as shown in Figure 1.1.9.2, a structure called ***Responsibility*** is created. ***Responsibility*** is subject of section 1.4.10.

Figure 1.1.9.2 has Sys-ADL element Test Team, represented by ***Stakeholder*** element. Test Team is connected to role Check if new volumes impacts component performance. This means that the responsible for test volumes in component is the Test Team.

Using Sys-ADL, it is possible to connect ***Responsibility*** and ***Concern*** structures, indicating in which components this responsibility is applicable as seen in Figure 1.1.9.3.



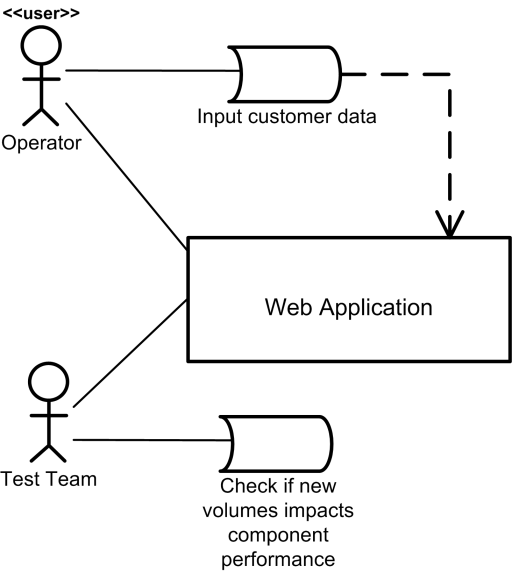
1.1.9‑2 - Test Team is responsible for component testing



1.1.9‑3 - Indicating which component will be tested

In Figure 1.1.9.3, the Test Team will test only Web Application system and not Legacy System, both represented by ***System*** element. ***System*** element is subject of section 1.1.12.

Sometimes, could be necessary a system or component to execute an activity. For example, to input data a user requires an application. This dependency can be expressed using ***Depends on*** relation as shown in Figure 1.1.9.4.



1.1.9‑4 - Including dependencies for roles

In Figure 1.1.9.4, ***Depends on*** relation indicates that to execute its role, element ***Stakeholder*** Operator requires the element ***System*** Web Application, or, in other words, to element ***Role*** Input customer data be accomplished element ***System*** Web Application is required.

A remarkable characteristic of ***Role*** element is that the ***Responsibility*** structure does not require a ***Concern*** structure and ***Stakeholders*** elements can be associated to ***Role*** elements freely and is not necessary to indicate which element is affected by the ***Role***. This is different of ***Requirement*** structure that is subject of section 1.4.9. Consequences of ***Responsibility*** structure characteristics will be discussed deeply in section 1.4.10.

### Sender

Since telegraph invention people have been preferred send message using cable instead of couriers because, for large distances, send messages using telegraph was faster and more reliable. We have advanced a lot since telegraph technology. We created Ethernet, token ring, coaxial cables, Transfer Communication Protocol and reached Internet era and today every single computer in world is able to be connected.

When such evolution is in network technology is considered, communication between systems that has no network sounds a bit weird. Nevertheless, in a deeper sight is possible to realize that we have some issues in network communication that must be considered in our analysis.

A communication network infrastructure has costs in building and maintenance. Although such costs pay for themselves as they are used, network communication requires message processing for transportation, interpretation and security checking. These factors impose a delay in communication that is not acceptable depending on situation. Moreover, if a message format changes a lot it is very difficult update systems to hold new format although XML and Enterprise Service Bus come to make such integration easier. Finally, some integration is made by users that manipulate data in a file and copies result in a shared directory and this file is read by another user or system.

***Sender*** element represents a communication that is made without a network infrastructure.

The symbol that represents ***Sender*** element in Sys-ADL is a rectangle with a tip on the right side as show in figure 1.1.8.1.

|  |
| --- |
|  |
| sender.png  1.1.10‑1 – Sys-ADL Sender symbol |
|  |

There are two structures that Sys-ADL is able to create: ***Device*** and ***Deposit***.

***Device*** structure is built when a ***Node*** element that is connected to another ***Node*** element through a physical port in target element. Physical port is represented by ***Sender*** element which is connected to ***Node*** element creating ***Device*** structure as shown in figure 1.1.8.2. ***Device*** element is section 1.4.4 subject.

|  |
| --- |
|  |
| ObjExamplesDevice.png  1.1.10‑2 – Device sample |
|  |

In figure 1.1.8.2 Sys-ADL represents a scanner device built connecting a ***Node*** element Scanner to a ***Sender*** element USB. ***Connection*** relation is used to connect both elements. ***Connection*** relation is detailed in section 1.3.2.

Using ***Installation*** communication is possible to represent that the scanner is installed in a ***Node*** element as shown in figure 1.1.8.3. Details about ***Installation*** communication are given in section 1.5.3.

|  |
| --- |
|  |
| ObjExemplosManualReferenciaChanneInstallation.png  1.1.10‑3 – Installation communication sample |
|  |

In figure 1.1.8.3 a scanner, built with ***Node*** element Scanner connected to ***Sender*** element USB, is installed into a desktop represented with a ***Node*** element Desktop A. Such installation is represented in Sys-ADL using an ***Installation*** communication. ***Installation*** communication is subject of section 1.5.3.

There are communications between systems that has no network either and such communication is represented with ***Deposit*** structure that is built connecting an ***Interface*** structure to a ***Sender*** element using a ***Connection*** relation. ***Deposit*** and ***Interface*** structures are subject of section 1.4.3 and section 1.4.6, respectively.

A common example is a system that needs information from other system and both systems are installed in the same database server, so is not necessary to use a network to transport information from one system to another. Picture a system called “System A” that needs information of a database from other “System B”, which allow “System A” access its information through a database view and query executing using Structured Query Language (SQL) and a Java Database Connectivity (JDBC) connection. This scenario is described using Sys-ADL in figure 1.1.8.4.

|  |
| --- |
|  |
| ObjExemplosManualReferenciaDepositCommunication.png  1.1.10‑4 – System A and B share same Server and they interchange information without network |
|  |

In figure 1.1.8.4 a ***System*** element System A receives information in SQL format, represented with ***Format*** element SQL, forming an ***Interface*** structure with ***System*** element System B. JDBC Communication protocol is represented with Sender element JDBC, which is connected to ***Interface*** structure. Finally, both systems are connected to a ***Node*** element building two ***Host*** structures meaning that both systems are hosted in same server, represented with ***Node*** element Database Server. ***Host*** structure is detailed in section 1.4.5.

***Sender*** element represents the way that structures make communication. There is a large set of situations where this communication is necessary and ***Sender*** element is able to represent them.

### Stakeholder

A system infrastructure is build by teams composed by professionals that act in all elements of physical and logical system structures. There are technical teams like system administrators that set up firewalls to allow external connections from other servers, developers construct systems, and database administrators define data design and many other technical people works in system project. But there are non-technical people that is considered in system development such users and sponsors which requirements and expects must be met.

***Stakeholder*** Sys-ADL element represents all persons or teams that must be considered in system architecture design.

***Stakeholder*** element is represented in Sys-ADL with a stickman as shown in figure 1.1.9.1.

|  |
| --- |
|  |
| stakeholder.png  1.1.11‑1 – Sys-ADL Stakeholder symbol |
|  |

A stakeholder has interests or roles in system architecture elements and such situation is represented using ***Concern*** structure which is created connecting a stakeholder in another Sys-ADL element through a ***Connection*** relation. ***Connection*** relation is subject of section 1.3.2 and ***Concern*** structure is detailed in section 1.4.2.

All stakeholders which have a concern in a Sys-ADL element are connected to such element. For example, picture a system called “System A” that will be developed by a software factory called “Acme”. This relation between “Acme” and “System A” is described with a ***Concern*** structure as shown in figure 1.1.9.2. “Acme” software factory is represented by Stakeholder Sys-ADL element Acme and its role has put in element between parentheses meaning that Acme is “System A” developer.

|  |
| --- |
|  |
| ObjExemplosManualReferenciaConcernSystem.png  1.1.11‑2 – Acme is concerned on System A |
|  |

Other example of ***Concern*** structure use is when a system administrator must to grant access to other computer through a system port as in figure 1.1.9.3.

|  |
| --- |
|  |
| ObjExemplosManualReferenciaConcernPort.png  1.1.11‑3 – System Administrator is responsible to firewall set up |
|  |

In the example in figure 1.1.9.3, Server A must be connected to Server B through a FTP port which will be configured by System Administrator, represented in Sys-ADL by Stakeholder element.

And, finally, an example that is not about technical. A business area that is the sponsor of a system development for attend a business case. Figure 1.1.9.4 shows Tax Payments being sponsored by Cash Management Area.

|  |
| --- |
|  |
| ObjExemplosManualReferenciaConcernObjective.png  1.1.11‑4 – Non-technical concern example |
|  |

***Concern*** structure helps system architect to map all people or teams that have participation in system development and generates information about teams that is enrolled in system financing, development, deployment, using, maintenance, sponsoring and other roles in system architecture.

### System

In classical description, a system is a set of elements which works together to reach a common objective. The question is: how large can a system be or, saying otherwise, how many components a system must have to be considered a system? In my own experience, I have already seen thousands of code pieces be considered only one system and I have been seen only one COBOL procedure with less than one hundred lines be considered a system. Ultimately, it is possible to measure how large or complex a system is using metrics such Halstead Complexity or Cyclomatic Complexity but how many lines of code it must necessary to a such code be a system is a personal choice.

Other question in is why only a set of code should be considered a system? Picture a file that was created yesterday and have been put in a directory in file system and a user opens the file and note that it is a plain text comma separated file with three information columns in it. In the next day, the same user opens the same file and realizes that there are five information columns and is clearly that a system have inserted new information in the file. But, it was not necessarily an automated task composed by a piece of code that changes the file. It may be that during the earlier night other user open the file, inserted the two new columns and save the modified file in the same directory and with the same format. The point is: only observing the file, it is impossible to know if a user or a piece of code has included new information on in.

Regarding questions above, Sys-ADL defines a system as a complete and self contained entity that is able to process and publish information, no matter how large or short it is or how it does such processing and publishing.

***System*** element is represented is Sys-ADL with a rectangle as shown in figure 1.1.11.1.

|  |
| --- |
|  |
| system.png  1.1.12‑1 – Sys-ADL System symbol |
|  |

As example, let’s detail the scenario which we have described earlier. A system called “System A” exports a file in Comma Separated Value (CSV) format and copies it to a directory and this file is captured by a user that uses a spreadsheet to include information captured from an Internet site and, finally, user send modified file to a system called “System B” in same CSV format. This scenario is represented in Sys-ADL in figure 1.1.11.2.

|  |
| --- |
|  |
| ObjExemplosManualReferenciaSystemConversation.png  1.1.12‑2 – User changing information scenario |
|  |

***Conversation*** represented in figure 1.1.11.2 is “System A” copying a CSV interface which is captured in file system that is represented in Sys-ADL diagram by ***Sender*** element FISYS, making a ***Conversation*** communication with ***System*** element User for user manually include public information from a site. User, that is has stereotyped ecosystem, makes a ***Conversation*** communication with Information Site file intending to get public information that user queries from a site represented in diagram by a ***System*** element. User sees information in a screen through a web application, represented by ***Sender*** element WEB. Using ***System*** element Spreadsheet, which dependency is denoted by ***Depends-on*** relation, User includes information that he has taken from Information Site. Finally, user copies modified file in file system and System B gets the information through ***Conversation*** communication represented by ***Channel*** element process completed information. ***Conversation*** communication is detailed in section 1.5.2 and ***Depends-on*** relation is subject of section 1.3.4.

***System*** receives information through conversations and publishes processed information no matter it is a computer or a user. Sys-ADL ***System*** element represents any kind of system.

### Transition

Consider the places where you live and where you work. When you are at home you take your breakfast and, when it is time to go, you move from home to office to be there in time. In this example, an event time had occurred and you move from home to office. If during the day you get sick, probably you will move from office to home due this new event.

In short, position changes responding to an event.

***Transition*** element represents an event that moves from a ***Layer*** element to another ***Layer*** element.

***Transition*** element is represented in Sys-ADL with an arrow with a filled point as shown in figure 1.1.12-1.

|  |
| --- |
|  |
| transition.png  1.1.13‑1 – Sys-ADL Transition symbol |
|  |

***Transition*** element is used to represent an event that makes an architecture moves from one ***Layer*** to another. Consider Sys-ADL diagram fragment in figure 1.1.12-2.

|  |
| --- |
|  |
| ObjExemplosAdicionaisTransitionLayer.png  1.1.13‑2 – Layer moving |
|  |

In figure 1.1.12-2 the event represented by ***Transition*** element user request makes layer move from ***Layer*** Presentation Layer to Business Logic Layer and the ***Transition*** data persistence makes layer move from Business Logic Layer to Data Layer. ***Transition*** element data processing makes layer move back to Business Logic Layer and, finally, ***Transition*** element view processed data moves layer to its initial layer Presentation Layer.

An event is able to make changes in a process or in the position of something and ***Transition*** element represents such events in Sys-ADL.

## Sys-ADL Element Data

### Concept

Sys-ADL elements are graphical representation of elements presents in System Architecture. Using them, it is possible to represent in diagram every single piece of architecture. Even though, only the graphical representation is not enough to represent every data of architecture. It is necessary to capture data related to these elements. In this section are introduced concepts used by Sys-ADL to capture and show data graphically using Sys-ADL element data.

Every element in Sys-ADL has data being four of them mandatory and for Sys-ADL user is possible to define additional data and which data are represented graphically and which are not.

When user selects a Sys-ADL element a form must be provided for user fulfillment.

### Mandatory fields

Sys-ADL determines four mandatory fields that must be presented and defined for all Sys-ADL elements. These fields must be present in every element of your system architecture and these fields are: **Stereotype**, **Namespace**, **Id** and **URL** Info.

First mandatory element data is **Stereotype** that is a field that is allowed to user left blank when element data is saving. This special field is used to create a graphical representation that allow elements of same nature be represented in distinct form. ***Stereotype*** is introduced in section 1.2.5.

Second mandatory element data is **Namespace**. This field is part of element identification allowing put elements in context. This field must have up to 200 characters. A good practice is separate element data in folders and namespace represent folder structure changing each folder slash character by dot.

Third mandatory element data is **Id**. This field creates unique identification by **Namespace** to recovery element data. This field must have minimum one and maximum 60 characters.

Fourth mandatory element data is **URL Info**. This field provides a Uniform Resource Locator (URL) that is accessible when element in diagram is clicked providing more information about element. There are no limits for this field length but it must represent a valid URL.

Table 1.2.2.1 shows a summary of mandatory fields.

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Blank allowed | Designed |
| Stereotype | Provide a differentiation to elements of same nature | Yes | Yes |
| Namespace | Provide a context for element | Yes | No |
| Id | Provide a unique Id for namespace context | No | No |
| URL Info | Provide a URL giving more information about element | No | No, but in design element has an action when clicked |

1.2.2‑1 – Short description of each mandatory field

Mandatory fields are ever requested in element form.

### Additional Fields

Sys-ADL user administrator staff is allowed to define further field definition in element other than mandatory fields. These fields are required to user provide data for them. User administrator must provide information about field metadata. Following these field metadata are introduced.

**Field name**: An identifier for field. Once a field is defined it is reusable for other elements.

**Field label**: A label is put in front of the field to show what information use must provide.

**Field description**: A short description of field that is shown for user help, if requested.

**Field message error**: A message that is shown to user when data provided for a field is not valid according to field validation rules.

**Field type**: Is the type of the field. Domain value is Date, Element, List, String, Time and Value.

**Validation**: According to type of element a validation is provided by user administrator. Table 1.2.3.1 shows validation data for each field type.

|  |  |
| --- | --- |
| Field Type | Validation |
| Date | Boolean: Past date allowed, Future date allowed and Present day allowed |
| Time | Minimum and Maximum time |
| Value | Minimum value, Maximum value and Allow only integer |
| String | Regular expression |
| Element | Allowed Sys-ADL element type |

1.2.3‑1 – Data validation information required to validate data provided by Sys-ADL user

After created, field is bound to element and user is able to provide data for this field because it is shown in the form of the element.

User administrator must define if the field is mandatory or not. User is able to bind a field to a pair of element and stereotype allowing that elements of same type but with different stereotype have different fields to be requested and is also able to define when field is mandatory or not. ***Stereotype*** is introduced in section 1.2.5.

User is able to set up if a field must be shown in diagram or not and in which cases. In next section is introduced how these element data are shown in diagram.

### Field graphical representation

Field definition provides information about how user will insert data in Sys-ADL Element form. User administrator is able to determine how a field must be present in Sys-ADL element graphical representation. Following are introduced data required to show element data in design.

Field design shows element data information in element text area. User is able to provide the order where information is presented and if data is presented between parentheses. User administrator is able to determine if additional field is presented in comment area if information is too long to fit in design area. Figure 1.2.4.1 shows an example of how this is presented in Sys-ADL. This picture can change depending on tool used to show diagram data.

|  |
| --- |
|  |
| StakeholderDataPresentation.png  1.2.4‑1 – Showing element data in design time |
|  |

In figure 1.2.4.1, user administrator determines that fields **Name** and **Role** must be shown in design and last one must be presented between parentheses. Fields presentation order is same fields form order, presenting **Name** first and **Role** after. Notice that field **Interests** is not shown with other fields, but is shown in comment area.

Sys-ADL forecast a mechanism of how information is input and show in diagram.

### Stereotype

***Stereotype*** creates a differentiation for elements of same type. With ***Stereotype***, user is able to create specific types of ***Systems***, ***Stakeholders***, ***Channels*** and any other Sys-ADL element. This difference could be only in design showing its   
***Stereotype*** in the text area reserved for this in Sys-ADL element graphic representation or user is able to determine a new set of element data for a pair element/stereotype if required.

***Stereotype*** allows user distinguish visually elements of same nature. Figure 1.2.5.1 shows an example of this in Sys-ADL.

|  |
| --- |
|  |
| StakeholderDataPresentation.png  1.2.5‑1 – Different stereotypes differentiates different system types |
|  |

In figure 1.2.5.1 there are three kinds of System elements and each one is distinguished by its ***Stereotype***. First, ***System*** element with no ***Stereotype*** is a common system with no specialization. Second, ***Stereotype*** ecosystem represents an area of company which may execute some process. Third, with ***Stereotype*** external, represents systems or entities outside of company.

Besides visual differentiation, Sys-ADL user administrator is able to define different fields for different stereotypes for same element type. Consider the scenario presented in figures 1.2.5.2 and 1.2.5.3.

|  |  |
| --- | --- |
|  | |
| StakeholderDataPresentation.png  1.2.5‑2 – Fields in no stereotype stakeholder… | StakeholderDataPresentationUser.png  1.2.5‑3 – … are different from <<user>> stereotype |
|  | |

In figures 1.2.5.2 and 1.2.5.3 are shown two types of ***Stakeholder*** elements and each one has its own set of data provided. First with blank stereotype has, besides Sys-ADL mandatory fields, fields **Name**, **Role** and **Interests**. Second with ***Stereotype*** user has **Name** and **Applications** that are quite different of its ***Stakeholder*** element brother. Notice that even visual presentation is different. Sys-ADL ***Stakeholder*** element with blank stereotype shows two fields in its design and one in comments and other ***Stereotype*** user shows only one field and presents the other field in comments area.

***Stereotype*** is a mechanism that allow user to create subtypes of Sys-ADL elements as he needs. It supports a series of process and increase design facilities.

## Sys-ADL Relations

### Concept

Sys-ADL Element is the smallest part of system architecture and has worth and data. Even though elements are very important, they do not have worth for System Architecture alone. To create value for these individual pieces, they must be related to each other generating new structures

This section shows the mechanisms that Sys-ADL provides to create relations between individual elements and the results of these relations. Sys-ADL relations are ***Connection***, ***Composed-by***, ***Depends-on***, ***Is-a*** and ***Represents***.

### Connection

Individual elements of a system architecture such as systems, computers, files, devices, components and others process information that has value for major objectives. Nevertheless, they are not enough to reach such objectives by themselves. Consider a system that process a set of information but does not publish it for a major objective. Such system has no value no matter his job is nicely done. To give value for others this system must turn public its processed information through an interface.

Another example, picture a computer that has a graphical user interface where a user inputs data and see process result and has no physical ports to input media devices neither is connected to a network. It would be great if the result viewed by user could be sent to another system make other processes based in such results, but the computer has no access to external world and such result cannot be exported. Now, although user has a marvelous system, all process result must be inputted in other computer manually. A computer must have open ports to be accessed to connect further computers.

***Connection*** relation is used to connect two individual elements creating structures that allow these elements work together increasing its individual values. There are five known structures that Sys-ADL is able to represent: ***Port***, ***Protocol***, ***Device***, ***Concern*** and ***Interface***. Structures are detailed in section 1.4.

***Connection*** relation is represented in Sys-ADL with a straight line as show in figure 1.3.2.1.

|  |
| --- |
|  |
| connector.png  1.3.2‑1 – Sys-ADL Connection symbol |
|  |

An example of ***Connection*** use is creating an ***Interface*** structure. Consider a system represented by ***System*** element System A that produces a plain text file with some information that must be published represented by a Format element TXT. Connecting these individual elements using ***Connection*** relation creates an Interface structure as show in figure 1.3.2.2.

|  |
| --- |
|  |
| ObjExemplosAdicionaisInterface.png  1.3.2‑2 – Interface structure with Connection |
|  |

Other example of ***Connection*** relation use is creating ***Port*** structure. Consider a Web Server called “Web Server A” that HTTP port must be configured as show in figure 1.3.2.3. Individual elements ***Node*** and ***Receiver*** are connected creating a ***Port*** structure

|  |
| --- |
|  |
| ObjExamplesPort.png  1.3.2‑3 – Port structure with Connection |
|  |

***Connection*** relation creates structures in Sys-ADL. Structures are detailed in section 1.4.

### Composed-by

Some structures are too large that is hard to understand it as a hole. A way to understand such structure is divide it into smaller parts that are easier to handle. This technique is used in network segmentation and subsystem division.

***Composed-by*** relation represents an all-part relation being relation start is all and relation end is the part.

***Composed-by*** relation is represented in Sys-ADL with a straight line with an arrow in the end, as show in figure 1.3.3.1.

|  |
| --- |
|  |
| composed-by.png  1.3.3‑1 – Sys-ADL Composed-by symbol |
|  |

***Composed-by*** relation only connects elements of same nature and the following elements:

* ***Objectives***
* ***Nodes***
* ***Systems***

***Composed-by*** is used to represent two elements interacting and the part cannot exist with its all. This is a strong relation once if the all no longer exist its part will not exist anymore.

A common use of ***Composed-by*** relation is to divide a system in minor components as shown in figure 1.3.3-2 Diagram fragment shows System A divided in specialized components Business Processing, User Application and Persistence.

|  |
| --- |
|  |
| ObjExemplosAdicionaisComposedBySystem.png  1.3.3‑2 – Composed-by specifying system components |
|  |

Let’s see another example of implicit dependency that is represented by ***Composed-by*** relation. Picture a system that receives an interface, process such information and sends modified information to another system, but, before it sends any information, it transforms the file to be published and it has a specialized component that converts and send a file. The explicit scenario is shown in figure 1.3.3.4.

|  |
| --- |
|  |
| ObjComposedByExplicitDependency.png  1.3.3‑2 – Explicit dependency between System B |
|  |

The scenario described in Sys-ADL diagram fragment shows a ***System*** element System A that sends a Format element TXT file to ***System*** System B using a ***Receiver*** FTP. The Conversion and Transmission component of System B executes a SQL query through JDBC local connection, represented by Sender element labeled JDBC to convert and send file. Finally, Conversion and Transmission creates a XML file with information taken from System B and sends this information to System C. Notice that the communication between System B and Conversion and Transmission has been described in an explicit way. Nevertheless, as Conversion and Transmission is part of System B, as described earlier, this relation is shown using ***Composed-by*** relation, as shown in figure 1.3.3.5 where interaction between System B and Conversion and Transmission is no longer detailed, being represented by ***Composed-by*** relation.

|  |
| --- |
|  |
| ObjComposedByImplicitDependency.png  1.3.3‑3 - Implicit dependency between System B |
|  |

Other example of ***Composed-by*** relation use is separate objectives to create an ***Objective*** map. All objectives are grouped into related parts as show in figure 1.3.3.6.

|  |
| --- |
|  |
| ObjExamplesObjectiveMap.png  1.3.3‑4 – Composed-by relation creating Objective Maps |
|  |

In figure 1.3.3.6 is represented a business segmentation with ***Composed-by*** relation used to create an ***Objective*** map diagram that helps system architect comprehends how systems attend business processes and how they are related.

Another use of ***Composed-by*** relation is in infrastructure description using ***Nodes*** as show in figure 1.3.3.7.

|  |
| --- |
|  |
| ObjComposedByNode.png  1.3.3‑5 – Server infrastructure using Nodes |
|  |

Figure 1.3.3.7 represents a server infrastructure, represented by ***Node*** element Server Infrastructure that is composed by ***Node*** element Production Server (main server), and COB Server (Continuity of Business server) that rises when main server down.

Another way to create a ***Composed-by*** relation is using ***Layers***. When you put a ***System***, a ***Node*** or a ***Layer*** element inside a ***Layer*** element, those elements has a ***Composed-by*** relation with this Layer element. For example, consider the Sys-ADL diagram fragment in figure 1.3.3.8.

|  |
| --- |
|  |
| ObjComposedByLayerExample.png  1.3.3‑6 – Composed-by relation using Layers |
|  |

Figure 1.3.3.8 has a set of elements that has ***Composed-by*** relation that are explained in table 1.3.3.1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Desktop Application   |  | | --- | | Composed-by Relation | | User Interface   |  | | --- | | Composed-by Relation | | Rich Application | | | User Input Validation   |  | | --- | | Composed-by Relation | | Field Content Validator | | | Rich Application | | Field Content Validator | | User Desktop | | Core System   |  | | --- | | Composed-by Relation | | Business Logic   |  | | --- | | Composed-by Relation | | Business Input Validator | | Business Rules | | | Persistence   |  | | --- | | Composed-by Relation | | Relational Database | | | Business Input Validator | | Business Rules | | Relational Database | | Application Server | | Database Server | |

1.3.3‑1 – Composed-by Layer relation result

***Composed-by*** relation allows system architect to identify related architecture elements helping not only creating diagrams that are easier to understand beside of all technology information infrastructure.

### Depends-on

Picture a system called “System A” that needs information from other system called “System B” to accomplish a process. As “System A” is not able to execute its tasks without information provided by “System B” it depends on “System A” to reach its goals. Regarding “System B” has no knowledge of “System A”, if “System A” disappears suddenly “System B” has no impacts in its functionalities because “System B” depends on nothing from “System A”. In such scenario, “System A” depends on “System B” but inverse is not true.

***Depends-on*** relation represents a situation where a system depends on another and is not practical or feasible to represent all interactions involved in such relation in an explicit way.

***Depends-on*** relation is created in Sys-ADL with a dashed line with an arrow in the end of relation. This symbol is represented in figure 1.3.4-1.

|  |
| --- |
|  |
| depends-of.png  1.3.4‑1 – Sys-ADL Represents symbol |
|  |

***Depends-on*** is different of ***Composed-by*** in relation nature because ***Depends-on*** creates a soft relation where the dependency target element have no knowledge of the element that depends on it. In other words, if the element that depends on another element has removed, the target element has no changes at all.

***Depends-on*** relation is used only between ***System*** Sys-ADL elements.

A very common use of ***Depends-on*** use is to represent technologies used by a system. For example, picture a system called “System A” that is deployed in a JEE® Application Server. Such relation is represented in Sys-ADL diagram fragment in figure 1.3.4.2.

|  |
| --- |
|  |
| ObjDependency1.png  1.3.4‑2 – System dependency represented by Depends-on |
|  |

Notice in diagram fragment in figure 1.3.4.2 that if “System A” has been removed from JEE Application Server this system remains serving other applications without modifications. “System A” depends on JEE Application Server but inverse is not true. Other important point is that there are a lot of conversations between these systems, but they are not shown because such interaction is not important in dependency description.

Other example of ***Depends-on*** relation is when a tool is used to help a system to reach its objective. In figure 1.3.4.3 there is a diagram fragment with an example of such relation.

|  |
| --- |
|  |
| ObjDependency2.png  1.3.4‑3 – User uses a spreadsheet to manipulate data in Manual Process |
|  |

In figure 1.3.4.3, user uses a spreadsheet to import a CSV file intending to change its content. This is a common use of spreadsheet and is not necessary to show how this interaction occurs and the layer denotes that user manipulates the file manually. This layer would not be necessary if a pattern has already been described and reused here because all process description could be described in other diagram and reused in this fragment. Architecture pattern design and reuse in Sys-ADL is created by ***Represents*** relation that is introduced in section 1.3.6.

There is a set of situations where is necessary to represent a dependence between two elements and at same time is not practical, feasible or necessary to represent all conversations between these elements and ***Depends-on*** relation is used to show such no detailed relation.

### Is-a

Occasionally there is a set of structures that has a lot of similar behavior. For example, picture a “System A” and a “System B” that create files with same XML tags meaning two files, although have different information, have same structure and content nature. Only looking inside both files anyone can figure out which system has generated the file because of similarity of system’s behavior. Because of this similar behavior, they are members of a class of systems that generate the same kind of XML file.

Is-a relation represents a situation where an element has the same behavior of other element. Nevertheless, elements have no behavior since they are static elements and they interact with no one. Thus, when an element has an Is-a relation with other it inherits all relations, structures and communications made by ***Is-a*** element relation target. This inheritance is valid no mater diagram relation has been described and this feature helps avoiding description duplication.

***Is-a*** relation is created in Sys-ADL using a filled line with a triangle in the end of relation. This picture is shown in figure 1.3.5-1.

|  |
| --- |
|  |
| is-a.png  1.3.5‑1 – Sys-ADL Is-a symbol |
|  |

As example, regard two ***System*** elements System A and System B both having same conversation with ***System*** element System C and both systems sends a file with same information, although its data is different. This scenario is described in Sys-ADL diagram fragment in figure 1.3.5-2.

|  |
| --- |
|  |
| ObjIsAExamplePart1.png  1.3.5‑2 – Systems A and B have similar interfaces with System C |
|  |

The scenario represented in figure 1.3.5-2 is represented in a cleaner way with Is-a relation as shown in diagram fragment in figure 1.3.5-3.

|  |
| --- |
|  |
| ObjIsAExamplePart2.png  1.3.5‑3 – System A and B interfaces have grouped into one type of interface |
|  |

In figure 1.3.5-3, a ***System*** element System C information provider is used to represent a class of systems that has communication with System C. There are two interesting points here. First, diagram is clear because symbols are saved avoiding unnecessary diagram pollution and second, such relation is reused in further diagrams, stimulating the use of Design Patterns, as shown in figures 1.3.5-4 and 1.3.5-5.

|  |
| --- |
|  |
| ObjDesignPatternExample.png  1.3.5‑4 – Design pattern description |
|  |

In figure 1.3.5-4 is represented all relations that a JEE Application has describing an Architecture Pattern which is reused in figure 1.3.5-5.

|  |
| --- |
|  |
| ObjDesignPatternExampleUser.png  1.3.5‑5 – Design pattern reuse |
|  |

The pattern use represented in figure 1.3.5-5 avoids representation of all relations for Customer Security Portal, which depends on Websphere and Oracle. Other advantage of this approach is when an update version is made in ***Objective*** element JEE Architecture Pattern all elements that inherit such relation will be updated at same time.

***Is-a*** relation allows Sys-ADL user creates a cleaner diagram, turns dependency relation management easier and improve pattern use.

### Represents

In 1999 the book “Design Patterns: Elements of reusable object-oriented software” from that have been known as Gang of Four, Erich Gama, Richard Helm, Ralph Johnson and John Vlissides published a set of solution used by programmers to solve a specific kind of solutions, but, more than this, they propose a way to document such solutions. Since them people have been realized they can document solutions for their own problems and publish this as a Design Pattern and with Sys-ADL designers can design such patterns and Represents relation allows design diagrams be tied to specific pattern solution.

***Represents*** is used when it designer wants to bind their own solution to a standard solution that is represented with another element. This relation has no semantic results being used just to connect architect designs to standard solutions.

***Represents*** is represented in Sys-ADL with a dashed line with a triangle in the end of relation as shown in figure 1.3.6-1.

|  |
| --- |
|  |
| represents.png  1.3.6‑1 – Sys-ADL Represents symbol |
|  |

***Represents*** is created in Sys-ADL connecting an element to another element that will be represented using a dashed line with a triangle in the end of relation as shown in figure 1.3.6-2.

|  |
| --- |
|  |
| ExemploRepresents.PNG  1.3.6‑2 – Objective being represented by a Receiver |
|  |

***Represents*** have no semantic results and is used when the designer wants to indicate that its solution uses a standard solution. This symbol can be used to indicate a path where information flows and is not necessary to reproduce it in each diagram. Figures 1.3.6-3 shows a pattern being represented by a ***Receiver***.

|  |
| --- |
|  |
| ObSendInformatinWebApplication.png  1.3.6‑3 – Entire solution is represented by a simple Receiver element |
|  |

In figure 1.3.6-3 a pattern has designed and bound to an ***Objective*** element Send information to Web Application that is represented by a ***Receiver*** element WEB. Once pattern is designed and an element has chosen to represent it this Sys-ADL element is reused in other diagrams as shown in figure 1.3.6-4.

|  |
| --- |
|  |
| ObjPatternUse.png  1.3.6‑4 – Pattern use with “Web” Receiver element |

In figure 1.3.6-4 a ***Protocol*** structure has created with Sys-ADL element ***Receiver*** element WEB which represents ***Objective*** element Send information to Web Application shown in figure 1.3.6-3. This diagram fragment is tied to the pattern solution.

There is another issue here. As there are no semantic results using ***Represents*** relation. This means that is not possible to derive the information that ***System*** element Debit Account sends information to ***System*** element Message Handling and Converter although it is obvious that this relation exists and is important. To solve this question, we need to create a ***Is-a*** relation between elements of target diagram as shown in figure 1.3.6-5

|  |
| --- |
|  |
| ObjPatternUseWithIsA.png  1.3.6‑5 – Pattern roles assigned |
|  |

In figure 1.3.6-5, ***System*** element Debit Account is related to Mainframe System with ***Is-a*** relation. This means that Debit Account changes interfaces with ***System*** element Message Handling and Converter. A similar ***Is-a*** relation has constructed between ***System*** element Balance Web Application and ***System*** element Web Application and, now, ***System*** element Balance Web Application inherits ***System*** element Web Application relation with Message Handling and Converter ***System*** element.

As you may guess, only ***Objective*** Sys-ADL element is able to be represented using ***Represents*** relation because only ***Objectives*** can represent a pattern designed in a Sys-ADL diagram. Only two Sys-ADL elements are able to represent ***Objectives***: ***Receiver*** and ***Sender***.

***Represents*** relation allows the use of design patterns in Sys-ADL and solutions built by architects can be tied to them.

## Sys-ADL Structures

### Concept

As introduced in section 1.1, Sys-ADL elements are pieces of system architecture that exists without connections or dependencies of other elements. They exist without connections or relations with each other.

***Connection*** relation has introduced in section 1.3.2. This relation is able to create new structures connecting individual elements. In this section are introduced in details all structures created by ***Connection*** relation.

Structures known in Sys-ADL are ***Concern***, ***Deposit***, ***Device***, ***Host***, ***Interface***, ***Port***, ***Protocol, Requirement*** and ***Responsibility***.

### Concern

A process exists before a system. Maybe this process already operates in real life in a department or company or it exists only in someone imagination. To get this process and implement it on a system, it is necessary that someone pays for it. After money is available, other people can start to work. We will have specialists, programmers, infrastructure, testers, data administrators and other technical staff working on system development. Meanwhile, future system users may operate the system in alpha or beta version to see its functionalities working in real life. Finally, as incredible that it may look, there are people enrolled on systems, not only hardware and software and they are very important to system architecture, because they will determine how a system is going to be robust, resilient, flexible, available, user friendly and other system characteristics. They are the ***Stakeholders*** and they have concerns on system architecture. ***Concern*** structure identifies how a ***Stakeholder*** is enrolled in an Architecture Element.

***Concern*** structure represents a stakeholder that has concerns on a System Architecture element. If someone or some group has any relation with an element such as sponsor, user, developer, settings and others this concern is represented by ***Concern*** structure.

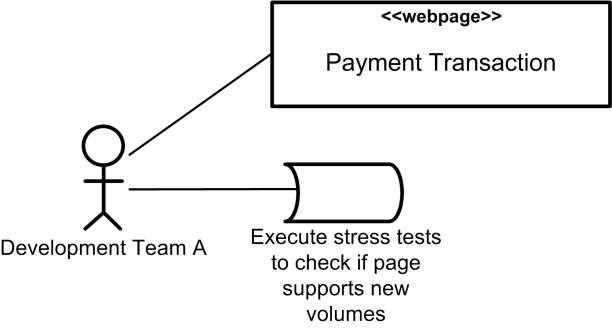
***Concern*** structure is created in Sys-ADL connecting a ***Stakeholder*** element to another Sys-ADL element, except other ***Stakeholder*** or ***Layer*** element, using ***Connection*** relation. An example is shown in Figure 1.4.2-1. ***Connection*** relation has introduced in Section 1.3.2 and ***Stakeholder*** has introduced in Section 1.1.11.

|  |
| --- |
|  |
| 1.4.2‑1 - Partial representation of a Concern structure  1.4.2‑1 – |
|  |

When indicated that a ***Stakeholder*** has a ***Concern*** on some element, it is necessary to indicate what the interest of the Stakeholder in the element is. Sys-ADL forecasts two situations when a ***Concern*** is indicated: ***Requirement*** or ***Responsibility***. ***Requirement*** is discussed in section 1.4.9 and ***Responsibility*** is discussed in section 1.4.10.

***Responsibility*** structure is independent of concern. ***Responsibility*** can be represented without a ***Concern*** structure. Although, opposite is not true since ***Concern*** can’t be represented without a ***Responsibility*** structure or ***Requirement*** structure.

***Responsibility*** structure can be used with a ***Concern*** structure to indicate the role of a ***Stakeholder*** on a Sys-ADL element. ***Stakeholder*** element is discussed in section 1.1.11. Figure 1.4.2.2 shows an example of this structure.

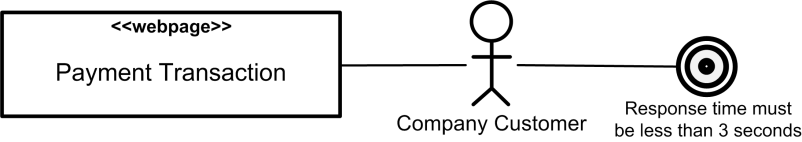


1.4.2‑2 - Development Team concern on Payment Transaction

As shown in Figure 1.4.2-2, ***Stakehoder*** element called Development Team has a concern on ***System*** element Payment Transaction. This concern is specified using ***Responsibility*** structure represented by ***Stakeholder*** element Development Team A and ***Role*** element Execute stress tests to check if page supports new volumes.

***Requirement*** structure, unlike ***Responsibility*** structure, can’t be represented without a ***Concern*** structure. This is used to justify a requirement of the design since the requirement is bind to the element and ***Stakeholder*** element that is the source of the requirement.

Figure 1.4.2-3 shows an example of ***Requirement*** structure related to a ***Concern*** structure.



1.4.2‑3 - Company Customer's requirement

In Figure 1.4.2-3, ***Stakeholder*** element Company Customer needs that response time of ***System*** element represented by Payment Transaction be less than 3 seconds. Such requirement is represented by ***Quality*** element Response time must be less than 3 seconds. ***Quality*** element is discussed in section 1.1.7. ***Requirement*** structure is discussed in section 1.4.9.

***Concern*** structure is used to identify all related stakeholder and their roles in an architecture solution. In figure 1.4.2-4 shows an example of ***Concern*** structure use.

|  |
| --- |
|  |
| 1.4.2‑4 - Example of Concerns |
|  |

In Figure 1.4.2-4, there is a ***System*** element called Payment Transaction. There are two ***Stakeholders*** that made a ***Concern*** structure with Payment Transaction: Company Customer, that has a requirement, and Development Team A that has a responsibility.

Company Customer can’t have a ***Concern*** structure with Payment Transaction without a ***Requirement*** or ***Responsibility*** structure that justifies this interest. Company Customer is connected to ***Quality*** element Response time must be less than 3 seconds.

***Stakeholder*** element Development Team A creates a ***Responsibility*** structure with ***Role*** element Execute stress tests to check if page supports new volumes. This ***Responsibility*** structure is related to ***Concern*** structure created by the connection between ***Stakeholder*** element Development Team A and ***System*** element Payment Transaction.

***Concern*** Sys-ADL structure is used to represent interests that ***Stakeholders*** has in some element of the architecture solution. ***Concern*** structure must be specified by a ***Requirement*** or ***Responsibility*** structure.

### Deposit

System sends interfaces to other system. In some cases there is a communication infrastructure between these systems such a network and in other cases don’t because is not feasible or practical to perform such conversation using a network. For these cases, Sys-ADL represents this conversation with ***Deposit*** structure.

***Deposit*** structure is used when a conversation is made without a network and is assembled connecting a ***Sender*** element to an ***Interface*** structure as shown in figure 1.4.3-1.

|  |
| --- |
|  |
| ObjDepositFigure.png  1.4.3‑1 – Sys-ADL Deposit set |
|  |

A ***Deposit*** structure use is data transfer through file copying. Figure a system that saves its information in a system folder that is shared with another system. Let’s represent first with a Sys-ADL ***System*** element System A and other we ***System*** element System B. See figure 1.4.3-2.

|  |
| --- |
|  |
| ObjDepositFileShare.png  1.4.3‑2 – File sharing represented with Deposit structure |
|  |

Both ***Systems*** System A and System B share a system folder in the same server and they need no network to communicate each other because they are installed in the same computer and this is represented by Sys-ADL ***Host*** structure. System A provides an interface represented by ***Interface*** structure composed by the connection between Sys-ADL ***System*** element System A and Sys-ADL element ***Format*** TXT, denoting System A generates a plain text file. ***Host*** structure is explained in section 1.4.5 and ***Interface*** structure is explained in Section 1.4.6. System A generates the plain text file and put it in a folder shared with System B and this folder is represented by Sys-ADL ***Sender*** element FSHAR, from “File Share”. Put action is made by ***Deposit*** structure. Once structure is designed, information is sent from System A to System B through ***Channel*** element process data.

Let’s see another example of ***Deposit*** structure. Picture a scanner that is used to get an image that is read by an application installed in a desktop. This scanner is connected in user’s desktop, so, the scanner’s program send the image to a system installed in user’s desktop. A proposal drawing in Sys-ADL for this situation is shown in diagram fragment in figure 1.4.3-3.

|  |
| --- |
|  |
| ObjDepositUSBpng.png  1.4.3‑3 – Another Deposit example |
|  |

In figure 1.4.3-3 ***System*** element Scanner System generates a JPEG file represented by ***Format*** Sys-ADL element JPG that is put in ***Sender*** Sys-ADL element USB creating a ***Deposit*** structure that sends the image stream to ***System*** element Desktop Application. Below is shown ***Device*** structure Scanner that is installed in User Desktop, represented by ***Installation*** communication. ***Device*** structure is explained in Section 1.4.4 and ***Installation*** communication is introduced in Section 1.5.3.

There are situations where systems have no network enrolled in their communication. For these situations ***Deposit*** structure must be used to denote this kind situation.

### Device

Some process requires special hardware to complete their tasks. Few examples of such kind of hardware are scanners, printers, data collectors, smart cards, barcode readers etc. These components are connected to servers, desktops and other hardware available in system solution for users.

***Device*** structure represents a device connected directly into a computer through its physical ports. This structure is created connecting a ***Node*** element to a ***Sender*** element through ***Connection*** relation as shown in figure 1.4.4-1. ***Node*** and ***Sender*** elements are introduced in section 1.1.5 and 1.1.8, respectively.

|  |
| --- |
|  |
| ObjDevice.png  1.4.4‑1 Sys-ADL Device representation |
|  |

***Device*** structure is plugged in ***Nodes*** denoting where it is installed. This notation creates a Sys-ADL communication called ***Installation***. A Sys-ADL diagram fragment showing this communication is shown in figure 1.4.4-2. ***Installation*** Sys-ADL communication is explained in section 1.5.3.

|  |
| --- |
|  |
| ObjDeviceInstallationExample.png  1.4.4‑2 – Installation sample |
|  |

In figure 1.4.4-2 shows a ***Device*** structure composed by a ***Node*** element Scanner and a ***Sender*** element USB. This structure is connected through a ***Channel*** element to a ***Node*** element User Desktop. This denotes that Scanner is installed into User Desktop creating the ***Installation*** communication.

***Device*** structure is Sys-ADL resource that allows the System Architect to represent all equipment requested to a system works.

### Host

Once a system is created it is necessary to provide some hardware infrastructure where this system will run. The infrastructure will be the hotel that will host his brand new invited.

***Host*** structure is used to represent a system that is installed in hardware and is constructed connecting a ***System*** element to a ***Node*** element through the ***Connection*** relation as shown in figure 1.4.5-1. ***Connection*** relation is explained in Section 1.3.2.

|  |
| --- |
|  |
| ObjHost.png  1.4.5‑1 – Sys-ADL Host representation |
|  |

***Host*** structure shows which systems are installed in each server. In figure 1.4.5-2 is shown an example of ***Host*** structure use.

|  |
| --- |
|  |
| ObjHostExample.png  1.4.5‑2 – Systems hosted in Nodes |
|  |

***System*** element Web Application creates a ***Host*** structure with ***Node*** element Web Server representing that there is a Web Application running in this Web Server. The same relation is created between ***System*** element Business Rules and ***Node*** element Application Server. Now, what if is a ***System*** element Database Server hosted in the same Application Server of the Business Rules? This scenario is shown in figure 1.4.5-3.

|  |
| --- |
|  |
| ObjHostExampleThree.png  1.4.5‑3 – Two systems hosted in same Node |
|  |

In figure 1.4.5-3, ***System*** element Database Server is running in the same ***Node*** Application Server of ***System*** element Business Rules.

***Host*** structure is used to represent the topology of a system architecture solution showing where each system is running.

### Interface

Systems have internal process to handle with its own data or information came from outside its boundaries and system designer ordinarily project systems to publish information that have processed by them in a determined format such as XML, plain text, Excel, return parameters, database queries etc.

***Interface*** is Sys-ADL structure used to represent system publishing information and is constructed connecting a ***System*** element to a ***Format*** element through a ***Connection*** relation as shown in figure 1.4.6-1.

|  |
| --- |
|  |
| ObjInterface.png  1.4.6‑1 – Sys-ADL Interface representation |
|  |

Any kind of information processed by system is able to be published in any format and ***Interface*** structure represents this in Sys-ADL. Figure 1.4.6-2 shows an example of ***Interface*** structure use.

|  |
| --- |
|  |
| ObjInterfaceExample.png  1.4.6‑2 – Interface structure sample |
|  |

Figure 1.4.6-2 shows ***System*** element Tax Payment System publishing its processed information in a XML file format and now this file can be read by another system through a ***Conversation*** communication. This is explained in Section 1.5.2. Information about file content is providing in element data. Element data is discussed in Section 1.2.

Ordinary systems use to process and publish data. Interface structure is Sys-ADL resource to represent this situation.

### Port

When two computers are connected, there are two items to be put in mind: how data will be sent from source computer and how it will be received by the destiny computer. Data is sent through a network and is received by a logical port configured in the computer that is the server of the connection.

***Port*** structure represents in Sys-ADL the logical ports of a server connection and is constructed connecting a ***Node*** element to a ***Receiver*** element using a ***Connection*** relation as shown in figure 1.4.7-1.

|  |
| --- |
|  |
| ObjPort.png  1.4.7‑1 – Sys-ADL Port representation |
|  |

***Port*** structure shows available logical ports of a ***Node*** through other ***Nodes*** can connect to send information to systems that are hosted there. In figure 1.4.7-2 is shown an example of this scenario.

|  |
| --- |
|  |
| ObjExamplesObjective.png  1.4.7‑2 – Web Server A has an HTTP Port avaiable to connection |
|  |

In figure 1.4.7-2 a Port structure is represented by ***Node*** element Web Server A and ***Receiver*** element HTTP ***Connection*** relation. This Sys-ADL diagram represents “HTTP” port configured in “Web Server A”. Through this port ***Node*** element User Desktop was able to create an HTTP connection to web server and, now, System element Web Server, which creates a ***Host*** structure with Web Server A, is able to send an HTML file represented by Format element HTML to a browser installed in Customer desktop. Notice ***Channel*** element between ***Port*** structure and ***Node*** element User Desktop which creates a Sys-ADL communication called ***Network*** that is detailed in Section 1.5.4.

Network is not enough to connect computers being necessary to set up communication ports in host system to receive client’s connections. Sys-ADL ***Port*** structure is used to represent such logical port that systems provides to connection allowing its hosted systems receives information from other systems hosted in other computers.

### Protocol

Systems interchange information each other to reach their objectives and network is a very important piece of this puzzle, but not the only because systems needs to establish a common way to receive and treat messages they receive. They need protocols and in this section is introduced how to represent them in Sys-ADL language.

***Protocol*** is Sys-ADL structure used to represent the protocol through information is received and treated and is made connecting a ***System*** element to a ***Receiver*** element as shown in figure 1.4.8-1.

|  |
| --- |
|  |
| ObjProtocol.png  1.4.8‑1 – Sys-ADL Protocol representation |
|  |

Though protocol a system is able to receive information from their feeders when a communication is established between information source and destiny. In figure 1.4.8-2 shows an example of such scenario.

|  |
| --- |
|  |
| ObjProtocolExample.png  1.4.8‑2 – System B receiving Interface from System A through HTTP Protocol |
|  |

In Sys-ADL diagram fragment shown in figure 1.4.8-2 a ***System*** element System A generates a XML file, represented by Format element XML, and sends it through ***Channel*** element send information until ***Receiver*** element HTTP that is connected to ***System*** element System B creating a ***Protocol*** structure, allowing it receive information through a HTTP port.

Other example of ***Protocol*** structure is shown in figure 1.4.8-3 which is already used to show other structures.

|  |
| --- |
|  |
| ObjExamplesObjective.png  1.4.8‑3 – Protocol used by Browser is the same of Web Server A turned avaiable for connection |
|  |

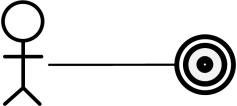
Let’s focus in HTTP protocol of figure 1.4.8-3. This protocol is used to ***System*** element Browser receive HTML data generated by Web Server system and show an HTML page to Customer, which is browser’s user.

Systems need protocols to receive data from other systems. Once established, they send it through networks. ***Protocol*** Sys-ADL structure is used to represent protocols that allow systems receive information.

### Requirement

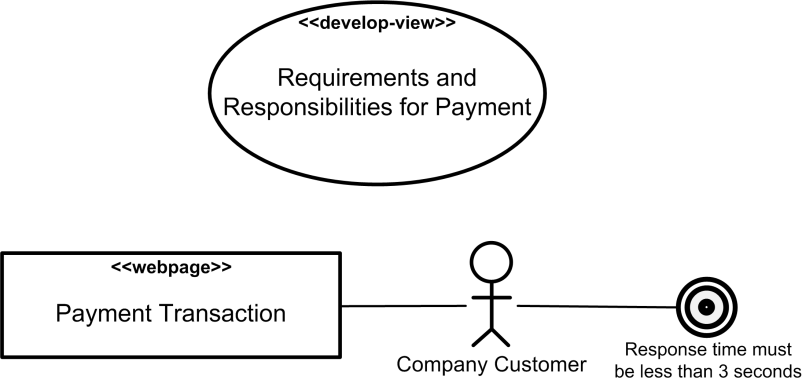
An element has one or more Quality attributes that it must attend. For instance, business can determine that a web application must respond requests from user within 20 seconds in 99,99% of the cases.

***Requirement*** is Sys-ADL structure that shows the quality attributes that an element must attend. ***Requirement*** structure is created when a ***Quality*** element is connected to a ***Concern*** structure. ***Concern*** structure is discussed in section 1.4.2. Figure 1.4.9-1 shows a partial representation of ***Requirement*** structure.



1.4.9‑1 – Partial representation of a Requirement structure

***Requirement*** structure relates each quality attribute to that element that must attend it. In figure 1.4.9-2 is an example of ***Requirement*** use. Very important: A ***Requirement*** structure can’t exist without a ***Concern*** structure.



1.4.9‑2 - Requirement use example

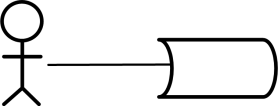
In Figure 1.4.9-2, a web page represented by ***System*** element called Payment Transaction made a ***Concern*** with Company Customer that has a ***Quality*** element Response time must be less than 3 seconds associated to him, creating a ***Requirement*** Structure. This means that Payment Transaction must respond to Company Customer’s requests in less than 3 seconds.

***Requirement*** structure allows to document architecture non-functional requirements. These requirements are associated to architecture elements when connected to ***Concern*** structure. All ***Requirement*** structure must be connected to a ***Concern*** structure.

### Responsibility

A system development and maintenance is a social activity. There are several teams that work together to build and keep the system running. Sometimes, these works has a great impact in architectural decisions. For example, an external entity can execute a task that is important to the system and should be represented.

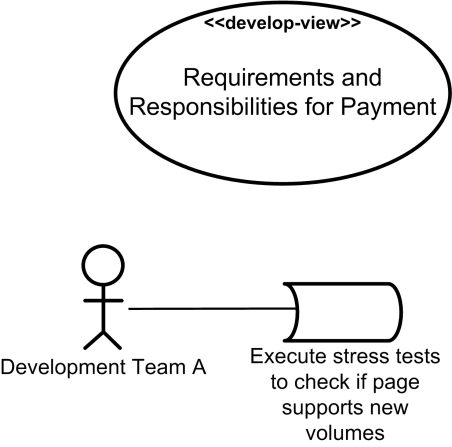
***Responsibility*** is a Sys-ADL structure that shows the ***Stakeholder*** ***Roles*** in a given system architecture. Figure 1.4.10-1 shows the representation of a ***Responsibility*** structure.



1.4.10‑1 - Representation of Responsibility structure

A ***Responsibility*** structure is created when a ***Role*** element is connected to a ***Stakeholder*** element. ***Responsibility*** structure can be, optionally, associated to a ***Concern*** structure. ***Concern*** structure is introduced in section 1.4.2. ***Role*** element is discussed in section 1.1.9. ***Stakeholder*** element is introduced in section 1.1.11.

***Responsibility*** structure is used to represent important functions or tasks. Figure 1.4.10-2 shows an example.

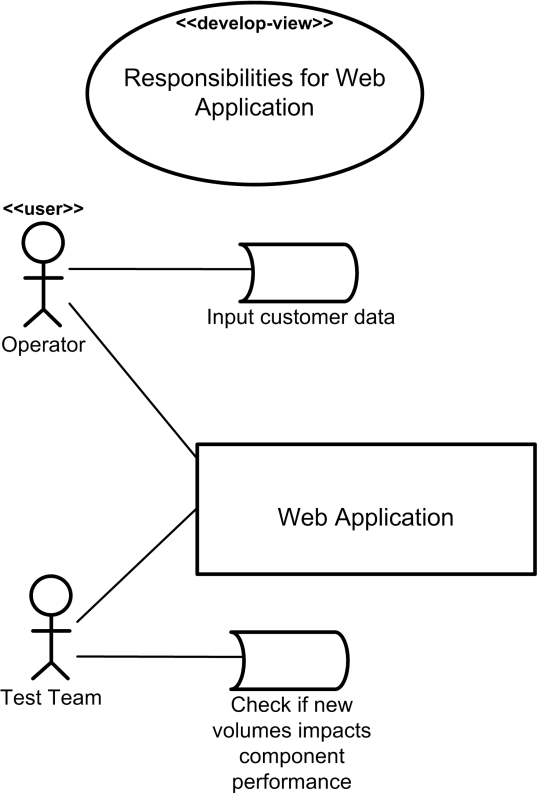


1.4.10‑2 - Example of Responsibility structure

In Figure 1.4.10-2, ***Stakeholder*** element called Development Team A is connected to ***Role*** element labeled Execute stress tests to check if page supports new volumes. This means that the team responsible for stress tests is Development Team A.

It is possible to indicate which components a ***Stakeholder*** element is responsible for. Creating a ***Concern*** structure is possible to indicate the element that is object of the responsibility as shown in Figure 1.4.10-3.

In Figure 1.4.10-4, there are two ***Stakeholder*** elements labeled Operator and Test Team that creates a ***Concern*** structure with ***System*** element labeled Web Application. Each ***Stakeholder*** element has its own ***Responsibility*** structure. Test Team is responsible for application testing according ***Responsibility*** structure created between ***Stakeholder*** element Test Team and ***Role*** element labeled Check if new volumes impacts component performance. Operator is a user that is responsible to input data according ***Responsibility*** structure created by ***Stakeholder*** element Operator and ***Role*** element labeled Input customer data. All these responsibilities are associated to ***System*** element labeled Web Application since these responsibilities are associated to both ***Concern*** structures.



1.4.10‑3 - Responsibilities associated to Concerns

***Responsibility*** structure allows architect to document the responsible for relevant tasks or activities in system architecture. Responsibilities can be associated to elements using ***Concern*** structure, although this is not mandatory.

## Sys-ADL Communications

### Concept

Communication is last part of Sys-ADL diagram. With communication, architect design is able to represent how individual structures will talk each other and, most important, why.

Individual structures are not capable to create a dynamic for system. A system that creates an interface and sends it to no one is a boring system. In this section, is introduced the mechanisms provided by Sys-ADL to show how systems works.

Sys-ADL provides three types of communication: ***Conversation***, ***Installation*** and ***Network***.

### Conversation

Systems publish their information generating files or streams known as interface that are sent to other systems that is received trough a combined protocol or being available to be caught for other system. There is an invisible line connecting them like processes and function calls.

***Conversation*** communication is used in Sys-ADL to represent information interchange between systems. System publishes an interface that is captured by other system and Sys-ADL uses ***Conversation*** communication to show this set processes or function calls. There are two ways to make a ***Conversation*** communication: Connecting a ***Deposit*** structure to a ***System*** element or connecting an ***Interface*** structure to a ***Protocol*** structure, both using a ***Channel*** element.

Figure 1.5.2-1 shows ***Conversation*** communication using ***Deposit*** structure and figure 1.5.2-2 shows ***Conversation*** communication using ***Interface*** and ***Protocol*** structures. ***Conversation*** communication using ***Deposit*** is performed without network different from ***Conversation*** that uses ***Protocol*** where a network is necessary.

|  |
| --- |
|  |
| ConversationDevice.png  1.5.2‑1 – Sys-ADL Conversation representation not using network |
|  |
|  |
| ConversationInterface.png  1.5.2‑2 – Sys-ADL Conversation representatin using network |
|  |

***Channel*** connects Sys-ADL structures representing the processes or function calls that moves interface from its source to its destiny creating a ***Conversation*** communication.

In figure 1.5.2-3 is shown a ***Conversation*** communication example.

|  |
| --- |
|  |
| ObjExemplosManualReferenciaConversation.png  1.5.2‑3 – Conversation sample |
|  |

In figure 1.5.2-3 ***System*** element System A creates a “TXT” file as described by ***Interface*** structure between elements. There is a ***Channel*** connected to this ***Interface*** up to the ***Protocol*** structure formed by ***Receiver*** element FTP and System B creating a ***Conversation*** communication between ***Interface*** structure and ***Protocol*** Structure. This conversation is performed through a network due ***Receiver*** presence. This scenario is interpreted as System A sends a text file to System B using a File Transfer Protocol”.

Consider other scenario. In section dedicated to ***Concern*** structure a Sys-ADL diagram shows a scenario for Tax Payments ***Objective*** element. Let’s see this diagram again but, now, we’ll get focus in ***Conversation*** between systems. This scenario is show in figure 1.5.2-4.

|  |
| --- |
|  |
| ObjTaxPayment.png  1.5.2‑4 – Tax payment scenario has several Conversations |
|  |

Consider Sys-ADL diagram shown in figure 1.5.2-4. Bank Customer uses Payment Web Application to do their payments. Payment Web Application captures user data and sends it to a product processor called Tax Payment, which is a Mainframe System. Tax Payment process payment data and sends an interface using a Mainframe Book to send information to Debit Account System intending to debit the value used by user to pay the tax. The payment process includes send an interface to Government Tax Payment Controller to notify tax payment to Government control payer tax payment. Table 1.5.2-5 is shown a description of all ***Conversation*** communications in diagram.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| From | To | Why | What | How |
| Payment Web Application | Tax Payment System | Process tax payment (payment data) | XML | HTTP (Receiver) |
| Tax Payment System | Debit Account System | Debit tax payment value (account number, tax value) | BOOK | CALL (Sender) |
| Tax Payment System | Government tax payment controller | Credit payment (tax payer code, payment code) | TXT | NDM (Receiver) |

1.5.2‑5 – Table representing Conversations shown in figure 1.5.2-4

Systems interchange information to reach its goals. ***Conversation*** communication represents processes or functions created to receive information from and send information to systems.

### Installation

Occasionally a process requires a special kind of equipment connected to a computer that is used to support a process. Examples of such equipment are scanners, bar code readers, smart card readers etc. They are represented in Sys-ADL using ***Device*** structure as introduced in section 1.4.4.

***Installation*** communication represents equipment installed in a computer. It is made connecting ***Device*** structure to a ***Node*** element using a ***Channel*** element as shown in figure 1.5.3-1.

|  |
| --- |
|  |
| ObjInstallation.png  1.5.3‑1 – Sys-ADL Installation representation |
|  |

***Device*** structure represents equipment installed in a node. ***Node*** element of structure represents the hardware and ***Sender*** element represents the physical port where equipment is installed. ***Node*** element in the other side represents the computer where device is being installed.

Any kind of equipment is represented using ***Device*** structure and ***Installation*** communication represents this equipment installed in a computer. In figure 1.5.3-2 Sys-ADL represents a common situation: a scanner installed in a desktop.

|  |
| --- |
|  |
| ObjDeviceInstallationExample.png  1.5.3‑2 – Scanner installed in a Desktop |
|  |

In figure 1.5.3-2 Sys-ADL represents that a scanner is installed in a user desktop through its USB physical port. Scanner is represented by ***Node*** element Scanner, USB port is represented by ***Sender*** element USB and, finally, the user desktop is represented by ***Node*** element user desktop. The connection between scanner and desktop computer is represented by ***Channel*** element scanner installation, which, in this case, represents the USB cable.

A characteristic of ***Installation*** communication is it deals with exclusive installation. Putting in other words, no other equipment is able to use this device without pass through ***Node*** where ***Device*** is installed. Consider a printer that is connected to a computer through a Line Print Terminal (LPT) port, also known as parallel port. No other computer is able to use that printer, except if it is connected to the computer where printer is installed. Sys-ADL diagram fragment in figure 1.5.3-3 shows this situation.

|  |
| --- |
|  |
| ObjInstallationPrinterUserScenario.png  1.5.3‑3 – Desktop uses a Printer through other Desktop |
|  |

In figure 1.5.3-3 there is a printer represented by ***Node*** element Printer which is installed in a desktop represented by ***Node*** element Desktop and printer is connected to desktop through LPT (parallel) port represented by ***Sender*** element LPT. Another desktop represented by ***Node*** element Desktop Printer Client that is connected through Server Message Block (SMB) protocol represented by ***Receiver*** element SMB as shown by ***Network*** communication between Desktop Printer Client and Desktop. ***Network*** communication is discussed in next section. Now, systems hosted in Desktop Printer Client are able to use Printer using the path Desktop Printer Client, Desktop and, finally, Printer.

Special devices are sometimes necessary to perform a process. ***Installation*** communication allows these devices installation be represented in Sys-ADL diagrams.

### Network

In technology information, network is a series of nodes interconnected by communication paths. Computers, printers, mobile phones, TVs and other modern devices are connected through networks and computers around the world are interconnected allowing systems interchange information through these network.

***Network*** communication represents in Sys-ADL a network connection and is represented in Sys-ADL connecting ***Port*** structure to a ***Node*** element using a ***Channel*** element as shown in figure 1.5.4-1

|  |
| --- |
|  |
| ObjNetwork.png  1.5.4‑1 – Sys-ADL Network representation |
|  |

A network connection is composed by a client and a server. The server of connection must provide a connection port in its operational system ready to receive requests of clients. ***Port*** structure represents the server of connection. ***Node*** element of ***Port*** structure represents the server hardware and ***Receiver*** element represents logic system port that will receive client requests. The ***Node*** element in the other side of ***Channel*** element connection is the client of connection.

In figure 1.5.4-2 a Network communication is shown in Sys-ADL diagram fragment.

|  |
| --- |
| ObjNetworkTopologyExample.png |
| 1.5.4‑2 – Network topology |
|  |

Diagram fragment in figure 1.5.4-2 represents system topology and network connections required. Company customer uses a company web application and this requires that his desktop is connected to the web server. To represent this ***Node*** element Customer Desktop has a ***Network*** communication, made with ***Channel*** element internet, with ***Port*** structure composed by ***Node*** element Web Server and ***Receiver*** element HTTP. This means that Customer Desktop is connected to an HTTP port from Web Server using an Internet connection and, in this case, Web Server is the server of this connection.

There is another connection in this process. Application Business Logic process Web Application requests and this requires a Remote Method Invocation (RMI) connection. There is a connection between ***Node*** element Web Application and ***Port*** structure composed by ***Node*** element Application Server and ***Receiver*** element RMI. ***Network*** communication is made through ***Channel*** element labeled lan. This time Web Application is the client of connection and Application Server is the server of connection through RMI port.

Finally, Customer Service Employee needs to access the same information that Company Customer accesses and, to do this, employee desktop is connected to Application Server. To represent this situation, ***Node*** element Employee Desktop has a ***Network*** communication with ***Port*** structure composed by ***Node*** element Application Server and ***Receiver*** element RMI connected with ***Channel*** element labeled lan.

Table 1.5.4-3 shows all connections represented in Sys-ADL diagram fragment.

|  |  |  |  |
| --- | --- | --- | --- |
| Client | Server | Protocol | Network |
| Customer Desktop | Web Server | HTTP | Internet |
| Web Server | Application Server | RMI | Lan |
| Employee Desktop | Application Server | RMI | Lan |

1.5.4‑3 – Network topology representation of diagram shown in figure 1.5.4-2

Distributed application is a very common system architecture pattern. This requires servers and desktops interconnected through networks. Sys-ADL ***Network*** communication is used to represent network topology in system solutions