4.2.6 Interoperability

Return to Non-Functional Requirements

- ☐ **[char]Please Review**

4.2.6.1 About

return to Top

https://www.isko.org/cyclo/interoperability#3.1

Interoperability is a characteristic of a product or system, whose interfaces are completely understood, to work with other products or systems, present or future, in either implementation or access, without any restrictions. However, there are different levels of interoperability. The lowest most basic level of interoperability is at the network level. The degree of interoperability increases in difficulty as one moves up the interoperability levels. The most difficult level is the semantic level, at this level two assets can communicate with little or no prep work using common detailed formal, machine readable Ontologies including a vocabulary of terms.

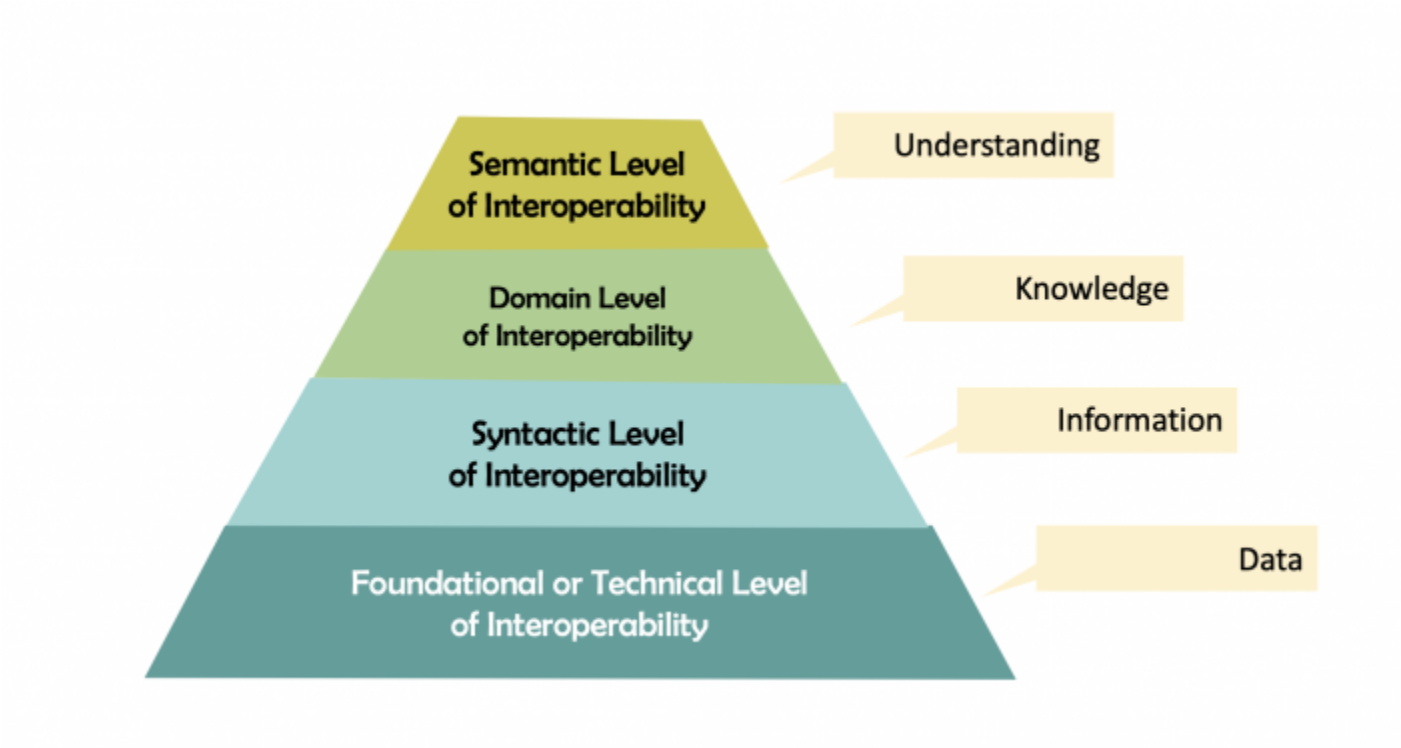


Figure 1: The Automation Pyramid and Interoperability

4.2.6.1.2 Foundational or Technical Level

return to Top

Foundational or Technical Interoperability Level is concerned with the most fundamental and basic kind of interoperability. It provides the “foundation” on which the other kinds of interoperability are built upon. It identifies the technical aspects of peer-to-peer interoperability regardless of hardware, operating system, or middleware platforms. There are two perspectives to the Foundational or Technical Interoperability Level: Data and System.

- From the Data perspective, there is a clear, shared expectation for the contents, context and meaning of that data (e.g., Big-Endian or Little-Endian, or size of an integer, or character set used)
- From the system perspective the differences in computer technology are non-existent (e.g., 16-bit, 32-bit and 64-bit processors. Reduced Instruction Set Computer (RISC) versus Complex Instruction Set Computer (CISC) computers, etc.). In a Distributed System, the network protocols are another part of the Foundational or Technical Level of interoperability (e.g., Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) over Internet Protocol Address (IP Address), etc.).

4.2.6.1.3 Software Manageability Issues

Return to the Top

Over the last few decades, many advances have been made in terms of Open Source Software (OSS) which has to change the way that software was developed, released and used. During this same time period, the traditional Waterfall Model of System and software development has also been largely supplanted with the Agile Model. Many of these changes also have to do with the evolution of systems (or projects) from being Greenfield to Brownfield development and from a “build the world” attitude towards “integrate and glue the world” mindset.

Successful OSS development and adoption not only has to produce products which are solid, strong and robust but also must meet the needs of a Community of Interest (CoI) that has coalesced around a single minded, purpose built, functionality (i.e., Apache Tomcat application server, PostgreSQL Database, Node.js an asynchronous event-driven JavaScript runtime, Docker containerized apps, Kubernetes orchestration engine for containers, etc.). Many of the OSS products are part of many of the successful projects today.

However, its not good enough to just write software and make it publicly available. At the heart of these successful efforts are the well governed, focused, supporting CoIs. There is a desire from almost all systems (or projects) to join the OSS trend but unfortunately, the need for strong governance and rigorous methodology is minimized or skipped in the name of expediency. Fortunately, there is an organization which can help with this called Talk Openly Develop Openly (TODO) (not to be confused with a `to-do`).

TODO organization, though focused on OSS, has written a series of white papers that are well worth studying and using even it your system (or project) is not OSS. One of these papers which is particularly germane to Software Manageability is Tools for managing open source programs [1)]. It is beyond the scope of this document to try to recreate the full content of this white paper. It does present a list of many of the tools available for managing software and how to use them. Here is the table of content from the document:

- Why you need special tools for open source program management
- How to select and plan your tools
- Elements of a basic toolset
- Tools for managing source code
- Tools for tracking project health
- Tools for communications and collaboration
- Tools for corporate-scale GitHub management

Software Manageability Issues

Return to the Top

Over the last few decades, many advances have been made in terms of Open Source Software (OSS) which has to change the way that software was developed, released and used. During this same time period, the traditional Waterfall Model of System and software development has also been largely supplanted with the Agile Model. Many of these changes also have to do with the evolution of systems (or projects) from being Greenfield to Brownfield development and from a “build the world” attitude towards “integrate and glue the world” mindset.

Successful OSS development and adoption not only has to produce products which are solid, strong and robust but also must meet the needs of a Community of Interest (CoI) that has coalesced around a single minded, purpose built, functionality (i.e., Apache Tomcat application server, PostgreSQL Database, Node.js an asynchronous event-driven JavaScript runtime, Docker containerized apps, Kubernetes orchestration engine for containers, etc.). Many of the OSS products are part of many of the successful projects today.

However, its not good enough to just write software and make it publicly available. At the heart of these successful efforts are the well governed, focused, supporting CoIs. There is a desire from almost all systems (or projects) to join the OSS trend but unfortunately, the need for strong governance and rigorous methodology is minimized or skipped in the name of expediency. Fortunately, there is an organization which can help with this called Talk Openly Develop Openly (TODO) (not to be confused with a `to-do`).

TODO organization, though focused on OSS, has written a series of white papers that are well worth studying and using even it your system (or project) is not OSS. One of these papers which is particularly germane to Software Manageability is Tools for managing open source programs [2)]. It is beyond the scope of this document to try to recreate the full content of this white paper. It does present a list of many of the tools available for managing software and how to use them. Here is the table of content from the document:

- Why you need special tools for open source program management
- How to select and plan your tools
- Elements of a basic toolset
- Tools for managing source code
- Tools for tracking project health
- Tools for communications and collaboration
- Tools for corporate-scale GitHub management

Software Manageability Issues

Return to the Top

Over the last few decades, many advances have been made in terms of Open Source Software (OSS) which has to change the way that software was developed, released and used. During this same time period, the traditional Waterfall Model of System and software development has also been largely supplanted with the Agile Model. Many of these changes also have to do with the evolution of systems (or projects) from being Greenfield to Brownfield development and from a "build the world" attitude towards "integrate and glue the world" mindset.

Successful OSS development and adoption not only has to produce products which are solid, strong and robust but also must meet the needs of a Community of Interest (CoI) that has coalesced around a single minded, purpose built, functionality (i.e., Apache Tomcat application server, PostgreSQL Database, Node.js an asynchronous event-driven JavaScript runtime, Docker containerized apps, Kubernetes orchestration engine for containers, etc.). Many of the OSS products are part of many of the successful projects today.

However, its not good enough to just write software and make it publicly available. At the heart of these successful efforts are the well governed, focused, supporting CoIs. There is a desire from almost all systems (or projects) to join the OSS trend but unfortunately, the need for strong governance and rigorous methodology is minimized or skipped in the name of expediency. Fortunately, there is an organization which can help with this called Talk Openly Develop Openly (TODO) (not to be confused with a `to-do`).

TODO organization, though focused on OSS, has written a series of white papers that are well worth studying and using even it your system (or project) is not OSS. One of these papers which is particularly germane to Software Manageability is Tools for managing open source programs [3)]. It is beyond the scope of this document to try to recreate the full content of this white paper. It does present a list of many of the tools available for managing software and how to use them. Here is the table of content from the document:

- Why you need special tools for open source program management
- How to select and plan your tools
- Elements of a basic toolset
- Tools for managing source code
- Tools for tracking project health
- Tools for communications and collaboration
- Tools for corporate-scale GitHub management

Syntactic Level ==== return to Top

Syntax Level is concerned with the correct combination and sequence of the elements in a language and in this context, as it applies to Data Structure or in a Programming Language. For example, in English, we do not say `ball red large plastic`, we would say `large, red plastic ball`. The first form is syntactically incorrect, the second form is not. Interoperability at the syntactical Level means two different systems can exchange information and structurally they are equivalent. The interpretation of

the syntactical data can be different (i.e., big to an ant is different than big to an elephant). The difference in interpretation is a semantic difference (see Semantic Level).

4.2.6.1.4 Domain or Structural Level

return to Top

Domain Level is knowledge of a specific, specialized subject area, discipline or field. For example, there is specific domain knowledge in Chemistry, Organic Chemistry, Physics, mathematics, or statistics. A specific example might be the use of the word `round`. There are completely different meanings for the word in poker, math, cheese, and military. This is in contrast to general knowledge, or domain-independent knowledge which in general would refer to the common usage of the word, sometimes referred to as dictionary usage of the word. The use of the word Domain or Domain knowledge is used within general purpose disciplines such as history, law, systems engineering or even computer science. People with knowledge in the generic area (i.e., law) might have specific domain knowledge in computers, electronics, computers or even specific computer architectures such s Reduced Instruction Set Computer (RISC).

Semantic Level

return to Top

Semantic Level is concerned with the meaning, relationship and restrictions on data and information described in the Domain or Structural and the Syntactical Levels. At this level, computer systems unambiguously exchange information. The information that is exchanged might not be identical but there is a shared understanding between the two systems about the meaning of the information. Some examples of Semantic Interoperability are described by the ability of:[4]:

- Two independent programs with reasoning capability to arrive at the same conclusions from the same data
- An information system, without additional human intervention, to perform the tasks for which it was designed through the exchange of relevant data with other systems[5], independently of how and for what purpose this data was originally collected, stored and managed.
- One system to use data from an external source (without prior planning) and output useful (though not necessarily perfect) results.

DDS Specifics

return to Top

Data Distribution Service (DDS) easily can help in the Interoperability starting at the Fundamental Level through to the Domain Level. It can support the use of tools and products that work at the semantic level, but the actual development of the semantic ontologies is not within the scope of DDS.

- **Foundational or Technical Level** DDS supports Publish-Subscribe over the Ethernet using Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) over Internet Protocol (IP) wired or wireless networks. There is vendor provided support available for other kinds of networks such as Bluetooth and ZigBee. The data is Marshalled across the network using Real-time Publish-Subscribe (RTPS), however, the details of how this are done are done automatically when Data-Centric Publish-Subscribe (DCPS) is used by the Publishers and Subscribers.

- **Syntactic Level** DDS supports a wide variety of scalar datatype as well as collections, Sata Structures and Unions in a standardized formal, software language agnostic method called the Interface Definition Language (IDL). The IDL is then compiled into any number of high-level programming languages such as ISO/IEC C, C++, DDS-Java, DDS-XML and JSON). Therefore, one publisher could be using data defined in C/C++ and a subscriber could be using Java without worrying about the data compatibility at each end. There is even a formal methodology for mapping other languages to IDL.
- **Note:** IDL 4.2 is not part of CORBA and has been elevated to an International Organization for Standardization (ISO) standard [6], however, it is backward compatible to the IDL defined as part of Common Object Request Broker Architecture (CORBA) standard.

- **Domain or Structural Level** DDS support for strong typing, scalar types, data structures and unions have allowed strong domain specific Community of Interest (CoI) to adopt and mandate the use of IDL and DDS. For example, there are specialized domain communities that have adopted IDL and DDS. For example, the mod has mandated a General Vehicle Architecture (GVA) and naesb have mandated an Open Field Message Bus (OpenFMB). See Section 03_mandates

- **Semantic Level** DDS support for semantics is limited to filtering the data that is received by a subscriber or by the matching of Quality of Service (QoS) Policies between publishers and subscribers. For example, a subscriber may apply a content filer to only receive published data within a specified geographic region or that concerns a particlar part number. Content filtering selects data defined within the data structure. There are a rich set of Quality of Service (QoS) Policies allowing filtering on DDS Metadata (see 02_quality_of_service). It is possible to hook up other kinds of filtering based on ontologies, rdf and sparql, but those have not been standardized yet.

[1] [2] [3]
Tools for managing open source programs, Talk Openly Develop Openly (TODO), Accessed 20 July 2020, https://todogroup.org/guides/management-tools/

[4]
Ontology Taxonomy Coordinating WG / OntacGlossary, Accessed 10 July 2020, http://colab.cim3.net/cgi-bin/wiki.pl?OntologyTaxonomyCoordinatingWG%2FOntacGlossary

[5]
Note: mediated by formally specific definitions and axioms

[6]
International Standards Organization (ISO) / International Electrotechnical Commission (IEC), Object Management Group® (OMG®), Information Technology, February 2020, ISO/IEC 19516:2020, https://www.iso.org/standard/65379.html

From:
https://omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://omgwiki.org/dido/doku.php?id=dido:public:ra:1.4_req:2_nonfunc:05_interoperability&rev=1605924610

Last update: **2020/11/20 21:10**