An Open System Architecture Framework for Interoperability (OSAFI)

Article *in* International Journal of Business Information Systems · September 2020

DOI: 10.1504/IJBIS.2020.10052134

3 authors:

Rashmi Jain
Montclair State University
64 PUBLICATIONS **342** CITATIONS
SEE PROFILE

Marina Evrim Johnson
Montclair State University
14 PUBLICATIONS **125** CITATIONS
SEE PROFILE

Abdullah Albizri
Montclair State University
21 PUBLICATIONS **166** CITATIONS
SEE PROFILE

An Open System Architecture Framework for Interoperability (OSAFI)

ABSTRACT

Interoperability of systems is a critical factor for firms to make informed operational and strategic decisions and achieve a competitive edge in the marketplace. As a result, open systems which have a higher level of interoperability with secured and stable operations have significant relevance in today's global economy. Interoperability is accomplished through appropriate system architecture and design. Thus, to achieve the open system interoperability, this paper proposes a framework that looks at system architecture at various levels of abstraction/implementation and identifies the required attributes at each of these levels. This framework can be used as a reference to analyse and determine interoperability requirements at all levels and prioritize the required aspects of interoperability.

1. INTRODUCTION

Many natural disasters and needless loss of human life due to lack of timely and critical communications and information sharing have repeatedly demonstrated the importance of having higher degrees of interoperability between systems and those relying upon them. The need for interoperability is historic, and lack of interoperability results in battlefield failures and substantial financial losses due to lack of common network protocols, shared message formats, and communication channels. Worldwide extended enterprise systems now distributing information, resources, and materials need to interoperate in secure and safe network environments. Also, systems must be deployed, redesigned, and enhanced at faster paces and in a shorter time to address the growing market demands and increased competition. This has driven the need to design systems possessing inherent interoperability, along with efficient deployment of services in networked environments. In order to obtain higher degrees of interoperability, architecture frameworks enabling open attributes is required. Existing literature has pointed out the need to address interoperability issues through architecture and design practices (Bhardwaj et al., 2018; Davis et al., 2002; Klaseen and Cunningham, 1994; Kiljander et al., 2014; Mills, 1993). Key principles of an interoperability architecture for public administration is proposed in European Public Administration Network (2004) (Kubicek and Cimander, 2009). The various types of interoperability for public administration are organizational interoperability, semantic interoperability, technical interoperability, and governance (Sarantis et al., 2008). Numerous Interoperability assessment systems, such as Levels of Information Systems Interoperability (LISI) and The Interoperability Score Model have also been established (C4ISR Architectures Working Group report, 1998; Ford et al., 2007).

As a system and its parts work together, interoperability is represented from different aspects within numerous different contexts such as data, process, applications, functions, components, information, resources, protocols, organizations, and enterprises. Arapi et al. (2007), Igamberdiev et al. (2018), Jardim-Goncalves (2012), and Ponis et al. (2012) discuss the multi-level problem of interoperability in terms of representations, objects, concepts, domains, contexts, and meta-contexts. They also state that high levels of interoperability characterize open systems from the aspects mentioned above, and the prerequisite for designing for interoperability is establishing interconnections. A universally accepted framework for interconnection is the OSI (Open System Interconnection) model. Proponents of the OSI model assert that system interoperability can be achieved through extrapolating or extending the OSI model (Abuelma'atti, et al., 2006). Because the OSI communication stack does not accommodate understanding in communications and system interconnectivity, many researchers emphasize the need for semantic interoperability with multiple layers (Alaya et al., 2015; Jabbar et al., 2017; Kiljander et al., 2014; Pollack and Hodgson, 2004; Yahia et al., 2012). For example, a model proposed by Pollack and Hodgson (2004) contains four layers as described below.

- Syntax Layer: binary format of the application layer message
- Schema Format Layer: physical structure of information classifications
- Referent Layer: relationships among conceptual and implementation schema
- Domain Context Layer: abstract model used to align context.

This paper investigates current interoperability issues and submits a new framework to achieve greater levels of interoperability. An abstract framework for an open system interoperability architecture is designed using a multi-level hierarchical approach similar to OSI model. The multi-levels of architecture abstraction include physical, functional, system, enterprise, and extended enterprise. At each of these levels, the various factors that address all aspects of interoperability such as data, process, and information are identified. The framework also groups these interoperability factors under different interoperability-related attributes of architecture such as security, scalability at each level of architecture abstraction.

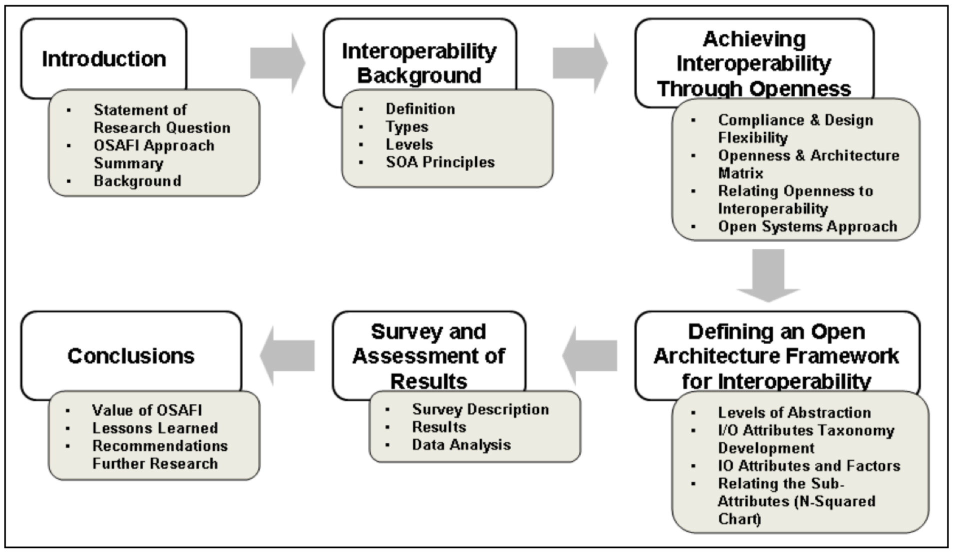

Introduction
Statement of Research Question
OSAFI Approach Summary
Background
Interoperability Background
Definition
Types
Levels
SOA Principles
Achieving Interoperability Through Openness
Compliance & Design Flexibility
Openness & Architecture Matrix
Relating Openness to Interoperability
Open Systems Approach
Defining an Open Architecture Framework for Interoperability
Levels of Abstraction
I/O Attributes Taxonomy Development
IO Attributes and Factors
Relating the Sub-Attributes (N-Squared Chart)
Survey and Assessment of Results
Survey Description
Results
Data Analysis
Conclusions
Value of OSAFI
Lessons Learned
Recommendations
Further Research

Figure 1. Research Methodology and Organization of Paper.

In the following sections of this paper, types and levels of interoperability are described, and architecture attributes supporting interoperability are examined. A framework identifying the required factors of interoperability and related architecture attributes at various levels of architecture abstraction is then proposed. Levels of architectural abstraction are defined along with interoperability related architecture attributes taxonomy. This framework is further converted into a survey tool to assess the level of interoperability at each level of architecture and the corresponding architecture attributes. This survey tool and its pilot validation results are then discussed, and recommendations are provided.

2. INTEROPERTABILITY: DEFINITION, TYPES, LEVELS, & CHALLANGES

The evolution of systems demands scalability and adaptability to interoperate with other existing and new systems. As a result, interoperability becomes a critical and pre-requisite system characteristic and must be of inherent quality. The growing usage of the term interoperability in systems engineering literature over the past few years shows the widespread understanding of its importance and criticality. In order to adequately and clearly define interoperability, the need and its importance have to be studied within various contexts specifically from the systems engineering perspective.

2.1 Interoperability Definitions

There are numerous definitions of interoperability, and this number is probably growing due to the continuing analysis of this problem domain in defence and commercial environments. In a survey on interoperability measurement, Ford et al. (2007**)** have catalogued thirty-four sources having interoperability definitions. A very general and context-independent interpretation of this definition is that interoperability allows components, sub-systems, and systems to (inter)-operate with each other. In other words, interoperability is a result of higher degrees of systems integration to obtain complete system functionality. Interoperability provides an ability of multiple systems or components to exchange information and to use that particular information that has been transferred (A Compilation of IEEE Standard Computer Glossaries, 1991).

Interoperability is the ability to interconnect business-aware software products irrespective of their suppliers, date, and origin, to provide access to corporate data and functionality by any authorized user, and to maintain theta interconnection and access over changes in suppliers, date, and origin, where business-aware software provides functions that are characteristics of that particular business (The Bellcore OSCA^{TM} Architecture, 1999). Interoperability is also defined as a collection of communicating entities to share specified information according to a common operational semantics (Alaya et al., 2015, Brownsword et al., 2004; Carney et al., 2005; Kundu and Tyagi, 2017).

The Authoritative Dictionary of IEEE Standards Terms (2000) distinguishes "interoperability" as "software interoperability and "hardware interoperability" and relates the interoperability to "**compatibility**" and "**conformance.**"

1. ***Software interoperability***: The ability of two or more systems or elements to exchange information and to use the information that has been exchanged.
2. ***Hardware interoperability***: The capability for units of equipment to work together to do useful functions.

3. The ***capability***: promoted but not guaranteed by joint conformance with a given set of standards, that enables heterogeneous equipment, generally built by various vendors, to work together in a network environment.
4. The ***ability*** of two or more systems or components to exchange information in a heterogeneous network and use that information.

These definitions given by IEEE focus on how to accomplish or a means of achieving interoperability. On the other hand, Hastings and McManus (2006) provide a solution-independent definition that captures all aspects of interoperability. According to these researchers, "Interoperability can be defined as the ability of the system to "play well with others," both with systems it was originally designed to work with, and with future systems; may be desirable in and of itself; also enhances versatility, flexibility, and evolvability of systems of systems".

2.2 Interoperability Types

There have been numerous types of interoperability documented in various contexts. For example, Ford et al. (2007) have provided a reference listing of sixty-four different interoperability types, including operational interoperability, technical interoperability, coalition interoperability, and constructive interoperability. Pollack and Hodgson (2004) emphasize data understanding and semantics and provide a different viewpoint regarding interoperability types as illustrated in Table 1.

Table 1. Interoperability (IO) Types Increase in Complexity Structure (Pollack and Hodgson, 2004, pp. 43-44)

IO Type	Definition
Data	Semantic interoperability of data enables data to maintain original meaning across multiple business contexts, data structures, and schema types by using data meaning as the basis for transformations.
Process	Semantic interoperability of process enables specific business processes to be expressed in terms of another by inferring meaning from the process models and contextual metadata and applying it in a different process model elsewhere or outside the organization.
Services/ Interface	Semantic interoperability of services enables a service to look up, bind, and meaningfully communicate with a new service without writing custom code in advance.
Application	Semantic interoperability of applications enables the granular interactions of methods, transactions, and API calls between heterogeneous software applications to be platform independent.
Taxonomy	Semantic interoperability of taxonomy enables any category to be expressed in terms of other categories by leveraging the intended meaning behind the category definitions.
Policy	Semantic interoperability of policies and rules enables businesses to protect valuable resources regardless of what technologies their security mechanisms have been implemented in or how complex the rights management issues have become.

IO Type	Definition
Social Network	Semantic interoperability of social networks enables people in different communities of interest to network, infer, and discover meaningful connections through previously unknown contacts and interests.

2.3 Interoperability Levels

Table 2 provides different levels of interoperability based on various interoperability assessment models. Usually, interoperability levels are associated with interoperability attributes used in the model (Wassermann and Fay, 2017). Please note that some models, such as the System of Systems Interoperability (SOSI) Model proposed by Morris et al. (2004) do not have any levels.

Table 2. Levels of interoperability (IO) differ amongst Assessment Models (Ford et al. 2007)

Level or Layer	**Levels of Information System IO (LISI) 1998**	**Layers of Coalition IO (LCI) 2003**	**NATO C3 Technical Architecture Reference Model for IO (NMI) 2003**	**Organizational IO Agility Model (OIAM) 2005**	**Levels of Conceptual IO Model (LCIM) 2006**
0	Isolated			Static	None
1	Connected	Physical	Unstructured data exchange	Amenable	Technical
2	Functional	Protocol	Structured data exchange	Accommodating	Syntactic
3	Domain	Data/Object Model	Seamless sharing of data	Open	Semantic
4	Enterprise	Information	Seamless sharing of information	Dynamic	Pragmatic
5		Knowledge/ Awareness			Dynamic
6		Aligned Procedures			Conceptual
7		Aligned Operations			
8		Harmonized/ Strategy Doctrines			
9		Political Objectives			

2.4 Challenges of Interoperability

Interoperability is a challenge and difficult to achieve in practice, whether the goal is to increase interoperability between systems that originally did not interact, or to architect new systems designed to interoperate at the enterprise level from inception (Petcu, 2011; Steel et al., 2012). The advent of the extended enterprise, where multiple disparate enterprise architectures must interoperate to achieve shared commercial objectives presents a complex interoperability challenge. Service-Oriented Architectures (SOA) is an approach poised to meet this challenge (Papazoglu and Van den Heuvel, 2007). Maintaining interoperability with legacy systems as new systems are deployed sometimes conflicts with achieving greater levels of interoperability among the newer systems. This necessitates trade-offs, and in some cases, decisions are made to accept reduced interoperability between older legacy and newer systems. Interoperability is also an issue for Commercial-off-the-shelf (COTS) based systems integration (Bhuta and Boehm, 2007; Jain et al., 2010; Tu et al., 2002). Standards and architecture play an important role in the interoperability of COTS-based systems integration. Architecture also influences the interoperability of both COTS and legacy components. Component interoperability has become a concern because companies integrate COTS products and assemble modules from various sources into a single application. Despite these challenges, interoperability has been addressed through some common guiding principles such as compliance to interface standards, deriving system interoperability requirements and testing against them, use of standardized communication interfaces and middleware, and implementation of SOA.

Within the context of a SOA, Erl (2016) describes the concept of inherently interoperable services. SOA allows architects to implement service descriptions and messages that are highly standardized, resulting in "intrinsic interoperability" where application customization is reduced, and the degree of modelling is enhanced. Erl (2007) establishes SOA Principles instead of interoperability levels or attributes and relates them to intrinsic interoperability as shown in Table 3.

Table 3. SOA Principles Supporting Intrinsic Interoperability (Erl 2007, pp. 363)

Principle SOA	Relationship
Standardized Service Contract	The fact that service contracts are consistently standardized guarantees a baseline measure of interoperability because of natural compatibility between data models defined in technical service contracts.
Service Loose Coupling	Reducing the amount of required service coupling fosters interoperability by making individual services less dependent on each other and therefore more open to sharing data with different service consumers.
Service Abstraction	Service Reusability considerations naturally increase interoperability as they outfit services with design characteristics geared for repeated usage by numerous service consumers (with which reusable services will need to effectively interoperate).
Service Autonomy	By increasing a service's autonomy, it establishes itself as a more reliable enterprise resource with predictable runtime behavior. This, in turn, increases its attainable level of interoperability
Service Statelessness	Through an emphasis on stateless design, the availability and scalability of services increase, allowing them to interoperate more frequently and reliably.

Service Discoverability	To enable interoperability between a service consumer and a service, the appropriate service must first be located. Therefore, the application of the Service Discoverability principle increases the chances for a service to maximize its interoperability potential.
Service Composability	For services to be repeatedly composable, they must be highly interoperable. Therefore, shaping each service into an effective composition member increases its native ability to interoperate with others.

SOA is a more specialized concept, applicable to the enterprise level of architecture and, as its name implies, the principles are strictly related to services without any supposition of a hierarchal structure of components. SOA services, although composable, are flat and assume support from the lower levels of the OSI model. The accomplishment of Erl's principles for intrinsic interoperability requires careful design activities to be made early in the SOA deployment process to attain the benefits.

3. INTEROPERABILITY THROUGH ARCHITECTURES: BALANCING COMPLIANCE AND DESIGN FLEXIBILITY

One typical pattern in addressing the need for interoperability in a system is through the compliance requirements and the system architecture. Thus, this section researches the relationships between interoperability, standards, system architecture, and integration. The primary goal here is to find out if adherence to standards leads to better interoperability, whether interoperability can be addressed through architecture, and how architecture influence interoperability. When these questions were studied in detail, it was found that interoperability can be best achieved by two approaches, one by maintaining compliance and another by implementing flexible designs.

Compliance can be defined as conformance to standards and regulations, which indicates that such systems or components meet the requirements specified by standards and regulations. The International Organization for Standardization (ISO) differentiates between standards and regulations. ***A standard*** is a document approved by a recognized body, that provides, for common and repeated use, rules, guidelines, or characteristics for products, processes or services with which compliance is not mandatory. On the other hand, ***a regulation*** is a document, which lays down product, process, or service characteristics, including the applicable administrative provisions, with which compliance is mandatory. Conformance (for software and hardware) to standards indicates that such hardware or software meets the requirements specified by a standard. COTS-based systems help this by introducing more standardized components into the marketplace that are widely adopted.

Compliance results in higher compatibility among systems and system components. Compatibility is sometimes used synonymously to "interoperability." Compatibility is defined as the ability of two or more systems or components to perform their required functions while sharing the same hardware or software environment (The Authoritative Dictionary of IEEE Standards Terms, 2000). Compatibility (for hardware) is defined as the degree to which devices may be interconnected and used, without modification, when designed as defined throughout a specified standard. (i.e., mechanical, electrical, or functional compatibility). Compatibility (for programmable devices) is the degree to which devices may be interconnected and used, without modification, when designed as defined throughout a specified standard. Flexible designs promote both "flexibility" and "adaptability". Flexibility characterizes system's ability to be easily changed or modified. In contrast, adaptability describes a system's ability to adapt itself towards changing operating environments.

Figure 2 shows the degree of interoperability that can be achieved by adopting the two approaches, namely, compliance and design flexibility. The y-axis shows the degree of compliance and x-axis shows the degree of "design flexibility." The resulting benefit or effect of compliance and flexibility are shown on the opposite side of the chart. For example, uniformity resulting from compliance is shown on the opposite side of the compliance (Y) axis. By having a greater level of compliance, a uniform design is obtained – conversely, a lower level of compliance results in a lower level of uniformity in architecture and design. Flexible designs promote innovation. Innovation is a by-product of flexible architecture and design.

In the left bottom quadrant (Figure 2) of the matrix, the degree of compliance and the degree of "design flexibility" is low, indicating that the systems are "isolated". Isolated systems are not designed based on standards, nor are their designs flexible and adaptable to achieve a level of interoperability. Once the degree of compliance increases, systems become "standardized," as shown in the upper left quadrant, indicating that systems and their components are designed conforming to standards. At this level of compliance, even though the design flexibility is low, standardized systems can provide some level of interoperability. On the other hand, once the degree of "design flexibility" increases while the degree of compliance is on the lower side of the matrix, the systems are "customized" as shown in the right bottom quadrant. This level of customization shows that systems are designed to maintain flexibility and adaptability but are not necessarily compliant with standards. At this level of customization, without a significant degree of compliance to standards, some level of interoperability can be obtained.

"Open" systems provide the maximum level of interoperability as shown in the top right quadrant of the matrix. Open systems comply with open standards allowing for design flexibility at the same time. Open systems provide the maximum level of flexibility and therefore are preferred over other levels of interoperability of the systems. To achieve openness, a system is required to comply with standards, regulations, and other regulatory requirements that support open systems while simultaneously adopting flexible system designs. Open systems support different aspects of openness, namely, interoperability, flexibility, adaptability, and adherence to open standards. According to this analysis and Figure 2, interoperability can be best achieved by two approaches, one by maintaining compliance and another by implementing flexible designs. Use of a local data vocabulary could be viewed as customization; therefore, the open architecture matrix as shown in Figure 2 is not perfect. In the context depicted above, openness is ideal – there is a realization that with autonomy, a single homogeneous architecture is impractical, and universal compliance to standards impossible.

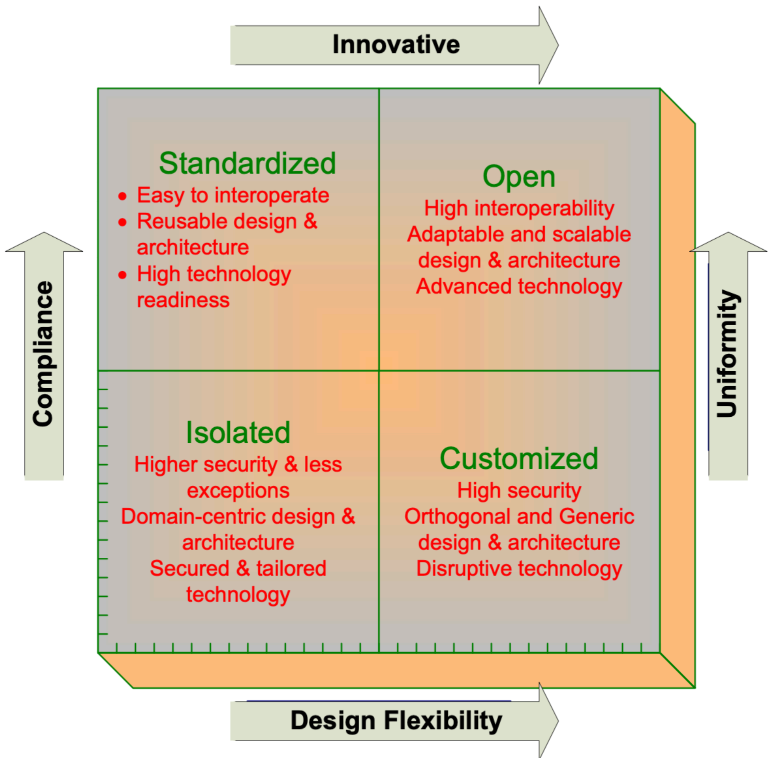

Innovative
Standardized
Easy to interoperate
Reusable design & architecture
High technology readiness
Open
High interoperability
Adaptable and scalable design & architecture
Advanced technology
Compliance
Uniformity
Isolated
Higher security & less exceptions
Domain-centric design & architecture
Secured & tailored technology
Customized
High security
Orthogonal and Generic design & architecture
Disruptive technology
Design Flexibility

Figure 2. Open Architecture Matrix

4. INTERCONNECTION TO INTEROPERABILITY ARCHITECTURES AND OPEN VS. CLOSED SYSTEMS AND THEIR RELATIONSHIOPS TO INTEROPERABILITY

Interoperability can be viewed from different levels. When two systems are capable of exchanging information, they are interconnected. When two interconnected systems are also capable of understanding and processing the information exchanged, they are defined as interoperable. Therefore, Interconnectivity is a means to an end, where interoperability is the goal (Gravina et al., 2018; Klaseen and Cunningham, 1994). A necessary step in achieving interoperability is to define an interconnection architecture comprising the protocols used to transfer data between systems. Additionally, an interconnection architecture must define various details relating to the infrastructure that supports the data transfer protocols, such as a directory schema, and system and network resource management (Czarnecki and Spiliopoulou, 2012; Guijarro, 2007; Klaseen and Cunningham, 1994). The second necessary step in achieving interoperability is to reach a common understanding of how data is to be interpreted and used, and what each function and sub-system can do (Haslhofer and Klas, 2010; Klaseen and Cunningham, 1994). Functions of one layer of architecture are decoupled from the functions of the other layer. A high enough level of decoupling can be expected to support interoperability (Alqaoud, 2010; Mills, 1993; OSCA^{TM}, 1999). In order to have integrated operations, a loosely coupled system is required to be frequently bound together by a common communication and processing protocol. Interoperability must be provided among building blocks (functional and physical) for better operational control (Mills, 1993). Interoperability also provides operational flexibility (Dai et al., 2015; Klaseen and Cunningham, 1994).

Interoperability can be considered as the next generation or a more advanced or evolved form of interconnectivity. To achieve open interconnections, the OSI model is predominantly used. But as the context of interconnection is moving from applications to systems, enterprises, and businesses, there is a need to extend this model to address the interoperability between the extended contexts. By studying the various existing levels of interoperability and the means by which interoperability is achieved through architecture, this study develops an Open System Architecture Framework for Interoperability (OSAFI), which is the focus of the following sections.

An open system is a system that can exchange energy, material, and information with its environment continuingly. Such exchange is enabled through the use of open (i.e., well defined, widely used and consensus-based) standards, protocols, languages, and data formats in developing systems. The focus of attention in an open system is on key interfaces (Gillis, 1999). An interface is designated as a key interface when the technology turnover is rapid and design risk is high on either side of the interface, and/or the system elements on one or both sides of the interface exhibit a high failure rate or are costly. Use of an open standard is the preferred method for implementing a key interface (OSJTF, 1996).

A closed system is characterized by closely held, privately owned standards, protocols, languages, and data formats that are either unavailable to outsiders or are available only at a very high license fee. Closed systems also include those that were designed by a single company for a single program or a small number of programs. In contrast, an *open* system is a system designed using a collection of interacting and integrated software, hardware and human components that are based on consensus-based, *de jure* or if not available, *de facto* standards that are easily accessible to all interested parties (OSJTF, 1996). Table 4 summarizes some critical distinctions between open and closed systems and relates the open system characteristic to interoperability by assessing the degree of correlation to interoperability along with the applicable architectural abstraction level.

Table 4. Characteristics of Open and Closed System

Characteristics	**Closed System**	**Open System**	**Correlation Between Characteristics & IO and Applicable Architecture Level**
Interfaces, languages, and data formats & protocols	Closely held and private	Openly and widely held and publicly available	High correlation at all levels of architecture. Being publicly available may violate security concerns and policies at system level and above.

Critical Importance	Given to unique design and implementation	Given to interfaces management and widely used conventions	High correlation at all levels of architecture. Existing Internet protocols provide commonality.
Modularity	Less emphasis	Heavy emphasis	Low Correlation at all levels of architecture. Mostly applicable to upgrade with minimum disruption.
Vendor and Technology Dependence	Very dependent	Very independent	Medium correlation at system level and below. COTS and IT standards ubiquitous and interoperable. Vendors must comply with standards.
Number of implementation and interfaces	Minimization of the number of implementations	Minimization of the number of types of interfaces	High Correlation especially at functional level for streamlined workflows. At enterprise level and above, this may enhance isolation and redundancy.
System Integration	Difficult and more costly integration	Easier and more cost-effective integration	High correlation at all levels. Interoperability built into integration procedures.
Portability, connectivity, interoperability, and scalability	Low	High	High Correlation at system level and above.
Vendor Usage	Use of sole-source vendor	Use of multiple vendors	Medium correlation at all levels. More vendor choices exist at lower architectural level.
Expansion and Upgrading	Requires considerable time, money, and effort	Easier, quicker, and less expensive	Low correlation at all levels. Maintaining interoperability usually limits choices and places. Constraints on expansion.
Cost of Ownership	Higher total cost	Lower total cost	Low correlation at all levels. But there is cost of interoperability at all levels.
Technology Transfer	Slower and more costly	Faster and less costly	Low Correlation at all levels. This is supported by modularity. Interoperability usually drives need for technology transfer.

Components, interfaces, standards, and implementations	Selected sequentially	Selected interactively	High Correlation at all levels. There is need to know compatibility with other systems and enterprises.
Life Expectancy	Shorter	Longer	High correlation especially at the system level and below. Better IO means longer life expectancy.
Adaptability	Less adaptable to change in threats and technologies	More adaptable to evolving threats and technologies	High correlation at all levels. IO link to changing standards.
Primary Focus	Focusing mostly on development cost and meeting present mission	Focusing on total costs of ownership, sustainment, and growth	Medium correlation at all levels of architecture.
User's Role	User as the producer of systems	User as the consumer of components	High correlation at all levels. Applicable to system level and above, especially SOA.
System Influence	Rigid and slow	Real time and cybernetic	High correlation especially where governance is manifested (e.g., above system level).
Relationship with prime contractors, suppliers, and vendors	Adversarial	Symbiotic	High correlation especially at Extended Enterprise level of abstraction.

5. OPEN SYSTEMS ARCHITECTURE FRAMEWORK FOR INTEROPERABILITY

The Open Systems Joint Task Force has defined an open systems approach in their draft (version 1.0, October, 2001) and subsequent versions as *"...an integrated business and technical strategy that employs a modular design and, where appropriate, defines key interfaces using widely supported, consensus-based standards that are published and maintained by a recognized industrial standards organization."* Some of the objectives that the open systems strategy aims to achieve are as follows:

- adapt to evolving requirements and threats
- facilitate systems integration
- reduce the development cycle time and total life-cycle cost
- ensure that the system will be fully interoperable with all the systems which it must interface, without major modification of existing components
- enhance commonality and reuse of components among systems
- mitigate the risks associated with technology obsolescence
- mitigate the risk of a single source of supply over the life of a system
- enhance life-cycle supportability

In order to address the openness of systems as described above, there are some architectural and design considerations that need to be provided during system design. It is these architecture and design considerations that will support and preserve interoperability as defined for a given system. The authors of this paper used such architectural and design considerations as the basis to develop a framework. The proposed framework could be used as a reference to explore the requirements of interoperability at each level of architecture abstraction.

As a first step in developing this framework a collection of various aspects of interoperability such as data, process, resources have been analysed by exploring the various interoperability models such as, Levels of Information System Interoperability (LISI) (C4ISR Architectures Working Group report, 1998), Organizational Interoperability Maturity Model (OIM) (Clark and Jones, 1999), NATO C3 Technical Architecture (NC3TA) Reference Model for Interoperability (NATO Allied Data Publication, 2003), Levels of Conceptual Interoperability (LCIM) (Tolk and Muguira, 2003), Layers of Coalition Interoperability, System of Systems Interoperability (SOSI) Model (Morris et al., 2004), and Basic Interoperability Data Model (BIDM) (IEEE 1420.1, 1995). Then, the authors have further explored the factors of architecture that can influence interoperability such as interfaces, protocols (standard), data, and layers (building blocks) (Klaseen and Sydir, 1995; Klaseen and Cunningham, 1994; Mills, 1993; Janssen, 2012). It is important to note that how these factors are addressed in architectures can vary. and some examples include communication between components and interfaces, transfer data between systems, as well as semantic integrity constraints/consistency. Afterward, the authors have categorized these factors in terms of how they would relate to different levels of architecture abstractions and finally identified the requirements for the factors at each level of architecture abstraction.

6. LEVELS OF ARCHITECTURE ABSTRACTION: INTEROPERABILITY & INTERFACES

A systems approach to architecture and design emphasizes on both abstraction and detailed design. Abstraction supports a holistic design and better understanding of interfaces and interoperability at each level. The authors adopt a 5-level architecture abstraction to address how to achieve and implement interoperability at each level. These levels of architecture abstraction are shown in Figure 3. In addition to the five levels of abstraction, interoperability related architecture attributes are listed. These attributes are applied to each level of architecture abstraction and will be discussed later in more detail. This method follows the approach taken by some other interoperability models as discussed, but instead of using levels of interoperability, architecture levels are employed. Since different architectural approaches are accommodated by this proposed method, the aspect of multiple views is shown (e.g., some views of system architecture contain an organizational architecture in addition to the physical and functional, but in this case, organizational aspects are reserved for the enterprise level).

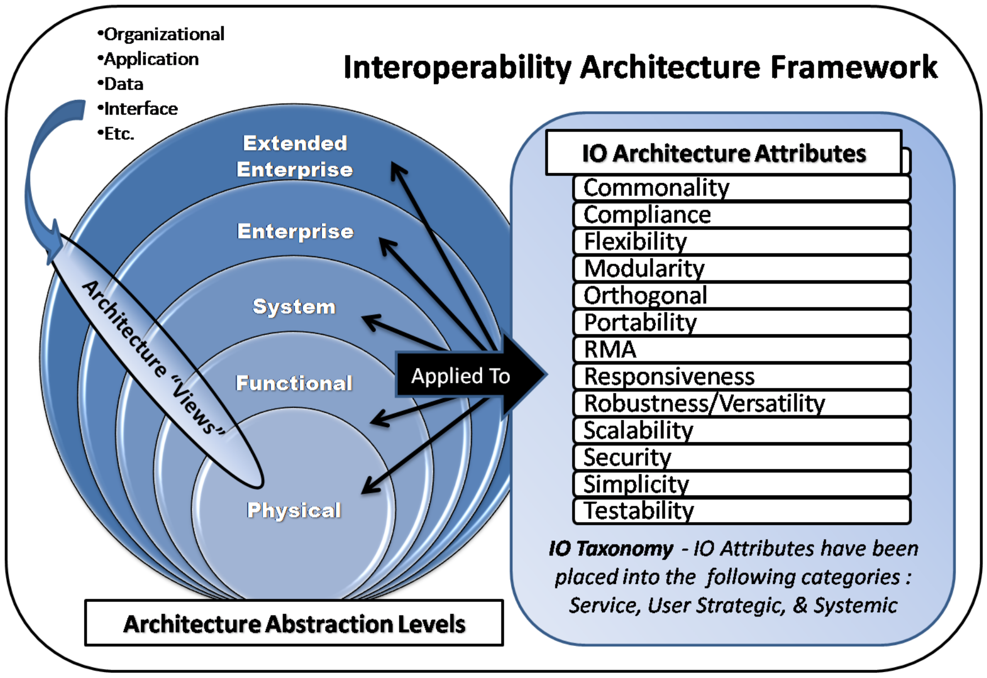


Figure 3. Architecture Levels of Abstraction and Attributes Comprise the OSAFI.

At the ***physical architecture, which is the lowest level of architecture abstraction,*** specifications of physical components and parts are provided, and most of the interoperability issues and implementations are addressed. Physical architecture defines how the physical components and parts work and integrate. It also describes the partitioned elements of the system with complete definitions of the performance characteristics of the resources. Physical interoperability occurs between the physical components and parts of the system. Physical interoperability is the capability for physical components and parts of the system to operate together to do useful functions.

The next level of architecture abstraction is ***functional architecture***. It is a logical model that captures the transformation of inputs into outputs using control information. This definition adds the flow of inputs and outputs throughout the functional decomposition; these items that comprise the inputs and outputs are commonly modelled via a data model. An IDEF0 model without any mechanisms can be used as the modelling technique to represent the functional architecture at this level of detail. In other words, IDEF0 is a logical model of a functional decomposition plus the flow of inputs and outputs, to which input/output requirements have been traced to specific functions and items (inputs, outputs, and controls). Functional interoperability is observed between the physical parts and components of the system when they are integrated and work together. Functional interoperability is the ability of physical components and parts of the system to function together, exchange information, and use the exchanged information.

The physical and functional architecture make up the ***system architecture***. The system architecture is the structure in terms of components, connections, and constraints of a product, process, or element (Maier and Rechtin, 2009). System architecture is a logical construct for defining and controlling the interfaces and the integration of all the components of the system (Zachman, 1987). It is the organizational structure of a system or component, their relationships, and the principles and guidelines governing their design and evolution over time (IEEE 610.12-1990). In other words, it is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution (Hilliard, 2000). Systems interoperability is the capability of systems to work together to do required functions, exchange information, and use the exchanged information when integrated with other systems and its environment. Systems interoperability requires compatibility and ability to function together with other systems, both with systems it was originally designed to work with, and with future systems.

An ***enterprise*** can also be viewed as a system of systems, and enterprise architecture describes how enterprise systems interoperate. Enterprise Architecture is about understanding all of the different elements making up the enterprise and how those elements inter-relate. An enterprise in this context is any collection of organizations that has a common set of goals/principles and/or single bottom line. In that sense, an enterprise can be a whole corporation, a division of a corporation, a government organization, a single department, or a network of geographically distant organizations linked together by common objectives. Enterprise interoperability is the level of interoperability between the systems within an enterprise. Enterprise interoperability demonstrates the ability of enterprise systems to provide services to and accept services from other systems within the enterprise and to operate effectively together.

Extended enterprise architecture describes how an enterprise and its suppliers and partners are linked together by information flows, integrating knowledge, design, and production. These arrangements are commonly long-term and/or permanent and represented as Extended Enterprises (E2) architectures. Extended enterprise architectures could take the form of a Business to Business (B2B) or Business to Customers (B2C) architecture or several others that exist today. Interoperability addressed through extended enterprise architecture facilitates business interactions beyond the boundaries of an enterprise. This may be termed as extended business interoperability. Due to emerging interconnectivity needs in the business and technology world, enterprise systems are required to integrate with the systems outside the enterprise. Business interoperability is the ability of enterprise systems to provide services to and accept services from other systems outside the enterprise and to operate effectively together.

In order to achieve higher levels of interoperability, all the requirements and factors of interoperability need to be addressed at all the levels of architecture abstraction discussed above. Therefore, the authors proposed a framework called "The Open System Interoperability Architecture framework (OSIAF)," which can be used to identify such requirements and factors of interoperability at various levels of architecture abstraction. To design this framework, first a subset of architecture attributes that can be utilized to resolve interoperability issues have been identified by reviewing the existing research regarding the factors that provide good and reliable architecture (Abuelma'atti et al., 2006; Arapi et al., 2007; EPAN, 2004; Jain et al., 2008; Keshav and Gamble, 1998; Klaseen and Cunningham, 1994; Mills, 1993) Then, the factors of good architecture have been compared with attributes of interoperability architectures. Based on this analysis, the authors have identified the attributes that focus on interoperability issues as listed in Table 5.

Table 5. Important Interoperability Attributes.

Attribute Name	**IO Applicability**
1. Commonality	Aims to establish commonality between systems
2. Compliance	Aims to help towards complexly developed business applications to be interoperable and ensure data and application integration across different platforms such as the cloud and on-premise data centres.
3. Flexibility	Allows introduction of new elements into the system; thus, allowing to maintain interoperability.
4. Modularity	Provides rapid and flexible development, thus allows interoperability among different systems
5. Orthogonal	Changes in one part of a system should occur without changing other parts
6. Portability	Enables organizations to move data among several resources and provide a degree of interoperability
7. Reliability, Maintainability, & Availability (RMA)	Provides the baseline for architecture components to interoperate with a larger system. It is expected that system architecture should reach the requires RMA level to achieve interoperability
8. Responsiveness	Allows a prompt reaction to changing business needs increase the level of interoperability (e.g., adaptation of cloud systems and their existence and co-operation with on-premise data centres)
9. Robustness/Versatility	Helps systems to operate near normalcy in the presence of errors and faults; thus, providing system interoperability during unexpected adverse environments
10. Scalability	Offers high security and massive scale between devices, applications, and systems. With the rise of Internet of Things (IoT) and Artificial Intelligence (AI), this is important to realize full interoperability
11. Security	Provides the rules and boundaries for data and information sharing (e.g., interorganizational) and facilitates security management operations; thus, enables interoperability of

	multiple systems at various levels.
12. Simplicity	Achieves IO with more straightforward and simpler set-up
13. Testability	Facilitates the definition of a method that enables testing the level of achievement of interoperability improvement

To form an interoperability architecture attribute taxonomy, the authors have categorized the above interoperability attributes into four higher-level categories (i.e., service, user, strategic, and systemic) as given in Figure 4 (Williams, 2000).

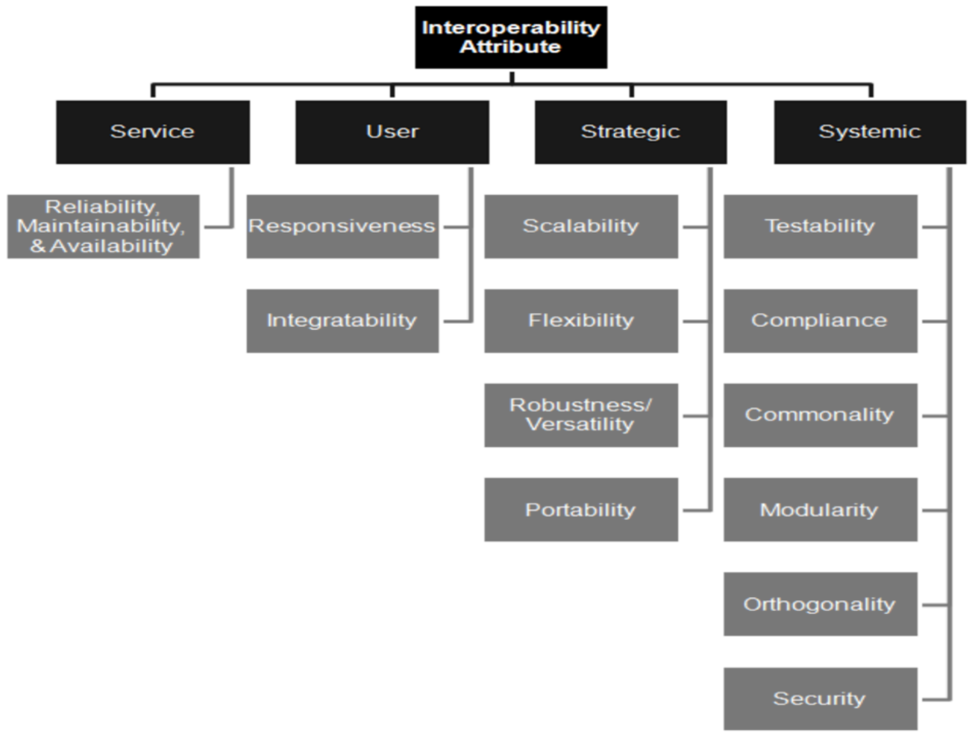


Figure 4. Top-Level Interoperability Architecture Attribute Taxonomy

The OSAFI framework as shown in Table 6 provides the architecture attributes and the factors and requirements of interoperability at five levels of architecture abstraction. For instance, the factors of interoperability such as uniformity, decomposition, and configuration level result in commonality at system-level architecture, leading to increased system interoperability. The columns in the table represent the different levels of granularity of architecture. The rows list the attributes of good architecture. The cells are the factors of interoperability needed for each attribute and its corresponding architecture to enable an open system design. Each level of architecture inherits the interoperability factors of the subset or lower-level architecture. However, if these interoperability factors are not addressed in the lower levels, they constrain the level of interoperability at the higher levels and may impact other attributes of the architecture. Please note that the scope of the table does not include cost and training.

Table 6. OSAFI: Interoperability Architecture Attributes and Factors

Attributes of Good Architecture	How interoperability of open system is addressed through attributes of architecture at different levels of granularity?				
	Extended Enterprise	**Enterprise**	**System**	**Functional**	**Physical**
Commonality	▪ Uniformity ▪ Organization structure & Processes ▪ Governance	▪ Uniformity ▪ Organization structure & Processes ▪ Governance ▪ Configuration	▪ Uniformity ▪ Decomposition ▪ Configuration	▪ Uniformity ▪ Decomposition ▪ COTS/Non-Development Items	▪ Uniformity ▪ Decomposition
Compliance	▪ Data ▪ Standards ▪ Throughput Levels of service agreements ▪ Ontology	▪ Data ▪ Standards ▪ Throughput Levels of service agreements ▪ Ontology	▪ Data ▪ Standards ▪ Throughput ▪ Levels of service agreements ▪ Semantics ▪ Safety	▪ Data ▪ Standards ▪ Throughput ▪ Levels of service agreements ▪ Semantics ▪ Safety	▪ Data ▪ Standards ▪ Throughput ▪ Levels of service agreements ▪ Syntactic ▪ Safety
Flexibility	▪ Configuration ▪ Organization structure & processes ▪ Governance	▪ Configuration ▪ Organization structure & processes ▪ Governance ▪ Maturity	▪ Configuration ▪ Decomposition ▪ Orthogonal ▪ Maturity	▪ Configuration ▪ Decomposition ▪ Orthogonal ▪ Technology	▪ Configuration ▪ Decomposition ▪ Orthogonal ▪ Technology
Modularity	▪ Organization structure & processes ▪ Governance	▪ Organization structure & processes ▪ Governance ▪ Configuration ▪ Maturity ▪ Vendor management	▪ Decomposition ▪ Configuration ▪ Maturity ▪ Vendor management	▪ Decomposition	▪ Decomposition
Orthogonal	▪ Integrated process ▪ Vertical dependency between goal, objects, and business requirements	▪ Integrated process ▪ Vertical dependency between goal, objects, and business requirements	▪ Services independent evolution	▪ Inheritance (dependency) ▪ Rationalized allocation ▪ Functional cohesiveness	▪ Inherence (dependency) ▪ Consistency in allocation
Portability	▪ Organization structure & process ▪ COP[1] ▪ Governance	▪ Organization structure & process ▪ COP ▪ Consistency	▪ COP ▪ Commonality ▪ User interface	▪ Reuse ▪ Modularity ▪ Common platform	▪ Reuse ▪ Modularity
RMA	▪ Dependencies ▪ Performance ▪ Serviceability ▪ Latency ▪ Recovery ▪ Capability	▪ Dependencies ▪ Performance ▪ Serviceability ▪ Latency ▪ Recovery ▪ Capability	▪ Dependencies ▪ Performance ▪ Serviceability ▪ Latency ▪ Recovery ▪ Capability	▪ Dependencies ▪ Performance ▪ Serviceability ▪ Latency ▪ Recovery ▪ Capability	▪ Dependencies ▪ Performance ▪ Serviceability ▪ Latency ▪ Recovery

[1] Common Operating Picture

Attributes of Good Architecture	How interoperability of open system is addressed through attributes of architecture at different levels of granularity?				
	Extended Enterprise	**Enterprise**	**System**	**Functional**	**Physical**
Responsiveness	▪ Disaster recovery ▪ Contingency plan ▪ Archiving ▪ Business continuity plan /Mission contingency plan ▪ Business impact analysis	▪ Disaster recovery ▪ Contingency plan ▪ Archiving ▪ Business continuity plan /mission contingency plan ▪ Business impact analysis	▪ Disaster recovery ▪ Contingency plan ▪ Archiving	▪ Disaster recovery ▪ Contingency plan ▪ Archiving ▪ Synchronization	▪ Disaster recovery ▪ Contingency plan ▪ Archiving
Robustness - Versatility	▪ Versatility in handling unpredictable environments ▪ Agility ▪ Adaptability	▪ Versatility in handling unpredictable environments ▪ Agility ▪ Adaptability	▪ Versatility in handling unpredictable environments ▪ Virtual prototyping ▪ Virtual testing ▪ Safe mode operation ▪ Exception handling ▪ Baseline operation ▪ Modelling and simulation ▪ Self-healing/ adapting ▪ Intelligence capability	▪ Virtual prototyping ▪ Virtual testing ▪ Safe mode operation ▪ Exception handling ▪ Baseline operation	▪ Virtual prototyping ▪ Virtual testing Protection layer (safeguard)
Scalability	▪ Performance ▪ Capability ▪ Serviceability	▪ Performance ▪ Capability ▪ Serviceability	▪ Performance ▪ Capability ▪ Serviceability ▪ Configuration	▪ Performance ▪ Capability ▪ Configuration	▪ Performance ▪ Configuration
Security	▪ Access control ▪ Assurance ▪ Resilience ▪ Policy ▪ Technical	▪ Access control ▪ Assurance ▪ Standards ▪ Resilience ▪ Policy ▪ Technical ▪ OPSEC (Operational Security)	▪ Access control ▪ Assurance ▪ Standards ▪ Resilience ▪ OPSEC ▪ Security services[2]	▪ Access control ▪ Assurance ▪ Standards ▪ Performance ▪ Recovery ▪ Auditing ▪ Security services	▪ Access control ▪ Standards ▪ Performance ▪ Recovery ▪ Security services
Simplicity	▪ User interface ▪ Organizational structure & processes	▪ User interface ▪ Organizational structure & processes	▪ User interface	▪ Interface ▪ Decomposition ▪ Manual intervention	▪ Interface ▪ Decomposition ▪ Manual intervention

[2] Security Services are Integrity, Non-repudiation, Confidentiality, Authentication, Availability, and Accountability

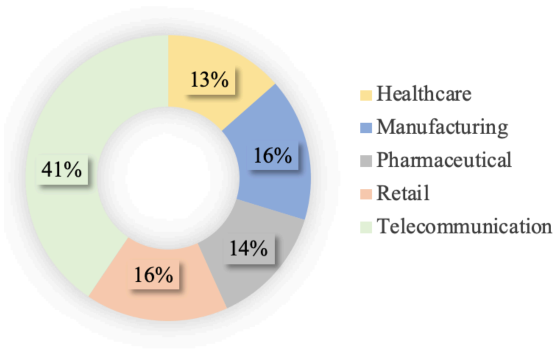
Attributes of Good Architecture	How interoperability of open system is addressed through attributes of architecture at different levels of granularity?				
	Extended Enterprise	Enterprise	System	Functional	Physical
	▪ Governance	▪ Governance		▪ Synchronization	▪ Synchronizatio n
Testability	▪ Traceability ▪ Interfaces ▪ Isolation ▪ Configuration ▪ Auditing ▪ Exception handling	▪ Traceability ▪ Interfaces ▪ Isolation ▪ Configuration ▪ Auditing ▪ Exception handling	▪ Traceability ▪ Interfaces ▪ Isolation ▪ Configuration ▪ Auditing ▪ Exception handling	▪ Traceability ▪ Interfaces ▪ Isolation ▪ Configuration ▪ Auditing ▪ Exception handling	▪ Traceability ▪ Interfaces ▪ Isolation ▪ Configuration

As an example, this section will elaborate on compliance and how it can be used at the five levels of architecture abstraction to address open system interoperability. In this context, compliance of architecture can be defined as the ability of the system to adhere to the standards, guidelines, and regulations to provide consistency among processes, components, and sub-systems (e.g., organizational process, development process, product lines). It also aims to maintain safety and security, reduce the communication interface and cost, and address interoperability. It is the degree to which a system or its architectural description is compliant with a given standard (McCabe and Pollen, 2004). Various interoperability aspects such as data, standards, throughput, and level of service agreements have to be addressed at all levels of architecture. To process, store, and transmit data as a precondition for interoperability, throughput and data related interoperability factors (e.g., format, type, and structure) need to be defined at all levels of architecture. Standards are required at all levels of architecture to address interoperability through interfaces, applications, data, protocols, and processes. To maintain the same level of interoperability throughout the system, sub-system, and organization, internal and external service level agreements need to be addressed at all levels of architecture. Safety-related issues of interoperability have to be addressed in physical, functional, and system architecture. These safety requirements are a part of regulatory compliance or organizational compliance. Syntactic interoperability in physical architecture ensures that the structure of inputs, outputs, and interfaces between the components that interoperate is well defined.

Semantic interoperability is addressed in both functional architecture and system architecture to exhibit compliance. Semantics is the use of domain knowledge to make systems and functions more intelligent, adaptive, and efficient (Lee, 2004). By using knowledge specific to the systems and functions, one can enhance their functionalities and optimize the performance. Semantic interoperability is achieved by sharing the semantics information about the applications and systems that interoperate. Semantic interoperability ensures that the requester and the provider have a common understanding of the ‘meaning’ of the requested services and data (Salah et al., 2005). To interoperate, they must agree on the semantics of concepts that enable them to exchange with a common understanding. Semantic interoperability in enterprise architecture and extended enterprise architecture is addressed through ontology mapping. Enterprise and extended enterprise ontology typically include information, data, meta-data, and meta-meta-data categorization. They describe relationships and structures between these entities that interoperate and their classes of information/data.

7. VALIDATION OF THE OSAFI FRAMEWORK THROUGH A SURVEY

The OSAFI framework has been converted into a survey tool (Appendix A) to assess the interoperability at five levels of architecture and examine the attributes of architecture in terms of interoperability factors. Thirty-four industry practitioners have responded to this survey. Figure 5 demonstrates the distribution of respondents by industry field and job title. The participants have different roles, including IT managers and IT director in various industry domains (e.g., telecommunications, healthcare), which reduces the bias in the dataset.



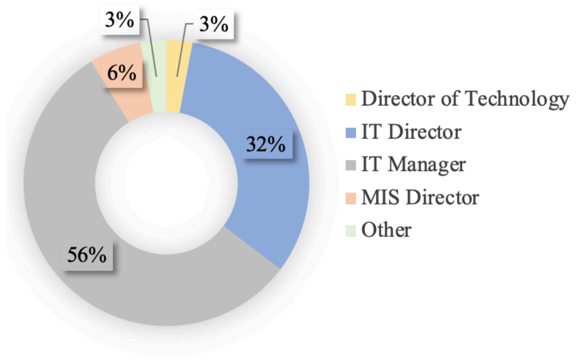


Figure 5. Distribution of Respondents by Industry Domain and Role

Table 7 summarises the most and the least essential interoperability characteristics at a given level of architecture abstraction. It should be noted that all of the interoperability factors studied in the survey received an average score of above 6. This indicates that the interoperability factors proposed in this study are critical attributes of good architecture that enable open systems.

Physical interoperability provides the physical connections allowing information to move from one place to another and sets the foundation for interoperable systems. Fault tolerance in the physical level specifically contributes to two primary attributes of good architecture design, namely availability and responsiveness. When physical links are under threat and not fault-tolerant, interoperability cannot be achieved due to being not available and responsive. Physical interoperability also defines how physical components and parts work; therefore, “Standards”, “Technology, and “Safety” factors ensuring reliable integration among the physical elements of an open system are considered crucial. It is important to note that these three factors primary assist open systems to be simultaneously compliant and flexible. Considering that the characteristics of physical components change so rapidly in the current technological environment, it is expected that compliance and flexibility are seen as essential to accomplish physical interoperability

Table 7. Survey Result Summary

Levels of Architecture Abstraction	Five Most Important Characteristics	Average Score	Five Least Important Characteristics	Average Score
Physical IO	Fault Isolation	9.48	Reuse	6.46
	Standards	8.58	Configuration	6.50
	Throughput	8.50	Uniformity	6.54
	Technology	8.08	Syntactic	6.58
	Safety	8.04	Levels of service agreements	6.77
Functional IO	Synchronization	8.81	Data	6.69
	Decomposition	8.65	Rationalized allocation	6.77
	Standards	8.54	Latency	6.77
	Semantics	8.46	Auditing	6.81
	Orthogonal	8.46	Common platform	6.85
System IO	Maturity	8.88	Configuration	6.73
	Common Operating Platform	8.77	Vendor management	6.73
	Safety	8.69	Versatility	6.73
	User Interface	8.54	Contingency plan	6.81
	Uniformity	8.42	Traceability	6.81
Enterprise IO	Governance	8.88	Maturity	6.58
	Standards	8.88	Configuration	6.73
	Data	8.81	Dependencies Impacting Cross-ilities	7.08
	Integrated Process	8.58	Serviceability	7.08
	Organization structure & Processes	8.42	Levels of service agreements	7.15
Extended Enterprise IO	Contingency plan	9.15	Organization structure & Processes	6.77
	Standards	9.08	Auditing	6.96
	User Interface	8.88	Uniformity	7.15
	Disaster Recovery	8.81	Latency	7.15
	Common Operating Picture	8.77	Business impact analysis	7.31

Functional interoperability aims to deploy integrations of components to function together, exchange data, and use the transferred data. Synchronization is a concern when multiple parts communicate to transport and utilize data, especially in today's systems operating in real-time. Synchronization specifically determines how responsive and simple an open system is and plays a pivotal role to achieve the maximum level of interoperability. Because fully synchronous parts function with low latency, modularity determining how components should be decomposed is necessary to guard the system against failures. Decomposition helps system designers identify the requirements to partition the whole system into components, allowing an uninterrupted flow of inputs and outputs. Thus, decomposition, within open systems, provides modularity that improves the level of interoperability. Another important factor worth mentioning in the functional level is orthogonal, which contributes to flexibility by allowing a procedure to be modified without a significant change in the workflow, other functions, and operations.

In the system level, maturity strengthens interoperability by enabling system elements (1) to be well-defined and non-overlapping (2) to be modified to do jobs not initially included in the requirements definition. Maturity grants two critical characteristics of good architecture design, namely modularity and flexibility. The user interface factor is crucial as it enables simplicity. The need for simplicity through user interfaces to enable complete systems interoperability has been addressed by Unified Communications Interoperability Forum (2010). According to the results, interoperability at the system level requires a "Common Operating Platforms", which assist interoperability by defining interfaces, incorporating data models and standards. Having Common operating platforms is also addressed as an essential interoperability factor because it provides various shared capabilities reducing the need to redesign systems and their components repeatedly. This factor also contributes to achieving interoperability even among future systems that have not been yet designed.

Governance providing simplicity and security is a foundational requirement for successful adoption of interoperable enterprise systems. Enterprise interoperability grants large organizations to link activities, including product and service delivery, supply chain management, and information storage, analysis, and reporting. The enterprise layer encompasses frequent use of data and requires data governance. Especially, considering several industry domains where data protection is essential, such as the healthcare and pharmaceutical industries, effective and standardized data governance processes to achieve interoperability is highly needed. Process integration provides portability enabling systems enterprises to move data among several resources. Process integration also allows automation and unification of systems across an organization. Therefore, it is a crucial interoperability factor supporting open systems at the enterprise level. The Organizational structure factor lays the foundation for how data and processes are shared and governed in addition to determining the ability of a particular enterprise to adopt technologies enabling interoperability. The Organizational structure factor provides portability and assists enterprise interoperability.

Extended enterprise architecture links an organization, its suppliers and partners together with information flow. In such a complex ecosystem where multiple parties are involved, having a contingency plan and disaster recovery becomes a critical factor for maintaining interoperability, especially during catastrophic events such as failures and cybersecurity attacks. A contingency plan and disaster recovery assist enterprises to stay responsive during system failures and allow an ongoing flow of inputs and outputs. For instance, two hospitals' systems that are not interoperable under normal conditions should become interoperable in the wake of a hurricane crisis. This can allow patients evacuated from one hospital to seek care in another facility.

8. CONCLUSION

This study has discussed how interoperability can be achieved through compliance and design flexibility. After defining five architecture levels, the study has proposed a framework by determining the most crucial interoperability factors at each architecture abstraction level to achieve open systems. In order to identify the critical interoperability factors, a survey has been conducted, and the results have been discussed. It has been observed that interoperability factors in achieving open systems vary depending on the architecture level. This framework can be extended by identifying the best practices and guiding principles in achieving the factors of interoperability discussed in each cell of the framework. For example, defining access control to have a secure and interoperable environment will need to be analysed using threat modelling (Anderson, 2001). Additionally, metrics for each of these factors of interoperability and overall system interoperability such as coupling, heterogeneity, synchronicity, boundedness, ownership, and usage patterns need to be developed. The proposed OSAFI can be employed to develop a framework for interoperability measurement.

REFERENCES

Abuelma'atti, O., Merabti, M., and Askwith, B. (2006), ‘A wireless networked appliances interoperability architecture’, 1st International Symposium on Wireless Pervasive Computing, Phuket, pp. 6, January 16 – 18, pp 6 – 12

Alaya, M. B., Medjiah, S., Monteil, T., and Drira, K. (2015) ‘Toward semantic interoperability in oneM2M architecture’, IEEE Communications Magazine, Vol. 53 No. 12, pp. 35 – 41

Alqaoud A., Taylor I., and Jones A. (2010) ‘Scientific workflow interoperability framework. International Journal of Business Process Integration and Management’, Vol. 5 No. 1, pp. 93 – 105

Anderson, R. (2001) ‘Security Engineering: A Guide to Building Dependable Distributed Systems, Wiley Computer Publishing, New York

Arapi P., Moumoutzis N., Mylonakis M., Christodoulakis S. (2007), ‘A Framework and an Architecture for Supporting Interoperability Between Digital Libraries and eLearning Applications’, Proceedings of the DELOS Conference on Digital Libraries, Pisa Italy, February 13 – 14

Bhardwaj S., Ozcelebi T., Lukkien J., and Lee K. M. (2018) ‘Semantic Interoperability Architecture for Smart Spaces’, International Journal of Fuzzy Logic and Intelligent Systems, Vol. 18 No. 1, pp. 50 – 57

Bhuta, J. and Boehm, B. (2007) ‘A framework for identification and resolution of interoperability mismatches in COTS-based systems’, In Proceedings of the International Workshop on Incorporating COTS Software into Software Systems: Tools and Techniques. IEEE Computer Science Society

Brownsword, L., Carney, D.J., Fisher, D., Lewis, G., and Meyers, C. (2004). Current Perspectives on Interoperability. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Carney, D., Fisher, D., and Place, P. (2005) ‘Topics in Interoperability: System-of-Systems Evolution’, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA

Clark, T. and Jones, R. (1999) ‘Organisational interoperability maturity model for c2’, Proceedings of Command and Control Research and Technology Symposium, Newport, Rhode Island. June 29 – July 1

Czarnecki, C., and Spiliopoulou, M. (2012). ‘A holistic framework for the implementation of a next generation network’, *International Journal of Business Information Systems*, Vol. 9 No,4, pp. 385-401.

C4ISR Architectures Working Group Report. (1998) Levels of Information Systems Interoperability (LISI), Department of Defence, Washington, DC

Dai, W., Vyatkin, V., Christensen, J., and Dubinin, V. (2015) ‘Bridging Service-Oriented Architecture and IEC 61499 for Flexibility and Dynamic Reconfigurability’ IEEE Transactions on Industrial Informatics, Vol. 11 No. 3, 771 – 781

Davis, L., Gamble, R., and Payton, J. (2002) ‘The Impact of Component Architectures on Interoperability’, Journal of Systems & Software, Vol. 61 No. 1, pp. 31 – 45

Erl, T. (2007) ‘Service-Oriented Architecture Concepts, Technology and Design’, 1st Edition, Prentice Hall/Pearson PTR

Erl T. (2016), ‘Service-Oriented Architecture Principles of Service Design’, 1st Edition, Prentice Hall

EPAN (2004) ‘Key Principles of an Interoperability Architecture’, European Public Administration Network, eGovernment Working Group. Retrieved on July 16, 2019 from https://circabc.europa.eu/webdav/CircaBC/eupan/dgadmintest/Library/3/2/8_ireland/meetingssduringsthesiris/27-28_may_2004/Principles_of_Interoperability__eGov.pdf

Ford, T., Colombi, J., Jacques, D., and Graham, S. (2007) ‘The Interoperability Score’, 5th Annual Conference on Systems Engineering Research, Hoboken, NJ, March 14 – 16

Gillis, M. (1999), ‘Open Systems Joint Task Force Gets the Word Out’, pp. 44-47

Gravina, R., Palau, C. E., Manso, M., Liotta, A., and Fortino, G. (2018) ‘Integration, Interconnection and Interoperability of IoT Systems’, Springer Berlin, Germany

Guijarro, L. (2007) ‘Interoperability frameworks and enterprise architectures in e-government initiatives in Europe and the United States’, Government. Informatics, Vol. 24 No. 1 pp. 89 – 101.

Haslhofer, B. and Klas, W. (2010). ‘A survey of techniques for achieving metadata interoperability’. ACM computing Surveys, Vol. 42 No. 2, pp. 1 – 42

Hillard, R. (2000) ‘Recommended Practice for Architectural Description of Software – Intensive Systems’. IEEE-std-1471-2000

IEEE (1990) ‘Standard Glossary of Software Engineering Terminology’, IEEE Standard 610.12

IEEE Dictionary. (1991) ‘IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries’, IEEE Std 610, Vol. 1, No. 1, pp.1 – 217

IEEE Standard for information technology – software reuse. (1995) ‘Data model for reuse library interoperability; Basic Interoperability Data Model’. IEEE std 1420.1

Igamberdiev, M., Grossmann, G., Selway, M., and Stumptner, M. (2018), ‘An integrated multi-level modelling approach for industrial-scale data interoperability’, Software System Model, Vol. 17 No. 1, pp. 269 – 294

ISO/IEC JTCI/SC21, Basic Reference models of open distributed processing – Part 2 Descriptive Model, Committee Draft, ISO/IEC CD 10746-2 1991-07-24.

Jabbar, S., Ullah, F., Khalid, S., Khan, M., and Han, K. (2017) ‘Semantic Interoperability in Heterogeneous IoT Infrastructure for Healthcare’, Wireless Communications and Mobile Computing, pp. 1 – 10

Jain, R., Chandrasekaran, A., Elias, G., and Cloutier, R. (2008) ‘Exploring the Impact of Systems Architecture and Systems Requirements on Systems Integration Complexity’ IEEE Systems Journal, Vol. 2 No. 2, pp. 209 – 223

Jain, R., Chandrasekaran, A., and Erol, O. (2010) ‘A framework for end-to-end approach to Systems Integration’, International Journal of Industrial and Systems Engineering, Vol. 5 No. 1, pp. 79 – 209

Janssen, M. (2012). ‘Sociopolitical Aspects of Interoperability and Enterprise Architecture in E-Government’ Social Science Computer Review, Vol. 30 No. 1, pp. 24 – 36

Jardim-Goncalves, R., Popplewell, K., Grilo, A. (2012) ‘Sustainable interoperability: The future of Internet based industrial enterprises’, Computers in Industry, Vol. 63 No. 8, pp. 731 – 738

Keshav, R and Gamble, R. (1998) ‘Towards a Taxonomy of Architecture Integration Strategies’, 3rd International Software Architecture Workshop, November 1 – 2, Orlando FL, pp. 49 – 51

Kiljander, J., D’elia, A., Morandi, F., Hyttinen, P., Takalo-Mattila, J., Ylisaukko-Oja, A., Soininen, J., and Cinotti, T. S. (2014) ‘Semantic Interoperability Architecture for Pervasive Computing and Internet of Things’, IEEE Access, Vol. 2 No.1, pp. 856 – 873

Klaseen, E. L., Levin, L. J., Denny, B. A., and Cunningham, J. (1994) ‘Achieving Open Interoperability in Tactical Remoted Systems with COTS Technology’, Proceedings of Technical Communications Conference - IEEE, Fort Wayne IN, May 12 to 14, pp. 225 – 233

Klaseen, E.L. and Sydir, J.J. (1995) ‘The definition of interoperability architectures for intelligent devices using abstract models, Factory Communication Systems’, Proceedings of WFCS, IEEE International Workshop, October 4 – 6, Leysin, Switzerland, pp. 237 – 245

Kubicek, H. and Cimander, R. (2009) ‘Three dimensions of organizational interoperability: insights from recent studies for improving interoperability frameworks’, European Journal of ePractice, Vol. 6 No. 1, pp. 3 – 14

Kundu, S. and Tyagi, K. (2017). ‘Selection and classification of common factors affecting the maintainability on the basis of common criteria’, International Journal of Business Information Systems, Vol. 26 No. 3, pp. 402-412.

Lee, J. (2004) ‘Introduction To Semantics Technology’, IBM T.J. Watson Research Center, Retrieved on July 13, 2019 From http://www.alphaworks.ibm.com/contentnr/introsemantics

Maier, M. W. and Rechtin, E. (2009) ‘The Art of Systems Architecting, Third Edition, CRC Press

McCabe, R. and Pollen, M. (2004) ‘Evaluating Architectures with System Attributes’, Software Productivity Consortium, Herndon VA, February 5th

McManus, H. and Hastings, D. (2006) ‘A framework for understanding uncertainty and its mitigation and exploitation in complex systems’, IEEE Engineering Management Review, Vol. 34 No. 3, pp. 81 – 94

Mills, J.A. (1993) 'Large Scale Interoperability and Distributed Transaction Processing', Journal of Systems Integration, Vol. 3 No. 3 – 4, pp. 351 – 369

Morris, E., Levine, L, Meyers, C., Place, P, Plakosh, D. (2004) 'System of Systems Interoperability (SOSI)', Technical Report. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA

NATO Allied Data Publication 34. (2003) 'NATO C3 Technical Architecture (NC3TA), Version 4.0," retrieved from http://www.nato.int/docu/standard.htm on July 21, 2019

Open Systems Joint Task Force Report. (2001). Acquired on July 7. 2019 from https://apps.dtic.mil/dtic/tr/fulltext/u2/a402560.pdf

Ouksel, A and Sheth, A. (1999) 'Semantic Interoperability in Global Information Systems', ACM Sigmod Record, Vol. 21 No. 1, pp. 5 – 12

Unified Communications Interoperability Forum (UCIF): Members Meeting, Los Angeles Convention Center, Los Angeles, California, October 5, 2010 Accessed on June 5th , 2019 from https:/web.archive.org/web/20130928001901/http:/www.ucif.org/portals/0/documents/2919_10_05_ucif_f2f.pdf

Papazoglou, M.P. and Van den Heuvel, W. (2007) 'Service oriented architectures: approaches, technologies and research issues', The International Journal on very Large Data Bases, Vol. 16 No. 3, pp. 389 – 415.

Petcu, D. (2011) 'Portability and interoperability between clouds: Challenges and case study', Towards a Service-Based Internet. ServiceWave. Lecture Notes in Computer Science, vol 6994. Springer, Berlin, Heidelberg

Pollack, J. T. and Hodgson, R. (2004), 'Adaptive Systems', John Wiley and Sons

Ponis, S. T., Van der Eijk, P., and Masselos, V. (2012). 'Supply chain interoperability for enhancing e-business adoption by SMEs: a case study from the European clothing sector', *International Journal of Business Information Systems*, Vol. 10 No. 4, 417-435.

Salah, H., Mahmoud, B., and Nacer, B. (2005). 'An architecture for the interoperability of workflow models', IHIS Proceedings of the first international workshop on Interoperability of heterogeneous information systems, November 4th Bremen, Germany, pp. 31 – 38

Sarantis, D., Charalabidis, Y., and Psarras, J. (2008) 'Towards standardizing interoperability levels for information systems of public administrations", The Electronic Journal for E- commerce Tools & Applications Special Issue on Interoperability for Enterprises and Administrations Worldwide, Vol. 1 No.1 pp. 1 – 15

Steel, J., Drogemuller, R., and Toth, B. (2012) 'Model interoperability in building information modelling', Software & Systems Modelling, Vol. 11 No. 1, pp 99 – 109

The Authoritative Dictionary of IEEE Standards Terms, Seventh Edition" (2000) IEEE Std 100-2000, pp.1-1362,

The Bellcore OSCATM Architecture; Technical Advisory TA-STS-000915, Issue 3, March 1999, Bellcore, Piscataway NJ

Tolk A, Muguira J. A. (2003) 'The levels of conceptual interoperability model'. Fall Simulation Interoperability Workshop, Simulation Interoperability Standards Organization, Orlando, Florida

Tu, S., Xu, L., Abdelguerfu, M., and Ratcliff, J. (2002) 'Achieving interoperability for integration of heterogeneous COTS geographic information systems', Proceedings of the 10th ACM international symposium on Advances in geographic information systems, McLean, Virginia, November 8 – 9, pp 162 – 167

Wassermann, E. and Alexander Fay, A. (2017) 'Interoperability rules for heterogenous multi-agent systems: Levels of conceptual interoperability model applied for multi-agent systems', 15th International Conference on Industrial Informatics, July 24 – 26, Emden Germany, pp. 89 – 95

Williams, J. (2000) 'Correctly assessing the -ilities requires more than marketing hype', IT Professional, Vol. 2 No. 6, pp. 65 – 67

Yahia, E., Aubry, A., Panetto, H. (2012) 'Formal measures for semantic interoperability assessment in cooperative enterprise information system', Computers in Industry, Vol. 63 No.5, pp. 443 – 457

Zachman. J. A. (1987) 'A Framework for Information Systems Architecture', IBM Systems Journal, Vol. 26 No. 3, pp. 276 – 292

APPENDIX A

Availability of architecture can be defined as the ability of the system to provide the intended functionality and performance during all periods of desired use. It is the degree to which a system or component is operational and accessible when required for use (IEEE 610.12, 1990). The parameters or constituent factors of availability are sub-system/component. Reliability and fault tolerance such as number of single points of failure, mean time between operational service interruption, and mean time to repair/recover, latent defects such as service interruption rate due to software malfunction and recovery time from software malfunction , and system recovery actions required per month (system IPLs and restarts, sub-system restarts, system warm starts, database recoveries), operational duty factor (e.g., downtime, uptime), fraction of uptime during normal 24/7 operation, fraction of customer transactions completed successfully after encountering recoverable errors, and fraction of customer-initiated activity completed by the system. Availability is typically characterized by a ratio of the number of system services provided correctly to the amount desired (McCabe and Pollen, 2004).

Compliance of architecture can be defined as the ability of the system to adhere to the standards, guidelines, and regulations to provide consistency among organizational processes, components, and sub-systems. Compliance aims to maintain required safety and security, reduce the communication interface and cost, and address interoperability. It is the degree to which a system (or its architectural description) is compliant with (or can be brought into compliance with) a given standard (McCabe and Pollen, 2004).

Evolvability can be defined as the ability of the system to serve as the basis of new systems to meet new needs and(or) attain new capability levels (McManus and Hastings, 2006).

Flexibility can be defined as the ability of the system to be modified to do jobs not originally included in the requirements definition. The modification may be in the design, production, or operation of the system; each has a unique flavour (McManus and Hastings, 2006). It is the ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed (IEEE 610.12, 1990).

Functionality of architecture can be defined as uniquely identified functions and capability provided by the system. It is a set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs ISO/IEC 9126, 1991). The parameter or constituent factor of functionality is the percentage of system objective requirements (functional and performance) satisfied by solution (e.g., the number of requirements fully satisfied, the number of requirements partially satisfied, and number of requirements not met)

Generality can be defined as using Multiple-function (sub)systems and interfaces, rather than specialized ones. A common example is a general-purpose processor/computer rather than specialized chip or machine. Bus interfaces, or fully switched networks connecting redundant elements (instead of the minimum interconnection necessary to swap in for a dead unit), are examples of "general" interfaces (McManus and Hastings, 2006).

Interoperability can be defined as the ability of the system to "play well with others," both with systems it was initially designed to work with, and with future systems. It may be desirable in and of itself; it also enhances versatility, flexibility, and evolvability of systems of systems (McManus and Hastings, 2006). It is the ability of systems, units, or forces to provide services to and accept services from other systems, units, or forces and to use the services so exchanged to enable them to operate effectively together.

Modularity of architecture can be defined as the extent to which the system is made up of well-defined, functionally non-overlapping, and modular elements with well-documented interfaces. These interfaces must allow updates or replacements of a portion of the system without affecting the remainder of the system. The parameters or constituent factors of modularity are system partitions completely defined by industry-standard interfaces, module orthogonality (non-overlapping functionality) in terms of Are functional requirements fragmented across multiple processing elements and interfaces?, Are capacity requirements satisfied by multiple synchronized elements?, and Are common specifications identified?, physical modularity - ease of upgrading system elements, functional modularity - ease of adding new functionality with minimum disruption, and abstraction - Does system architecture provide for proper information hiding. Modularity, open architectures, and standard interfaces: Functions grouped into modules and connected by standard interfaces in such a way that they can "plug and play." Not independent strategies, but greatly helps redundancy, generality, upgradeability, and makes testing easier (sometimes). Modularity is the degree to which a system is structured as a configuration of smaller, self-contained functional units with well-defined interfaces and interactions (i.e., independently testable), moderating design complexity and enhancing its clarity, and enabling design and functional flexibility and variety for the system as a whole (McCabe and Pollen, 2004; Open Systems Joint Task Force , 2005)

Responsiveness of architecture can be defined as the ease with which the system can be updated in response to changing business needs. The parameters or constituent factors of responsiveness are ease of updating solution to reflect updated business process requirements, amount of coordination and consensus required with other business areas prior to or concurrent with system update, amount of customized design incorporated in solution, development time required to incorporate changes, integration and test effort required for system updates and operational training cost and complexity.

Robustness can be defined as the ability of the system to do its basic job in unexpectedly adverse environments. Well understood for non-aerospace products; aerospace products tend to be designed for expected adverse environments already, leading to minor ambiguities. Worse is the common tendency to use this word for any of the attributes on this list (McManus and Hastings, 2006). It is the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions. See also: error tolerance; fault tolerance (IEEE 610.12, 1990)

Scalability of architecture can be defined as the ease with which the system can grow to accommodate increased performance (e.g., higher transaction rates, more customers, etc.), expanded functionality (e.g., additional pricing methods) or scaled back to cost-effectively support reduced levels of performance or functionality. The parameters or constituent factors of scalability are total cost to add system capacity to accommodate more users, higher transaction volumes, additional service types, additional products, etc…, architectural limitations on growth/scalability - ability of application software to accommodate infrastructure changes (greater capabilities and capacities or different structure) without incurring degraded performance or unreliable operation, reach - architecture limits on expansion of solution into multiple geographic regions, total additional cost incurred to upgrade infrastructure to accommodate functionality and capacity growth and operational impacts of scaling performance capacities.

Serviceability/Upgradeability can be defined as the degree to which the (Sub) systems can be modified to improve or change function. Obvious difficulty in non-retrievable vehicles like satellites, although there are options even in this case (software, swarm components). Combining general hardware with upgradeable software is very powerful (McManus and Hastings, 2006).

Simplicity of architecture can be defined as the degree of complexity (how complex is the system) in terms of partitioning, interfaces, manual intervention, and synchronization relative to other systems of similar functionality. The parameters or constituent factors of simplicity are number of independent functional interfaces types, number of interfaces, number of independent functional partitions, manual intervention required, operator skill-level required, system timing synchronization required, data Synchronization requirements and infrastructure requirements.

Versatility can be defined as the ability of the system, as built/designed, to do jobs not originally included in the requirements definition, and/or or to do a variety of required jobs well. Often confused or combined with Robustness and Flexibility in discussions (McManus and Hastings, 2006).

SURVEY

Questions 1 Through 5:

Please provide rating ranging from 0 to 10 on the extent to which factor of interoperability listed below is needed and addressed (for the corresponding **architecture** abstraction level) to enable an open system design. Please provide ratings based on your experience of a specific system. Note that 0 stands for "Not addressed at all" while 10 indicates "completely addressed."

Physical IO	Uniformity (1), Decomposition (2), Data (3), Standards (4), Throughput (5), Levels of service agreements (6), Syntactic (7), Safety (8), Configuration (9), Orthogonal (10), Technology (11), Inheritance (Dependency) (12), Consistency in allocation (13), Reuse (14), Modularity (15), Dependencies Impacting cross-ilities (16), Performance (17), Serviceability (18), Latency (19), Recovery (20), Disaster Recovery (21), Contingency plan (22), Archiving (23), Virtual prototyping (24), Virtual testing Protection layer (safeguard) (25), Access control (26), Security services (27), Interfaces (28), Manual intervention (29), Synchronization (30), Traceability (31), Fault Isolation (relevant for regression testing) (32)
Functional IO	Uniformity (1), Decomposition (2), COTS/Non-Development Items (3), Data (4), Standards (5), Throughput (6), Levels of service agreements (7), Semantics (8), Safety (9), Configuration (10), Orthogonal (11), Technology (12), Inheritance (Dependency) (13), Rationalized allocation (14), Functional cohesiveness (15), Reuse (16), Modularity (17), Common platform (18), Dependencies Impacting cross-ilities (19), Performance (20), Serviceability (21), Latency (22), Recovery (23), Capability (24), Disaster Recovery (25), Contingency plan (26), Archiving (27), Virtual prototyping (28), Virtual testing (29), Safe mode operation (30), Exception handling (31), Baseline operation (32), Access control (33), Assurance (34), Auditing (35), Security services (36), Interfaces (37), Manual intervention (38), Synchronization (39), Traceability (40), Fault Isolation (relevant for regression testing) (41), Fault Isolation (relevant for regression testing) (42)

System IO	Uniformity (1), Decomposition (2), Configuration (3), Data (4), Standards (5), Throughput (6), Levels of service agreements (7), Semantics (8), Safety (9), Orthogonal (10), Maturity (11), Vendor management (12), System evolution based on reduced functional dependencies (13), Common Operating Platform (14), Commonality (15), User Interface (16), Dependencies Impacting cross-ilities (17), Performance (18), Serviceability (19), Latency (20), Recovery (21), Capability (22), Disaster Recovery (23), Contingency plan (24), Archiving (25), Versatility in handling unpredictable environments (26), Virtual prototyping (27), Virtual testing (28), Safe mode operation (29), Exception handling (30), Baseline operation (31), Modeling and simulation (32), Self healing/adapting (33), Intelligence Capability (34), Access control (35), Assurance (36), Resilience (37), OPSEC (Operational Security) (38), Security services (39), User Interface (40), Traceability (41), Interfaces (42), Fault Isolation (relevant for regression testing) (43), Auditing (44), Exception handling (45)
Enterprise	Uniformity (1), Organization structure & Processes (2), Governance (3), Configuration (4), Data (5), Standards (6), Throughput (7), Levels of service agreements (8), Ontology (9), Maturity (10), Vendor management (11), Integrated Process (12), Vertical Dependency between goal, objects, and business requirements (13), Common Operating Picture (14), Process Consistency (15), Dependencies Impacting Cross-ilities (16), Performance (17), Serviceability (18), Latency (19), Recovery (20), Capability (21), Disaster Recovery (22)
Extended Enterprise	Uniformity (1), Organization structure & Processes (2), Governance (3), Data (4), Standards (5), Throughput (6), Levels of service agreements (7), Ontology (8), Configuration (9), Integrated Process (10), Vertical Dependency between goal, objects, and business requirements (11), Common Operating Picture (12), Dependencies Impacting Cross-ilities (13), Performance (14), Serviceability (15), Latency (16), Recovery (17), Capability (18), Disaster Recovery (19), Contingency plan (20), Archiving (21), Business continuity plan /Mission contingency plan (22), Business impact analysis (23), Versatility in handling unpredictable environments (24), Agility (25), Adaptability (26), Access control (27), Assurance (28), Resilience (29), Policy (30), Technology (31), User Interface (32), Traceability (33), Interfaces (34), Fault Isolation (relevant for regression testing) (35), Configuration (36), Auditing (37), Exception handling (38)

Question 6: What is your Current role within your company?

Question 7: In which year was your company founded?

Question8: In which Primary Industry/Products/Services that your company operates?

Question 9: Where is the location of major operations of your company?