Int. J. Emerg. Sci., 2(4), 619-631, December 2012
ISSN: 2222-4254

Challenges in Achieving Interoperability in Distributed Systems: a Survey of Literature

Kennedy O. Ondimu, Geoffrey M. Muketha

Department of Computer Science and Information Technology,
Mombasa Polytechnic University College, P. O. Box 90420-80100 Mombasa, Kenya,
Department of Computer Science, Masinde Muliro
University of Science and Technology, P.O. Box 190-50100 Kakamega, Kenya,

k.ondimu@gmail.com, gimuchiri@gmail.com

Abstract. As need for collaboration and distributed systems among organizations increase, there is the challenge of different standards in almost all communication issues. This calls for scrutiny of systems used by different organizations or units of the same organization in an effort to achieve distributed systems. Available literature on this topic is limited and there is lack of an assessment framework to determine which system has attained what level of interoperability. We discuses four strategies or approaches used in achieving some degree of interoperability, as well as issues in distributed systems interoperability. This paper conducts a detailed literature review on interoperability in distributed systems and then proposes a ranking framework to assess interoperability. Currently XML is widely used implementing systems that need to communicate with others. However technical challenges such as semantics, security and legacy databases, together with many managerial issues remain a hindrance to achieving complete interoperability.

Keywords: Interoperability, Distributed systems, Legacy Systems, XML.

1 INTRODUCTION

Most of today's software systems are complex consisting of smaller components which are systems on their own right. The system could also be part of a more complex system consisting of independently managed and operated components that depend on other systems outside the administrative control of their owners, developers and users [7]. The components could also have different structures and designed for different purposes.

Unlike traditional systems, such systems display emergent behaviour, a behaviour that cannot be localized to a single component but instead produce effects that arise from the cumulative effect of contributions from all involved components. The emergent behaviours can lead to the success or failure of the distributed systems. Such systems are referred to as systems of systems by some authors and as distributed systems by others; the rest of this paper uses "distributed systems" to mean the same.

A distributed system is defined as one in which different components use messages only to communicate and coordinate actions between themselves in a networked environment [5]. Its characteristics include; lack of a global clock and autonomy of individual components. When a component fails, it does not have to affect the relationship between those that remain functional. The internet and intranets are the best examples of distributed systems, though this can apply to even home environments [21].

A distributed system consists of sub-systems, which in turn consist of components. Sub-systems focus on local problems but communicate with other sub-systems in the distributed environment when need be, while components are mainly concerned with transformation.

Interoperability is defined as the ability of a collection of communicating entities to share specified information and operate on it according to shared operational semantics in order to achieve a specified purpose in a given context [2]. It is achieved when all components and sub-systems in distributed systems work together seamlessly to achieve a set objective. In software engineering, interoperability is defined as the ability to exchange functionality and interpretable data between two software entities [8]. In this case interoperability is divided into Application and semantic interoperability. Application interoperability addresses communication issues normally handled by the TCP/IP communication protocols. Semantic interoperability deals with data interpretation (schemas) and knowledge representation and exploitation by means of ontologies and agents [22].

In this paper, we suggest a ranking framework to assess interoperability in distributed systems. This is an extension of [15], in which the extent to which a particular distributed system has achieved interoperability is not addressed. However, only six of the eight interoperability issues proposed in [15] are used in this case since motivation and funding are closely related..

This paper addresses the following four research questions:

RQ1. What are the current strategies/approaches and technologies in distributed systems interoperability?

RQ2. What are the issues in distributed systems interoperability?

RQ3. How can we rank the interoperability of a distributed system in such a way as to be able to tell the level of interoperability attained?

The rest of this paper is structured as follows. Section 2 presents findings of RQ1, Section 3 presents findings of RQ2, and Section 4 discusses the findings of RQ1, RQ2 and suggests an interoperability ranking framework for distributed systems. Section 5 gives the conclusion of the paper, implications, and future directions.

2 KEY STRATEGIES, APPROACHES AND TECHNOLOGIES FOR MANAGING INTEROPERABILITY

A common strategy for achieving interoperability is to have a common format/Standard to ease future interpretation, open database; independence of hardware, operating system and programming languages. It is important to adhere to some standard which makes it easy for others to predict and understand your database. Such databases are easy to access even in the face of technological change since the model is known. The database needs to be open, given that it will be accessed by others whom you do not even know yet. Some information needs to be communicated throughout the distributed system such as semantics of a database. Tight coupling ideal at the constituent level does not support interoperability; however some loose coupling is encouraged between constituent sub-systems [4]. The following four examples are typical efforts towards achieving interoperability.

2.1 e-Government initiatives

Two strategies used in implementing e-government in Europe and USA have been outlined by [8]. Governments have set up frameworks that ensure interoperability by setting specifications and policies to be used between its agencies and service delivery to the public. The specifications and policies are revised and updated frequently to keep up with developments in technological change and the environment. The frameworks consist of specifications and policies covering interconnectivity, data integration, e-service access and content management. The standards include semantic interoperability that address data interpretation, and knowledge representation. To remain current provisions have been made for revisions and updates. The United Kingdom's e-GIF uses a Technical standard catalogue which is revised and updated every six months for this purpose. Other Governments with a similar approach are France, Germany, and Denmark. The European Union has its own framework IDABC (Interoperable delivery of European e-Government services to public Administrations, Business and Citizens). Unlike the individual governments, IDABC is only a guideline that does not prescribe any specific architecture or standard catalogue. It mainly deals with issues not addressed in the individual government frameworks to ensure interoperability throughout the Union. This approach is akin to tight coupling within the constituents with loose coupling between constituents.

2.2 Enterprise architecture

Enterprise architecture [8] goes beyond technology to also address organizational issues such as human resources, business location, and motivation among others, which really define an enterprise. The architecture aims at aligning the IT solution with the enterprise business processes and goals. In USA, FEA (Federal Enterprise Architecture) sets out five reference models for government business transactions. The models define business, performance, data, service component and technical reference. All government funded projects are required to adhere to the FEA model(s) to qualify for funding by the OME (Office of Management and Budget). In this case there is clear motivation for compliance else funding will be withdrawn (a big stick and money) [11].

2.3 Navigator

Navigator [1] provides a framework that helps organizations improve their sub-systems towards attaining distributed systems interoperability. It is based on the paradigm that future SOS will not have an overall architecture standard since the constituent will just evolve. Even those which start with an overall architecture and standards will not last long due to evolution of the constituents that lead to emergence. The approach underpins a number of aspects, the first one being establishing a common understanding for the overall goal for the sub-system or the vision, and the role of every constituent towards achieving it. Secondly is to establish how the constituent sub-systems affect or influence each other and thirdly have a common understanding (semantics) of the data interchanged between the constituents. Finally, agreements on aspects that affect constituents' relationships that needs to be managed.

2.4 Web Services

Web services are self-contained and modular applications that can be described, published, located and invoked over the web. Web services use established open standards and infrastructure such as Extensible Markup Language (XML) over Hyper Text Transfer Protocol (HTTP), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI). Using WSDL, the service provider describes itself giving information such as network location, data types, binding information and operations. The service description is sent to the service registry, where a service requester can find it and use the service description information to locate and bind with the service provider so as to use the service. XML acting as an interface hides the implementation details of the service, allowing communications between different software and hardware platforms. This also facilitates inclusion of legacy systems in any new service [19; 20].

XML, is a technology that facilitates different programs to bridge the gaps between themselves [11]; a registered trademark of Massachusetts Institute of Technology. It is

closely related to HTML in the aspects of displaying and hyper linking content. However it goes beyond to provide structural and semantic information about the data involved; a feature described as metadata. XML is a metalanguage; an open text-based Markup language that is applicable both in the front and back-end processing. Its derivatives or closely related products are the most widely used applications in the web environment that include XUL, ebXML and RDF among others.

The ability of XML to describe data content using metadata makes it possible to retrieve and process data from legacy systems. The order of fields, their length etc is available from the metadata. It is also flexible on naming conventions that allows later resolving in case of concurrent development before linking up. XML offers a content-based structure as opposed to format-based structure only that makes it impossible to work with legacy and heterogeneous databases.

XML is widely recognized by industry leaders such as Microsoft, Netscape and IBM among others as standard for data interchange [16] It specifies strict rules or protocols that have to be consistently applied across interested parties to facilitate interoperability. The protocols called Document Type Definitions (DTD) are platform and database independent making XML highly scalable [13]. The DTD is the vital link between the data file given to XML processor and the application. XML is therefore a standards based implementation that makes sure that there is interoperability in a distributed environment. Data mining in distributed health care systems is an example where XML facilitates interpretation by the receiver who may not necessarily be using the format as the source [23].

There are two approaches to dealing with legacy systems. The first one is to develop a set of XML standards and re-write or redesign all its legacy databases to comply. The other alternative is to convert the existing database to XML structure and standards. In the first approach, the users will have to agree on the data that will be shared and its definitions (taxonomy) before embarking on developing the actual data, its structures and standards [10].The conversion approach is cheaper though temporary by employing tools such as wrappers and mediators.

3 ISSUES IN INTEROPERABILITY

The following eight issues have been identified as a hindrance or challenge in the adoption and implementation of interoperability [15]. They include Ownership, funding, Legacy systems/databases, security, emergent behaviour, motivation, tools, and ambiguity in terminologies. Each individual issue is discussed below.

3.1 Ownership

Ownership refers to possession, authority or control and responsibility over something [2]. It is not possible for any individual or organization to possess a distributed system. By definition each proprietary constituent sub-system could be owned by an individual

or organization, but to remain part of the larger distributed system it has to forgo some of its freedom for example to make changes for its enhancement, but compromise its role as part of the larger system. Likewise authority and control cannot be exercised throughout the distributed system by a sub-system or component. A sub-system has limited localized control while making sure that it remains coherent with the rest of the system. Possession and authority/control that places responsibility on the owner, lacks in distributed systems [11]. This makes it difficult for responsibility to be exercised throughout the distributed system. Responsibility could make the owner take risk mitigation initiatives in case of failure or loss which is impossible in a distributed system. For example if there is an exponential build up of some fault/error, no single entity can take the blame and even its source may be difficult to trace.

3.2 Funding

Traditionally organizations budget for and execute projects that are of strategic importance to their survival or profitability. In a distributed system environment, what you provide for may not be for your immediate benefit as an entity but quite crucial, say for the industry in a cooperative environment [22]. For example a research station may provide for others to access its database for purposes of comparison with other research done elsewhere. The top management would be more interested in new findings from the station and its priority would be geared towards an efficient processing and storage system locally. However, the researchers in the station need to know and compare their findings with others elsewhere hence the necessity for interoperability. The question which arises at this stage is who is to provide the funds and their control [11].The funding of the interoperability features would in most cases be an afterthought putting a strain in the project budget and resources and are better avoided. It is even worse when funds are required for just upgrading a system for interoperability with no internal gains. The best approach for funding would be through an oversight body that funds and ensures that all constituents of a distributed system adhere to set interoperable standards.

3.3 Legacy systems

Most software is developed with a focus on immediate local problems. Even within one organization differences of focus exist. Such systems, either as a department or a system which forms part of a distributed system is likely to sub-optimize at the expense of synergy in the whole distributed system. The architecture and schema of information in each system may not be understood by the other systems making interoperability impossible [14]. Legacy systems are a direct product of technological changes that would always arise in a large organization with collaborative applications [17]. The old system becomes a legacy system once a new technology is adopted. Interoperability is more challenging regarding software and databases as opposed to hardware where

standards have been established for some time now. Changing the old software may require source code and a lot of man hours. In databases, the progression has been through hierarchical, relational and now object oriented. The architecture and schema of such databases is different and interoperability is compromised. Some intermediate intervention is necessary to facilitate the database interoperability between current and legacy databases.

3.4 Security

The security issues in distributed systems involve four areas namely integrity /privacy of: data, infrastructure, participant computer resources and application result correctness [26]. A database that can be accessed by others, not responsible for its security, poses a security risk. The immediate owners will therefore embark on a number of measures such as passwords and encryption to safeguard the database. Such measures achieve the exact opposite as regards ease of use by others to ensure interoperability or openness. Security standard applied throughout distributed systems can ensure that the measures taken have the same interpretation and that accessibility is not compromised. Even then, same standards may not be applicable across countries which have different data control legislations. Adopting an international standard such as ISO 9000 may be the answer in such cases.

3.5 Emergent behaviour

Emergent properties are characteristics that arise from the cumulative actions and interactions of autonomous constituents of a distributed system and cannot be localized to any constant number of constituents. Distributed systems display certain global properties that cannot be accounted for as the result of the sum of actions and properties of their constituents. Emergent behavior arises naturally and predictably from influence mechanisms, cascade effects, and other emergent phenomena that are inherent in distributed systems [7]. In operating systems this is experienced as a priority inversion scenario, whereby a high priority task is blocked because it needs a resource held by a lower-priority task. A poorly configured load balancer (with a short timeout) can report application server(s) dead just because they have grown in latency [25].

3.6 Tools for building distributed systems

The traditional approaches such as tight coupling with less cohesion, hierarchical architectures and top-down would work in a proprietary system but curtail flexibility and capability that define a distributed system. Attempts to use the tools currently available in most cases results in cost overruns, yielding systems that are not scalable and would fail anytime a change is introduced in one of the parts. According to [1] the current tools where performance expectations are known and build by single focused

teams has reached a threshold. New tools are required that will scale the ever changing requirements of distributed systems. An application or tool is considered interoperable if it has the ability to executes multiple programming models concurrently over diverse back-ends [24]

3.7 Lack of Motivation

Motivation for player effort towards enhancing their systems to interoperate with others is very limited. According to [11], Program Offices and contractors often concentrate on delivering a product that meet specific local requirements; little effort is invested in achieving interoperability. Without some incentives as in business/economic collaboration nobody will spend extra resources to achieve interoperability [18]. These calls for an oversight body that sets and enforces interoperability standards. The body can also introduce incentives for best practice achievements.

3.8 Ambiguous terminology

Terms used have different meanings (semantics) to different constituent players in a distributed system. Different professions and sources treat different topics and approach them from deferent viewpoints, techniques and objectives [6]. Even if systems are able to access and retrieve information from each other, they still find it difficult to handle ambiguity in the absence of a universal dictionary among professions and organizations [18; 22]. This calls for agreement on terminology used a distributed systems among the players.

4 DISCUSSION

As [11], points out, standards are necessary but not sufficient for guaranteeing interoperability. Standards that can take care of problems such as semantic interoperability have yet to be developed. An example is in the American armed forces where different forces have a different meaning for the same terminology [11]. Suggestions by some researchers such as [6] and others to handle semantics are still under development. Database interoperability alone cannot guarantee quality of service. The databases are linked via the internet, which despite the convergence on TCP/IP for connectivity, has a number of shortcomings such as: Many users are competing to transmit information resulting in variable speed. Breakdowns are common for at least short periods and occasional loss of transmissions. The database and data while on transmission are always exposed to security risks if extra measures are not taken [4; 3]. Malicious content can corrupt data and programs, reformat complete disks or even shut down systems among other forms of damage. In such events the system may not be available.

From section 3, significant issues in achieving interoperability include: Ownership, funding, legacy, security emergent behaviour, tools, motivation and ambiguity. To a large extent, motivation is closely linked to the funding in a project and can be effectively represented as part of the funding system. Emergent behaviour cannot be represented on a scale like the other issues given its complexity. We therefore remain with six/features issues, which can form the variables of a simple ranking framework. The framework consists of a ranking scale based on a quantification of the presence or absence of the issues/features identified.
Using a scale of 3, the issues are tabulated in Table 1.

Table 1: Issue measurement scale

	Measurement Scale		
	0	1	2
Ownership	No control	Limited Control	Total control
Funding	No funding	Limited funding	Heavy funding
Legacy	Legacy systems	Mixed legacy and modern systems	Modern systems
Security	No standard	Local standard	International standard
Tooling	No tools	Limited tooling	Tool optimized for distributed systems
Ambiguity	ambiguous	limited effort to harmonize semantics	high level of harmonization of semantics
Total			

Semantic interoperability remains a major challenge in digital libraries and all systems using XML [9]. As a result in distributed systems as in the digital library, a search may yield material that is not relevant to the user's needs. The challenge of interoperability in subject searching and browsing involving distributed digital libraries [12] can be ranked using our framework as follows.

Different libraries may be using different catalogue schemes such as LCC (Library of Congress Classification) and DDC (Dewey decimal classification) among others. The same schemes can have extensions such that two libraries using the same scheme may not be exactly the same. In most cases libraries are owned by institutions which also fund or have a major role in funding and management. In some cases such organizations agree to share resources with other libraries which are mostly the initial

goal of the organization and little or no funding is channeled towards achieving interoperability. The libraries would most often than not have legacy files or Unique material whose schema may not be global since it is not subject to agreements with others, making it difficult to be accessed from outside. The collaborating institutions agree to use XML.

Using our ranking framework, the library project is ranked in Table 2.

Table 2: ranking a library distributed system

	Measurement Scale			**Library Project**
	0	1	2	
Ownership	No control	Limited Control	Total control	1
Funding	No funding	Limited funding	Heavy funding	1
Legacy	Legacy systems	Mixed legacy and modern systems	Modern systems	1
Security	No standard	Local standard	International standard	1
Tooling	No tools	Limited tooling	Tool optimized for distributed systems	2
Ambiguity	ambiguous	limited effort to harmonize semantics	high level of harmonization of semantics	1
Total				**7**

The minimal possible score is 0 while the maximum possible is 12. In this case, while the framework scores more than 50%, a lot needs to be done regarding the other issues.

5 CONCLUSION

Achieving interoperability in systems of systems is a challenging task. A number of issues, technical and more so none technical remain to be dealt with. Some approaches such as Navigator can be used to achieve some degree of interoperability. XML, the most widely used application is able to handle a number of technical issues but lacks

semantic interoperability, which hopefully will be addressed by semantic web development in future. So far the traditional tools used in implementing interoperability are inadequate leading to poor products and budget overruns. Above all there are more management related issues affecting interoperability than technical ones. Issues such as ownership and funding have far reaching implications, underpinning the need for an overseer to among others enforce discipline and fund features that support interoperability. We have proposed a framework that can be used to rate interoperability in distributed systems; however, the research findings are based on literature hence we intend to empirically test the framework on a later date.

Both technical and management issues need to be addressed as organizations strive to achieve Interoperability in distributed systems. Issues such as ownership and funding are just as important as tools and security, hence the need for an oversight body to address interoperability.

The proposed framework will help organizations intending to venture into distributed system projects to generally rate their current status and also identify issues that need to be addressed for better interoperability.

REFERENCES

1. Brownsword, Lisa, et al. *System-of-systems navigator: An approach for managing system-of-systems interoperability*. No. CMU/SEI-2006-TN-019. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2006.
2. Carney, David, William Anderson, and Patrick Place. *Topics in Interoperability: Concepts of Ownership and Their Significance in Systems of Systems*. No. CMU/SEI-2005-TN-046. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2005.
3. Coetzee, Marijke, and Jan HP Eloff. "An access control framework for web services." *Information management & computer security* 13.1 (2005): 29-38.
4. Connolly, Thomas M., and Carolyn E. Begg. *Database systems: a practical approach to design, implementation, and management*. Addison-Wesley Longman, 2005.
5. Coulouris, George F., Jean Dollimore, and Tim Kindberg. *Distributed systems: concepts and design*. Addison-Wesley Longman, 2005.
6. Da Silva, Catarina Ferreira, et al. "Semantic interoperability of heterogeneous semantic resources." *Electronic Notes in Theoretical Computer Science* 150.2 (2006): 71-85.
7. Fisher, David. "An emergent perspective on interoperation in systems of systems." (2006).
8. Guijarro, Luis. "Interoperability frameworks and enterprise architectures in e-government initiatives in Europe and the United States." *Government Information Quarterly* 24.1 (2007): 89-101.
9. Hasselbring, W. and weigand, H., "Languages for the electronic business communication: State of the art", *Industrial management & data systems* 2001; 101(5): 217-226.

10. Lakshmanan, Laks VS, and Fereidoon Sadri. "XML interoperability." *ACM SIGMOD Workshop on Web and Databases (WebDB), San Diego, CA*. 2003.
11. Morris, Edwin, et al. *System of Systems Interoperability (SOSI): final report*. No. CMU/SEI-2004-TR-004. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2004.
12. Nicholson, Dennis, and Ali Shiri. "Interoperability in subject searching and browsing." *OCLC Systems & Services* 19.2 (2003): 58-61.
13. Seng, J., Liu, Y., Wang, J. and Yu, J., "An analytical study of XML database techniques", *Industrial management & data systems* 2003; 103 (2): 111-120.
14. Singh, Valdew. "Systems integration-coping with legacy systems." *Integrated Manufacturing Systems* 8.1 (1997): 24-28.
15. Smith, James D. "Topics in Interoperability: Structural Programmatics in a System of Systems." (2006).
16. Smith, Alan D. "Exploring potential strategic impacts of XML-related technologies." *Information Management & computer security* 11.2 (2003): 92-100.
17. Iqbal, Rahat, Anne James, and Richard Gatward. "A framework for interoperability of heterogeneous systems." *Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on*. IEEE, 2003.
18. Chituc, Claudia-Melania, Américo Azevedo, and César Toscano. "A framework proposal for seamless interoperability in a collaborative networked environment." *Computers in Industry* 60.5 (2009): 317-338.
19. Huang, Ying, and Jen-Yao Chung. "A Web services-based framework for business integration solutions." *Electronic Commerce Research and Applications* 2.1 (2003): 15-26.
20. Benguria, Gorka, and Xabier Larrucea. "Data model transformation for supporting interoperability." *Commercial-off-the-Shelf (COTS)-Based Software Systems, 2007. ICCBSS'07. Sixth International IEEE Conference on*. IEEE, 2007.
21. Miori, Vittorio, et al. "DomoNot: a framework and a prototype for interoperability of domotic middlewares based on XML and Web Services." *International Conference on Consumer Electronics (ICCE'06)*. 2006.
22. Yahia, Esma, Alexis Aubry, and Hervé Panetto. "Formal measures for semantic interoperability assessment in cooperative enterprise information systems." *Computers in Industry 63.1* (2012): 443–457
23. Kazemzadeh, Reza Sherafat, and Kamran Sartipi. "Interoperability of data and knowledge in distributed health care systems." *Software Technology and Engineering Practice, 2005. 13th IEEE International Workshop on*. IEEE, 2005.
24. Sehgal, Saurabh, et al. "Understanding application-level interoperability: Scaling-out MapReduce over high-performance grids and clouds." *Future Generation Computer Systems* 27.5 (2011): 590-599.
25. Mogul, Jeffrey C. "Emergent (mis) behavior vs. complex software systems." *ACM SIGOPS Operating Systems Review*. Vol. 40. No. 4. ACM, 2006.
26. Cappello, Franck, et al. "Computing on large-scale distributed systems: XtremWeb architecture, programming models, security, tests and convergence with grid." *Future Generation Computer Systems* 21.3 (2005): 417-437.