IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 24, Issue 2, Ser. II (Mar. –Apr. 2022), PP 32-39
www.iosrjournals.org

Survey on the development of an Insurance Application Systems Interoperability toolbox for business organizations

Fungai Jacqueline Kiwa* Shakemore Chinofunga** Mary Shangwa***
**Department of ICT and Electronics, School of Engineering Sciences & Technology Private Bag 7724, Off-Chirundu Road,+263 67 29467, Zimbabwe*
***Department of Information Systems Engineering, School of Engineering Sciences & Technology Private Bag 7724, Off-Chirundu Road,+263 67 29467, Zimbabwe*
****Department of ICT and Electronics, School of Engineering Sciences & Technology Private Bag 7724, Off-Chirundu Road,+263 67 29467, Zimbabwe*
*****Department of Mechatronic Engineering, School of Engineering Sciences & Technology Private Bag 7724, Off-Chirundu Road,+263 67 29467, Zimbabwe*

Abstract
This paper covers segment of available solutions and methods as well as the advancements interoperability system technology that has been made within the area of concern that is cross-platform interoperability. The literature review focuses on previously used techniques to achieve sharing of resources between ICT systems. For decades, computer program development requires the utilization of measured useful components that perform a particular work in different places inside an application. As application integration and component-sharing operations got to be connected to pools of facilitating assets and conveyed databases, endeavors required a way to adjust their procedure-based improvement model to the use of inaccessible, conveyed components. The proposed system covers a lot of gaps that came from the close analysis of previous systems performance. The paper includes proposed system test cases with acceptance testing. The content in literature has aided the researchers in adopting the trending techniques in the interoperability of ICT systems.

Keywords: *Distributed computing, cross-platform, inter-process communication, interoperability, message passing, procedure call, programming languages, remote procedure call, service-oriented architecture*

Date of Submission: 26-03-2022 Date of Acceptance: 06-04-2022

Abbreviations

SOA-	Service Oriented Architecture
RPC-	Remote Procedure Call
CORBA-	Common Object Request Broker Architecture
DCOM-	Distributed Component Object Model
COM-	Common Object Management
OMA-	Object Management Architecture
ORB-	Object Request Broker
OMG-	Object Management Group

I. Introduction

One method to several of the issues in systems integration is to supply interoperable software frameworks (Potter 1992, Potter et al. 1994, Otte 1996). Interoperability is the capability of two or more computer program components to take part by trading services and data with one another, despite the conceivable heterogeneity in their dialect, interface, and hardware platform (Heiler 1995, Wegner 1996). Interoperable computer program models provide a standard that advances communication between components and gives for the integration of bequest and recently made components. Within the previous decade, organizations have been moving mainframe-based systems toward open and dispersed computing circumstances. A passed-on computing environment is an environment where various computers are organized together and are allowed to share data and handle duties. The request for interoperability has been driven by the enlivened improvement of large-scaled dispersed frameworks for operational use and by extending the use of the Web cross-platform (Manola, 1995).

In this chapter, the researchers will analyze the current cross-platform interoperability systems that have to do a lot about communication between different systems that run on different platforms besides service-oriented architecture (SOA). The main aim is to come up with the analysis so that we know what each system does which will help us to achieve our objectives. The techniques which have been applied by other researchers on this topic will also be discussed. A full description of the proposed system will be explained. Interoperability has been proven to be the best technique in today's world to save development costs and to allow systems to communicate (Walsh &Winer 1998, Udell 1999). The literature below unveils the work that has been done in ensuring sharing of resources between different systems running on different platforms.

II. Remote Procedure Call (RPC)

RPC is a computer communication tradition that supports interoperability on the program running on one computer to call subprograms on another computer, without coding for inaccessible interaction (Randy J., 2000). RPC is a synchronous operation requiring the asking program to be suspended until the results of inaccessible strategy are returned. Hence, the client is blocked whereas the server is preparing the call and only continued execution after the server is wrapped up. RPC components are utilized when a computer program causes a method or subroutine to execute in a distinctive area, which is coded as a normal method call without the software engineer especially coding the subtle elements for the blocked off interaction, thus RPC enables the utilization of the applications within the dispersed environment, not only within the nearby environment. RPC is a communication procedure and it is utilized for server-client applications or vice versa. This method call moreover manages low-level transport conventions, such as Client Datagram Convention, Transmission Control Convention, etc. It is utilized for carrying the message and data between programs (Kohun F.,2008).

RPC framework has five main components shown in the diagram below:

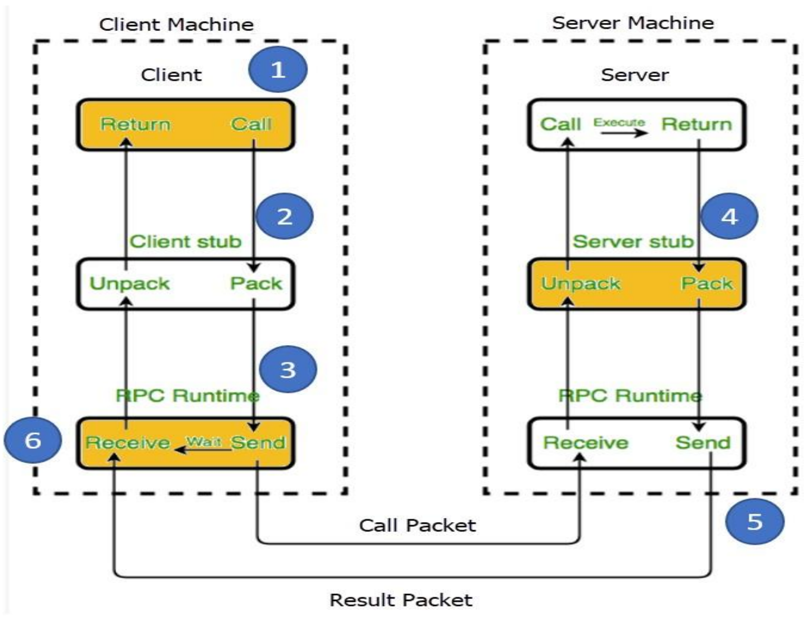

Figure 1: Architecture of RPC (Onsman, 2018)

When an inaccessible method call is conjured, the requesting environment is suspended, the method variables are exchanged over the internet to the environment where the technique is to execute, and the method at that point is executed in that environment. When the methodology wraps up, the results are traced back to the calling environment, where execution resumes as if returning from an ordinary strategy call.

Steps during the RPC process:

i. The client stub, the client, and one occasion of RPC runtime running on the client computer.

ii. A client starts a client stub handle bypassing variables inside the ordinary way. The client stub stores interior the client's claim address space. It additionally asks the adjacent RPC Runtime to send back to the server stub.

iii. In this arrangement, RPC got to the client by making a uniform Neighborhood Procedural Call. RPC Runtime oversees the movement of messages between the network over server and client. It moreover performs the work of steering, retransmission, encryption, and affirmation

Iv. After finishing the server technique, it returns to the server stub, which packs the return values into a message. The server stub then sends a message back to the transport layer.

v. In this step, the transport layer sends back the result message to the client transport layer, which returns a message to the client stub.

vi. In this phase, the client stub de-marshalls (empty) the return parameters, inside the resulting allocation, and the execution handle returns to the caller.

Remote procedure calls help clients to communicate with servers through the conventional use of method calls in tall level languages and can be utilized in a disseminated environment, as well as the nearby environment. RPC underpins thread-oriented and process-oriented models thus it covers up the inside message-passing instrument from the client and requires negligible exertion to redevelop and revamp the code; reflects, i.e., message-passing nature of arranging communication is covered up from the user; and overlooks numerous of the convention layers to move forward performance. RPC permits the utilization of the applications in a conveyed environment that's not as it was within the neighborhood environment and with RPC; code redeveloping and re-vamping exertion are reduced (A. Chakraborty, 2019).

In conclusion, RPC specification itself doesn't provide any degree of well-defined interoperability between clients and servers from different implementations. The server and client utilize distinctive execution situations for their particular schedules, and the utilization of assets e.g., records are too more complex. There's no uniform for RPC, it can be actualized in an assortment of ways.

III. Common Object Request Broker Architecture (CORBA)

CORBA is an architectural system created by the Object Management Group (OMG) in 1991 as a portion of the standard in Object Management Architecture (OMA) to provide interoperability among distributed objects. The OMA set of principles comprises object services, Object Request Broker (ORB) work, common facilities, application objects, and space meddle. CORBA is the world's driving middleware arrangement empowering the exchange of data, independent of hardware platforms, programming dialects, and operating systems. CORBA is a plan determination for an ORB, where an ORB gives the component required for disseminated objects to communicate with one another, whether locally or on inaccessible gadgets, composed of completely distinctive languages, or at different areas on a network.

CORBA is organized to allow the integration of a wide range of challenge frameworks and give components to find object execution of atask, to arrange usage to induce the request, and to communicate with the information that produces the ask. CORBA gives a component to characterize interfacing between components and demonstrates standard administrations such as tireless protest services, catalog and naming services, and exchange services which delineate the interoperability feature for CORBA compliant applications. OMG's determinations separate into two major parts: CORBA (Figure 4), which gives the object-based interoperability establishment and built on this establishment, the conceptually layered (but not dynamic) Object Management Architecture (OMA: Figure 5). The figure underneath shows a request passing from a client to an object execution inside the CORBA design. Two viewpoints of this design stand out:

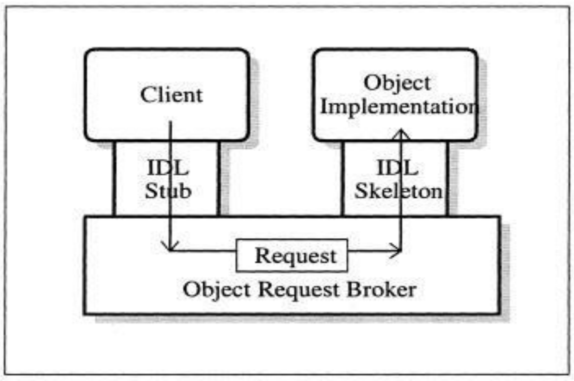

Figure 2: CORBA architecture (Siegel, 1999).

Client and object execution are disconnected from the ORB by an OMG IDL interface. CORBA requires that each object's interface be communicated in OMG IDL. Clients only see the object's interface; never any utilization detail. This guarantees the substitutability of the users behind the interface of our plug-and-play component program environment. The request does not pass particularly from the client to protest utilization; instep, demands are persistently directed by an ORB. Each conjuring of a CORBA object is passed to the ORB; the frame of the invocation is the same whether the target object is adjacent or inaccessible (if inaccessible, the conjuring passes from the ORB of the client to the ORB of the object execution). Conveyance points of interest dwell only within the ORB where they are taken care of by the computer program you bought, not the computer program you built. Application code, freed of this regulatory burden, concentrates on the issue at hand.

Building on CORBA, OMG's (Object Management Architecture) (Figure 2) gives the premise for venture computing. The CORBA services deliver system-level administrations required by nearly any object-based system, though the CORBA facilities, basically in industry-specific (vertical) zones, allow standards-based get to common data types and usefulness required in endeavor computing. Together, they empower an endeavor computing to appear composed of interoperating objects from numerous merchants and designers - the component computer program insurgency.

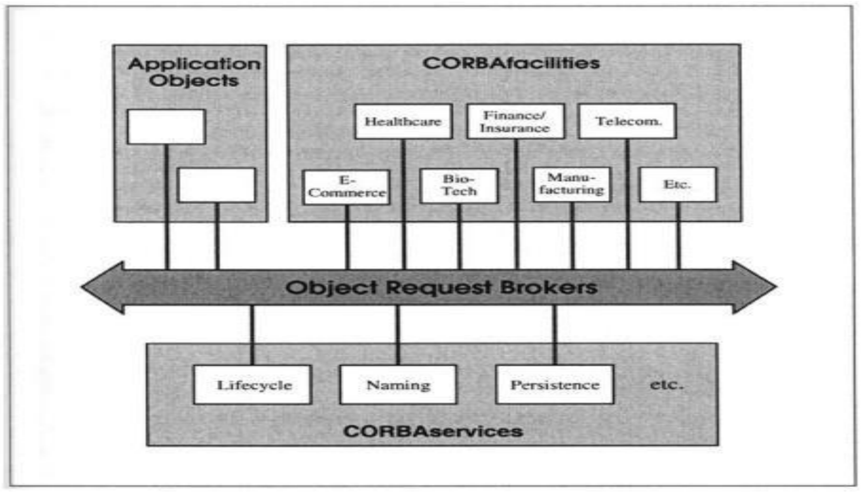

Figure 3: The Object Management Architecture (Siegel, 1999).

The basic ORB interoperability prerequisite is to permit clients to utilize such protest names to conjure operations on objects in a neighborhood ORB or an inaccessible one. Giving straightforwardness can be exceptionally included, and naming modes are vital to it. An ORB gives the components by which an object's straightforwardness makes and gets demands and reactions. In so doing, the ORB provides communication between applications on diverse machines in heterogeneous conveyed situations. ORB interoperability expands this definition to cases in which servers and clients' objects on particular ORB's transparently make and get requests. Note that a coordinate result of this straightforwardness necessity is that bridging must be bidirectional: that is, it must work as reasonably for dissent references passed as parameters as for the target of an object conjuring. We're bridging unidirectional (e.g., on the off chance that one ORB appears only a client to another), at that point straightforwardness would not have been given since protest references passed as parameters would not work precisely: ones passed as "callback objects," for illustration, seem not to be utilized. Without the misfortune of simplification, most of this determination centers on bridging in as it were one course. This can be absolutely to disentangle talks and does not infer that a unidirectional network fulfills essential interoperability necessities (Object Management Group, 2012).

In conclusion, the CORBA interface might solve interoperability by giving interoperation features for overseeing disparate systems, moreover, it has some disadvantages which are not perfect for execution in a smart home environment. The gap in this technology is that it allows passing generic objects which can have methods and there is no real CORBA standard to bind an ORB and its clients to a port, meaning there are only specific options. CORBA has standardized interfaces and a standardized protocol too. In addition, during the standardization, it has been overlooked to decide the harbor of the communication. Subsequently, network directors had to open a modern port for each CORBA communication accomplice within the firewall. CORBA requires data on the inaccessible systems sometime recently working and inevitably leads to adjusting the legacy systems that don't comply with cross-platform interoperability (Helmat P, 2004).

IV. Distributed Component Object Model (DCOM)

The development of DCOM began with the introduction of a concept introduced with the earlier graphical user interfaces. This concept placed a heavy emphasis on the interoperability and seamless exchange of objects related to documents. The concept was known as Dynamic Data Exchange (DDE). DDE was designed to allow applications to exchange data of any particular type (Rodent, 1992). DCOM is one of the developing models for disseminated objects. DCOM gives distributed component communications. The DCOM model is a set of extensions of the COM (Common Object Management) environment and was developed and shipped after the COM system was deployed into the marketplace. The DCOM architecture points to intercepting all the local inter-process communication messages and substituting them with a network protocol based on the Distributed Computing Environment (DCE). Additionally, the protocol might also be responsible for routing the request through a network environment. For DCOM to work, the COM object has to be arranged accurately on both computers, thus it conveys network administrators to COM, permitting DCOM-enabled computer program components to communicate over the internet in a comparative form to the technique by which COM components communicate among themselves on a single computer. DCOM works natively with Internet advances like TCP/IP, Java languages, and HTTP network conventions. DCOM gives the object stick that permits business applications to work over the web.

DCOM objects client make demands for administrators from DCOM server objects on distinctive machines on the network employing a standard set of interfaces. The client object cannot call the server protest straightforwardly; instep, the working framework intervention the DCOM request and uses the inter-process communication strategies such as remote procedure calls (RPC) to supply a straightforward communication instrument between the client and server objects. The COM run time gives the basic object-oriented administrations to the client and server objects. The COM run-time moreover employs the security supplier and RPCs to form network outlines that accommodate the DCOM standard (Finin et al. 1994, Genesereth & Ketchpel 1994). This is shown in the figure below:

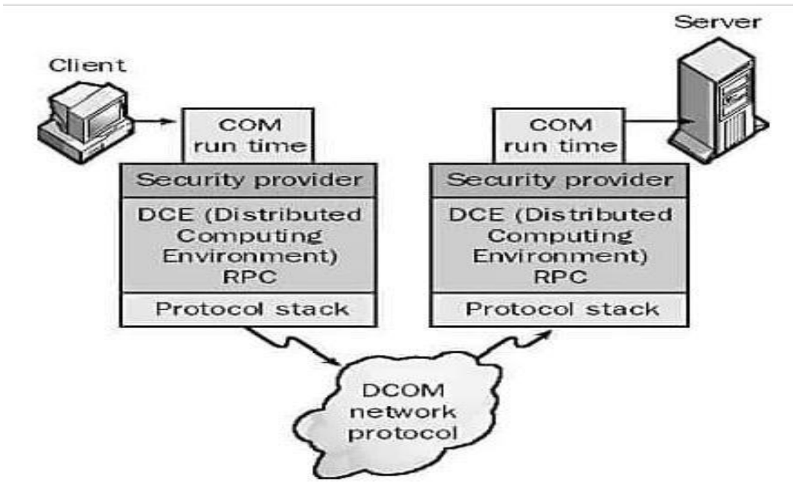

Figure 4: Distributed Component Object Model (DCOM), (Finin et al. 1994).

DCOM provides standard determinations for accomplishing dialect interoperability and platform autonomy. They characterize their interface measures to bargain with the quirks of legacy applications. They bolster dispersed processing, object reuse, and the Internet. This architecture is well archived and the documentation materials are effortlessly available to the open online as well as through books and diary articles. In conclusion, DCOM provides standard determinations for accomplishing dialect interoperability and platform independence. They characterize their guidelines to bargain standards to deal with the quirks of bequest applications. They bolster object reuse, dispersed processing, and the Internet.

V. Proposed System

Lack of cross-platform interoperability is a major problem in computing. Most software's are platform-dependent meaning they run only on windows, android, etc. Therefore, the communication between the different systems is hindered and could not be achieved effectively using the above-mentioned technologies. To solve the research gap, we want to develop a consumer black box that complements the currently adopted Interoperability systems. This toolbox allows the users of SIAM to use services that are available on a network to communicate with their clients without depending on the platform to be used. Services communicate with other applications through common language which means it is independent of the platform on which the application is running and services can be used by multiple applications at the same time.

VI. Proposed system test cases

During the project implementation, major test cases were conducted based on the testing process design and certain areas of concerned were rectified of any errors.

Table 1: Test Cases performed on the proposed system

Test case	Action to execute	Expected Outcome	Actual Result	Comment
Show Client Registration page	Navigate to the web URL localhost/ or open the Mobile App	Display User Register View by default when there no session present for any user on the browser or app.	Display User Login View by default when there no session present for any user on the browser or app.	PASSED

Test case	Action to execute	Expected Outcome	Actual Result	Comment
Show Client Login View	Navigate to the web URL localhost/ open the Mobile App	Display User Login View by default when there no session present for any user on the browser or app	Display User Login View by default when there no session present for any user on the browser or app	PASSED
Agent and Client Login into the application.	Input user credentials in the text area and click the login button.	If input credentials are correct, then present page else return a message showing Invalid Username or Password.	If input credentials are correct, then present page else return a message showing Invalid Username or Password.	PASSED
Client navigate through the activities	Upon login, the client can navigate through the activities (that is choosing what he or she wants to do)	The client must see a page with various activities such as Request for Insurance Policy, Q & A and Rates	A View with various activities must be shown to the client	PASSED
Agent navigate through the activities	Upon login, the Agent can navigate through the activities (that is choosing what he or she wants to do)	The Agent must see a page with various activities such as Exchange Rates, Weather Forecast, Q & A, Requests and Insurance Companies	A View with various activities must be shown to the client with the following tabs: Exchange Rates, Weather Forecast, Q & A, Requests and Insurance Companies	PASSED

Test case	Action to execute	Expected Outcome	Actual Result	Comment
Requesting of Insurance Policies by the client	Once the client login to the system, can select the page named Request for Policy and then provide their username, select the type of insurance and optionally provide the amount budgeted and then click the request button	The client should be able to provide their username, select insurance type and then click the request button. The client must be able to see insurance information about policies based on what they searched.	The client was able to provide their username, select insurance type and click the request button. After clicking the button, the client was provided with responses based on their searches.	PASSED
Asking Questions by Clients	Navigate to Questions page, provide the following: username, phone number and message and click send button. A client will see a message: Message Sent Successfully.	The client must be able to provide username, phone number and message and click send button. Lastly, a client must be able to see a notification which shows that a message sent successful.	The client was able to provide username, phone number and send message and see the notification "Message Sent Successfully".	PASSED
Viewing Exchange Rates by Agents	Agents navigate to Exchange Rates tab, choose the option to use in selecting rates either: Currency or Currency and Day. Click Search button and see the result presented in the grid view.	The agents must be able to choose the option to use in selecting rates either: Currency or Currency and Day. Click Search button if option 2 is selected and see the result presented in the grid view.	The agents were able to choose the option to use in selecting rates either: Currency or Currency and Day. Click Search button and see the result presented in the grid view.	PASSED
Test case	Action to execute	Expected Outcome	Actual Result	Comment
Viewing Weather Conditions of a certain area by Agents	Agents to navigate to Weather Forecast tab, enter the location name and send request. After sending, see the result on the provided area.	The agents must enter valid location name and send the request, then weather conditions details will be provided.	The agents were able to enter location name and send the request, then weather conditions details were provided.	PASSED
Replying Clients Questions by Agents	Navigate Q & A tab, select reply on certain client message, a pop-up form is provided, under that enter clients phone number and message and send the message.	The Agents must be able to send replies to clients by providing client's phone number and message.	The Agents were able to replies to clients by providing client's phone number and message.	PASSED
Requesting New Policies by Agents	Navigate to Requests tab, select Insurance Company, Type of Insurance and Search.	The agents must be able to search for new policies from an insurance company.	The agents were able to search for new policies when published by Insurance Companies.	PASSED

VII. System Testing Results

The researchers tested the whole system by creating user accounts and logging in to the system. After log in, they navigated through the system to check if they clicked menu buttons, the system would direct them to the desired pages. They tested if they were able to give required input for requesting policies, asking questions and to see if the system would give results. They also tested if the system was able to connect to other services that are available online such as SMS pro for sending notifications and replies to clients, Weather Forecast for searching the weather conditions of a certain area, searching exchange rates and the system was able to give the desired outcome. The table below shows the results of testing the whole system.

Table 2: System testing requirements results

Functional Requirements	**Result**
Create a user account for future use of the system.	Pass
Requesting insurance policies	Pass
Asking Questions	Pass
Sending Replies	Pass
Searching weather conditions	Pass
Searching for Exchange rates	Pass

VIII. Acceptance Testing

This is the last level of testing which involves users of the system. Potential users are allowed to test the software measuring their satisfaction, the systems friendliness and usability. The driving question behind this test is to evaluate to what extent the users are seeing the solutions to be impactful, useful and acceptable as a dependable product. The researchers managed to deploy and test their system at Smart Insurance Agents Marondera(Desktop System) and with some of their clients (Mobile App). The researchers showed the Agents and clients how the system works. The Agents and their clients tested the system, thus they created their accounts and also navigated through the system to familiarize with the system functionality.

IX. Validation and Verification Testing

Validation testing seeks to answer the questions “Are we building the right software system?” and “Is the deliverable fit for purpose?” The main purpose of validation testing is to deliver ensure and deliver quality. System Validation was achieved through the comparison of data entered and the output. This was done to ensure that the data captured is as required and the output is as required. Validation mainly checks values entered in the textboxes if they correspond to the specified field type. This means that the system only accepts data that is compatible with the destination field.

X. Conclusion

Remote Procedure Call(RPC) is a computer communication tradition that supports interoperability on the program running on one computer to call subprograms on another computer, without coding for remote interaction (Randy J, 2000). RPC is a synchronous operation requiring the asking program to be suspended until the results of inaccessible strategy are returned. Hence, the client is blocked whereas the server is preparing the call and only continued execution after the server is wrapped up. RPC enables the utilization of the applications within the dispersed environment, not only within the nearby environment. However, The RPC specification itself doesn't provide any degree of well-defined interoperability between clients and servers from different implementations. The server and client utilize distinctive execution situations for their particular schedules, and the utilization of assets e.g., records are too more complex.

Secondly, Common Object Request Broker Architecture **(**CORBA) is another method that is a standard created in 1991 by the Object Management Group (OMG) to provide interoperability among disseminated objects. CORBA is the world's driving middleware solution empowering the trade of information, autonomous of hardware platforms, working frameworks, and programming languages. The gap in this technology is that it allows passing generic objects which can have methods and there is no real CORBA standard to bind an ORB and its clients to a port, meaning there are only specific options. CORBA has standardized interfaces and a standardized protocol too. Other than, during the standardization, t has been overlooked to decide the harbor of the communication. Subsequently, network directors had to open an unused port for each CORBA communication (Helmat P, 2004).

Lastly, Distributed Component Object Model (DCOM) is the last solution we found which is one of the rising guidelines for dispersed objects. DCOM provides distributed component communications. The DCOM aims to intercept all the local inter-process communication messages and substitute them with a network protocol based on the Distributed Computing Environment (DCE). DCOM provides the protest stick that permits commerce applications to work over the Net. Apart from what was done on DCOM, the technology has a challenge that it failed to solve such as platform dependence, DCOM does not support multiple inheritances, and rather it supports multiple interfaces.

Lack of cross-platform interoperability is a major problem in computing. Most software's are platform-dependent meaning they run only on windows, android, etc. Therefore, the communication between the different systems is hindered and could not be achieved effectively using the above-mentioned technologies. To solve the research gap, we want to develop a consumer black box that complements the currently adopted Interoperability systems. This toolbox allows the company to use services that are available on a network to communicate with their clients without depending on the platform to be used.

The existing systems for cross-platform interoperability do not provide the degree of well-defined interoperability between the system clients and the services that are available online without considering the platform used. Most of these systems are platform-dependent they give standard determinations for accomplishing dialect interoperability and sharing of resources in hybrid platform apps is not well defined though CORBA has managed to allow hybrid platform apps to communicate, thus our objectives are not fully met by these systems since the foremost goal of this project is to demonstrate that sharing of resources between ICT systems could be achieved using a consumer black box. Therefore, this toolbox allows the users of SIAM to access weather information and advice motorists to avoid risks associated with transportation and to be able to use other services that are available on a network to communicate with their clients without depending on the platform to be used since SOA provides us a room to demonstrate to increase adaptability and productivity. Services communicate with other applications through common dialect which suggests it is autonomous of the platform on which the application is running and administrators can be utilized by numerous applications at the same time.

Modern innovations are continually being created to make strides in existing computing approaches. The zone of conveyed computing is no exemption. For standard clients that require a cross-platform interoperability solution nowadays, SOA gives the finest approach. The literature reflected above points to the significance of interoperability by SOA. Of course, it may not be the driving approach in a long time but that's the way it goes in a tall advancement field; alter comes rapidly. In addition, the literature confirms that other methods besides SOA allow us to demonstrate that cross-platform interoperability could be achieved. That is why we chose the above methods as they helped us in understanding interoperability.

Reference

[1]. George, T. Aphrodite, and P. Michael (2006.), "Interoperability among Heterogeneous Services," in Services Computing, SCC '06. IEEE International Conference on, pp. 174-181.
[2]. Gustavo A, Fabio Casati, Harumi Kuno and Vijay Machiraju (2004), Web Services: Concepts, Architectures, and Applications, Springer-Verlag Berlin Heidelberg
[3]. Grid Wise™ Architecture Council (2008), "Interoperability Context-Setting Framework".
[4]. Cox, D. E., and Kreger, H. (2005). "Management of the Service-Oriented-Architecture Life Cycle," IBM Systems Journal 44(4), 709-726.
[5]. Marks, E. A., and Bell, M. (2006). Service-Oriented Architecture: a Planning and Implementation Guide for Business and Technology, John Wiley & Sons, Hoboken, NJ
[6]. Papazoglou, M. P., and Georgeakopoulos, D. (2003). "Service-Oriented Computing," Communications of the ACM 46(10), 25-28
[7]. Jeffrey Kaplan et al (2005)., "Roadmap for Open ICT Ecosystems," Berkman Center Publication Series, p. 4 - 5.
[8]. Jonathan Zittrain (2008), The Future of the Internet – and How to Stop It (Yale University Press).
[9]. Jonathan Zittrain (2006), "The Generative Internet," 119 Harv. L. Rev. 1974.
[10]. Eric von Hippel (2002), Open-source projects as horizontal innovation networks - by and for users, MIT Sloan School of Management.
[11]. Clayton Christensen (1997), The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail (Harvard Business School Press).
[12]. The Component Object Model Specification, Oct. 1995
[13]. N. Brown and C. Kindel (1996), Distributed Component Object Model Protocol -- DCOM/1.0, Nov.
[14]. Y. Huang and C. Kintala (1993), "Software implemented fault tolerance: Technologies and experience", Proc. IEEE Fault-Tolerant Computing Symp., pp. 2-9, -June
[15]. Windows NT Magazine (1997) "Clustering Solutions for Windows NT", pp. 54-95.

Fungai Jacqueline Kiwa, et. al. "Survey on the development of an Insurance Application Systems Interoperability toolbox for business organizations." *IOSR Journal of Computer Engineering (IOSR-JCE),* 24(2), 2022, pp. 32-39.