



## Software Compliance Management Model towards Portability Faults using RESTful Web Service Exchanges in Distributed Systems

<sup>1</sup>Kanchana Natarajan, <sup>2</sup>Sarala Subramani

<sup>1</sup>Research Scholar, <sup>2</sup>Assistant Professor

Department of Information Technology  
Bharathiar University, Coimbatore, India

**Abstract:** The objective of the research work is to propose a software service compliance management model towards portability fault tolerance (PFT) using RESTful web services exchanges in distributed systems. The compliance management is achieved through correct identification of the various non-compliances while deploying and executing software services on various hardware and system platforms. The portability faults are due to operational platform changes (OPC), service audit constraints (SAC) and multiple service contracts (MSC). These portability faults can be detected and identified through the proposed set of services, namely SARALA that are based on computational situations, resources and locations of the needed services. The software services in HIPAA framework for mobile health care system are to be portable but also accountable where both are also to be compliant with existing government policies and regulations. The incorrect portability attributes which lead to non-compliances in the network protocol, the third party software and data format are checked to detect the causes of policy violations and activate the necessary enforcement actions to meet the healthcare acts.

**Keywords:** software service, compliance management, portability faults, situation awareness, resource awareness, location awareness, fault tolerance

### I. Introduction

In the world of software services through virtual networks across heterogeneous devices, the applications or services rendered by software is to be conformance to the user expectations. The software is to have a high degree of fitness with functional and cost benefit requirements at minimum defect level in all its operational and transition phases. The software has to meet its quality expectations in its totality of features and characteristics in all environments. The challenge in achieving the software functionalities in a different computational platform or environment depends on its portability with sensitivity and selectivity. The software may be portable and operational in one environment and may not deliver the same quality of service in a different environment. The software community is expecting this feature of maximum usability of an application or a service or a virtual machine as a whole as the right form of portability. The earlier models discuss these dispersed portability concepts and terms into a standards-based reference framework of system portability requirements [1]. For example, the framework within the context of a concrete healthcare scenario in which obligations incurred from Health Insurance Portability and Accountability Act (HIPAA) are delegated and refined into software requirements. Most of the earlier research works focus on the privacy and security issues of e-health or mobile healthcare systems but failed to contribute much when the software systems are in transition phase due to improvised standards and policies in the healthcare system. The data that has to be kept confidential may sometimes need to be shared or ported to the authorities and similarly the entire machine has to be migrated into another physical host due to sudden catastrophic incidents detected in the computational environment [2] [3]. Information privacy needs to be controlled on the data (PII) level, the user needs to be able to determine the sensitivity and context of the PII provided, privacy-preserving data portability can only. There are works on Digital Rights Management where the problem of portability of applications between different software-hardware platforms were addressed [4]. From a machine to a set of binary codes, the portability is too emphasized and the economical part of the virtual machine usage by the end user also deserves special attention. Portability of the source code is potentially applicable to a wider class of platforms. In a number of cases, the use of cross-platform libraries may be helpful; another alternative is to use interpretable languages. The improving of application portability generally increases cost of its development; however, the earlier the designers and architects think of this problem, the less the cost of porting this application to a new platform [5]. However, there is no evidence that HIPAA is associated with an overall improvement in transaction delivery time or quality [6]. The portability of designs generated by front-end tools is much greater, and such designs can be used to generate systems written in COBOL or 4GLs without the use of hardware-specific features. The adaptability of the software is closely linked with the maintainability of the code [7]. Product Transition in case of portability has sub-factors such as software system independence, modularity and self descriptive. Portability

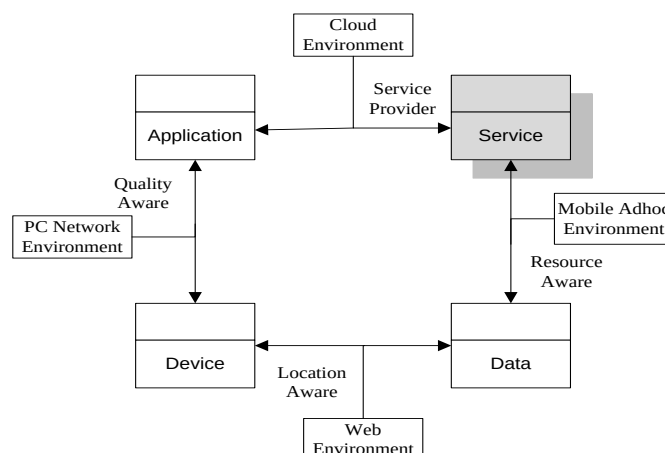
requirements specify the environments in which the software system has to be operable, the operating systems, and hardware and communication equipment standards, among other variables. The portability test to be carried out will verify, validate and test these factors as well as estimate the resources required for transfer of a software system to a different environment [8]. With all these information explored above, the main focus of the paper is to propose a portability compliance aware framework with the help of exchange servers through RESTful web services not only to minimize the portability faults but also improve the overall governance by reducing the data or service non availability risks.

The paper is organized as follows: Section 2 brings the various existing software product standards in the context of portability of data or application or the platform across web services. Section 3 discusses about the need for RESTful web services towards achieving the portability compliance in a distributed software based system. Section 4 proposes a situation aware, resource aware and location aware (SARALA) services that will monitor and manage the detection of portability issues in the distributed system. Section 5 brings the execution and implementation of the proposed portability compliance management model using RESTful services and proposes the portability compliance metrics. Section 6 concludes the work on software compliance management model with its limitations and the relevant inferences from the simulation for the service interactions.

## II. ISO/IEC Software Product Quality Standards - Portability

The service or software system standards help the provider and also the user to know the limited functionality of the requested service and also the expected behavior of software services in terms of the individual component services. For example, the ISO/IEC 9126 mentioned the standard states of three approaches to quality (internal quality, external quality and quality in use). Internal and external quality share a set of six high-level factors or characteristics (functionality, reliability, usability, efficiency, maintainability and portability), each of them divided in a few sub characteristics. Quality in use is determined by a set of four characteristics (effectiveness, productivity, safety, and satisfaction). The standard also defines as final target the definition of the lowest-level quality factors or attributes together with the metrics to use in measurement. Similarly, the ISO/IEC 12119 standards describe the set of requirements for software packages and instructions on how to test a software package against the requirements. And the ISO/IEC 24765 – The standard describes the system-related portability requirements as a system or component that can be transferred from one hardware or software environment to another. The standard states the portability in a system environment refers to a transfer between software and hardware or both. The ECSS Standards series includes the number of portability requirements at the system level is considered as the capability of the system to be transferred from one environment to another. Portable web services can be built by implementing through platform independent languages. It is hard to determine the portability with quantifiable effort with system. Portability is the usability of the same software in different environments. ISO defines usability as the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use. The adaptability is the extent to which a software system adapts to change in its environment that includes installability that means the capability of the software product to be installed in a specified environment. The replaceability or the capability of the software product to be used in place of another specified software product for the same purpose in the same environment as per ISO 9126: 2001 is also playing an important feature in determining the portability of the software. The portability faults or the portability non compliances as per HIPAA implicitly are focusing on the failures in the adaptation, installation and replaceability of the application or service software or even the data in various operational and maintenance or revision phase of the software product or service. If portability is the usability of the same software in different environments, then usability is a complex mixture of multiple features like adaptability, installability, co-existence and replaceability.

**Figure. 1 Entity-Environment-Relationship**



The Entity-Environment-Relationship (E2R) flow shown in figure.1 indicates the relationship between entities like data, applications, services and devices in different environments. For example: Mobile environment is needed to port an entire service or needed data through agents. Cloud environment is needed to port the application as a service into another server or migrate the virtual machine from one host to another host. PC Network environment is needed to port the application through different adaptable devices. Web environment is needed to port the data through devices.

**Table 1: Portability Factors and Web Services**

| Portability Sub-Attributes  | RESTful Web Services        | SOAP Web Services           |
|-----------------------------|-----------------------------|-----------------------------|
| Adaptability                | Device Based Mobile or PDAs | Network and Interface Based |
| Installability and Exchange | Architecture Principles     | Communication Protocol      |
| Co-existence                | Simple Transaction          | Complex Transaction         |
| Replaceability              | Stateless Services          | Stateful Services           |

### III. REST Vs SOAP Web Services

REST describes a set of architectural principles by which data can be transmitted over a standardized interface such as HTTP and URI. A client can access the resource using the unique URI and a representation of the resource is returned. While accessing RESTful resources with HTTP protocol, the URI of the resource serves as the resource identifier like GET, PUT, DELETE, POST and HEAD are the standard HTTP operations to be performed on that resource. REST separates user interface concerns from data storage concerns. It improves portability of interface across multiple platforms and simplifies server components by making them stateless. SOAP defines a standard communication protocol (set of rules) specification for XML-based message exchanges.

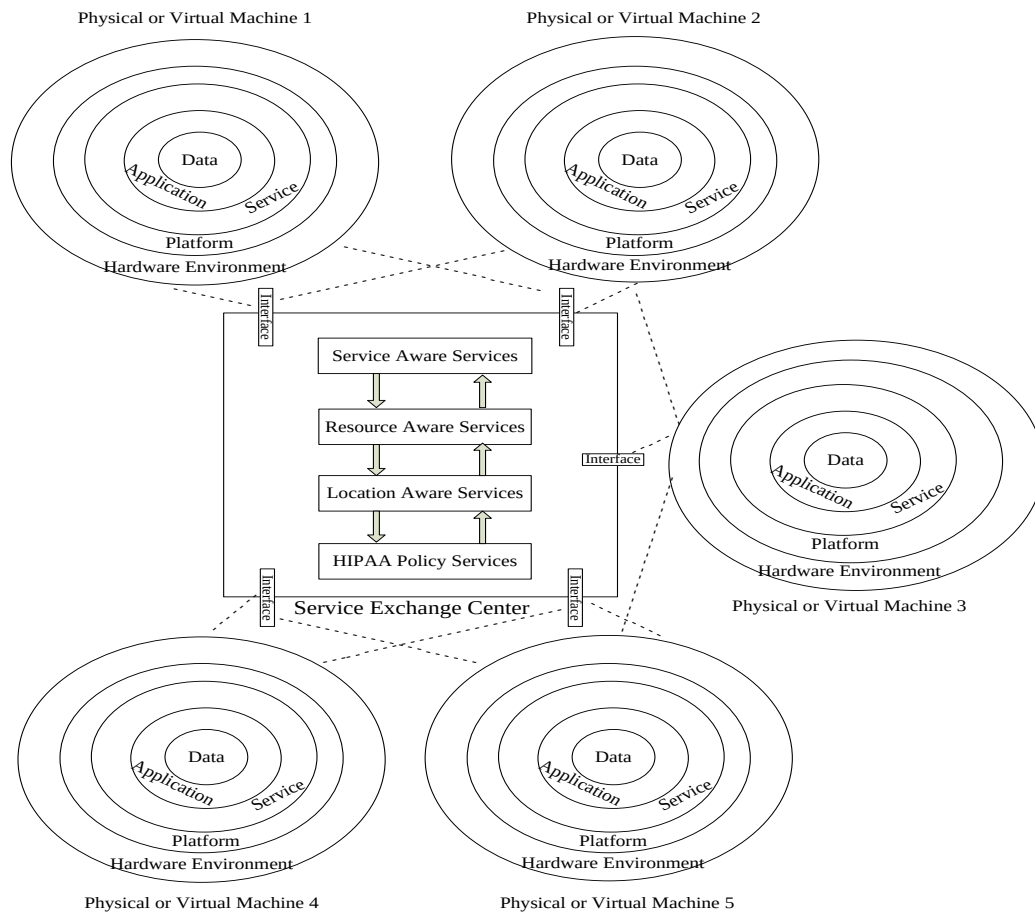
SOAP uses different transport protocols, such as HTTP and SMTP. SOAP is very much tied to HTTP POST messages. Restful services provide a good caching infrastructure over HTTP GET method (for most servers). This can improve the performance, if the data of the Web service returns is not altered frequently and not dynamic in nature. REST does not enforce message format as XML or JSON or etc. But SOAP is XML based message protocol. Because of the XML format, SOAP can be considerably slower than REST architecture. SOAP is strongly typed, has strict specification for every part of implementation. But REST gives the concept and less restrictive about the implementation. On the operational side, it is known how to scale a stateless RESTful Web service to serve a very large number of clients, thanks to the support for caching, clustering and load balancing built into REST. In case of SOAP, the interoperability problems can occur for instance, native data types and language constructs of the service implementation are present in its interface. This weakness can be mitigated by stating and enforcing certain design and coding guidelines such as contract-first development. The Table 2 shows the portability entities in the software, applications, service, system, cloud and mobile devices.

**Table 2: Software Service Portability Entities**

| Software       | Application      | Service          | System       | Cloud                   | Mobile Device   |
|----------------|------------------|------------------|--------------|-------------------------|-----------------|
| Requirements   | Server           | Service Provider | OS           | Cloud Service Provider  | Hardware        |
| Coupled with   | Interoperability | Scalability      | Timeliness   | Security, Composability | Performance     |
| Limitations    | Number of users  | Down-time        | Installation | Agreement               | Memory          |
| Non-compliance | Co-existence     | Installability   | Adaptability | Replaceability          | Transferability |

### IV. Portability Compliance Management Model through SARALA Exchange Server

The portability compliance exchange server model for porting of data, application, platform and service over physical or virtual machines has been proposed which is shown in figure.1. In case of data portability compliance (DPC) the application and service has to interact to ensure the data is portable across different virtual machines. While the network environment and service has to interact in order to ensure the application portability compliance (APC). The service portability compliance (SPC) can be ported when network environment fits for the service porting. The platform portability compliance (PPC) can be ensured when the infrastructure and operating systems interact across virtual machines. The Virtual Portability Compliance (VPC) may be ensured by virtual machines, virtual servers and hypervisor.

**Figure. 2 Software Cloud Portability Compliance Exchange Server Model using RESTful Services**

The platform that permits the all forms of portability with its attributes to other platforms through portability services is represented as “1” and those who do not are marked as “0” as shown in the OPC matrix below.

**Operational Platform Change Matrix**

|      | OPC1 | OPC2 | OPC3 | OPC4 |
|------|------|------|------|------|
| OPC1 | 1    | 0    | 1    | 0    |
| OPC2 | 0    | 1    | 0    | 1    |
| OPC3 | 0    | 0    | 1    | 0    |
| OPC4 | 0    | 0    | 0    | 1    |

The service that draws a contract with other services can not only port all the applications within that service but also the data to other services using multiple service contracts. Those services which entered into the SLC are marked as ‘1’ and those who do not are marked as ‘0’ as shown as a MSC matrix below:

**Multiple Service Contract Matrix**

|      | MSC1 | MSC2 | MSC3 | MSC4 |
|------|------|------|------|------|
| MSC1 | 0    | 0    | 1    | 1    |
| MSC2 | 1    | 0    | 1    | 1    |
| MSC3 | 0    | 1    | 0    | 1    |
| MSC4 | 1    | 1    | 0    | 0    |

If one service audit, constraints an another audit on other service, it means that the first service cannot port all the applications within that service but also the data to other services using multiple services contracts. Those audits which entered into the SAC are marked as ‘1’ and those who do not are marked as ‘0’ as shown in SAC matrix below.

| Service Audit Constraint Matrix |      |      |      |      |
|---------------------------------|------|------|------|------|
|                                 | SAC1 | SAC2 | SAC3 | SAC4 |
| SAC1                            | 0    | 1    | 0    | 0    |
| SAC2                            | 0    | 0    | 0    | 0    |
| SAC3                            | 1    | 0    | 0    | 0    |
| SAC4                            | 0    | 0    | 1    | 0    |

$$\text{Portability Failure Rate} = \text{pr}(t) = 1 - P [\text{Max.Failures}\{\text{Adaptability} + \text{Installability} + \text{Co-existence} + \text{Replaceability}\}] \rightarrow (1)$$

The portability relies on the maximum usability of the concerned data or application or service in the given environment with all forms of suitability towards attributes and sub attributes. Portability failures depend on the usability failures which are due to non-installable, non-compliant, non-existence and non-adaptable components which are in the form of data or services. Therefore the maximum usability of the data is declared as the  $i^{\text{th}}$  data in  $j^{\text{th}}$  application of  $k^{\text{th}}$  services which is represented in the form as shown below:

$$\begin{aligned} & \text{Max}[Usability(Data_i@Application_j@Service_k)] \quad (\text{or}) \\ & \text{Max}[Usability(Data@Device@Network)] \\ & = \text{Max Usability}(di@aj@sk) | \text{Physical Layer} + \text{Max Usability}(di@aj@sk) | \text{Virtual} \end{aligned} \rightarrow (2)$$

$$\begin{aligned} & = \sum \sum \sum \sum Usability(\text{Virtual}(d+a+p+s)) \\ & \text{Max Usability} = [D_o@A_o@S_o + D_o@A_1@S_1 + D_o@A_n@S_m] \end{aligned} \rightarrow (3)$$

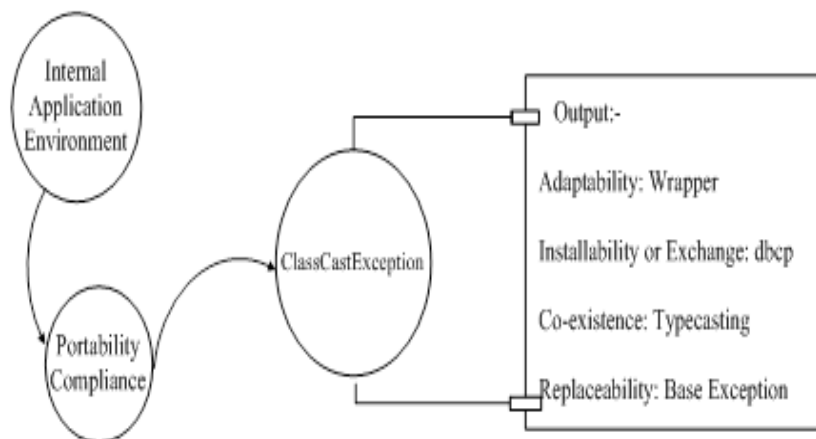
The timestool is considered in order to simulate the each task and its processing time taken to complete. All the tasks are assigned as scheduled tasks with its priority among each task, computing time, deadline and time taken for the complete process.

**Table 3: Tasks and Assigned Priorities**

| Names                  | B | Pr | C | D  | T  |
|------------------------|---|----|---|----|----|
| Running_Platform       | S | 1  | 1 | 30 | 30 |
| Requesting_App         | S | 2  | 1 | 30 | 30 |
| Exchange_Server        | S | 3  | 1 | 30 | 30 |
| RESTFUL_Web Service    | S | 4  | 1 | 30 | 30 |
| Service_Aware Service  | S | 5  | 1 | 30 | 30 |
| Resource_Aware Service | S | 6  | 1 | 30 | 30 |
| Location_Aware Service | S | 7  | 1 | 30 | 30 |

The figure 3 shows the class exception that occurred during the application deployment with different web servers. The portability non compliances arise due to the class cast exception of wrapper class, type casting and base exception.

**Figure 3: Application Deployment in Different Web Servers and its Exception**



$$\text{Portability} = \text{Min}(\text{Detection Time } t_d + \text{Identification Time } t_i + \text{Exchange (Conversion) Time } t_e) \rightarrow (4)$$

## V. Architectural View Mapping and Portability Compliance Metric

A software portability fault is due to architectural and aspect. The below shown table emphasis the SARALA services i.e., service aware, resource aware and location aware over RESTful web services.

The table 4 list outs the portability faults that arise in the web applications due to the failure of portability requirements. The different web services development technologies and its comparison has done with portability mappings of adaptability, co-existence, installability and replaceability.

**Table 4: Conceptual Portability Non-compliances and Mapping**

| Portability Fault due to       | REST | Other WS | Portability Mapping |
|--------------------------------|------|----------|---------------------|
| Resource Identification        | v    | ×        | Adaptability        |
| URI Design                     | v    | ×        | Adaptability        |
| Resource Interaction Semantics | V    | ×        | Co-existence        |
| Resource Relationships         | v    | ×        | Co-existence        |
| Payload Format                 | v    | ×        | Installability      |
| XML (POX)                      | v    | ×        |                     |
| XML (RSS)                      | v    | ×        |                     |
| JSON                           | v    | ×        |                     |
| YAML                           | v    | ×        |                     |
| MIME                           | v    | ×        |                     |
| Service Description            | v    | ×        | Replaceability      |
| Textual Documentation<br>WADL  | v    | ×        |                     |

Portability metrics quantify the effort of porting or rewriting the software-oriented services which is defined as an architectural view by the following equation:

$$\text{Prob (AA)} = 1 - \frac{\text{No of direct architectural aspects}}{\text{Total No of direct architectural aspects} + \text{No of aspects linked}}$$

$$\text{Critical State Transfer (CST) Factor} = \text{Number of Critical States} / \text{Total Number of States} \rightarrow (5)$$

$$\begin{aligned} &\text{Portability Compliance (PC)} \\ &= \text{Probability of Successful Critical State Transfer} / \text{Probability (Total number of States Transfer)} \end{aligned} \rightarrow (6)$$

$$\begin{aligned} &\text{Portability Compliance (PC)} \\ &= \text{Probability (Critical State Transfer)} * \text{Probability (Aspects Linked)} \\ &\quad \text{(OR)} \\ &1 - \text{Failure Probability (Critical State Transfer * Aspects Linked)} \end{aligned} \rightarrow (7)$$

Critical State Transfer (CST) is a function of the direct architectural aspects and the total number of links. So,  $\text{CST} = \alpha * \text{Number of Architectural Aspects}_{aa} * L_a$

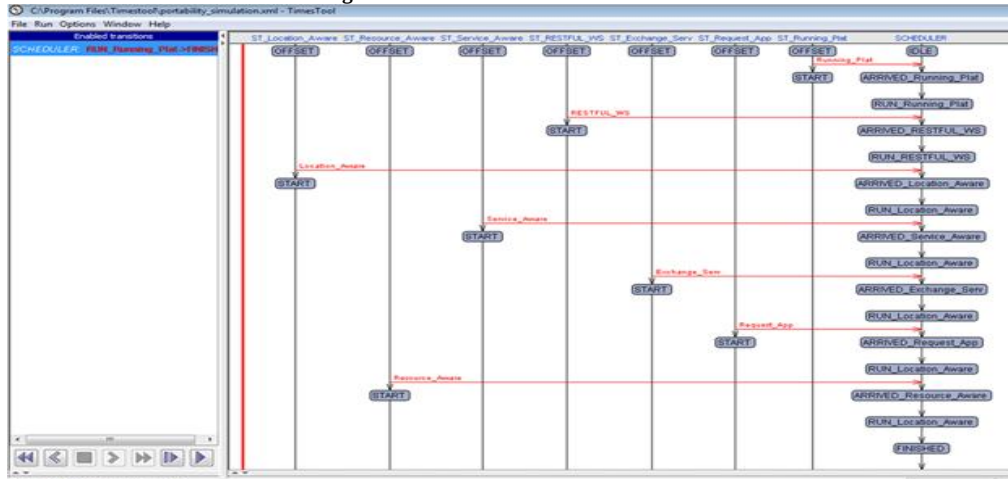
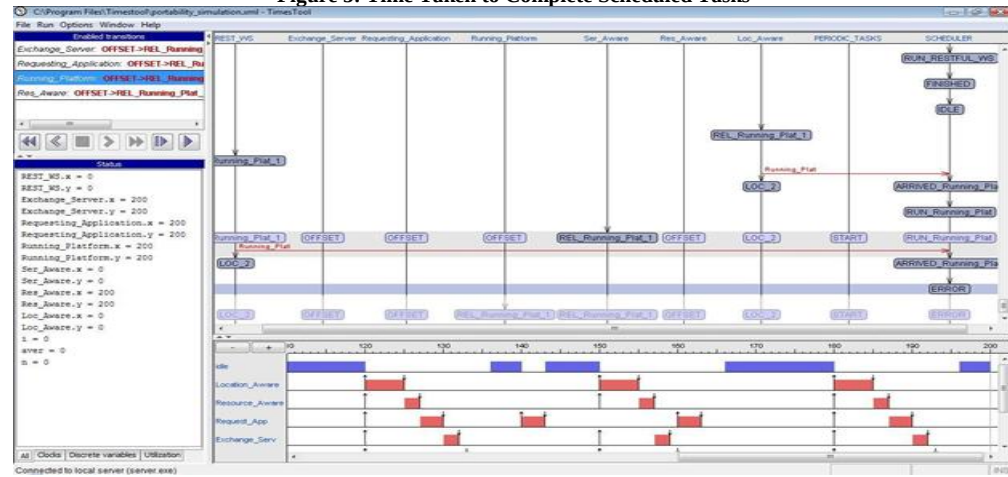
$$\text{Portability}_{\text{Non-Compliance}} = 1 - \text{Portability}_{\text{Compliance}} \rightarrow (8)$$

$$\text{Portability}_{\text{Compliance}} = \text{Probability (Architectural Aspects | State Transfer)} \rightarrow (9)$$

Portability compliance metric can also be measured in terms of the number of state transfer with respect to total number of system service states transferred.

## VI. Simulation Results and Discussions

The simulation results while executing the scheduled tasks as per the given timeline and automation processes derived. The interacting executable tasks in the checking of portability compliance are as follows: running platform, requesting\_application, exchange\_server, RESTFUL\_webservice, service\_aware, resource\_aware and location\_aware and the processing schedule is shown in the figure 4. The figure 5 shows the state of the each task and time taken for the completion of each task.

**Figure 4: Scheduled Tasks and its Process****Figure 5: Time Taken to Complete Scheduled Tasks**

## VII. Conclusion

The software portability fault due to architectural aspect is detected using service aware, resource aware and location aware over RESTful web services. The service portability compliance (SPC) is verified when network environment fits for the service porting with the help of different matrices. The platform portability compliance (PPC) is ensured when the infrastructure and operating systems interact across virtual machines. The performance of each and every compliance verification service is validated through the proposed cloud exchange server model.

## References

- [1] Alian Abran, Khalid T. Al-Sarayreh, Juan J. Cuadrado-Gallego, "A standards-based reference framework for system portability requirements", Journal of Computer Standards and Interfaces, Vol.35, pp 380-395, 2013.
- [2] Travis D. Breaux, Annie I. Anton, Eugene H. Spafford, "A distributed requirements management framework for legal compliance and accountability", Journal of Computers and Security, Vol. 28, pp 8-17, 2009.
- [3] Jiankun Hu, Hsiao-Hwa Chen, Ting-Wei Hou, "A hybrid public key infrastructure solution (HPKI) for HIPAA privacy/security regulations", Journal of Computer Standards and Interfaces, Vol. 32, pp 274-280, 2010.
- [4] Stefan Weiss, "Privacy threat model for data portability in social network applications", International Journal of Information Management, Vol. 29, pp 249-254, 2009.
- [5] D.V.Silakov, V. Khoroshilov, "Ensuring Portability of Software", Journal of Programming and Computer Software, Springer Pub, Vol. 37, No.1, pp 41-47, 2011.
- [6] Albena R. Iossifova, Susan Meyer-Goldstein, "Impact of standards adoption on healthcare transaction performance: The case of HIPAA", International Journal of Production Economics, Vol. 141, pp 277-285, 2013.
- [7] Alan C Gillies, "Software Quality Theory and Management", Second Edition, Cengage Learning India Private Limited, 2012.
- [8] Daniel Galin, "Software Quality Assurance from Theory to Implementation", Pearson Education Ltd, 2013.