Dynamic Performance-based Decision Support for Service Reusability

Tehreem Masood

N° d'ordre NNT : 2018LYSE2095

THESE de DOCTORAT DE L'UNIVERSITÉ DE LYON

Opérée au sein de

L'UNIVERSITÉ LUMIÈRE LYON 2

École Doctorale : ED 512 Informatique et Mathématiques

Discipline : Informatique

Soutenue publiquement le 26 octobre 2018, par :

Tehrem MASOOD

Aide à la décision dynamique basée sur les performances pour la réutilisation des services.

Devant le jury composé de :

Bernard ARCHIMEDE, Professeur des universités, École Nationale d'Ingénieurs de Tarbes, Président

Yves DUCQ, Professeur des universités, Université de Bordeaux, Rapporteur

Lilia GZARA, Maître de conférences HDR, Université Grenoble Alpes, Rapporteure

Andras GABOR, Professeur d'université, Cornivus University of Budapest, Examinateur

Nejib MOALLA, Maître de conférences HDR, Université Lumière Lyon 2, Co-Directrice de thèse

Chantal BONNER-CHERIFI, Maître de conférences, Université Lumière Lyon 2, Co-Directrice de thèse

Contrat de diffusion

N°d'ordre NNT : xxx

THESE de DOCTORAT DE L'UNIVERSITÉ DE LYON

Opérée au sein de

L'UNIVERSITÉ LUMIÈRE LYON 2

Ecole doctorale : ED 512

Informatique et Mathématiques

Discipline : Informatique

Soutenue publiquement le 26 Octobre 2018, par :

Tehreem MASOOD

Dynamic Performance-based Decision Support for Service Reusability

devant un Jury composé de :

Yves DUCQ, Professeur des universités, Université de Bordeaux, Rapporteur

Lilia GZARA, Maître de conférences-HDR, Université Grenoble Alpes, Rapporteur

Bernard ARCHIMEDE, Professeur des Universités, École nationale d'ingénieurs de Tarbes, Examinateur

András GABOR, Professeur des Universités, Corvinus University of Budapest, Examinateur

Néjib MOALLA, Maître de conférences-HDR, Université Lumière Lyon 2, Directeur de thèse

Chantal BONNER CHERIFI, Maître de conférences, Université Lumière Lyon 2, Co-encadrante de thèse

UNIVERSITÉ LUMIÈRE LYON 2

EÉCOLE DOCTORALE INFORMATIQUE ET MATHÉMATIQUES

THESE

Pour obtenir le grade de

DOCTEUR EN INFORMATIQUE

Présentée par

Tehreem MASOOD

Dynamic Performance-based Decision Support for Service Reusability

Préparée au sein du laboratoire

Laboratoire Décision et Information Pour les Systèmes de Production

Directeur de thèse : **Dr. Néjib Moalla**

Co-encadrante de thèse : **Dr. Chantal Bonner Cherifi**

Soutenue publiquement le 26 Octobre 2018, devant un jury composé de :

Yves DUCQ	Rapporteur	(PR, Université de Bordeaux)
Lilia GZARA	Rapporteur	(MCF– HDR, Université Grenoble Alpes)
Bernard ARCHIMEDE	Examinateur	(PR, École nationale d'ingénieurs de Tarbes)
András GABOR	Examinateur	(PR, Corvinus University of Budapest)
Néjib MOALLA	Directeur de thèse	(MCF-HDR, Université Lumière Lyon 2)
Chantal BONNER CHERIFI	Co-encadrante de thèse	(MCF, Université Lumière Lyon 2)

Résumé

Mots clés: Web service; Service reuse; SOA; SBS; Performance; Ontology; Risk; Maturity; Dynamicity; Decision support.

Abstract

Reuse of services in supporting new business processes, in addition to alignment of IT with business functions, is a key motivation in using Service-Oriented Architecture (SOA) for developing business solutions. In a service-oriented architecture, it is important to smooth the selection, configuration and composition of existing services to deal with the runtime changes or the evolution of End User requirements. In contrast to other traditional software systems, the dynamic behavior of service-based systems requires up-to-date quality of service (QoS) information for its proper management in the different stages of the lifecycle. Organizations need to know the performance of Web services and business processes to maintain their sustainability for reuse of services. The three key benefits of service reuse are improving agility of solutions by quickly assembling new business processes from existing services to meet changing marketplace needs, reducing cost by not developing new services for enabling similar business functions across multiple business processes, but also spanning service deployment and management in runtime environments throughout the SOA lifecycle. However currently, there are many challenges related to the sustainability and governance of service behavior during its lifecycle. Among those challenges, one can mention level of performance, persistence of the requirements and adaptability of the service. Moreover, there are some limitations of monitoring tools. They lack of anticipation in problem detection, and they are passive and neither reactive nor predictive. This thesis focuses on providing assessment and recommendations for performance and governance of information systems for suggesting service reuse during its evolution. The aim is to maintain sustainability, robustness, adaptability, reusability and evolvability of information systems. For this purpose, we evaluate the performance of service-oriented architecture. There are several existing monitoring solutions designed to support a specific layer of SOA. Particularly, BAM is a business activity monitoring tool for monitoring the flow of data for business processes. However, BAM monitoring do not provide the performance evaluation for recommending services and processes to reuse. There are very few approaches that support monitoring of SOA layers together. Furthermore, the solutions are partially dynamic with limited decision support. Therefore, we propose performance-based decision support for service-oriented architecture. It consists of four layers as specification, data management, data mining and decision layers. The specification layer identifies the requirements from the End User and process through the proposed ontology. The data layer analyzes technical indicators that are compliant to the latest quality standard, ISO 25010. Quality characteristics are related to performance efficiency, reliability and reusability. The data mining layer generates specific decisions based on service instances by applying the machine learning algorithms. It uses the proposed ontological concepts and semantic inference rules of service, business process, server and integration

layers. The data mining layer returns to ontologies with these specific decisions where more refined rules have been generated from new ontological concepts. The decision layer processes these results and generates a global decision in terms of recommendations. It provides multi-viewpoints decision to reuse existing services or suggesting their composition.

To motivate the proposition of this approach, we illustrate the implementation of the proposed algorithms for all the four layers by a business process use case and data set of public repositories of shared services. Validation has been made based on the evaluation of cost, confidence, precision and support. As a result, we recommend reuse of atomic service, composite service and resource allocation provisioning. In this way, we ensure the sustainability, adaptability, reusability and evolvability of service-based systems by handling new business requirements, performance efficiency, reliability in terms of availability, maturity and risk, resource management and dynamicity issues.

Keywords: Web service; Service reuse; SOA; SBS; Performance; Ontology; Risk; Maturity; Dynamicity; Decision support.

Table of Contents

CHAPTER 1. INTRODUCTION 14

1.1. Research Context 14

1.2. Research Objective 20

1.3. Scientific Problems 21

1.4. Research Problem 21

1.5. Justification 22

1.6. Contributions 22

1.7. Organization of Chapters 23

CHAPTER 2. SERVICE REUSE AND PERFORMANCE BASED SOLUTIONS 27

2.1. Analytic Methodology 27

2.1.1 Systematic Analysis 28
2.1.2 Research Classification 28
2.2. Service Reuse Solutions 29

2.2.1 Reusability in SDLC 30
2.2.2 Reusability in Service Life Cycle 30
2.2.3 SOA Governance 31
2.2.4 Summary 32
2.3. Service Performance 33

2.3.1 Quantitative 33
2.3.2 Qualitative 37
2.4. Performance based Service Structuring 44

2.4.1 Ontologies 44
2.4.2 Summary 46
2.5. Performance based Service Reasoning 46

2.5.1 Decision Support Systems 47
2.5.2 Summary 48
2.6. Discussion 48

CHAPTER 3. PERFORMANCE ORIENTED DECISION SUPPORT 50

3.1. PODSF 51

3.1.1 PODSF Environment 53
3.1.2 PODSF Process 55
3.1.3 PODSF Components 58
3.2. Discussion 75

CHAPTER 4. IMPLEMENTATION OF PODSF 76

4.1. PODS Research Design 77

4.1.1 PODS Ontology Structuring 77
4.1.2 PODS Reasoning 84
4.2. PODS Research Prototype 85

4.2.1 Data Management 86
4.2.2 Decision Support 91
4.3. PODS System 94

4.3.1 PODSS High Level Architecture 95
4.3.2 Classes of PODSS 96
4.4. Discussion 99

CHAPTER 5. EVALUATION OF PODSS 101

5.1 Service Reuse Use Case 102

5.2. Use Case Implementation 104

5.2.1 Data Mining Analytics 105
5.2.2 Decision scenarios for Service Reuse 112
5.2.3 Recommendations 118
5.3 Discussion 119

CHAPTER 6. CONCLUSION AND FUTURE WORKS 121

6.1 Conclusion 121

6.2 Future Works 123

REFERENCES 125

List of Figures

Figure 1: SOA Reference Architecture [4] 14
Figure 2: ISO/IEC 25010 quality model for software products [8] 16
Figure 3: Extended SOA Functionality with Oracle Fusion Middleware [17] 18
Figure 4: TOGAF [22] 19
Figure 5: Research Classification 29
Figure 6: Dimensions of Welke's SOAMM [80] 39
Figure 7: Mapping of CMMI Models and SOAMMs [81] 40
Figure 8: Service Lifecycle [82] 40
Figure 9: Methodical Building Blocks [82] 41
Figure 10: Web service Ontology [111] 46
Figure 11: Position of PODSF in SOA based organization 51
Figure 12: PODSF 52
Figure 13: PODSF Process 55
Figure 14: Data flow of PODSF 58
Figure 15: PODSF Components and their interaction 58
Figure 16: Data Management in PODSF 59
Figure 17: GetSupplier Service Information in WSO2 server 64
Figure 18: GetSupplier Service Operation in WSO2 server 65
Figure 19: Server information in WSO2 server 65
Figure 20: GetSupplier Service Summary using WSO2 server 66
Figure 21: Methodology to Create Performance Profile 68
Figure 22: SOnt Ontology based on Technical Indicators of WSO2 Server 69
Figure 23: Inference Rules using SWRL in Protégé 71
Figure 24: SOA Layers Ontological Concepts 78
Figure 25: Ontology of Performance at Process Layer 79
Figure 26: Ontology of Performance at Integration Layer. 80
Figure 27: Ontology of Governance Layer. 81
Figure 28: Performance Oriented Decision Support Ontology (PODSOnt) 82
Figure 29: Performance Profile 83
Figure 30: Performance Management Algorithm 87
Figure 31: Maturity Evaluation Algorithm 88
Figure 32: Risk Management Algorithm 91
Figure 33: Performance Evaluation for Specific Decision based Algorithm 92
Figure 34: Impact Analysis Algorithm 93
Figure 35: Decision Evaluation Algorithm 94
Figure 36: Package Diagram of PODSS 96
Figure 37: High level Schema of PODSS Evaluation 101
Figure 38: CollaborationAnalyzer Schema 103
Figure 39: SDScan Application 104
Figure 40: Use Case Implementation 105
Figure 41: Performance Criteria 106
Figure 42: Basic Rules for the Assignment of Recommendation Weights 107
Figure 43: Analytics of Classification Algorithms 108
Figure 44: Parametric Result of Classification Algorithms 108
Figure 45: Results from 3DScan Application 109
Figure 46: Decision Tree Model 110

FIGURE 47: RESULTS FROM DECISION TREE MODEL 111
FIGURE 48: COST MATRIX FOR DECISION TREE MODEL 111
FIGURE 49: RECOMMENDATION OF SERVICE BASED ON RESPONSE_TIME 112
FIGURE 50: RECOMMENDATION OF SERVICE BASED ON SERVICE_INSTANCE_TIME 113
FIGURE 51: RECOMMENDATION OF SERVICE BASED ON THROUGHPUT 113
FIGURE 52: DECISION RESULTS BASED ON SERVICE INSTANCES 114
FIGURE 53: MATURITY EVALUATION BASED ON INSTANCEID 115
FIGURE 54: RISK EVALUATION BASED ON INSTANCEID 115
FIGURE 55: SERVICE TIME EVALUATION BASED ON INSTANCEID 116
FIGURE 56: RECOMMENDATION PREDICTION AND COST ANALYSIS MATRIX 117
FIGURE 57: PARAMETRIC ANALYSIS MATRIX 117
FIGURE 58: PREDICTIVE CONFIDENCE 117

List of Tables

TABLE 1: COMPARISON OF THE EXISTING PERFORMANCE MANAGEMENT TECHNIQUES (1/2) 36
TABLE 2: COMPARISON OF THE EXISTING PERFORMANCE MANAGEMENT TECHNIQUES (2/2) 37
TABLE 3: A SAMPLE REQUIREMENT FROM AN END USER 53
TABLE 4: A SAMPLE TOPOLOGY OF RECOMMENDATION: THE CALL FOR BLOOD EXAMPLE 54
TABLE 5: QUANTITATIVE TECHNICAL INDICATORS 60
TABLE 6: OBJECT PROPERTIES OF SONT 69
TABLE 7: DESCRIPTION OF OBJECT PROPERTIES OF SONT 70
TABLE 8: DESCRIPTION OF DATATYPE PROPERTIES OF SONT 70
TABLE 9: PERFORMANCE MEASUREMENT 72
TABLE 10: SERVICES PROFILE FOR HOSPITAL MANAGEMENT 73
TABLE 11: A SAMPLE OF SPECIFIC DECISION 73
TABLE 12: FIRST LEVEL RESULTS FOR SERVICE INSTANCES 74
TABLE 13: SECOND LEVEL RESULTS FOR SERVICE RECOMMENDATION 75
TABLE 14: INFERENCE RULES (IR) (1/2) 84
TABLE 15: INFERENCE RULES (IR) (2/2) 85
TABLE 16: MATURITY EVALUATION BASED ON CMMI LEVELS 89
TABLE 17: RISK ASSESSMENT AND MITIGATION 90
TABLE 18: DECISION RESULT FOR EACH SERVICE 116

List of Abbreviations

ADM	Architecture Development Method
BPEL	Business Process Execution Language
BPM	Business Process Management
BPMN	Business Process Model and Notation
BPMS	Business Process Management System
BP	Business Process
DDS	Distributed Database System
EA	Enterprise Architecture
ESB	Enterprise Service Bus
FEAF	Federal Enterprise Architecture Framework
KPI	Key Performance Indicator
OWL-S	Ontology Web Language for Web Services
OEAF	Oracle Enterprise Architecture Framework
QoS	Quality of Web Service
SAWSDL	Semantic Annotations for WSDL
SBA	Service Based Application
SBS	Service Based System
SDLC	System Development Lifecycle
SLA	Service Level Agreement
SOC	Service Oriented Computing
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SWRL	Semantic Web Rule Language
TOGAF	The Open Group Architecture Framework
UDDI	Universal Description, Discovery, and Integration

UML	Unified Modeling Language
VMM	Virtual Machine Monitor
WS	Web service
WSDL	Web Service Description Language
WSMO	Web Service Modeling Ontology
XML	Extensible Markup Language

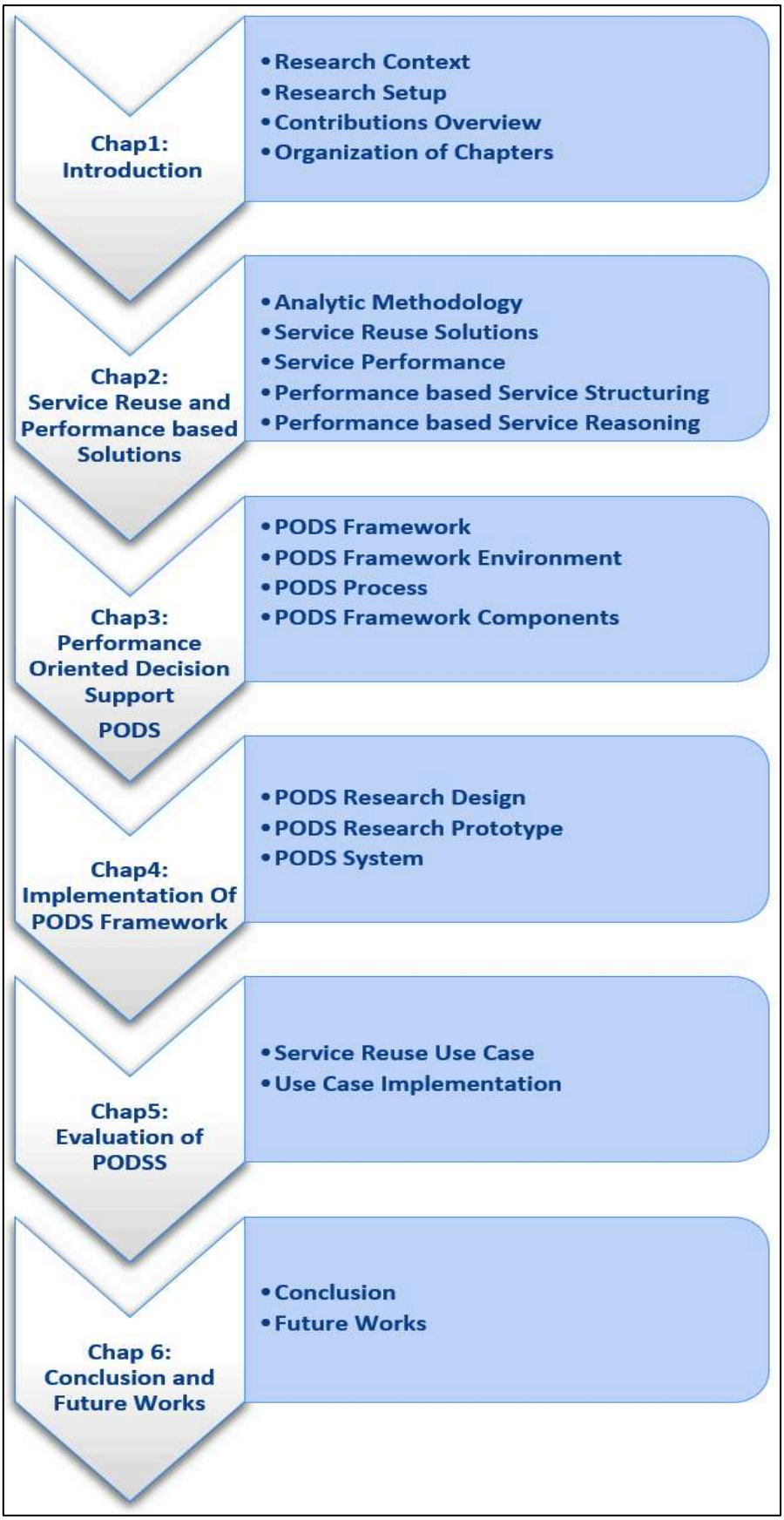


Figure A.1: Structure of Thesis

CHAPTER 1. Introduction

1.1. Research Context

Enterprise performance can be improved only by providing reactive and predictive monitoring tools which anticipate in problem detection. It requires advanced approaches for creating more agile, adaptable and sustainable information systems that are able to adapt themselves to new trends. Service oriented architecture (SOA) is one such approach that has received significant attention among information system practitioners. SOA is emerging as a powerful paradigm for organizations that need to integrate their applications within and across organizational boundaries [1]. It has to be noted that the advanced organizations are more likely adopting SOA because of its several advantages like loose coupling, modular, non-intrusive and standard's based. Web services provide a functionality following web standards such as Simple Object Access Protocol (SOAP) [2], Web Service Description Language (WSDL) [3]. Figure 1 introduces a set of logical layers of the SOA reference architecture [4]. The five horizontal layers are the main functional layers that describe the functionality of the SOA solution. The four vertical layers define non-functional support that is produced across the functional layers.

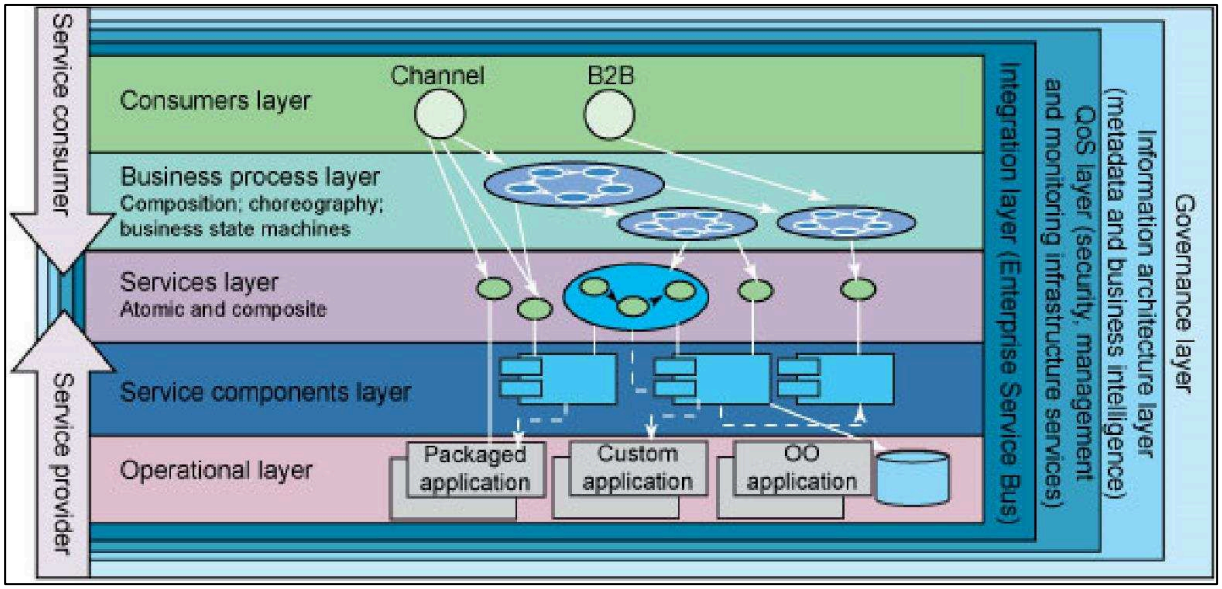


Figure 1: SOA Reference Architecture [4]

The five horizontal layers are operational layer, service components layer, services layer, business process layer and consumers layer. Operational layer contains existing software application systems. This includes customer applications, transaction processing system, legacy system, database and packaged applications and solutions. Service components layer provides software components that are the implementation of services or service operations. Service components reflect the definition of

services, both functional and non-functional properties. Services layer contains all the services inside the SOA. A service is defined by its operations. A service specification describes the invocation information about a service and description of the abstract functionality. A service specification may include also a policy document, SOA management description and a document about service dependencies. Business process layer defines compositions and choreographies of services exposed in the service layer. Services are combined or choreographed into flows that create composite services from atomic services. This layer defines the process representation, composition methods, and building blocks for aggregating loosely coupled services as a sequence of processes associated with business goals. Consumer's layer provides interfaces that allow the communication between applications. It can also provide the capabilities required to deliver IT functions and data to End User.

The four vertical layers are integration, quality of service, information architecture and governance layers. Integration layer transports service request from a service requester to the service provider. This layer allows the integration of services through point-to-point, protocol mediation and other transformation mechanisms. Quality of service layer deals with the non-functional requirements. It captures, monitors, stores and indicates non-compliance with the requirements provided in the service level agreement (SLA). Non-functional requirements are related to reliability, availability, manageability, scalability and security. Information architecture layer captures cross industry and industry specific data structures, Extensible Markup Language (XML) based metadata architectures and business protocols for exchanging business data. Governance layer covers all aspects of business operations life cycle management in the SOA, capacity, performance, security and monitoring. Guidance and policies for making decisions about SOA solution are provided in this layer.

A system following SOA is known as service-based system (SBS). This system is composed of several services. Services are self-contained functional entities with well-defined interfaces that contain their functional and non-functional specifications. Interfaces are of two types: provided to and required from other services. Functional specifications are related to the service operations while non-functional specifications are related to Quality of Service (QoS). One specific type of service is Web service. Web services can be combined as building blocks for the composition of larger SBS's [5]. The advances in modern technology and the constantly evolving requirements implied by dynamic business environments imposes new challenges for engineering and provisioning SBS. SBS should be able to operate and evolve in highly dynamic environments to identify and react to various changes or new requirements. Moreover, they require up-to-date QoS information for their proper management at the different lifecycle stages starting from the construction until decommission [6]. SBS rely on SLA provided by service providers to ensure that the services comply with the agreed QoS [7].

QoS is usually structured in the form of a quality model. Quality models are useful for specifying requirements, establishing measures and performing quality evaluations. There exist many proposals of general-purpose quality models for software systems. They differ on the terminology that they use, the set of quality attributes they define, and the structure of the quality model. ISO/IEC series of quality standards, 25010 is the recent quality model as shown in Figure 2 [8]. This model includes a concrete quality model that classifies software quality into a structured set of high-level characteristics and sub characteristics. The major characteristics of this model are functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability and portability. Among these quality characteristics, performance efficiency, reliability and maintainability related to reusability play an important role to ensure performance of service life cycle. ISO/IEC 25010 might not accommodate entirely into the Web service domain. The high-level quality characteristics of ISO/IEC do not provide quantitative or qualitative measurements. Therefore, it is important to divide them into concepts such as response time, latency or execution time. When these concepts are clearly defined in measurable terms, they are usually known as quality metrics. As defined by Burnstein, "a quality metric is a quantitative measurement of the degree to which an item possesses a given quality attribute" [9]. An example of quality metric is the average response time during a time interval.

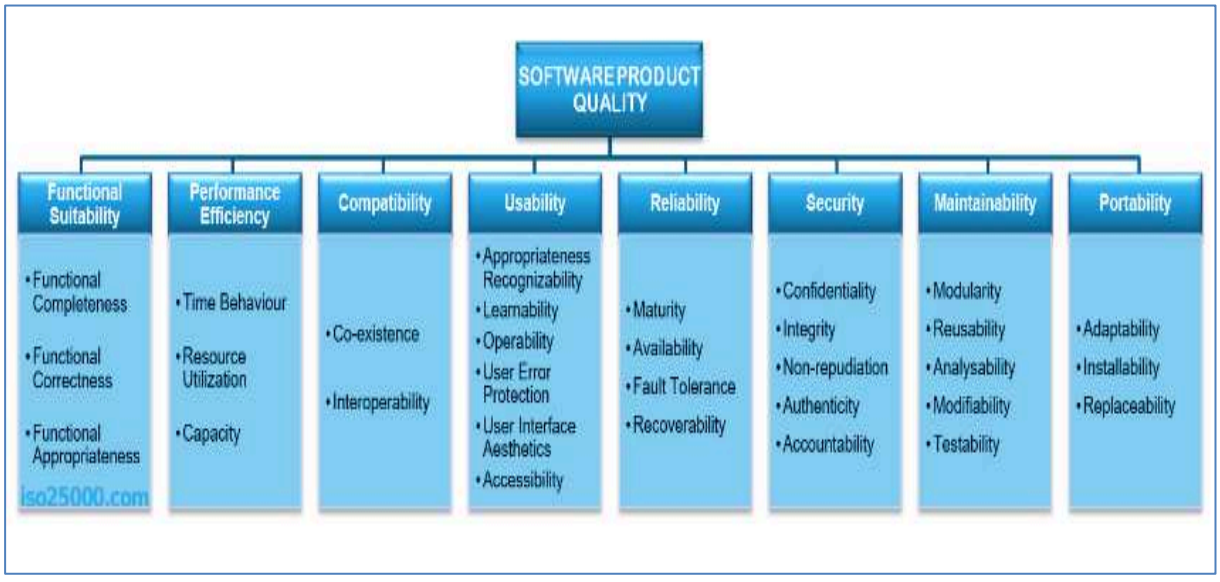


Figure 2: ISO/IEC 25010 quality model for software products [8]

ISO/IEC 25010 quality characteristics that play very important role for achieving performance of SBS are performance efficiency, reliability and maintainability. Under these quality characteristics, the important quality sub characteristics are time behavior, resource utilization, capacity, availability, fault tolerance, recoverability, maturity and reusability. Time behavior is further analyzed based on technical indicators such as response time, service or process start and end times. Resource utilization is based on CPU frequency, RAM size, storage device and maximum CPU load. Capacity is measured by

throughput and bandwidth of service instances. Availability is measured by checking that a service or system is operational and available when needed. This can be analyzed based on quality technical indicators like service up time, service down time, request count and response count. Fault Tolerance is analyzed by checking that a service or a system operates as planned despite the incidence of faults. This can be ensured by analyzing the associated risks and providing some mitigation actions against risks. Different notions related to risks exists in the literature such as threats, vulnerabilities, threat probabilities and their impacts on the organization. Threats related to the technical problems are service failure, network failure, loss of service and specification changes. After mitigating the associated risks, system re-establish its stable state and recover the data. This is called recoverability. Maturity is another important quality sub characteristic that affects performance of an enterprise. It is used to evaluate that systems or services are reliable under normal operation. However, it is not widely used in the existing research as quality attribute to evaluate performance. Different maturity models have been proposed to date. Capability maturity model integration (CMMI) [10] is most commonly used because of its efficient framework for assessing and providing guidelines. Existing maturity models lack prescriptive properties to determine the level of maturity from one level to another for SBS's.

Reusability is a very important sub characteristic for SBS to analyze reuse of service or process. Reuse of services helps in achieving business agility to meet changing marketplace needs. It can be achieved by quickly assembling new business processes from existing services and even creating new business processes from existing services [11]. There is very less concrete data on service reuse in current SOA engagements. Rather, a lot of recent writings have pointed out the challenges in achieving service reuse [12, 13].

ISO/IEC 25010 lists quality characteristics but we need to store them with respect to SBS. QoS attributes are stored in ontologies in general. Ontologies are used because of their several advantages. The first advantage is the modelling and structuring of performance knowledge related to SBS. Ontologies offer a formal expressive description of concepts and their existing relationships in a coherent way. So, it is important that the stored QoS information is structured, managed and reused in a reliable and standardized manner. The second most important advantage is to infer new knowledge by reasoning on ontologies. Reasoners are used to check the uniformity of ontologies that whether some classes are inadequate, and to manage the order of classes and relations. The third very important advantage is the dynamic nature of ontologies. They have the ability to evolve over time by accumulating new classes, concepts and instances. Ontologies also support the decision by generating decision rules and interacting through queries. Rules are developed using Semantic Web Rule Language (SWRL) [14], and queries are published by using SPARQL [15]. Several ontologies have been proposed to date in order to store Web service properties, both functional and non-functional.

However, to our knowledge, no complete QoS ontology exists. Existing QoS ontologies are neither formulated to infer new knowledge nor evolve with the evolution of service life cycle. Yet, service life cycle evolves at run time with changed or new business requirements.

Monitoring all component services and processes constantly and inspecting the entire SBS during runtime is difficult due to excessive resource and time consumption required, especially in large-scale scenarios [16]. SOA has been extended with Oracle fusion middleware to provide more technological solutions, and to that end monitor component plays an important role to monitor performance of SBS, as shown in Figure 3 [17]. Gateway is a set of modules that encompasses Web service interfaces to allow routing, transformation and security. Orchestrate deals with composite applications and orchestration of business process. Interact and access provides a common interface for multiple dissimilar applications. Monitor and optimize provides access to critical business performance indicators in real time. The most commonly used toolsets for monitoring applications and service-oriented networks are Java Management Extensions (JMX) [18]. JMX is a Java technology that supplies tools for managing and monitoring applications, system objects, devices and service-oriented networks while Business Activity Monitoring (BAM) [19] is widely used to monitor business activity. BAM tools allow managers to monitor the status of their business processes and the global business, all from the same point. It is a toolset that allows the monitoring of Key Performance Indicators (KPI). The indicators that BAM tools monitor are mainly related to the payload such as real time tracking of number of transactions, number of process events, number of changes in records, and velocities. However, there is a need of maximizing QoS by providing more key performance indicators such as time behavior, resource utilization, capacity, availability, maturity, reusability and risk. Evolution of these indicators in the form of quantitative and qualitative with time dimension along SOA layers is very important to measure in order to ensure the sustainability of information system.

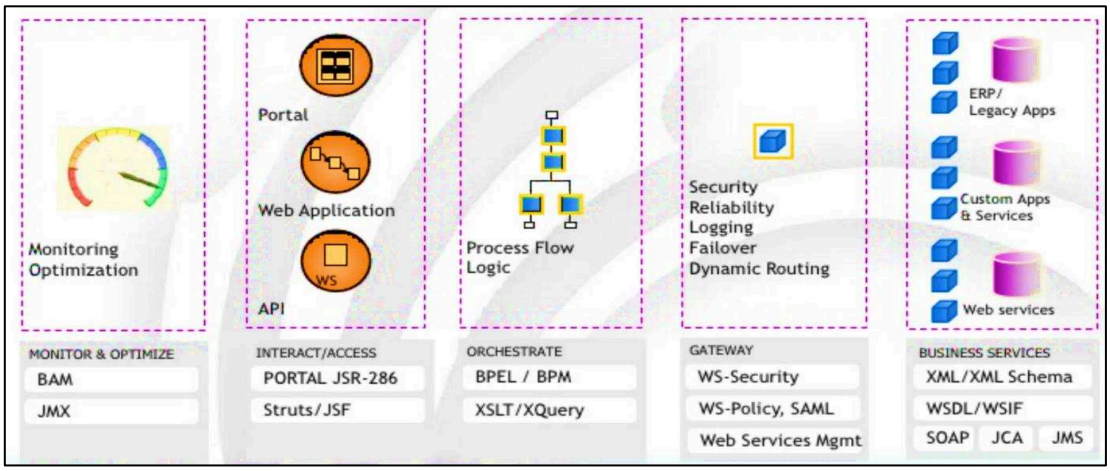


Figure 3: Extended SOA Functionality with Oracle Fusion Middleware [17]

As stated previously, QoS represents the non-functional properties of a Web service. Along with QoS, services' reputation, service provider's information and service's accessing information are also non-functional properties of a Web service. The functional properties of a Web service define its operations that are specified by input and output parameters [20]. QoS can be integrated in the Web service descriptions. Web services can be syntactically described by using Web service description languages such as WSDL. WSDL allows syntactic matching when searching for Web services [21].

In order to create information systems-based solutions for respective business needs, The Open Group Architecture Framework (TOGAF) provides a well-defined set of guidelines [22]. TOGAF is shown in Figure 4. TOGAF is a framework that provides a detailed method and a set of supporting tools for developing enterprise architecture within an organization. It helps to utilize resources more efficiently, effectively, to realize a greater return on investment. One of the most important phase of TOGAF is Information Systems Architectures. This architecture describes the development of Information Systems Architectures containing the development of Data and Application Architectures.

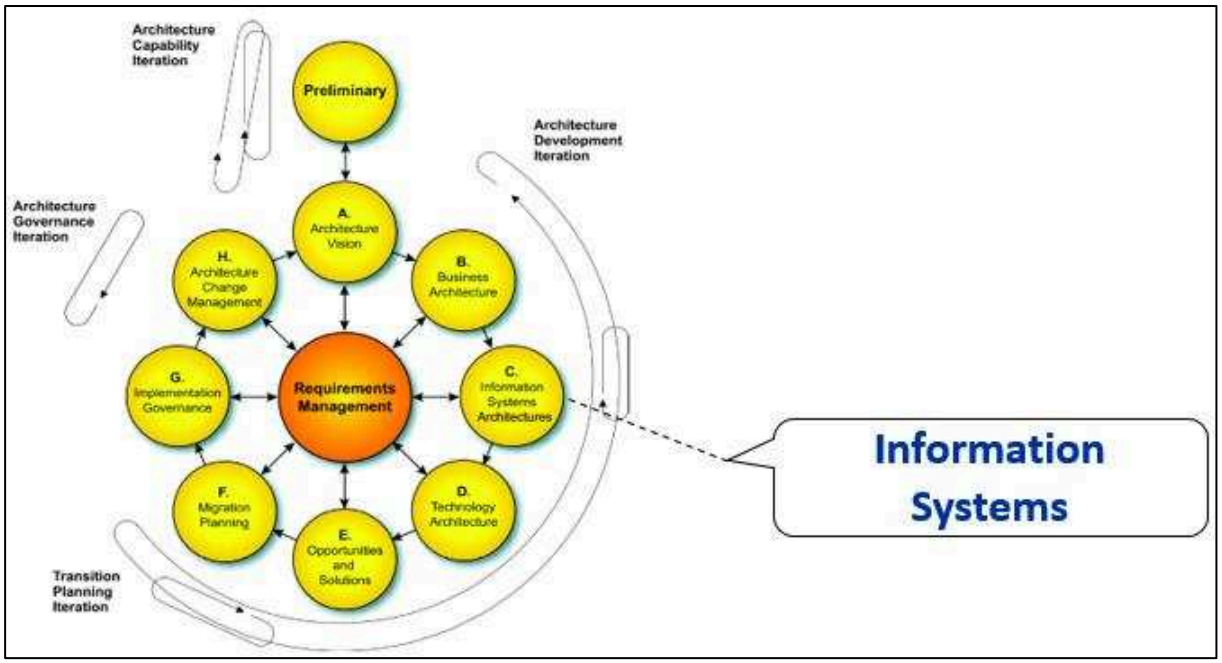


Figure 4: TOGAF [22]

Advances in computer technology are dynamic, and they impact information system applications including Decision support systems (DSS) [23]. Any application that involves decision making in any mode, is often named as a DSS. The consequence is a set of DSS applications that is dynamic and constantly evolving. This phenomena highlights that information systems will evolve accordingly into various directions. The dynamic nature of information systems makes it difficult for information practitioners and other managers to provide a stable set of DSS applications. However, the selection of DSS applications plays an important role in forecasting organizational policies for the deployment

of information technology. Most common DSS applications are artificial intelligence, machine learning, and business intelligence. Several DSS's have been proposed to date covering different domains. However, no DSS has been provided to cover the performance of SBS and evolving nature of service lifecycle.

In order to provide efficient DSS applications, data mining concepts play an important role. The key perspective of data mining is to extract useful information from the data [24]. Data mining involves six common classes of tasks such as anomaly detection, association rule learning, clustering, classification, regression and summarization. In terms of data analytics, machine learning is a method used to create complex models and algorithms that impart themselves to prediction. Machine learning is also combined with data mining and it is called as unsupervised learning. In machine learning and statistics, classification is the problem of identifying a set of a new observation on the basis of a training set of data containing observations whose category membership is known. Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree to go from observations about an item to conclusions about the item's target value.

Several research challenges ranging from sustainability of service behavior during its lifecycle to the limitations of monitoring tools, still remain open. In terms of sustainability of service behavior, challenges are related to the acceptable level of performance, persistence of the requirements and adaptability of the service. Monitoring tools are passive and neither reactive nor predictive in terms of performance and service reuse. Ontology is neither used to infer new knowledge nor used to evolve with the changed or new business requirements. Also, it is difficult to anticipate problem detection. The definition of a more comprehensive and holistic approach is crucial for delivering high performance, robust, reusable, and highly adaptable SBSs [25]. Moreover, no decision mechanism for handling new or changing business requirements, anticipating problem detection and suggesting reuse of service or process have been provided so far.

1.2. Research Objective

With reference to the above context, this research aims to create a support system for accelerating monitoring of Web services and decision making for their reuse. Developers usually prefer to develop new Web services for answering End User requirements. The End User requirements here refer to new requirements and modified requirements. In addition, developing new Web services costs money and

time. Therefore, reusing available Web services is a better solution for developers than to create new Web services.

Our objective is to provide assessment and recommendations for service reuse during the evolution of the service while preserving acceptable performance and conforming governance rules. For this purpose, we aim to provide dynamic decision support for SBSs while considering the performance of service, business process and integration layers of SOA. In this way, we will be able to ensure sustainability, evolvability, reusability and adaptability of SBS. The End User of the system are the business organizations that are motivated to SOA concepts.

1.3. Scientific Problems

In order to achieve the above stated objective, there are several challenges. Today's software systems are becoming increasingly integrated into the lives of their End Users and their ever-changing environments and needs. These demands lead to a growing complexity of systems. The development of adaptive systems is a promising way to manage this complexity. Adaptive systems are able to adapt their behavior at run time while considering the changing operational environment to maximize the satisfaction of End User needs.

It is difficult to compose dynamically performance based technical indicators at the different SOA layers to ensure information system sustainability. There is a lack of performance management in terms of its evolution for both quantitative and qualitative level indicators with time dimension along SOA layers. Along with that, there is no guidance for management of service or process reuse, handling changing or new business requirements and effective resource utilization. Moreover, estimation of the impact of new consumption of services and handling governance problems are still remained as challenges.

1.4. Research Problem

Provide efficient decision support for information system evolution in terms of service reuse with acceptable performance level while conforming to system governance rules. The main issue consists of how to manage dynamically performance, risk and maturity of SOA layers to ensure sustainability of information systems. The above-mentioned research problem leads to the following research questions to guarantee efficient performance based decision support for service reuse:

RQ1. Are technical indicators compliant with new quality standards?

RQ2. How to evaluate the impact of increased consumption of services on performance?

RQ3. How to incorporate changing or new End User requirements with the evolution of service life Cycle?

RQ4. How to handle resource demands of the system at different workloads?

RQ5. How to provide efficient performance-based decision support for service reuse?

1.5. Justification

As very little work has been done particularly for monitoring of performance of SOA layers with respect to the latest quality model, it is important to monitor and examine performance of SOA layers in terms of latest quality characteristics. Another important aspect is that existing works do not evaluate the impact of increased consumption on performance measurement over different time stamps. This leads to monitor and examine performance over different time stamps by adding or consuming more data. There is very less concrete data on service reuse in current SOA engagements. Rather, a lot of recent writings have just pointed out the challenges in achieving service reuse.

1.6. Contributions

The main contribution of this research is to provide performance management, maturity evaluation, risk mitigation actions and reusability of services. Reuse of services in supporting new business processes, in addition to alignment of information technology with business functions, is a key motivation in using SOA for developing business solutions. Based on the current research challenges for service reuse, we highlight the contributions for the research gaps highlighted in part 1.5. The major contribution is to provide assessment and recommendations for service reuse during its evolution while preserving acceptable performance, maturity and risk levels. We present the list of contributions hereafter:

C1: Maximizing performance of SBSs by adding new quality characteristics with reference to latest quality standards.

- Performance Efficiency
- Reliability
    - Availability

- Maturity Evaluation
- Risks Management

C2: Semantic Performance Profile: Performance profile will provide insights about performance-oriented decision support concepts and technical indicators. First step is to investigate the definition and structure of the performance based technical indicators of SOA (i.e. what to monitor). Second step is to investigate the different features required to support the activities of the whole SBS lifecycle (i.e. how to monitor). Last step is to investigate the evolution of performance based technical indicators with time dimension along SOA layers and decision support (Semantic performance profile).

C3: Development and implementation of a knowledge-based decision support for service reuse with maximized performance.

C4: Application of machine learning for generating efficient decision and optimized decision by increasing the consumption of data.

C5: Ontology revision for the generation of semantic rules to get global decision for service reuse.

C6: Conforming governance for new End User requirements and access rights.

C7: Dynamic QoS provisioning.

C8: Definition of a case study where performance-based decision support system is applied to validate and measure whether it can be reasonable for the suggesting service reuse.

1.7. Organization of Chapters

This thesis document consists of 5 chapters. After the introduction in Chapter 1, the remainder of the content is organized as follows:

Chapter 2: Service Reuse and Performance based Solutions

This chapter is dedicated to literature review. It introduces current available solutions to the problems confronted in Chapter 1 from existing research works. It discusses the lack of current proposed solutions and point out possible contributions that can be done. In this study, very recent research papers have been considered. The first area of research is related to service reuse solutions. We evaluate service reuse solutions as reusability in SDLC, reusability in service life cycle and SOA governance. The second area of research is related to SBS performance. Performance of SBS is

analyzed based on qualitative and quantitative quality characteristics. We analyze performance monitoring solutions in terms of quantitative quality characteristics. Performance based monitoring can be performed at the SOA layers. Qualitative quality characteristics analysis comprises of risk management and maturity models. Risk management includes literature on business process risk analysis, business process risk management and business process compliance risk. Maturity models analysis consists of literature on process management maturity, CMMI and service oriented architecture maturity model (SOAMM). After analyzing maturity models, we scrutinize mapping of CMMI process areas with SOAMM dimensions and SOA maturity levels and methodical building blocks. The third area of research is service structuring. Services are modeled in terms of service domain and QoS. The most common model that exists in the literature is ontology modeling. We analyze literature that models service domain and QoS in terms of ontologies. The fourth area of research is dedicated to service reasoning. We explore existing literature that provides efficient decision support systems, and data mining algorithms that support to provide effective decisions for SBSs. The literature review is enriched by discussions to critical analyze the limitations of these areas. Finally, we present the conclusion of this chapter.

Chapter 3: Performance Oriented Decision Support

This chapter presents an overview of performance-oriented decision support for SOA layers. For this purpose, we propose a framework named as performance-oriented decision support framework (PODSF). PODSF is divided into two parts. First part describes framework environment. Framework environment comprises of requirements as input, resources required by framework, and output in the form of recommendations. The second part provides details about the PODSF process in the form of components. We explain the details of each component with the help of a small example. PODSF process is composed of six components. Those components are data management, traces management, ontology modelling, reasoner, analytical assessment and impact analysis. Data management deals with ISO 25010 quality characteristics and provides measurement mechanisms for both quantitative and qualitative evaluation of atomic services as a components of composite services. Traces management provides quantitative indicators statistics by deploying and analyzing atomic and composite services in application servers. Ontology modelling helps to store the traces of services and provide the dynamic evolution of services with different instantiations. Reasoner component extracts the concepts of ontologies and provides semantic rules to evaluate performance. Analytical assessment exploits these semantic rules and provides assessment by applying classification algorithms. This component selects the most optimum results. Impact analysis evaluates the overall performance of services by increasing the consumption and taking into account the governance policies of services. Impact analysis provides decision in terms of a decision matrix. It also evaluates

the trend of overall performance. PODSF ensures evolution of service lifecycle with new and changed business needs, and provides assessment for performance-oriented service reuse, by the help of its components.

Chapter 4: PODSF Implementation

This chapter describes the implementation of the PODSF. It is divided into three parts. The first part is based on the PODSF research design, the second part explains the PODSF research prototype, and the third part discusses PODS system. PODSF research design is supported by the implementation of ontology structuring and reasoning rules. Ontology structuring includes the proposed ontologies for service domain, SOA layers and risk. We implement service network ontology highlighting the service domain concepts and adding performance profile concepts. Performance profile concepts are integrated in detail by implementing ontologies at SOA layers. SOA layer for performance profile are service layer, integration layer, process layer and governance layer. Moreover, we implement risk ontology by integrating risk types concepts. Reasoning part describes semantic rules and queries. Semantic rules are implemented in SWRL from the ontological concepts. Semantic rules mainly deal with the ontological concepts of service profile and performance profile. To ensure the consistency of concepts and values, queries have been implemented following SPARQL. PODSF research prototype is aided with different algorithms that we have developed. Research prototype is composed of two parts. The first part includes the algorithms implemented for the data management. The second part deals with the algorithms implemented for the decision support. Data management part is further divided into three parts based on the type of algorithms. The first part of data management shows and explains the algorithm implemented to evaluate and manage performance of atomic and composite services. The second part of data management demonstrates and explicates the algorithm implemented to evaluate maturity of services. Last part of data management presents and explains the algorithm implemented to analyze risk impacts of services. Decision support part is divided into three parts based on the type of algorithms. The first part is performance evaluation for specific decision that shows and explains the algorithm implemented to evaluate and manage global performance evaluation. The second part of decision support demonstrates and explicates the algorithm implemented to evaluate impact analysis of performance with increased consumption of services. Last part of decision support presents and explains the algorithm implemented to analyze and evaluate decision in terms of recommendations. PODS system is further divided into parts. These two parts are high level architecture of PODS system and PODS system classes.

Chapter 5: PODS System Evaluation

This chapter describes the validation of PODS system with the help of the business process use case. At first, this chapter explains the objective of the use case in terms of service reuse. The second step explains the implementation of use case. This step begins by explaining data set repositories and is further expanded to three parts. First part explains the data mining analytics while second part is dedicated to the decision scenarios. Finally, third part explains recommendations provided by PODS system for reuse scenarios of atomic service, composite service and resource utilization.

CHAPTER 6: Conclusions and Future Works

This chapter presents a summary of this research work and research contributions. The conclusions part explains five models that are used in the proposed decision support system. These models are related to performance based technical indicators, ontologies, decision support algorithms, evaluation and validation mechanisms. The proposed decision support system provides recommendations in terms of service and process reuse while accumulating performance, maturity and risk management. Future works part introduces a list of perspectives for directing future related research works.

CHAPTER 2. Service Reuse and Performance based Solutions

In this chapter, we present the works that are relevant to our research. We focus on four areas namely service-based reuse solutions, service performance, performance-based service structuring and performance-based service reasoning.

In the first area, we analyze various solutions that exist in the literature for service reuse. Service reuse is about reusing atomic services as a component of composite services. Composite services can in turn be reuse in larger service compositions. The management of service reuse can rely on technical indicators. SOA governance also intend to manage service reuse.

In the second area, we analyze performance-oriented solutions for SBS. Performance oriented solutions for SBS include qualitative and quantitative perspectives of performance monitoring. Quantitative analysis deals with performance management while qualitative analysis is related to risk management and maturity models.

In the third area, we analyze literature related to performance service structuring in terms of both service domain and QoS. Services and QoS have mainly been structured under the form of ontologies.

Last area of research is related to service reasoning. Service reasoning helps to provide efficient decisions. For this purpose, we analyze existing decision support systems that help to provide effective decisions for SBS. While analyzing existing support systems, we also provide data mining algorithms that play an important role to provide efficient decisions.

The above-mentioned research areas triggered an important body of research. Consequently, we followed a thorough methodology to retrieve and analyze the related works. We hence start this chapter by the analytic methodology. We then present the state of the art related to various existing solutions for service reuse, performance-oriented solutions for SBS, service structuring, and finally service reasoning. We conclude the chapter by a discussion.

2.1. Analytic Methodology

The analytic methodology consists of two steps. The two steps are systematic analysis and research classification. In the systematic analysis step, we provide an analytic approach to retrieve papers

systematically. The analytic approach that we follow is a systematic mapping study. In the research methodology step, we classify related work based on different areas of work retrieved from the systematic analysis step.

2.1.1 Systematic Analysis

Systematic mapping study is a method that has initially been used in classification of medicines. Recently, it has also been applied in software engineering field. The systematic mapping study allows to show the frequency of publication, to determine the scope in the certain field, and to combine the results in answering the research questions. There are five steps in systematic mapping study including defining the research question, searching the relevant papers, filtering the papers based on the abstract, and mapping the data extraction. The review is carried out using Scopus, Web of Science and ISI Web of knowledge with different search criteria. Our first search criterion is service performance. For that, we extracted 3103 research papers from year 2004 to 2018. We selected 2842 papers relevant to computer science domain in the second analysis. 994 papers related to SOA have been selected in the third analysis. The fourth analysis resulted in 486 papers based on Web services. The fifth analysis included 97 papers related to monitoring of Web services. We eliminated papers published before year 2010 which resulted in 28 papers in the final analysis. We categorized these papers based on performance management. The second search criterion is service reuse. For that, we extracted 711 research papers from year 1991 to 2018. Most of the papers under this category is not related to service-based systems but rather focus on computer networks or some other domains like water and energy consumption. Therefore, we selected papers that are relevant to service-based systems that results in 16 papers. The fourth search criterion is service performance-based decision support. For that, we extracted 103 research papers from year 2000 to 2018. We select 10 papers that are relevant to service-based reasoning.

2.1.2 Research Classification

The results of the systematic analysis provide a systematic structure allowing to build research classification. We build a research classification based on the systematic analysis of relevant research areas as shown in Figure 5. This classification is based on service reuse and performance-based solutions. There are four main classifications for this research area. The first classification consists of research works that proposes architectures or systems for service reuse. These works highlight some important concepts related to quality evaluation and quality engineering. Quality is evaluated on the basis of time, cost, performance, resource, capacity and efficiency. Quality is structured on the basis

of ontologies and managed through frameworks and knowledge representations. The second classification consists of research works that focus on service performance. Performance of service is measured or analyzed at both quantitative and qualitative levels. Quantitative level provides quantitative technical indicators for performance management while qualitative level analyzes maturity and risk. The above mentioned two classification approaches yield to two other major classification defined as performance-based structuring and performance-based reasoning for services. Structuring is performed by using ontologies while reasoning has been made by decision support systems followed by some data mining algorithms for efficient decision support.

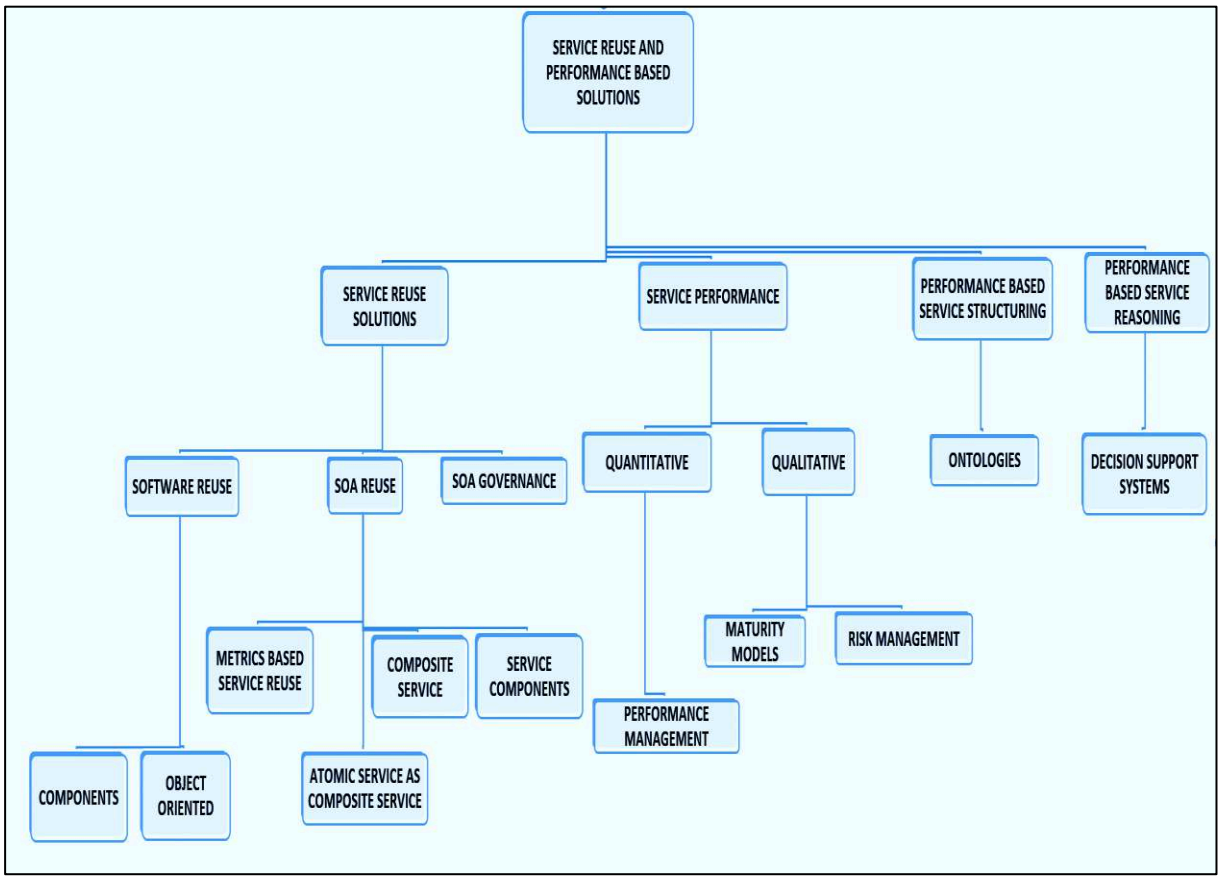


Figure 5: Research Classification

2.2. Service Reuse Solutions

The concept of reusability has been used along various axes. We focus here on reusability in software development life cycle (SDLC), Service life cycle and SOA governance.

2.2.1 Reusability in SDLC

The first dimension in which reusability plays a vital role is SDLC. Software reuse has been used in the context of object-oriented programming and component-based development. The primary mechanism for achieving reuse in object-oriented programming is inheritance. Inheritance creates strong dependencies such as coupling among application objects. A variety of approaches have emerged to guarantee reusability for SDLC phases. Aversano et al [26] propose an approach to identify reusable components in software systems by analyzing the business processes that use them. The authors intend to obtain services from existing pieces of code. They extract component's code from the existing software systems by identifying those ones that support the business process and candidate them for implementing a service. For this reason, they exploit the recovery of the links existing between the business process model and the supporting software systems.

2.2.2 Reusability in Service Life Cycle

The second dimension is to guarantee atomic services reuse and composite services. Feuerlicht and Lozina [27] list three principles for service reuse: service coupling, service cohesion and service granularity. They define service reuse as the ability of a service to participate in multiple service compositions. They closely relate service reuse to service composability. Perepletchikov et al. [28] propose an approach that measures cohesion of service. This approach is applied at design time for service interface. They highlight cohesion in the context of service interface data, usage, implementation and sequential. This approach does not consider XML-based service description language and business process definition languages. This approach mainly concerns service consumer for the utilization of the service. BPEL process reuse is also promoted in organizations and SOA solutions with the aim of reducing effort and change. Xue et al [29] propose a technique of process partitioning. The authors construct decentralized service compositions from the code and provide a graph transformation-based approach. They also discuss some issues about decentralized service compositions and performance tests of service compositions. From the experimental results, the authors show that this technique have lower average response time and higher throughput in runtime environment as compared to centralized composition approaches. Khoshkbarforoushha et al. [30] propose an approach for evaluating reuse of a composite service. This approach provides analysis based on logic and description mismatch. They propose a metric formula to decide the probability of a service to be reused. The authors applied this metric on a BPEL process. Choi et al. [31] present a model for reuse of atomic services in SOA. This model is based on the metrics of business commonality, modularity, adaptability, standard conformance, and discoverability of services in SOA. The authors

perform evaluation based on the feedback of service consumers and provide an analysis report. Doultsinoua et al. [32] provide a procedure for selecting requirements and mapping with service reuse from existing repository of services for product-service systems. The authors describe the service issues and service knowledge that has an impact on product design. Lee et al [33] provide a feature-oriented approach, to analyze and identify reusable services. Feature analysis and modeling are employed to identify and group units of features to provide services at the right level of granularity in a SBS. Ahmed-Kristensen and Vianello [34] propose reuse service knowledge (RSK) model based upon the findings and the understanding from a general framework for developing a knowledge management strategy. The RSK model was developed based on a case study from a customized industry. The authors describe a case study from the oil industry investigating the transfer of knowledge within the service phase and also between the service and design phases. Allen et al. [35] provide a detailed description of the behavior of the network communication broker. The authors propose a method that incorporates smart reusable integration, automation and End User controllability. This work influences the convergence of services and providers to the End User the required services.

In order to provide effective decisions in the perspective of the lifecycle of service components, services and business processes, SOA governance plays an important role. In the next coming part 2.2.3, we analyze papers that focus on governance for service reuse.

2.2.3 SOA Governance

In order to promote service reuse and to create business agility, SOA governance mechanisms have been proposed. In this context, the Open Group [36] proposes a governance model. This model consists of two categories of processes that are governance processes and governed processes. The authors divide governed processes into four parts. These parts include the management of service portfolio, service lifecycle, solution portfolio and solution lifecycle. Niemann et al. [37] propose SOA governance model. This model is composed of five main building blocks. These building blocks are organizational governance entities, governance policies, best practices catalog, compliance observation and SOA maturity measurement. Filho and Azevedo [38] extends Niemann's governance model and propose an approach named as a common governance (CommonGov) model for SOA. CommonGov consists of four groups that are strategy, compliance, execution and support. The purpose of these groups is to ensure governance at their respective levels. Strategy group focuses on achieving governance at strategic level. Compliance group ensures policies. Execution group handles service portfolio, service life cycle, solution composition cycle and solution portfolio. Support group

handles versioning, monitoring and problem management. This work addresses and defines processes of governance model. Joachim et al [39] classify governance into three broad categories. They identify corresponding existing governance mechanism for each category. The authors have made this classification based on a survey of 81 SOA based companies in Germany. They highlight the different kinds of governance policies used in these companies at various departments. The three categories are structure, processes and employees or relations. Governance mechanism for the category of structure includes decision making bodies, standards and roles or responsibilities. Governance mechanism for the category of process includes service management, service development and performance measurement. Governance mechanism for the category of employees or relations includes qualifications, IT or business communications, collaboration and incentives. Dan et al. [40] highlight advantages and challenges of reuse of services. The purpose of this research is to apply different practices of SOA governance to address facets like terminology, service discovery, creation, and service entitlement. This paper discusses the importance and challenges of reuse in SOA. The authors define three key benefits of service reuse such as improving agility of solutions, reducing costs, and reducing risks. In this survey, they list the properties or fundamentals of governance used in the surveyed companies. Kim et al. [41] propose a decision model to evaluate the services based on prioritization mechanism. This approach identifies optimum service portfolio after making prioritization. The model considers prioritization of technical feasibility, business needs, development and maintenance cost. This model decides about the potentially realizable services based on the priority of each company.

2.2.4 Summary

Reuse has gained much attention and has been considered very important in the IT industry. Initially, reusing code and runtime components came into existence. As the IT industry evolves with new architectures and technologies, SOA emerged with the key benefit of runtime reuse. SOA promotes reuse of atomic services as composite services and reuse of composite services in composition of larger business processes. From the analysis of SOA governance approaches, we conclude that two areas are important to promote service reuse. These two areas are service compliance or policy and service performance measurement or monitoring. For this purpose, service portfolio can be enhanced. It is important to align performance metrics with business goals such as increase in flexibility and reduction of business process costs. However, it is important to note that SOA governance varies according to the size and function of the organization. Existing metrics have not been pragmatically validated. It is required to suggest reuse of services that are efficient in terms of performance and are compliant with governance policies.

2.3. Service Performance

Web services can be monitored automatically or semi automatically at both atomic and composite level. They can also be monitored at any phase of service life cycle. Monitoring quality involves evaluating current performance based on some standards or expected level of performance. Service performance can be measured at both quantitative and qualitative levels. We explain the quantitative and qualitative service performance evaluation below.

2.3.1 Quantitative

As mentioned in the introduction chapter, ISO 25010 proposes quality characteristics. These quality characteristics are measured by quantitative indicators. In this thesis, we focus on performance efficiency and reliability characteristics of ISO 25010. Reliability is measured based on availability sub-characteristic and performance efficiency is measured based on time behavior and capacity sub-characteristics. Quantitative technical indicators related to time behavior and capacity are response time, service start, end times, throughput, transporting time (up time), CPU time and bandwidth. Quantitative technical indicators related to availability are request count, response count, up time, execution or processing time and down time. The performance of a service can be analyzed based on these quantitative technical indicators.

A. Performance Monitoring

Services are directly involved at the service, business process and integration layers of SOA, therefore performance can be monitored at these three layers. Performance of service can be analyzed based on quality characteristics such as performance efficiency and reliability. These quality characteristics are measured by different technical indicators like response time, throughput, availability, execution time, network bandwidth, CPU time, server CPU, task CPU, server memory utilization and task memory utilization. Oriol et al [42] propose a framework named as SALMon. It is supported by two services: the monitor service and the analyzer service. The monitor service measures the values of response time, throughput and availability. The analyzer service detects whether SLA is being or going to be violated. SALMon supports passive monitoring and testing. It supports any type of service technology. This framework ensures quality guarantees in SLA. The authors perform tests on 11 services with 30 invocations per second. In this paper, the dynamicity and real-time constraints of QoS is not analyzed. Garcia Valls et al [43] propose iLAND as a service-oriented approach for timely reconfiguration of real-time systems. Monitoring of time, network bandwidth and power consumption is performed in this

paper. In this paper, the authors provide a virtualized infrastructure where service-based applications can execute. Asadollah and Thiam [44] propose monitoring of Web services. For this purpose, they propose a method to calculate the response time of Web services by using a proxy that is connected to the requested services. The authors have validated their approach by three tests for three different Web service invocations. The first test is used to measure the processing time. The second test measures the processing time and transporting time. The third test measures queuing time for Web services, where two End Users may invoke a Web service at the same time. The proposed monitoring method has not been implemented. Avila et al [45] propose an optimization model that performs service selection based on historical QoS data. Historical QoS data includes execution time traces. The authors use fuzzy logic to dynamically define the level of QoS that can be delivered. Garcia Valls et al [46] propose reconfiguration of service compositions for distributed real-time systems. Monitoring of execution time is performed in this paper. They present an algorithm that target embedded real-time systems. Zheng et al [47] investigate quality of service based on real world Web services. They conduct three large-scale distributed evaluations on real world Web services and collect comprehensive Web service QoS data sets. First, 21,358 Web service addresses are obtained by crawling Web service information from the Internet. Then, three Web service evaluations are conducted. In the first evaluation, failure probability of 100 Web services is assessed by 150 distributed service End User. In the second evaluation, response time and throughput of 5,825 Web services are evaluated by 339 distributed service End User. And in the third evaluation, response time of 4,532 Web services is evaluated by conducting 30,287,611 invocations. The authors have used PlanetLab global research network for monitoring of QoS. Kahlon et al [48] conduct a survey of existing research papers based on the activities of Web service life cycle. They highlight different activities involved in the service life cycle as specification, requirement, analysis, deployment, execution, monitoring, recovery and testing. McKee et al [49] monitoring current system state within a workflow execution. In this work, system performance is monitored by server CPU, task CPU, server memory utilization and task memory utilization. Experimental validation is performed on real server utilization data from Google Cloud and their own local server cluster. The authors apply their probabilistic model to a single service in the workflow. Boumahdi et al [50] propose model named as SOA+d. They integrate elements into two dimensions such as conceptual and methodological. They model decision, intelligence, design and the choice activities.

Performance of business process can also be analyzed based on quality characteristics such as performance efficiency and reliability. These quality characteristics are measured by different technical indicators like response time, and CPU time. Aschoff and Zisman [51] monitor the response times of services. They propose proactive adaptation of service composition (ProAdapt) based on the

exponentially weighted moving average (EWMA). Decision has been made by considering both response time and the cost value. Moreover, ProAdapt is dynamic and automated. Fan [52] proposes an approach to measure computation cost and CPU time. The author uses particle swarm optimization algorithms for approximating execution plans of composite services. He performs experiments to monitor computation cost and CPU time on 30 tasks and 50 services. Sheng et al [53] provide a survey of Web service composition and Web service technologies. They evaluate service composition platforms on the basis of some parameters. These parameters include ease-of-use, simulation, adaptability, optimization, security, administration and monitoring. The authors found that many of the proposed automatic composition systems are unable to adapt to dynamic environments.

Integration layer mainly deals with the communication mechanisms. Messages are exchanged through SOAP, HTTP, TCP/IP protocols. Messaging through SOAP provides the ability to perform the necessary message transformation to connect the service requestor to the service provider and to publish and subscribe messages and events asynchronously. The following research for integration layer focuses only on response time, network load (bandwidth) and throughput. Tari et al. [54] provide a benchmark of different SOAP bindings in wireless environments. The experimental results show that UDP binding has the lower overhead which results in a reduction in the response time and an increase in the total throughput. Then Tari et al. [55] propose a similarity-based SOAP multicast protocol (SMP) which reduces the network load by reducing the total generated traffic size. In the next paper, Tari et al. [56] propose a tc-SMP technique as an extension of SMP providing the performance improvement of tc-SMP of about 30% higher network traffic reduction than SMP at a small expense of up to 10% rise in the response time. Yoon et al. [57] propose a mechanism to identify a suitable QoS combination for a specific system or a communication environment. However, it is difficult to find optimal QoS combination and their values for a certain system or service amongst many combinations.

B. Summary

We define parameters to perform a systematic analysis of the above-mentioned research papers. For each paper, we provide an overview of the content. We resume the approach taken by the authors and list the performance based technical indicators used in these papers. We mention if the approach is dynamic and using real time data or not. We also mention the number of services used while performing tests. We highlight the use of a decision support. For each paper, we make categorization as service, business process, integration and server level. Comparison of the existing papers based on these parameters is shown in Table 1 and Table 2. We observe that the existing approaches do not cover all the layers of SOA discussed in Chapter 1. Moreover, these approaches do not provide the decisional aspect for SBS in order to cover the service reuse capability, managing of resources and

suggesting service compositions. Approaches are primarily static in nature, which makes them unsuitable for assuring runtime and emerging system qualities. Distribution of the quality models along the time dimension and the identification of their relationships are missing. There is a need to maximize performance by adding more performance based technical indicators for SOA layers. Existing approaches are not adaptable to new or changed business requirement. We will investigate the definition and structure of the performance based technical indicators of service-oriented architecture from latest release of ISO/IEC. We will develop and deploy services in WSO2 and oracle server to monitor the behavior of services with respect to selected performance based technical indicators. Along with that we will create a performance profile of quantitative and qualitative technical indicators with time dimension along SOA layers. To make the whole process dynamic, we will use ontologies as a modelling paradigm and propose a decision support algorithm. We will recommend service reuse based on the performance evaluation of several Web services.

Table 1: Comparison of the Existing Performance Management Techniques (1/2)

	Boumahdi 2016	Mckee 2015	Buga 2015	Oriol et al 2015	Rastegari et al 2015	Kahlon et al. 2014	Sheng et al. 2014	Zheng et al 2014	Whitham 2014
Overview	Bridge the gap between SOA and decision automation	Monitoring current system state within a given workflow's execution window	Decentralized monitoring architecture	Model-based and Invocation-based configuration of SALMon, Re-selection, Redeployment	Fuzzy-based dynamic reconfiguration of SBA at runtime	A technique based on web Services monitoring life cycle process	An overview on the latest development in the field of WS composition	Evaluations on user-observed Qos of services from distributed locations	Analyze performance of within service methods. Service is regarded as black box
Approach	Meta model based on Intelligent design choice model of Simon	Probablistic methods	Abstract state machines	Model based/invocation based strategies	Context model Fuzzy logic controllers Jmeter load simulator Zabbix monitoring system	A survey	A survey	Qos evaluation using Axis2. Monitor Qos using PlanetLab	Comparison between scratchpad memory and a method cache
Perfomance Indicators	NA	Server CPU utilization, server memory utilization, task cpu, task memory, task execution time	Availability Network bandwidth	Response time Throughput	AVG Response time, Throughput, AVG CPU Load, Load level, AVG Execution Time	Not defined	Not defined	Failure probability Response time Throughput	Cache utilization Execution time
Dynamic Qos	NA	Yes	No	Yes	Partially	NA	NA	NA	No
Real-Time	NA	Yes	No	Yes	No	NA	NA	Yes	Yes
No of services	NA	1 no of service, 6 no of servers, 15000 service calls	Not defined	11 services 30 invocations per second	Not defined	NA	NA	Failure of 100 services, RT and TP of 5,825 services, QoS changing of 4,532 services. 30 million invocations	Not defined
Decision support	Yes	No	No	No	No	NA	No	No	No
Service level	✓	✓	×	✓	×	✓	×	✓	×
Communication level	×	×	×	×	×	×	×	×	×
Business Process level	✓	×	×	×	×	×	✓	×	×
Server/ Resource level	×	×	×	×	✓	×	×	×	✓

Table 2: Comparison of the Existing Performance Management Techniques (2/2)

	Garcia-valls 2013	Fan 2013	Garcia-valls 2012	Zou 2012	Zarghami et al. 2012	Asadollah et al. 2012	Zheng et al. 2011	Oriol et al. 2010
Overview	Service-based reconfiguration from a system perspective	A method for personalized composite service to find feasible solution	Separates Qos into data centric and resource centric layers	Model Web service Qos. Impact of underlying resources on service execution is not considered	Decision as a service	An architecture to measure and monitor response time of web services	A calculation method for prediction of performance	A tool assessing the satisfaction of service level agreement
Approach	Schedulability analysis algorithm. Service composition algorithm	Particle swarm optimization	Service workflows	AI Planning, probablistic methods	Request-response manner	Client server architecture using HTTP	Probabilistic	Passive and active monitoring of services
Perfomance Indicators	Execution Time	Computation cost, CPU time	CPU time, period, deadline, priority, network bandwidth, power consumption	Availability, reputation, execution time, execution cost	Not defined	Response time, Processing time, Transporting time	Execution time	Response time, Availability
Dynamic Qos	Yes	No	No	Partially	NA	No	Yes	No
Real-Time	Partial	No	Yes	No	NA	No	Yes	No
No of services	4 no of services	30 tasks, 50 candidate services	Not defined	30 Web service repositories with 7, 275	Not defined	Not Defined	7 no of services, 1 composite service with 10000 invocations	Not defined
Decision support	No	No	No	No	✓	No	No	
Service level	✓	×	✓	×	✓	✓	×	✓
Communication level	×	×	Partial	×	×	×	×	×
Business Process level	✓	✓	×	✓	✓	×	✓	×
Server/ Resource level	×	×	Partial	×	×	×	×	×

2.3.2 Qualitative

Performance of a service can also be analyzed based on qualitative characteristics. Qualitative characteristics proposed by ISO 25010 are explained in the introduction section. The following research focuses on reliability characteristics of ISO 25010. Reliability is measured based on maturity and risk management.

A. Maturity Models

A maturity model is a way to assess and improve business processes constantly. It describes the typical patterns in the evolutionary process of technology and business development of an enterprise [58]. It is used to rate the capabilities of maturity elements and to select the processes for improving their maturity [59]. Maturity models are widely applied in the field of information systems as a means of

benchmarking and as an approach for continuous improvement [60]. ISO/IEC 15504 [61] uses (ISO/IEC 12207) [62] to identify process capability for process improvement. It acts as a base in conducting an assessment for process definitions. ISO/IEC 15504 [63] defines process assessment as "the systematic evaluation of an organization's processes against a process reference model". Röglinger et al. [64] provide an analysis of a set of BPMMs. They provide an exhaustive analysis of ten BPMMs with respect to general design principles. These models adequately address the principles for a descriptive purpose of maturity model use. BPM-CF [65] develops a holistic BPM maturity model. It facilitates the assessment of BPM proficiencies. It has five maturity stages. BPMM-OMG [66] and vPMM [67] provide a maturity model and the assessment model. BPMM-OMG has five maturity levels and defines process areas at each level. This work refers to models that are publicly available. PMMA [68] provides a holistic evaluation of all areas relevant to BPM based on a complete set of criteria. The authors provide a five steps model as an indicator for process maturity. This model corresponds to the implementation topics of the business process management. BPMM-OMG [69] defines an intelligent maturity model tool. The Capability Maturity Model Integration (CMMI) models are the most well-known generic maturity models. The CMMI Product Suite has three different models. These models are CMMI-DEV, CMMI-SVC, and CMMI-ACQ for development, services, and acquisition, respectively [70]. These models provide the essential elements that describe the characteristics of effective processes [71]. The CMMI-DEV model is used to assess and improve the software engineering processes of an enterprise that develop a product, whereas CMMI-SVC model is used to assess and improve the management of service delivery. CMMI models have two representations, staged and continuous. SCAMPI [72] is a standard CMMI appraisal method for process improvement. It is a process assessment method to identify maturity level of a software organization. It evaluates compliance with CMMI models. Hirschheim's et al. [73] publish a study report. The authors analyze SOA aspects from the perspectives of different stakeholders and propose Welke's SOAMM to oversee the enterprise SOA adoption. IBM provide Service Integration Maturity Model (SIMM) [74]. SIMM focuses on the maturity of services and their integration. This model has a method dimension that discusses the maturity characteristics of service development methodologies at different levels. Later, it has been adopted by the Open Group [75]. Hensle and Deb of Oracle Corporation [76] highlight SOAMM as a guide to accelerate the enterprise SOA adoption. This work describes a road map based on Oracle SOAMM. It does not provide any guidance on the methods being followed for service system engineering. Baghdadi [77] proposes the SOA maturity framework to guide the SOA adoption process. The author analyzes four SOAMMs, compare them with CMMI-SVC and proposes a maturity framework. This work is limited to the framework and can be further extended by providing methods for SOA adoption including models and tools. There are several SOAMM contributed by both academia and industry such as IBM SIMM/OSIMM, HP SOAMM [78], Oracle SOAMM, iSOAMM [79], and Welke's

model [80]. Welke's model focuses on the maturity of enterprise architecture. There are two facets for this model that are SOA attributes and SOA motivations. The SOA motivations are infrastructure efficiency, reuse, application and data composition, business analytics and processes, flexibility and agility, and enterprise transformation. Each of these motivations is associated with a maturity level. Infrastructure efficiency and reuse are at the bottom level, composition and business process management are at the middle level, and flexibility and enterprise transformation are at the highest level of maturity. Figure 6 depicts the six SOA attributes of Welke's model.

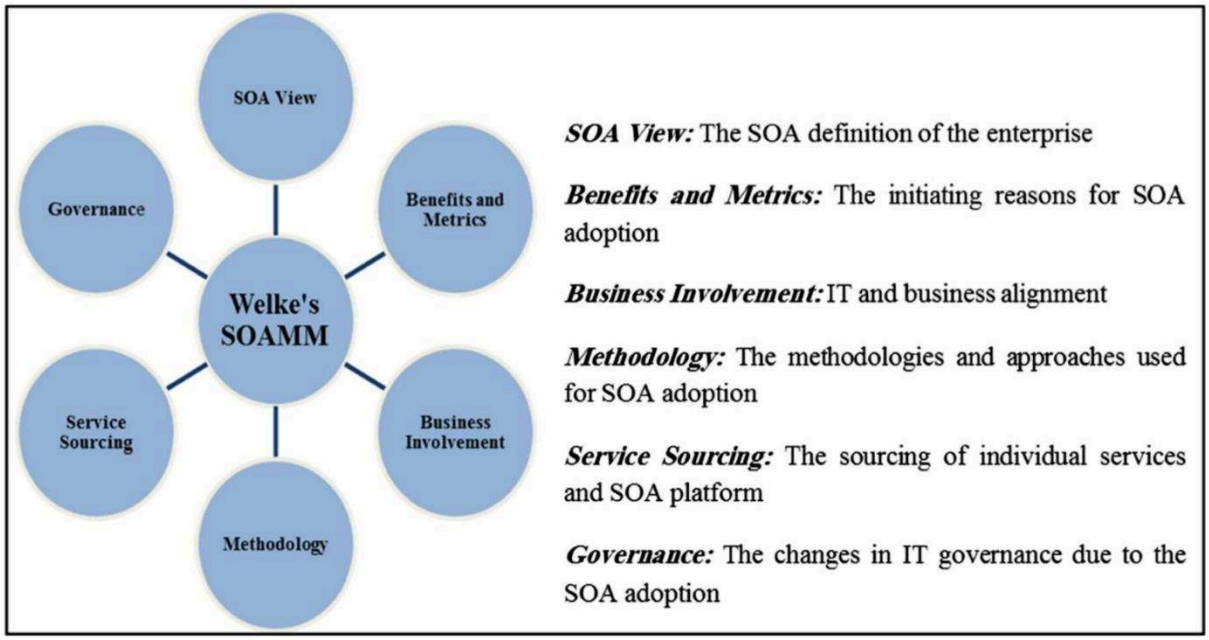


Figure 6: Dimensions of Welke's SOAMM [80]

In the following part, we explain research related to the mapping of CMMI process areas and SOAMM dimensions. For this purpose, Pulparambil et al [81] propose a set of overlapping categories between CMMI and SOAMM. However, they are using a different criterion to define the maturity level. Most of the SOAMMs adopted the same terminologies and levels of CMMI models. The authors highlight the commonalities between CMMI process areas and SOAMM dimensions as we can see in Figure 7. The assessment areas can be broadly classified into five categories. In contrast to CMMI models, SOAMMs bring the architecture view and the business involvement under the scope of maturity assessment. The service establishment and delivery, support, and work management are common to both the models. In addition to these common process areas/dimensions, there are some sets of dimensions that are unique to SOAMMs to focus on different aspects of enterprise SOA adoption.

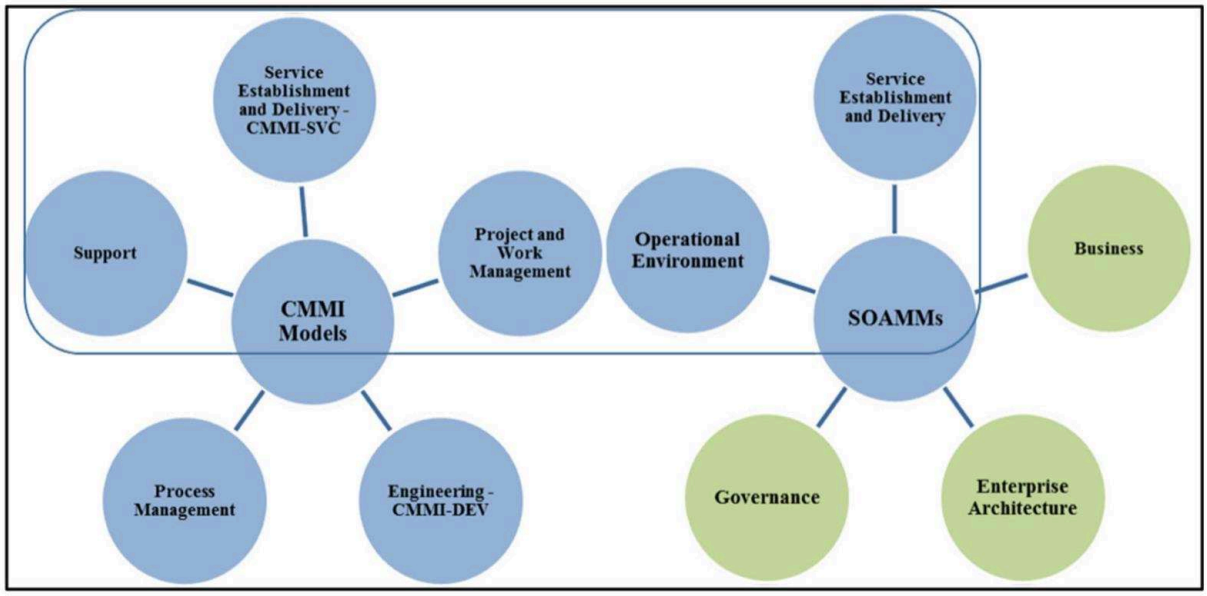


Figure 7: Mapping of CMMI Models and SOAMMs [81]

After mapping CMMI process areas and SOAMM dimensions, Pulparambil et al [82] propose an approach based on SOA adoption initiative for maturity levels. The authors model the service lifecycle activities as shown in Figure 8. Service lifecycle activities include service identification, service contract design, and the service discovery phases. The governance of service defines a set of processes to manage the activities of service lifecycle. This includes design-time rules and enforcement for service creation and run-time governance policies for service usage and operation policies.

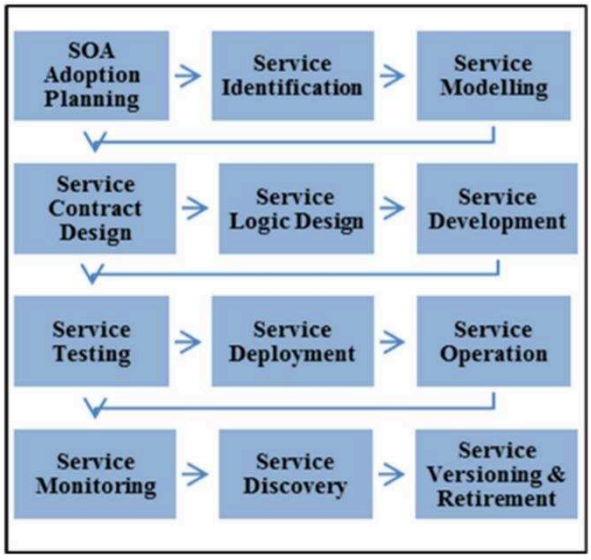


Figure 8: Service Lifecycle [82]

The authors define building blocks for the five levels of maturity. They are graphically represented in Figure 9. The bottom layer represents the entire corporate IT. The Initial level uses existing software development methods such as object-oriented or component-based. The Managed level uses service-oriented software engineering methods, enhances the method and practices to address the creation, implementation, and deployment of services, and adopt SOA project methodology. The Defined level

implements business process modeling, a formal method across the enterprise such as service-oriented modeling, composite application management, business process and business rules governance. The Quantitatively managed level implements intra and inter-organizational service definition; formal methods are used to create and manage both internal and external services, and processes are quantitatively managed to drive business value and leverage business activity monitoring. The Optimized level focuses on business process improvement, adaptive enterprise and support virtualization, focuses on business optimization, refines and improves standards.

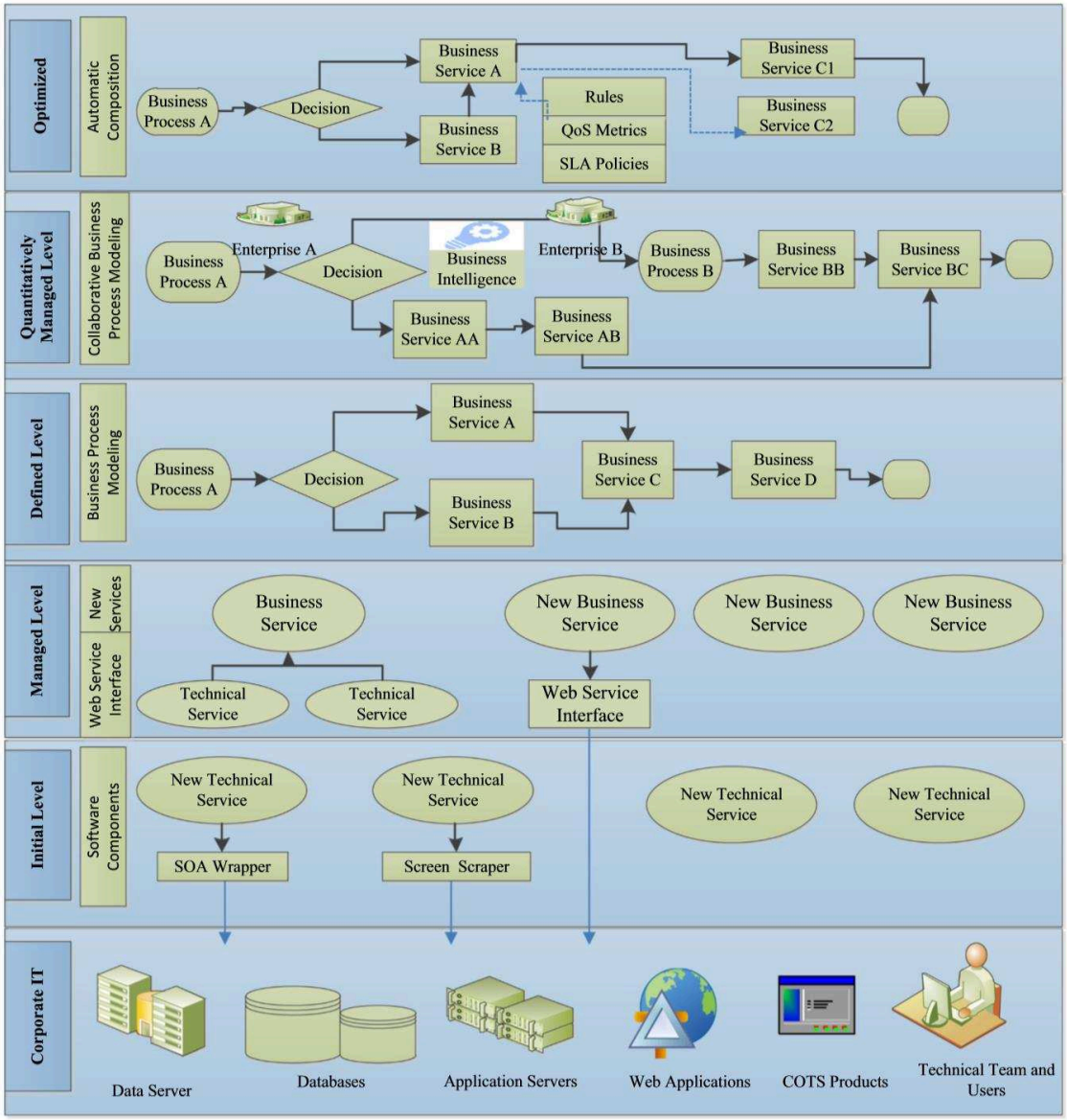


Figure 9: Methodical Building Blocks [82]

B. Risk Management

Risk management plays an important role for addressing the issues of threats. There exist diverse classifications of these threats [83,84,85] ranging from accidents to natural catastrophes and to deliberate acts. In the past years, risk management also highlights incident, disaster recovery and

business continuity management [86, 87, 88, 89] along with threats. Over the last years, scientific community increased its research efforts for highlighting the importance of business risks. In order to do so, several research papers as mentioned below highlights business process risk analysis, business process risk management and business process compliance risk.

The application of existing business process risk analysis techniques is an important area. However, the number of existing methods that endeavor to apply existing business process risk analysis techniques is low. CORAS [90] is a method for conducting security risk analysis. CORAS provides a customized language for threat and risk modeling. It provides guidelines to capture and model relevant information during the various stages of the security analysis. The CORAS approach includes seven steps that are introductory meeting, high-level analysis, approval, risk identification, risk estimation, risk evaluation and risk treatment. Jallow et al. [91] propose a framework for risk analysis in business processes. The framework consists of the steps that model the activities of the business process as identify risk factors and probability of occurrence and effect. The authors provide a prototypical framework implementation by using Microsoft Excel. The framework consists of the following six steps such as model the activities, determine dimensions, identify risk factors, impact of risks, calculate each identified risk and calculate forecasts for each activity. There are several research results regarding the integration of risk aspects and security requirements into business process analyses.

Sackmann [92] proposes a model as IT risk reference model for business process-oriented view. This model builds the bridge between the economic and more technical layers including vulnerabilities. The author defines the relations between causes of IT risks and their effects on business processes or a company's returns. Sackmann [93] extends his work and expresses these relations in a matrix-based description. This model consists of four interconnected layers that are business process layer, IT applications or IT infrastructure layer, vulnerabilities layer and threats layer. Zur Muehlen and Rosemann [94] propose an approach to tackle the topic of risk management in the context of business process management. Additionally, they propose a taxonomy for business processes that includes five clusters such as goals, structure, information technology, data and organization. They propose two distinguished lifecycles such as build-time and run-time, and provide the classification of both, errors and risks.

Sadiq et al. [95] propose a method for business process compliance. They highlight the dependency and interconnection between business and control objectives. They handle the modelling of control objectives along with their transmission onto business process models. Weber et al. [96] describe a method for validating whether the states reached by a process are compliant with a set of constraints.

This will check the compliance of a new or altered process against the constraints base, and the whole process repository against a changed constraints base. The authors formalize a knowledge base that consists of compliance rules and annotated process models respectively. Jakoubi et al. [97] propose a method as risk-oriented process evaluation (ROPE). They represent risk elements by the help of graphical notations. The elements of risks include threats, resources, counter measures, and recovery actions for business process activity. Based on the ROPE methodology, Tjoa et al. [98] and Suriadi et al. [99] propose a reference model for risk-aware evaluation. Milanovic et al. [100] present a method for demonstrating availability related to services, underlying ICT infrastructure and human resources. They use a fault model of two failure modes as temporal and value. The authors provide an analytical assessment method that follows seven steps such as define business process, refine activities, create an infrastructure graph, map services to infrastructure components, map business processes to atomic services, transform the Boolean expressions into reliability block diagrams or fault trees and calculate the availability of business process and services.

C. Summary

The existing literature based on maturity presents a conceptual framework of methodical building blocks at different levels of maturity. There is a lack of prescriptive properties as a guiding approach. The existing models are generic maturity models for business processes with different viewpoints. It is required to construct methods based on the conceptual framework and provide prescriptive properties in order to guide the evolution from current maturity level *i* to next maturity level *i+1* for SBS's. We aim to fill this gap by proposing methods to measure the level of maturity and we will provide a prescriptive guiding approach to evaluate the evolution from one maturity level to another.

The existing literature is based on risk analysis and risk-aware phases of process modelling. Types of risks with respect to SBS have not been mentioned in the above studied research papers. Along with that, impact of risk is also missing. There is also a lack of prescriptive properties in terms of mitigation actions for risks. Moreover, none of the above-mentioned approaches specifies dynamic allocation of resources with respect to risk. We can provide for example, in-depth consideration of service level management threats. There is a need of formalizing risk constructs in the form of semantics. We will identify types of risks related to SBS and formulate the impact of these types of risks on SBS. We will also propose mitigation actions for the identified risks and their impacts. As a result of these mitigation actions, we will recommend dynamic allocation of resources.

2.4. Performance based Service Structuring

In order to structure the services and quality of services, ontologies have been widely used for the creation and elicitation of domain knowledge [101]. Ontologies represent formal specifications about the component of the systems and their relationships in a machine understandable and processable manner [102]. They have played an important role in both semantic web applications and knowledge engineering systems [103]. Several tasks such as information storage, processing, retrieval, decision making are done on the basis of ontologies. In this section, we provide analysis of existing ontology-based research that focuses on QoS. Ontologies archive current or historical decision-making processes and their outputs, outline decision-making rules, provide semantic representation and a tree structure that provide better searching time. Ontologies explicitly represent the data along with their semantics to facilitate the transfer of information. They are used to describe semantic representation of Web services such as domain concepts and terminologies of QoS properties.

2.4.1 Ontologies

As explained in the introduction, QoS is a non-functional property of a service. In this part, we provide the analysis of various QoS ontologies that exist in the literature. Giallonardo and Zimeo [104] propose the onQoS ontology defined with OWL language. It is composed of three extensible complementary layers: upper, middle and lower ontology. The upper ontology introduces the QoS ontological language that provides “the words” to describe and formulate the information of QoS. The middle ontology describes the standard vocabulary of ontology such as QoS parameters, QoS based technical indicators and QoS scales. The lower ontology describes the concepts, the properties and the constraints of a specific domain. Lin et al. [105] proposed an ontology based QoS-Aware support for semantic Web services. They have composed their ontology into upper and lower level property. Tran et al. [106] propose QoS ontology to store the information and constraints of QoS properties at different quality levels and fine-grained service level. This QoS ontology allows storing QoS values from End User and system brokers. The QoS based technical indicators for this ontology are throughput, latency, response time, MTTR, uptime, failure, authentication, failover, disaster and cost. The QoS properties of the attributes are divided into two groups: required and optional. Each group can contain many sub level properties that can have different data type values such as single value types (string, boolean, enumeration and numeric), boolean and string-based type (for non-measurable QoS properties),

numeric based type (for both measurable and non-measurable QoS properties) and multiple value types (range, set, list, vector). D'Mello and Ananthanarayana [107] propose an extension to the OWL-S to support the storing of QoS and business offering. The authors create functionality ontology to store all functional concepts. The authors proposed a service selection algorithm that performs the functional matching by comparing the functional concept provided by End User with the concepts of Web services based on the functionality ontology, and a matching degree is determined. Benaboud et al. [108] propose a semantic Web service discovery approach based on agents and ontologies. The framework is modelled by adding semantics of QoS attributes with Web service profiles. It describes the design and implementation of a Web service matchmaking agent. Agent uses an OWL-S based ontology and an OWL reasoner to compare ontology-based service descriptions.

Moraes et al. [109] propose an ontology named as MonONTO. They have considered network performance technical indicators such as response time, availability and throughput. Their system monitors the performance of advanced internet applications. This ontology serves as a support to a decision recommendation system by providing high-level information to the End User about the compliance of the network facing the service level demands. This process is primarily accomplished through the match making of Network Characteristics against Service Characteristics individuals. These individuals are essentially concepts of QoS technical indicators. Pakari et al. [110] propose a hybrid semantic matching approach for service discovery from OWL-S ontology. The authors have enhanced OWL-S ontology by creating concepts to store End User requirements for services. They have calculated global matching score from the sum of multiplication between weights with the scores obtained from three different comparisons. These comparisons are syntactic similarity, structural similarity and semantic similarity. Chhun et al. [111] propose Web service ontology (WSOnto) as shown in Figure 10. WSOnto is composed of two parts. The first part presents services categories and the second part presents service information. The service's category refers to the value of tModel in the UDDI registry. Service information part is composed of functional and non-functional properties. Functional properties are related to the input, output of service and service operations, while non-functional properties are related to QoS. QoS is further composed of performance and security. Performance based technical indicators includes availability, execution time and totalCalled, while indicators related to security are encryption, authentication and authorization.

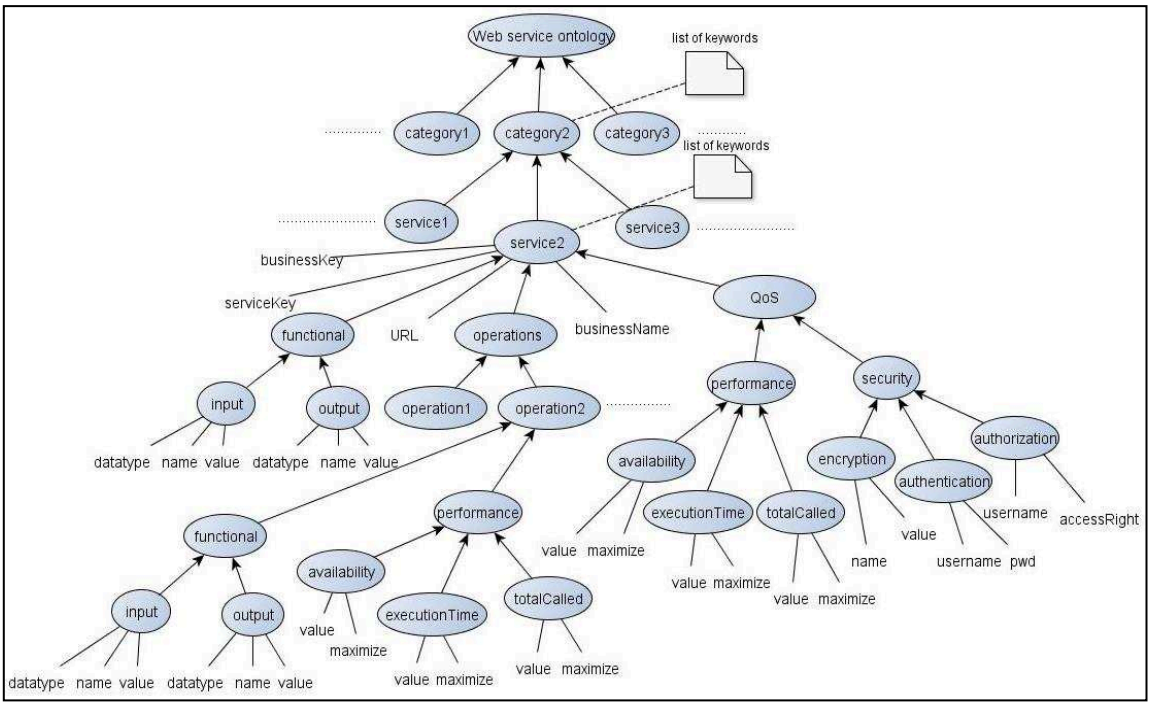


Figure 10: Web service Ontology [111]

2.4.2 Summary

The use of ontologies facilitates the representation of shared concepts in a domain or across domains by specifying a set of terms to ensure proper communication between the enterprises. At present, different organizations are developing their own ontologies, in most cases independently, to describe the same, different or overlapping domains. There are several ontologies developed for service domain, QoS and performance of SBS explained in the above part. All of these ontologies focus on specific QoS related to network, services and business processes. Few ontologies perform selection of Web services based on End User specified QoS. Above mentioned QoS ontologies are not adaptable for new or changed End User requirements. Moreover, QoS ontologies do not evolve with time. Since several QoS information is stored in the above-mentioned literature, there is need of composite semantic ontology. This analysis will help to create a composite or complete QoS ontology.

2.5. Performance based Service Reasoning

Performance based service reasoning can be performed by providing efficient decision support systems. A variety of decision support tools has emerged during the last decade to provide more

systematic reasoning in different domains. With the passage of time, decision support systems are integrated with advanced concepts such as data mining.

2.5.1 Decision Support Systems

The Analytic Hierarchy Process (AHP) is a Multi Attribute Decision Making (MADM) method developed by Saaty [112] in 1980. The method has broad applications widely used in different fields including engineering, business management, government, education, telecommunication, construction, health, and others. The method focuses on prioritizing selection criteria and distinguishing the more important criteria from the less important ones. AHP is made up of suitable techniques for prioritizing critical management problems [113]. The steps of calculation that are considered in AHP include Hierarchy Construction, Comparative Judgment Matrices; Normalization Procedure; and Weight Synthesis and Consistency Test [114]. It utilizes the judgments of decision makers to structure decision problems into hierarchies. That is, AHP constructs ranking of decision items utilizing comparisons or correlations between every pair of items constituted as a matrix. The matched comparisons generate weighting scores that measure the amount of significance items and criteria have with one another. Matrix algebra is then used to sort out variables to arrive at the best choice [115].

Wua et al. [116] propose Fuzzy AHP (FAHP) and adopt the three MCDM analytical tools of SAW, TOPSIS, and VIKOR to rank the banking performance and improve the gaps with three banks. First, the authors estimate the relative importance of the chosen balanced score card indices by fuzzy AHP and then they adopt MCDM tools. Büyüközkan et al. [117] propose service quality framework. They examine the concept and factors of service quality. The authors use fuzzy AHP) for structuring to evaluate the proposed service quality framework. They present a case study in healthcare sector in Turkey to clarify the methodology. Büyüközkan et al. [118] examine the electronic service quality concept and determine the key components of electronic service quality. The authors propose electronic service quality framework by using service quality (SERVQUAL) methodology as the theoretical instrument. They provide a Web service performance example of healthcare sector in Turkey by using a combined multiple criteria decision making (MCDM) methodology containing FAHP. The work presented in this paper shows the applicability of the electronic service quality framework in explaining the complexity of aspects observed in the implementation of healthcare services via Internet.

Data Envelopment Analysis (DEA) is a mathematical programming-based approach that is used to measure the relative efficiency of decision-making units which may have multiple inputs and outputs [119]. The main aim of DEA is to provide benchmarking guidelines for inefficient decision-making units. For such inefficient decision-making units, DEA identifies efficiency units, namely the reference set.

The latter constitutes its benchmark, since it provides the necessary information on how much the unit needs to be enhanced to be considered efficient [120]. Ho et al. [121] review the literature of the multi-criteria decision making approaches for supplier evaluation and selection. They analyze related articles appearing in international journals. This research provides evidence that the multi-criteria decision making approaches are better than the traditional cost-based approach.

2.5.2 Summary

There is no decision support for handling new End User requirements and reuse of services and processes. There is a lack of approach that provides decisions based on performance evaluations of one service instance as well as for the consumption of large number of services. Governance is also not used to make effective decision for service reuse.

2.6. Discussion

In this chapter, we analysed existing research works that are extracted from a systematic analysis. Based on the systematic analysis of existing research, we classified the research areas. We discussed each area of research identified in research classification and analysed them. These areas of research are broadly categorized as performance-based service reuse solutions. First area of research is service reuse. We analysed different dimensions of service reuse such as software reuse, SOA governance and service reuse. Second area of research is service performance approaches at quantitative and qualitative level. Quantitative level is analysed by evaluating existing performance monitoring techniques while qualitative analysis includes the evaluation of maturity models and risk management. We evaluated these approaches based on the performance based technical indicators. Third area of research is performance-based service structuring. This research area is important as service performance must be modelled to make efficient use of it. For this purpose, we analysed existing ontologies that are based on QoS or performance level technical indicators. Fourth area of research is performance-based service reasoning. In this area, we analysed existing decision support systems to provide efficient decisions in terms of services.

The mechanism for maximizing service reuse in the context of highly distributed SOA based applications with services developed by autonomous services providers is not fully understood at present. Reusability is regarded as a key concept in the existing work. However, the definition of formal technical indicator that can directly be used within typical SOA based modeling language is still missing.

The operation unit of traditional service composition is atomic service, and existing works seldom consider the reuse of service or process in any granularity.

From the above analysis, we propose following research directions.

- Reuse of services in terms of performance and are compliant with governance policies.
- Enhancing service portfolio in terms of performance monitoring and service policies.
- Distribution of the quality models along the time dimension.
- Maximizing performance by adding more technical indicators.
- Adaptation of new or changed business requirement.
- Evaluation of the service evolution from one maturity level to another.
- Formalization of risk constructs in the form of semantics.
- Creation of a composite and complete ontology that can evolve with time.
- Efficient decision support for the increased consumption.

CHAPTER 3. Performance Oriented Decision Support

From the state-of-the-art on performance-based service reuse solutions, we analyzed that currently complete performance based technical indicators definition that is compliant with latest international standards is still missing in service portfolio. Moreover, the perspective of new or changing business needs with time dimension is not addressed. In addition, no decision support has been provided for determining performance trends based on technical indicators with time dimension and increased consumption. Therefore, efficient decision support for reuse of atomic services or composite services in terms of performance and resource utilization remains a big challenge. This analysis highlighted the need of fully automatic performance-based methodology for service reuse in the dynamic SOA based organization, where efficient decisions are required to be made on-the-fly. Our goal is to ensure adaptability, reusability, evolvability, agility, dynamicity and sustainability of SOA based information system.

For this purpose, we propose PODSF. PODSF maximizes service portfolio in terms of performance profile and is compliant with latest quality standards. In addition, PODSF is aided with ontology structuring and reasoning to achieve the dynamic evolution of business needs while considering time dimension. Data mining algorithms of classification and machine learning have been effectively utilized to provide efficient decision support for performance-based service reuse.

Before explaining PODSF, we show its position in an SOA based organization in Figure 11. In this figure, we highlight the interaction of SBS actors and the flow of the information from one actor to another under the form of inputs and outputs. The actors are End User, ServiceRepository, ServiceOwner and IT Department. First of all, the End User will request to use a service. If the service is available in the ServiceRepository, then the service is provided to the End User along with SLA, otherwise IT Department will start the process of developing a new service. If the service has some performance degradations, then there is a need of performance evaluation. In this scenario, PODSF is required to provide performance assessment report and recommendations. If the ServiceOwner allows to use existing services, the services will be provided to the End User, otherwise, it is required to request IT Department to create a new version of the same service and SLA. Hence, a new version of the existing service will be provided to the End User.

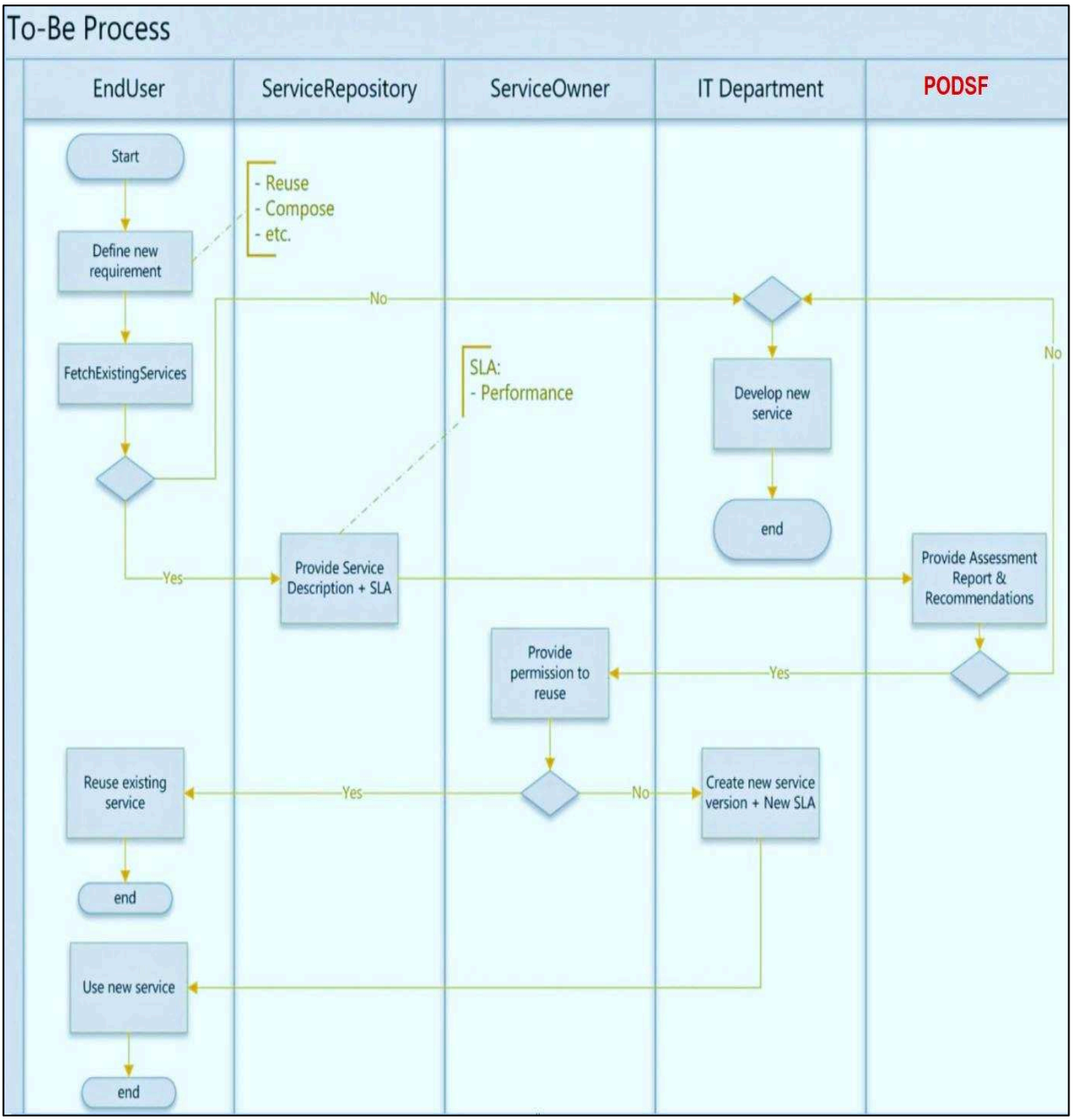


Figure 11: Position of PODSF in SOA based organization

The above discussion yields to integrate all proposed solutions together in the form of framework called PODSF. We discuss PODSF below in detail, including its technical environment, components interactions in the form of input, and output and description of components in the form of conceptual implementation.

3.1. PODSF

We propose PODSF to provide efficient decision support based on performance as shown in Figure 12. This figure is separated into two parts. The upper part of the figure shows conceptual implementation of the components while the lower part presents the s technical environment of PODSF.

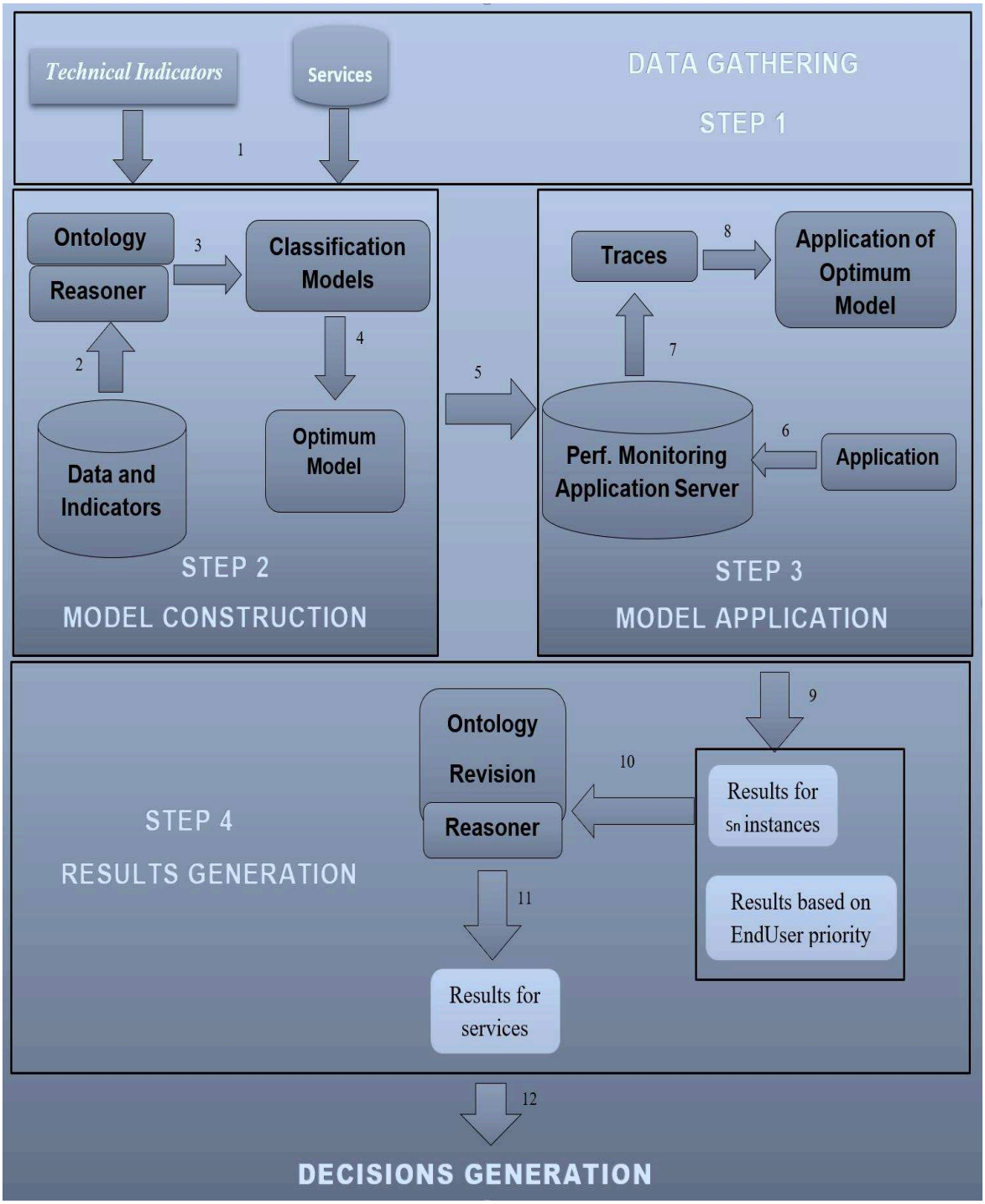


Figure 12: PODSF

PODSF works in four steps while the final step is the decision generation. The first step is data gathering. This step involves selection of services from repository and of technical indicators originated from ISO 25010 quality standard characteristics. The second step is model construction. In this step, service domain concepts and technical indicators concepts are stored in ontologies, and the reasoner generates semantic rules based on developed ontological concepts. After this, data are tuned for classification models. Classification models are applied on these data and results are generated based on the recommendation rules. As a result, the model that generates optimum results is selected for

step 3. The third step is model application. In this step, existing services are monitored in the application server and they generate traces. The selected classification model is applied on these traces and generates results for service instances. The results from both steps are aggregated and transferred to step 4. Step 4 is defined as results generation. In this step, results are instantiated in the ontologies,

and the reasoner generates results for each service based on semantic rules. Finally, decisions are generated in the form of recommendations to reuse existing service, create new service, check other repositories and check for resources.

PODSF can be seen from various entry points namely contextual, static and dynamic. Hence, we identify the elements of the environment that are required for the framework components. In addition, the framework process is detailed in terms of the interaction flow of different steps with the help of inputs and outputs. Finally, abstract execution of each component is illuminated by using an example.

3.1.1 PODSF Environment

PODS framework environment is composed of three parts. These parts are input, output and resources. The first part is the input in terms of End User requirement. The second part is the output in the form of recommendations. The third part is the resources that are used in PODSF.

A. Input

PODSF takes End User requirements as input. A sample of requirement is shown in Table 3. In this sample, we show one example of request for service and possible actions for this requirement. Possible actions include reuse existing service, create new service and check for other repositories. The End User sends a request to the IT department of a particular company for a service. If the service is performing well and if it is available, then IT Department will provide the service to the End User. If the service is not performing well, then IT Department needs performance evaluation from PODS framework.

Table 3: A Sample Requirement from an End User

Requirement	**Actions**		
Requests for a service	Reuse existing, check number of requests	Create new service	Check resources

B. Output

PODS framework provides a decision matrix of recommendations for service reuse or process reuse as output. A sample topology of recommendation is shown in Table 4. In this sample, we show different topology levels that we consider such as business, functional, applicative and technical. All of them have a particular aspect, measurement, type of answers and result. For instance, governance is related to the business aspect, call for blood is the functional aspect, while execution time is an applicative aspect. Technical aspects include availability, risk, maturity and time. The measurement mechanism for the governance of services at business level is the checking of policy compliance. At the functional level, we ask permission from the relevant organization to use the service. We measure the delay at applicative level. Measurements at technical level are in the form of assigning threshold levels. Type of answers are OK, MayBe and NO that are evaluated based on the actions such as compliant or not, available or not and others. Finally, one kind of recommendation is reuse existing service.

Table 4: A Sample Topology of Recommendation: the Call for Blood example

Topology	Aspect	Measurement	Type of Answers	Recommendation
Business	Governance	Check Policy compliance	Compliant or not	Reuse existing service
Functional	Call for blood	Allowed to use	Available	
Applicative	Execution time	Not delayed	OK	
Technical	Availability Time Risk Maturity	Threshold values	OK MayBe	

C. Resources

The main resources that we use in PODSF are the ISO 25010 quality model, repository of services, WSO2 server [122] and database of traces. ISO 25010 quality model provides both qualitative and quantitative characteristics to evaluate the performance of services. We select a list of services from the existing repository and deploy them in WSO2 server to get the traces of technical indicators. With the help of traces, the status of a service or a process can be quickly known. WSO2 server expands SBS development efforts by integrating data stores, creating composite data views, and hosting data services. With the help of this, we can easily deploy services and analyse them by integrating technical indicators. To make efficient use of WSO2 server capabilities and to exploit quality characteristics, we trail following steps.

- Deployment of services and processes in WSO2 server
- Collection of the traces at run time
- Storage of traces in the database
- Evolution of traces with timestamp
- Dynamic evolution of traces by ontology creation and semantic rules.

The above-mentioned elements of the environment plays an important role in order to execute the process flow of PODSF. Therefore, we propose steps to explain the overall process of PODSF in the below part.

3.1.2 PODSF Process

In this part, we explain the overall flow of the PODSF process as shown in Figure 13. PODSF process comprises of six major steps. The six steps of PODSF process are data management, traces management, ontology modelling, inference rules-based reasoning, analytical assessment and impact analysis. The output of the first five steps are the inputs of the following steps while the output of the last step is provided to the IT Admin who has requested for a service. The overall process of PODSF is illustrated below step by step.

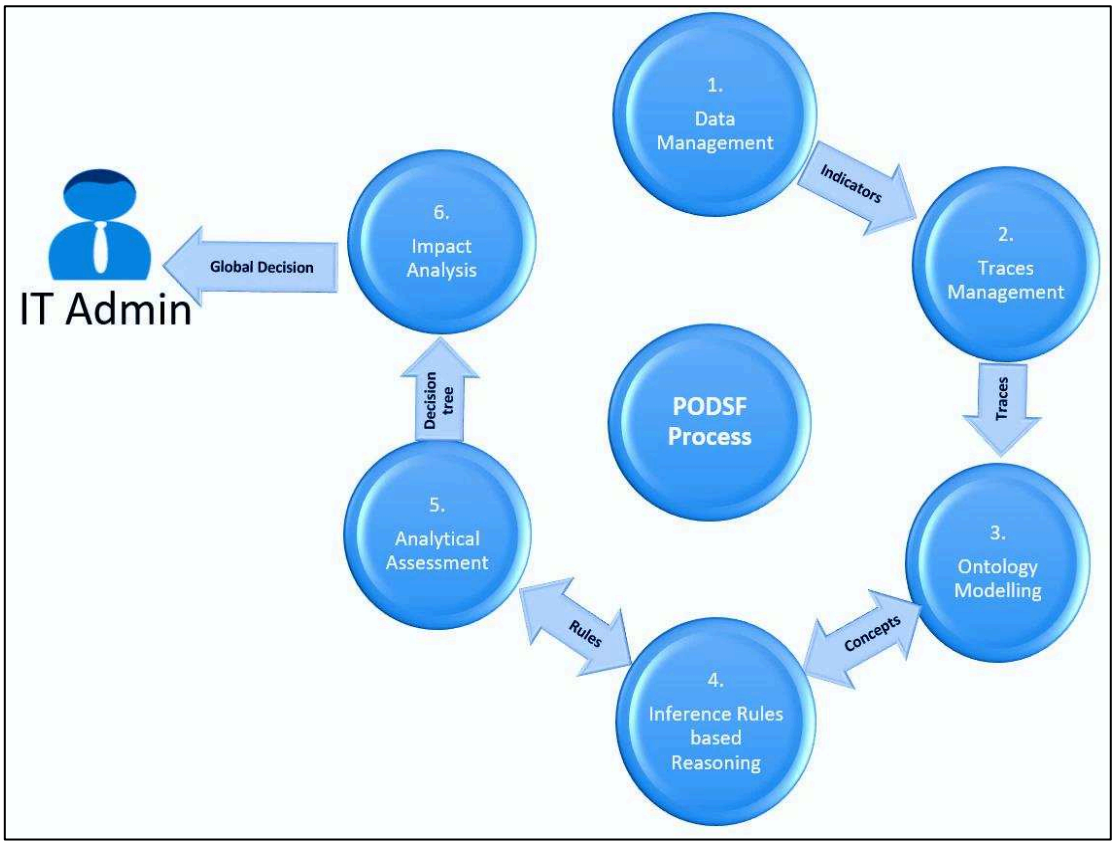


Figure 13: PODSF Process

Data management step performs monitoring and evaluation of SBS in order to guarantee performance. The ultimate goal of data management step is to ensure efficient performance and reliability of SBS. Performance efficiency and reliability are evaluated on the foundation of quality characteristics in the form of quantitative and qualitative based technical indicators. This foundation of quality characteristics leads to three different categories of data management. These three categories are performance monitoring, risk analysis and maturity analysis. Data management step provide measurement based technical indicators to the next following step of traces management.

Traces management step exploits application servers to make efficient use of them for evaluating performance trends or fluctuations of atomic or composite services. These application servers have some embedded metrics that are monitoring by this step. Metrics are mainly based on evaluating the trends and fluctuations. They are different from the technical indicators of data management step. Traces management perform tests by deploying services in the application server and analyze the performance trends and fluctuations based on the built-in metrics of application server for different SOA layers. Traces management also stores technical indicators taken from data management step to analyze performance. Traces management step stores the performance trends and fluctuation values of metrics and technical indicators in the repository of traces for the later step of ontology modelling.

Ontology modelling step creates concepts related to SBS and performance. This step develops ontology for service domain and ontologies of technical indicators at SOA layers. In addition, ontology modelling involves the creation of data properties and object properties for the developed concepts. Moreover, it includes the creation of relationships between different concepts. The dynamic nature of ontology allows it to evolve with time and changing business needs. This step stores ontological concepts, relationships, object properties and data properties and delivers them as input to the next coming step.

Inference rules-based reasoning step develops inference rules based on the ontologies. The ontological concepts and relationships created at the former step, allows the reasoner to make efficient use of them. Hence, reasoning step creates a knowledge base of inference rules in order to perform efficient decision making. This step generates the knowledge base of rules and make available to the succeeding step of analytical assessment.

Analytical assessment follows two steps to evaluate the performance of atomic and composite services. The first step is to train the data set of knowledge base while the second step involves the assessment of this training data set with respect to service instances, services and composite services. Analytical assessment applies classification algorithms of data mining. After applying all the

classification algorithms, these step selects the most optimum result. This step generates its result in the form of a decision tree which contributes as input to the later step.

So far, the performance is evaluated for priority based technical indicators that yields to specific decisions. However, recommending a service that is efficient in terms of performance requires global evaluation of performance by increasing the consumption of services. This requires performing impact analysis. This step trails two steps. The first step involves the application of selected optimum classification algorithm to evaluate the performance of all service instances, while the second step includes the evaluation of performance for each service. This step also generates results based on the cost, confidence, support and accuracy. As a result, a report is generated and deliver to the IT Admin who has requested to provide performance assessment of service.

Now, we explain the data flow of PODSF as shown in Figure 14. The upper most row of the figure shows the flow of data that begins with the event of risk. Risk analysis generates risk chart and stores in the form of ontology structures. Ontology structure stores the concepts related to risk such as risk type and mitigation action. This mitigation action is inferenced in the form of semantic rules, and specific decision for risk is evaluated by performing data analytics. The middle row of the figure starts with the event of maturity. Maturity analysis formulates maturity level chart and store this information as ontological concepts in respective ontology. Mitigation action concept is advanced by the inference engine to generate semantic rules. These semantic rules are analyzed to create specific decision for maturity. The lowest row of the figure demonstrates the event of performance monitoring. Metrics are stored in the repository and extracted by respective ontologies. Ontologies are used to generate concepts and are stored in performance measurement repository. Based on these concepts semantic rules are generated by inference engine. Specific decisions for performance are made by tuning these data. All the specific decisions are formalized by running classification algorithms. Because of these specific decisions, a knowledge base is created to further tune it for generating global decisions. For this purpose, machine learning model is used.

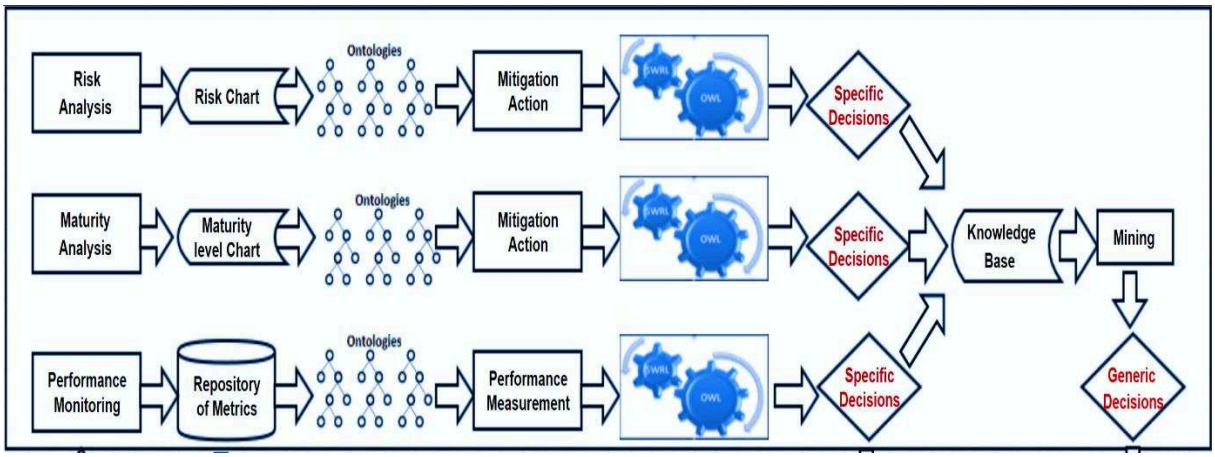


Figure 14: Data flow of PODSF

The above enlightenment of PODSF process yields to implement the functionality of the proposed steps conceptually in the form of components. Consequently, we discuss the components of PODSF in the below part.

3.1.3 PODSF Components

PODSF is made of six major components. These components are data handler, traces handler, ontology builder, reasoner, process assessment analyser and impact analyser. The interactions between the components of the framework are shown in Figure 15.

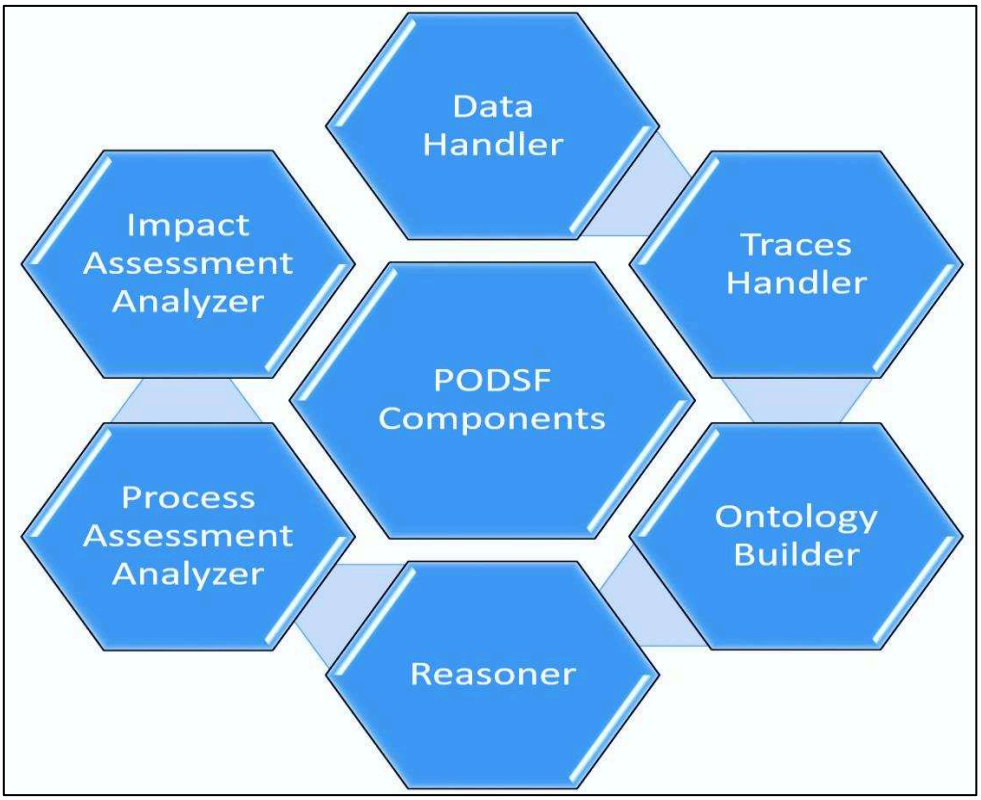


Figure 15: PODSF Components and their interaction

We explain the methodology of PODSF components and conceptual implementation of each component step by step in forth coming parts A, B, C, D, E and F. In addition, we explain the input, output, technology, models and algorithms that are used by each component.

A. Data Handler

Data handler component manages both quantitative and qualitative technical indicators. As mentioned in Chapter 1, performance can be measured, analysed and guaranteed by characteristics provided by ISO 25010. These characteristics are performance efficiency and reliability. Performance efficiency is analysed based on sub characteristics time behaviour, resource utilization and capacity. Reliability is analysed based on availability, maturity and fault tolerance in terms of risk. All of these sub characteristics are evaluated by respective technical indicators. Based on the characteristics of ISO 25010, we classify data management into three parts. These parts are performance monitoring, maturity analysis and risk analysis as shown in Figure 16. The first step of this component is performance monitoring. Performance monitoring is performed based on the quantitative technical indicators. The second step is maturity evaluation. Maturity is evaluated by using CMMI model and quantitative technical indicators. The last step is based on risk analysis. Risk is analyzed by following on existing risk management steps and measuring quantitative indicators. These steps are identifying risk, measuring the level of risk, calculating the probability percentage, making assessment and mitigating action.

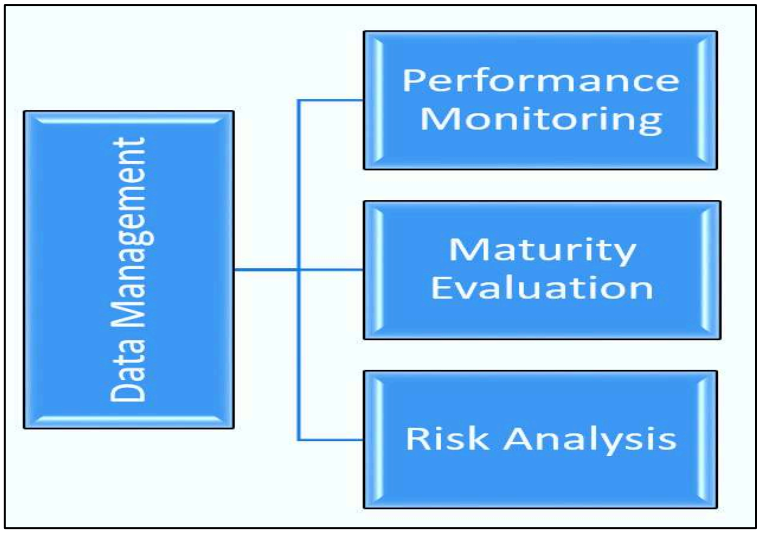


Figure 16: Data Management in PODSF

Quantitative technical indicators for performance monitoring are response time, service instance start time, service instance end time, throughput, bandwidth, availability, service down time, service up time, request count, response count, failure rate, RAM size and Mean time to recover (MTTR). Qualitative indicators are RAM type, CPU type, storage device, risk level and maturity level. Qualitative

characteristics are measured by some quantitative indicators. Quantitative indicators for maturity and risk are availability, service instance start time, service instance end time, probability, gravity and severity. Quantitative technical indicators are demarcated below in Table 5, along with their respective units and formula.

Table 5: Quantitative Technical Indicators

Technical Indicator	Unit	Formula
Response Time	Ms	Service End Time-Service Start Time
Throughput	Sec	Number of active requests processed per unit time
Bandwidth	Int per sec	Service_Instance per time unit
Availability	%	(Response_count/ (total * 100))
Delay	%	Service Deployed Time-Service Up Time
service_up_time	%	(100 - service_down_time)
Failure rate	%	Response_count-Request_count
MTTR	Seconds	service_down_time/ no_intervals
Service instance start time	minutes	service_deploy_time in minutes
Service instance end time	Seconds	Stop time in seconds

Availability is a sub characteristics of reliability that may include other quality attributes to measure the degree to which the service is operational. The formula to calculate availability is shown below.

$$\text{Availability } = (\frac{Response_{count}}{(total * 100)})$$

Throughput is a metric used to evaluate capacity sub characteristics of performance efficiency. It is calculated based on the number of active service, process and service instances in a particular time interval. The formula to calculate throughput is shown below.

$$\text{Throughput } = (\frac{no\_of\_active\_request}{(time)})$$

Delay is a metric used to evaluate time behavior. It is calculated on the basis of service, process or service instance deployed time and the time in which it is in use. The formula to calculate delay is shown below

$$\text{Delay } = (Service\ Deployed\ Time - Service\ Up\ Time)$$

Response Time is a metric used to evaluate time behavior. It is calculated on the basis of service, process and service instance ending and starting times. The formula to calculate response time shown below.

$$\text{Response Time } = (Service\ End\ Time - Service\ Start\ Time)$$

Bandwidth is a metric used to evaluate capacity sub characteristics of performance efficiency. It is calculated based on service instance used in a particular time interval. The formula to calculate bandwidth is shown below.

$$\text{Bandwidth } = \frac{(service_{instance})}{time}$$

Failure Rate is a metric used to evaluate capacity sub characteristic of performance efficiency. It is calculated based on the response and request counts. The formula to calculate failure rate is shown below.

$$\text{Failure Rate } = (Response_{count} - Request_{count})$$

MTTR is defined as mean time to recover. It is a metric used to evaluate recoverability sub characteristics of reliability. It is calculated based on service, process and service instance down time in a particular time interval. The formula to calculate MTTR is shown below.

$$\text{MTTR } = (\frac{service_{down_{time}}}{no_{intervals}})$$

Service Time is a metric used to evaluate time behavior sub characteristics of performance efficiency. It is calculated based on service, process and service instance start and end times. The formula to calculate service time is shown below.

$$\text{Service Time } = (service_{start_{time}} + service_{end_{time}})$$

Total number of service instances is calculated by adding all number of instances. The formula to calculate the total number of instances is shown below.

$$TotalNumberOfServiceInstances = \sum_{k=1}^{n} ServiceInstace\ k$$

Now, we explain qualitative indicators that are available memory, risk and maturity. Available Memory is a metric used to evaluate resource utilization in terms of time behavior. It is calculated on the basis of used and available memory. The formula to calculate available memory is shown below.

$$\text{Available Memory } = (Memory\_Usage - Memory\_Allocated)$$

Risk is defined as the probability of occurrence of a threat at run time, that will have a negative impact on the system. Risk level is measured based on risk gravity. Formula to calculate risk gravity is shown below.

$$\% \text{ Risk Gravity} = Probability \times Severity$$

Where

Probability = (Number of Failures) / (Number of Executions)

Severity of service = n, where n is number of operations

Severity of process = 1/no of services

Severity n = 1 / n

With n = number of services or operations

Maturity is a qualitative technical indicator used to evaluate reliability. It may include other quality attributes to measure the degree to which the service is operational. Maturity is evaluated through the CMMI method which defines the level of service or process control at a particular instance. We consider that a mature process is generally one that moves from an unstable state to a stable state. A higher level of maturity in the service or process will result in better control of results, improved goal prediction, and greater effectiveness in achieving goals. We evaluate maturity based on the availability of a service or a process using CMMI levels. We choose the CMMI levels because of the efficient method for assessment and evaluation of the service or process. The formula to calculate maturity at each level is shown below.

$$Maturity\ level\ 0: service - instance - time = 0$$

$$Maturity\ level\ 1: service - instance - time >= 0.00000001ms$$

$$Maturity\ level\ 2: if\ availability\ >= 98.999$$

$$Maturity\ level\ 3: if\ availability >= 99.99$$

$$Maturity\ level\ 4: if\ availability = 100$$

Data analyzer component provided the measurement formulas for technical indicators in order to evaluate performance of SBS. These measurement formulas help to monitor performance, analyze risk and maturity. Furthermore, this component stored the information of technical indicators, their units and corresponding measurement formulas in database. This information is important for the later component of traces handler in order to perform tests in real time.

B. Traces Handler

Traces handler provides status report of services and processes at run time. Traces are helpful for information systems to extract and analyze a set of knowledge that helps in decision making. This component stores execution traces from WSO2 server in PODSF. It uses the result of the execution traces as input to ontologies. Existing methods are generally static and do not analyze the performance

of a service and process with different time stamps. Traces are stored in the database during different time stamps. These data help the framework to analyze performance degradations at run time.

We study the medical domain where any hospital or medical organization wants to look-up an optimal solution via selecting several Web services from the Web service repository. Consider a small example of an urgent medical patient who arrived at a hospital. There are several precautionary and immediate service measures for the hospital management, such as call for blood, immediate doctor consultation, arrangement of necessary products and equipment's, registering patient, providing him a room according to his/her severity and more. To manage all those aspects, we deploy services in WSO2 server to get the traces. Following is the detail of each of the Web services involved.

1. Call_for_Blood: This service gets the blood group of a patient, matches blood group in the repository of blood bank, and returns delivery date time of available bloods with their quantity.
2. Check_Doctor_Availability: This service gets patient complications and specialty of a doctor as an input, matches with the list of staff members of a hospital, and checks their available time slots. Finally, it returns unique id of a doctor with its available timings.
3. GetSupplier: This service gets a list of products with their specification, checks the name of suppliers who can provide these products. Finally, it returns supplier id, delivery time and cost of each of products.
4. Find_Room_Availability: This service gets ward number and type of room as an input, matches the room requirements with the available rooms. Finally, it returns the room number as a response.

We have performed different tests and analyze performance of service, service operation and server. Figure 17 illustrates GetSupplier service information in WSO2 server. The upper part of the figure shows the statistics of the service while the lower part displays configuration for QoS. Statistics of services represent different attributes of this service, such as name, description, service group name, deployment scope, service type, service deployed time and service up time. Configuration of QoS demonstrates the status of the service, security, reliable messaging, response caching, policies, transports, modules, operations and configuration parameters.

Figure 17: GetSupplier Service Information in WSO2 server
(Upper part: Service Statistics. Lower part: QoS Configuration)

We perform a first test to evaluate the performance of one operation of GetSupplier service. Figure 18 shows information about GetSupplier service operation in WSO2 server. The upper part of the figure provides the information and statistics of the operation while the lower part displays KPI analysis graph. Information of the operation includes WSDL file versions and the endpoints used to perform this test. Statistics of operation represent different KPI analyzed in this test and their values. This test is performed to evaluate the performance efficiency in terms of time frequency and faults. For this purpose, three KPI's and three metrics are analyzed. The three KPI's are request count, response time, and fault count while three metrics are maximum response time, minimum response time and average response time. KPI analysis graph shows the average response time of the service in different time intervals. The values of average response time and time instance correspond to the unit of milliseconds. The analysis shows that no fault is encountered during this test.

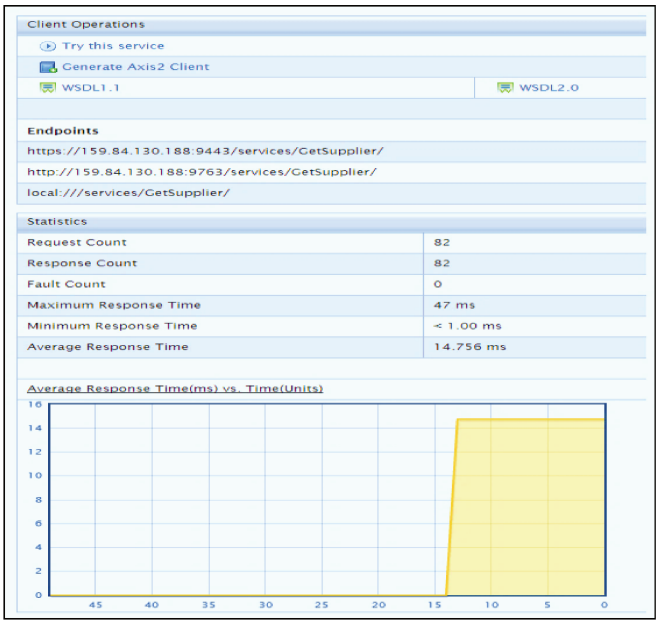


Figure 18: GetSupplier Service Operation in WSO2 server
(Upper part: Operation Statistics and Lower part: KPI Analysis Graph)

Similarly, we have analyzed server level performance in the second test. Figure 19 illustrates the server information in WSO2 server. The upper part of the figure shows the information of host and statistics of server while lower part displays KPI analysis graph. Statistics of server represent different KPI analyzed in this test and their values. This test is performed to evaluate the resource allocation in terms of memory with time frequency. For this purpose, four KPI's are scrutinized. The four KPI's are server start time, server up time, memory allocated and memory usage. KPI analysis graph shows the distribution of allocated and used memory in a particular time frame. The blue part represents the memory allocated while orange part shows the used memory. The values of memory are evaluated in megabyte (MB) unit while time instance correspond to the unit of milliseconds.

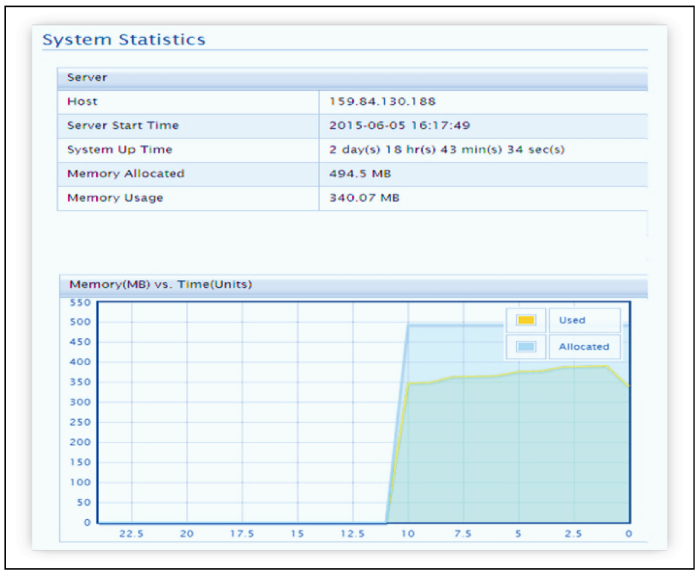


Figure 19: Server information in WSO2 server
(Upper part: Server Statistics. Lower part: KPI Analysis Graph)

The last test demonstrates the performance of GetSupplier service as a whole. Figure 20 displays the summary of GetSupplier service information in WSO2 server. The upper part of the figure provides the statistics of KPI's at service level while lower part shows KPI analysis graph. Statistics of the service represent different KPI analyzed in this test and their values. This test is performed to evaluate the performance efficiency in terms of time frequency and faults. For this purpose, one KPI and six metrics are analyzed. The one KPI is active number of services. The six metrics are average response time, minimum response time, maximum response time, total request count, total response time and total fault count. KPI analysis graph shows the average response time of the service in particular time frame. The values of average response time and time instance correspond to the unit of milliseconds. The analysis shows that no fault is encountered during this test.

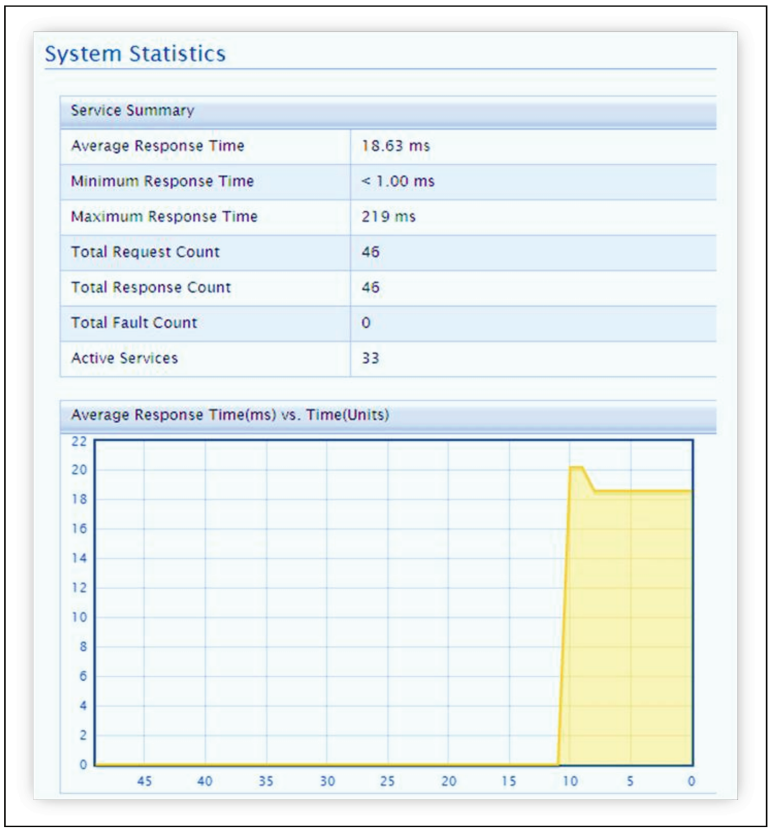


Figure 20: GetSupplier Service Summary using WSO2 server (Upper part: KPI Statistics. Lower part: KPI Analysis Graph)

However, WSO2 server do not support all the technical indicators that are compliant to latest quality characteristics necessary for the evaluation of performance. For this purpose, we develop ontologies to generate semantic rules for the evaluation of performance based latest quality characteristics.

Traces analyzer component evaluated the performance trends and fluctuations based on built in metrics of WSO2 server as well as incarnation of technical indicators originated from the data analyzer component. Furthermore, this component stored the values of metrics and technical indicators

originated from the tests in the database to make it available for the next component of ontology builder.

C. Ontology Builder

Ontology builder generates ontological structure in Protégè [123]. Protégè structure permits the addition of plug-ins to interact with other tools. Its graphical capabilities can draw concept trees and ontology flow diagrams that help in the visualization and generates OWL code automatically. Ontologies provide a formal description of properties, concepts, and their relationships in a coherent way. Another important step is the creation of parent child classes. Ontology has two views. The first view is the static view of domain knowledge. It is usually represented or created in the form of classes. This view is commonly referred as Tbox. The second view is the dynamic view of ontology instantiation. This view is commonly referred as Abox. This is usually represented by assertion knowledge.

Ontologies are able to evolve with different time stamps. Knowledge base containing ontology concepts and instances can evolve with different time stamp as well. Therefore, it is possible to define or cumulate new concepts. However, it is important to verify that the ontology classes, concepts, relationships and properties are correctly defined and are consistent with the basic rules of ontology creation. For this purpose, ontology reasoner is built in Protégè to check the consistency of ontologies. Reasoner inspects consistency by class checking, verifying relationships and eliminating conflicting definitions. Ontology builder uses reasoner of Protégè to verify its consistency to avoid all types of ontological error.

Ontology builder generates an ontology for service domain and ontologies of performance at SOA layers. Methodology to create performance profie from *n* ontologies is shown in Figure 21. Performance based ontologies are developed at process defined as composite service, services, integration and governance layers of SOA. These ontologies stores the concepts, properties and relationships related to the technical indicators of the corresponding layers. Ontology builder generates a performance profile of these ontological concepts, properties and relationships. They can also evolve with time frequency.

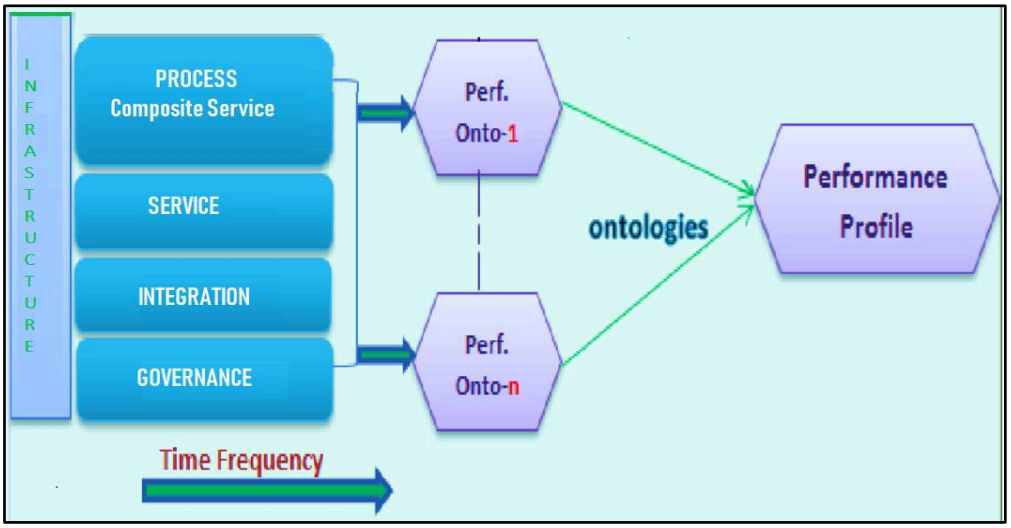


Figure 21: Methodology to Create Performance Profile

As an example, we explain the structure of the service ontology (SOnt), and we show how it stores the traces of events gathered from the execution of services in WSO2 server. The domain owl classes of proposed SOnt are represented in Figure 22. This ontology has already been published [124]. SOnt facts base comprises instances of some SOnt classes and values from statistics of WSO2 application server. However, performance-based ontologies at relevant SOA layers and a composite ontology for performance-oriented decision support, will be implemented in implementation chapter. Service provider, service consumer and service host are the concepts that have already been used in the literature. Technical indicators like response time and delay are also used [125]. Further, we classify different performance levels. Each of the service and its operations are monitored. Performance level is a level at which a service network can be monitored. Performance level has various sub concepts to monitor the performance. These sub concepts are binding Level, service level, business process level and server level. Technical Indicators concept records the value and description of each of the performance based technical indicators. Each of the service and its operations are monitored. Therefore, performance indicators are recorded for both service as a whole and its operations. The various performance based technical indicators that we use are described hereafter. Response_Time is the response time of a service or operation. It has three sub concepts to record, Maximum, Minimum and Average. Request_Count shows the number of invocations of a service. Response_Count is the number of replies for an invocation of a service. Fault_Count is the number of invocations the service has not replied to. Deploy_Time is the time at which the service has been deployed on the server. Up_Time is the time period the service is available since its deployment. Down_Time is the time period of un-availability of a service since its deployment. Delay is the average response time of a service. Loss express the unavailability of the service so that it cannot be invoked.

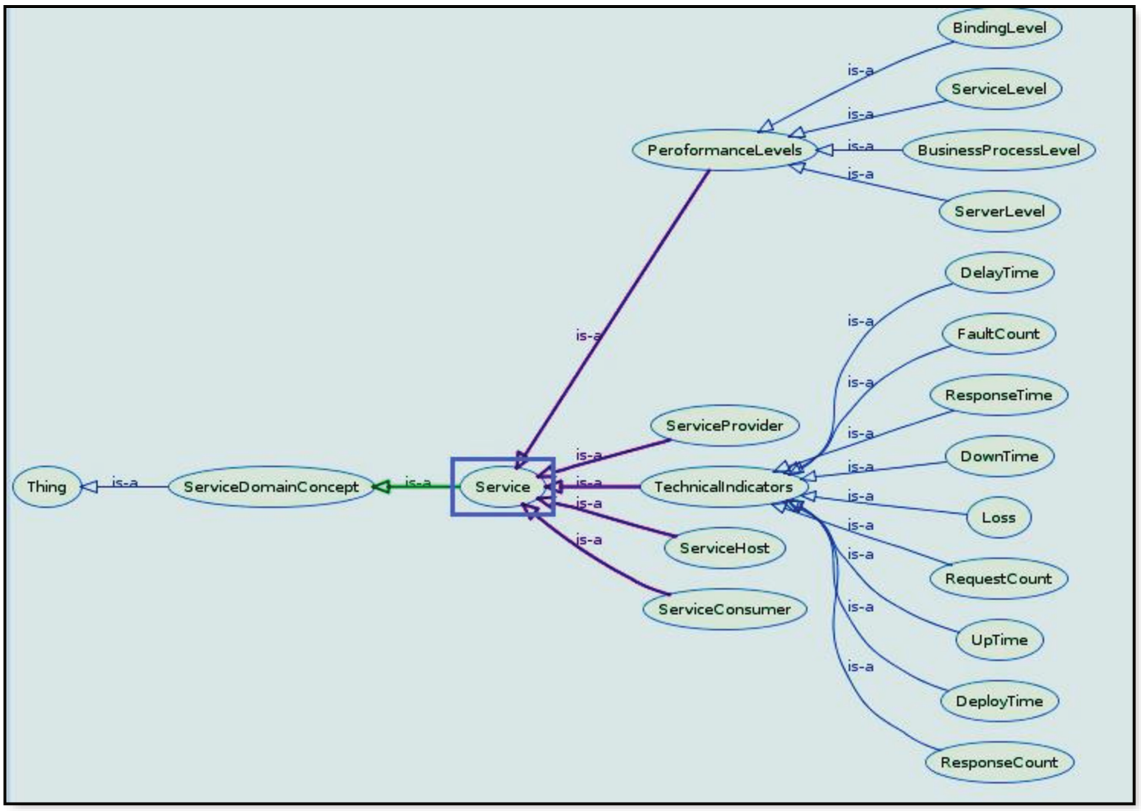


Figure 22: SOnt Ontology based on Technical Indicators of WSO2 Server

We develop relationships between different concepts of SOnt by object properties as shown in Table 6.

Table 6: Object Properties of SOnt

Concepts	Relationships
Service	Consumed by
Service consumer	Consumes
Service	Has Indicators
Service	Hosted By
Service host	Host
Performance Levels	Measured At
Service	Measures
Service Provider	Provides
Service	Provided By

The description of the above-mentioned object properties that links service concept to other concepts is presented in Table 7.

Table 7: Description of Object Properties of SOnt

Object Properties Description	
Consumed_By	Captures the relationship of each service consumed by its consumers.
Has_Attributes	Each service has some attributes (Estimated_Attributes). It is linked to the attributes by this property.
Has_Indicators	The performance of each service is monitored via various KPI, and this property links service with KPI concept.
Has_Performance_Level	Makes the relation between service concept and the performance level.
Hosted_By	It captures the relation between each service and its host.
Measured_At	Each service is monitored and performance is measured at the performance level (Performance_Level).
Measured_By	Each service QoS is measured by the QoS metrics (QoS_Metrics) via this property.
Provided_By	Captures the relation of each service provided by its provider.

A service concept has some datatype properties to capture different attributes in the SOnt as shown in Table 8.

Table 8: Description of Datatype Properties of SOnt

Datatype Properties Description	
Name	Records the name of the service.
Description	Captures the description about the service.
Group_name	Records the Name of the group to which the service belongs.
Deployment_Scope	Captures the deployment scope of the service.

By the help of these concepts and relationships, reasoner component will generate semantic rules to infer new knowledge. These rules are identified in relative to the diverse usage scenarios. Dynamic nature of ontology allows to integrate or aggregate new concepts and can evolve with time. As a result, semantic rules can also evolve.

D. Reasoner

Reasoner component generates and publishes inference rules in SWRL. It uses SPARQL queries to support the decision making. Ontologies support the decision through interaction via queries. This component uses ontological concepts values and generate inference rules. It customs aggregated QoS functions like divide and difference, that can be easily defined and managed. These rules help to detect performance degradations. Since we create inference rules by using ontological concepts, therefore inference rules can also evolve with different time stamps and with the new requirements. We create a knowledge base of inference rules. For instance, we present inference rules based on the ontological

concepts defined for technical indicators at WSO2 server in. Figure 23. Note that these inference rules have already been published [126]. We will expand this knowledge base in the implementation section.

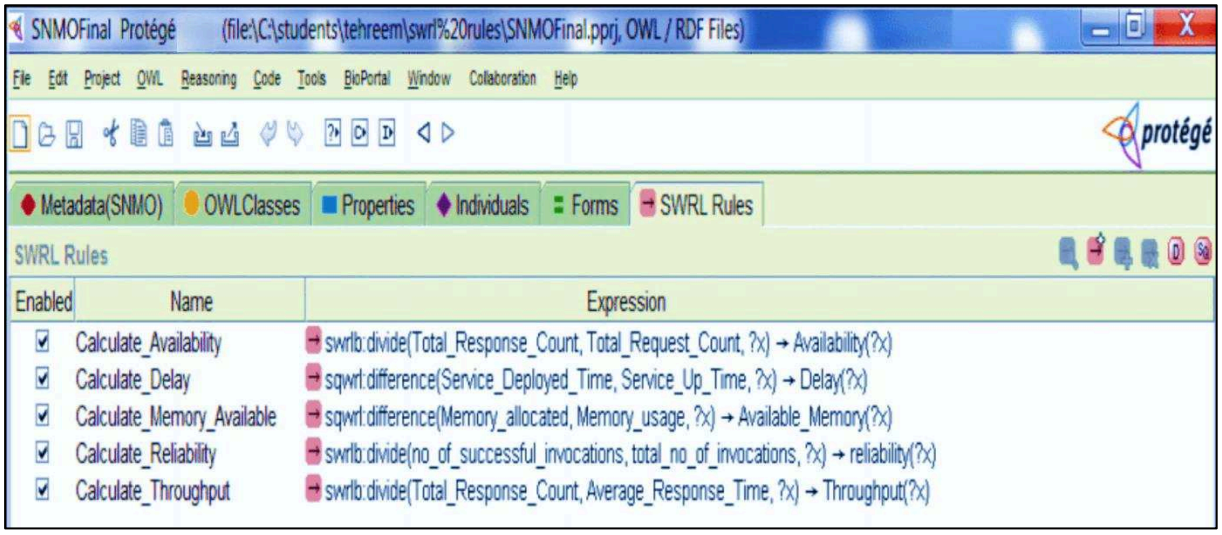


Figure 23: Inference Rules using SWRL in Protégé

We show the inference rules that use the concepts of SOnt ontology, and that are mainly based on WSO2 server metrics. Those SOnt concepts are total response count, total request count, service deployed time, service up time, memory allocated, memory usage, no of successful invocations, total no of invocations, total response count and average response time. Inference rules are generated for the computing of availability, delay, memory available, reliability and throughput. Following is the description of these rules. Availability is calculated as the total response count of a service or process divided by total request count of a service or process. If the requested service is available, then we can suggest reusing this service. Delay is measured as the difference of service_deployed_time and service_up_time. Memory available is measured as the difference of memory usage and memory allocated. Reliability is calculated as no_of_successful_invocations divided by total_no_of_invocations. Throughput is calculated as the total_response_count divided by average_response_time.

Reasoner component provided the knowledge base of semantic inference rules. This knowledge base is useful to evaluate performance of services and helps to provide decision support in dynamic environments. Furthermore, this component stored this knowledge base so that the next component of process assessment analyzer can retrieve it for data discovery and data analysis.

E. Assessment Analyzer

Assessment analyzer component encompasses two steps. The two steps are data discovery and data analysis. Data discovery step involves the discovery of inference rules from database and the training of these data. Data analysis step evaluates the deployed services and processes on the basis of this

training set. A sample of training set created by data discovery step is shown in Table 9. This table presents for two quality characteristics, the sub characteristics, the type of technical indicators, either quantitative or qualitative, and the corresponding measurement formulas.

Table 9: Performance Measurement

ISO/IEC Quality Characteristics	ISO/IEC sub-Quality Characteristics	Type of Technical Indicators		Measurement Formula
		Quantitative	**Qualitative**	
Performance Efficiency	Time behavior	Response time		(Average_response_time= (service_instance_end_time) - (service_instance_start_time))
				Service_deploy_time in minutes
				Stop time in seconds
	Resource utilization		RAM Size/Type, CPU	
	Capacity	Throughput		Number of active requests processed per unit time
		Bandwidth		Service_Instance per time unit
			Max CPU load	
			Storage device	
Reliability	Availability			(Response_count/ (request_count * 100))
		Requests count		Number of requested
		Response count		Number of answered
		Failure rate		Response_count-Request_count
		MTTR		service_down_time/ no_intervals
	Risk		Risk level	Risk Chart
	Maturity		Maturity level	Maturity Level Chart

Let's come back to the hospital example. To be able to manage the patient, the hospital management service has some requirements for each one of the implied Web services. Those requirements are expressed under the form of inputs and outputs. Table 10 shows the services' profile of the hospital management. This figure illustrates inputs and outputs of the corresponding Web service.

Table 10: Services Profile for Hospital Management

Service Profile	Input	Output
Call_for_Blood	Blood Group	Quantity, Delivery Date Time
Check_Doctor_Availability	Patient complications, specialty	Doctor UID, Available timings
GetSupplier	List of products	Supplier id, Delivery Time, cost
Find_Room_Availability	Ward, Type	Room Number

Data analysis step deploys test services based on the training set. Based on the input, output and QoS values of hospital management, data are analyzed and stored in the database. Classification algorithms are applied in order to make decisions based on these data. A sample of specific decisions is shown in Table 11. For instance, we list five technical indicators that are risk level, maturity, failure rate, service time and available memory. This scenario evaluates specific decisions based on risk level. Service is recommended, if the risk level is Very Good and Good. It is not recommended, if the risk level is Bad. Specific decisions are made based on the threshold values as shown in table. In this scenario, we define three threshold values that categorize risk level. Risk level is considered as good, if the threshold value is 0.01120. Risk level is defined as very good, if the threshold value is 0.00134 while it is evaluated as bad, if the threshold value is 0.02380. Maturity is calculated as good, if it is at the 3rd level of CMMI while failure rate is defined as good, if the threshold value is 0.05. Service time is evaluated as good, if the threshold value is 0.01 while available memory is defined as good if it is 0.05. We only explain few technical indicators in this scenario however, we will explain the details in the implementation section.

Table 11: A Sample of Specific Decision

Technical Indicators	Specific Decision	Threshold Value
Risk_level	Good	0.01120
Risk_level	Very Good	0.00134
Risk_level	Bad	0.02380
Maturity	Good	3
Failure_Rate	Good	0.05
Service_Time	Good	0.01
Available_Memory	Good	0.05

Assessment analyzer component provided specific decisions in the form of recommendations. These specific decisions are provided to the IT Admin on the basis of their priority of technical indicators. Furthermore, this component generates decision tree of specific decisions related to corresponding technical indicators. This decision tree is provided as input to the next component, impact analyzer. Impact analyzer will evaluate a global decision by using generated decision trees.

F. Impact Analyzer

Impact analyzer component evaluates the decision in two steps. The first step involves the application of optimum algorithm on the traces generated in traces handler part above and generates results based on service instances. The second step instantiates the ontologies and rules to generate results for each service, based on the results of step one. It is also important to evaluate or analyze global performance of services by increasing the number of services. With the help of this, impact analyzer component generates performance trends for the generic decisions. As a result of this approach, we provide a decision matrix as output. A sample of first level results based on service instances is shown in Table 12.

Table 12: First Level Results For Service Instances

Technical Indicators	Level one Result	Threshold Value
Risk_level_service_instance_1	Good	0.01120
Risk_level_ service_ instance_2	Good	0.01120
Risk_level _service_ instance_3	Bad	0.02380
Maturity_level _service_ instance_3	Very Good	4
Maturity_level _service_ instance_3	Very Good	4
Maturity_level _service_ instance_3	Bad	1

First level results for service instances provide results based on all instances for all technical indicators. We show a sample result for risk level and maturity level evaluation for service instance 1, 2 and 3. We remark that two results are marked as Good while one is marked as Bad, based on the threshold value defined for risk. Similarly, maturity level is evaluated as Very Good for two instances while it is marked as Bad for one instance. These results vary according to the threshold values defined for each technical indicator.

After generation of results for all the instances of the services, results are calculated for each service. A sample of second level results for service recommendation is shown in Table 13. In this table, we calculate the most repeated result among all instances. Based on the illustration of Table 12, risk level for service is recommended as Good as it is repeated two times. We observe the same configuration for maturity level which is recommended as Very Good based on its repetition of two times. We complete this evaluation for all technical indicators and generate a global performance decision.

Table 13: Second Level Results for Service Recommendation

Technical Indicators	Level two Result	Threshold Value
Risk_level_Service	Good	Choose the most repeated result among all instances
Maturity_level _Service	Very Good	Choose the most repeated result among all instances

After evaluating results for services, we estimate the cost, predictive confidence and accuracy of the performance evaluations. Finally, recommendations have been made in terms of reuse atomic service, composite service and resource allocation. We do not provide any new mechanisms for governance policy creation nor define any new policies. However, the proposed approach takes into account existing governance policies while using services. It also elaborates enforcement strategies that PODSF will follow for the performance evaluations in terms of resource utilization. We will explain the details in the next coming chapters dedicated to implementation and evaluation.

3.2. Discussion

In this chapter, we discussed the proposed framework, PODSF, dedicated for effective decision support in terms of performance-based service reuse. PODS framework exploits ISO 25010 performance-based characteristics at SOA layers. The different components of PODS framework are presented in this chapter. We have explained PODS framework with the help of an example related to hospital management domain. We have provided mechanisms to evaluate performance, risk and maturity. We have proposed performance-based ontologies for SOA layers and developed inference rules from these ontologies. We have analyzed the traces of performance based technical indicators on services deployed in WSO2 sever. As a result, performance profile is provided as input to the analytical assessment component. Analytical assessment component has deployed services, executed performance profile, analyzed decisions by using classification algorithms and selected optimum decision. This component provides decision chart as output which becomes the input of impact analysis component. Impact analysis component has analyzed the overall performance by increasing services load to evaluate a global decision for service reuse. This component provides decision in the form of recommendation. In the next chapter, we will discuss the implementation of PODS framework. With the help of PODSF, we perform monitoring of service, process and integration layers.

CHAPTER 4. Implementation of PODSF

This chapter is dedicated to the implementation of PODSF. It brings together the integration of all the components of the framework to build a performance-oriented decision support system. The main functionality is to implement DS Framework automatically to provide efficient decision support to recommend the most optimum service in terms of performance and compliance to its governance policy.

The PODSF implementation is discussed in detail by including its three phases. The three phases are PODS research design, PODS research prototype and PODS system. PODS research design phase is aided with the implementation of PODS ontology structuring and PODS reasoning. PODS research prototype phase is dedicated to the implementation of the algorithms of the system. PODS system phase is committed to the automation of PODS research prototype.

PODS research design phase is composed of two parts. These two parts are PODS ontology structuring and PODS reasoning. PODS ontology structuring discusses the implementation of service domain ontology and performance-based ontologies. Performance based ontologies are implemented for process, integration and governance layers of SOA. Finally, a composite performance-oriented decision support ontology is implemented to generate a performance profile and to provide decision support. This ontology aggregates proposed ontologies and extends the performance and decision support concepts at service layer. PODS reasoning generates a knowledge base of semantic inference rules by using the ontological concepts mentioned in PODS ontology structuring phase. This knowledge base is used to generate decisions in different iterations.

PODS research porotype phase is composed of two parts. These two parts are data management and decision support. Data management part is supported by the implementation of three algorithms. These three algorithms are performance management algorithm, maturity evaluation algorithm and risk management algorithm. Decision support part is aided by the implementation of three algorithms. These three algorithms are performance evaluation for specific decision algorithm, impact analysis of new consumption algorithm and decision evaluation algorithm.

PODS system phase is organized into two parts. These two parts are high level architecture of PODS system and PODS system classes. High level architecture of PODS provides the architecture detail of

the automation process. PODS system classes discuss the implementation of all the classes to build the system.

We hence start this chapter by the PODS research design phase. We then present the PODS research prototype. We explain the PODS system in the last part. We conclude the chapter by a discussion.

4.1. PODS Research Design

Research design traces the complete methodology that is used to integrate the different components of the framework in a rational and coherent way. In this way, the research problem is effectively addressed. The most important criterion is that the research design must be appropriate for testing the particular hypothesis of the study. This research study uses quantitative approach. A quantitative approach develops the knowledge and inquires the strategies on the basis of experiments or tests.

In this part, we explain the research design of the PODS by highlighting the interaction of internal and external actors. PODS research design is composed of two parts. These two parts are PODS ontology structuring and PODS reasoning. PODS ontology structuring is supported by different ontologies implemented at the domain level as well as for SOA layers performance. PODS reasoning is aided with the implementation of semantic inference rules. We show and explain the implementation of these two parts below.

4.1.1 PODS Ontology Structuring

Ontology builder generates ontologies based on a set of basic concepts containing several instances. This promotes the reuse of ontology knowledge in many use cases. We have developed ontologies in Protégè and used DL Reasoner to verify their consistency. The verification avoids all types of ontological error. Protégè supports the definition of classes, sub-classes, the relationships between them, their individual properties and instantiation. SOA has nine layers that have already been defined in the introduction chapter. SOA layers ontological concepts are shown in Figure 24. This ontology defines concepts for consumer, component, service, process, QoS, operation, integration, governance, information architecture layers.

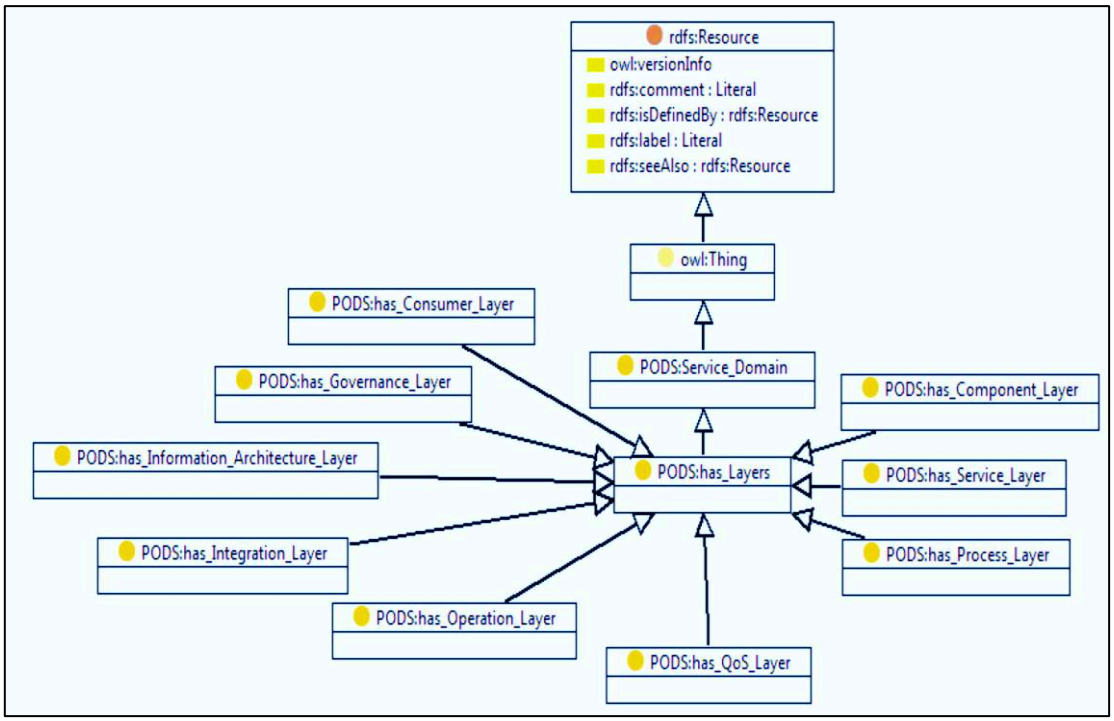


Figure 24: SOA Layers Ontological Concepts

In order to maximize service reuse along service lifecycle, we develop performance and decision-based ontologies. For this purpose, we target layers that are related to performance and service reuse concepts for decision support. These layers are service, process, integration and governance layer.

A. Performance Ontologies for SOA layers

We now explain the ontological concepts of the performance at process layer, integration layer governance layer and service layer. Ontologies for all these layers are shown below step by step. We start with ontology of performance at process layer. Note that process is a combination of two or more atomic services as a composite service to complete the desired functionality requested by the consumer. Therefore, the performance of process relies on the underlying services that are composed to have a process. So, the value for the technical indicator at process level is calculated based on all the atomic services that are involved in the process. The work on ontology of performance at process layer has already been published [127]. Figure 25 shows ontology of performance at process layer. The six technical indicators at process layer are explained below.

- **Process-Response-Time**: captures the response time of a composite service. It evaluates response time of atomic service that forms the composite service and measure the response time as a whole. We evaluate composite service response time by calculating the average response time of all underlying atomic services. It has three sub concepts to record Maximum, Minimum and Average response time.

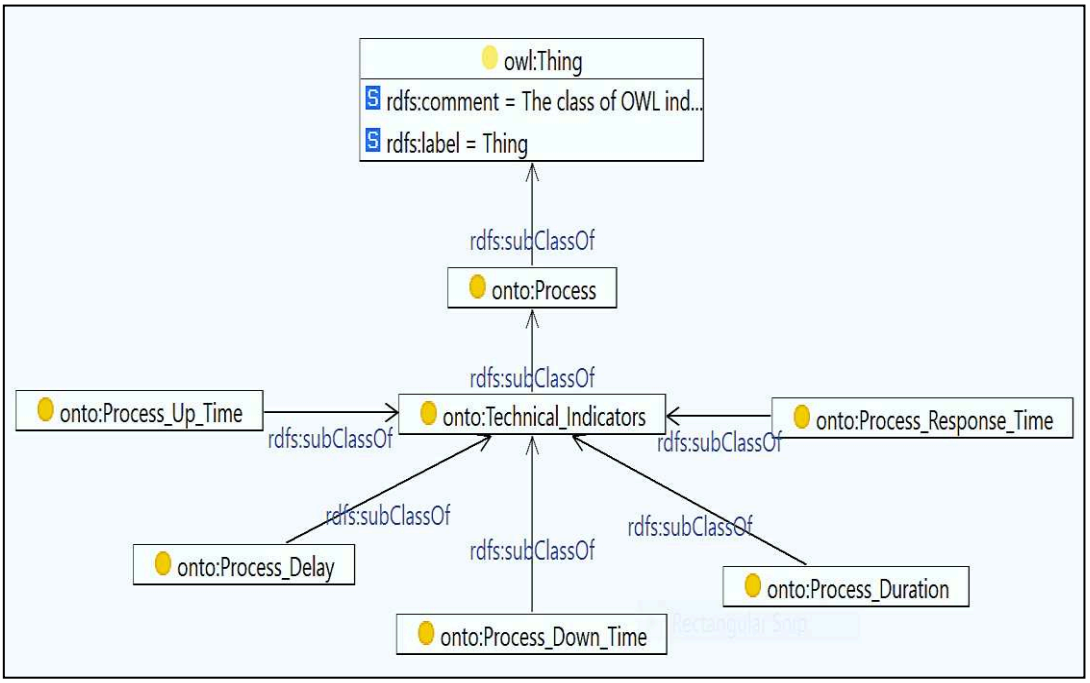


Figure 25: Ontology of Performance at Process Layer

- **Process-Up-Time:** illustrates the time period the composite service is available since its deployment. We evaluate composite service up time by calculating the average up time of all underlying atomic services. It has three sub concepts to record Maximum, Minimum and Average Up time.

- **Process-Down-Time:** stores the time period of un-availability of a composite service. We evaluate composite service down time by calculating the average down time of all underlying atomic services. It has three sub concepts to record Maximum, Minimum and Average down time.

- **Process-Delay:** shows the delay of a composite service. Delay is calculated based on the up time and down time of process as explained above.

- **Process-Loss:** specifies that the composite service is un-available (i.e., it cannot be invoked). If any of the atomic service of the process is not available, then it means that whole process cannot function properly without alternate service allocation.

- **Process-Duration:** expresses the time duration of a composite service since it is deployed, executed and remained in process. We evaluate composite service duration by calculating the average duration of all underlying atomic services. It has three sub concepts to record Maximum, Minimum and Average duration.

After the process layer, we now focus on the ontology at integration layer. Integration layer means communication at the messaging level. This ontology is represented in Figure 26. Technical indicators concepts at integration layer are hasThroughput, hasReliableMessaging, hasBandwidth and hasResponseTime. We have already published this ontology of performance at integration layer in a book chapter [128]. These technical indicators are explained below.

- **Throughput:** shows the throughput handled by protocol. It is calculated by the average response time of the protocol by number of requests.
- **Protocol-Response-Time:** captures the response time of the underlying protocol used for the transfer of messages. It has three sub concepts to record Maximum, Minimum and Average response time.
- **Bandwidth:** represents the bandwidth handled by protocol. It is measured by calculating the number of instances handled by protocol in time unit.
- **Binding-Reliable-Messaging:** Messaging through protocol is measured by necessary message transformation to connect the service requestor to the service provider.

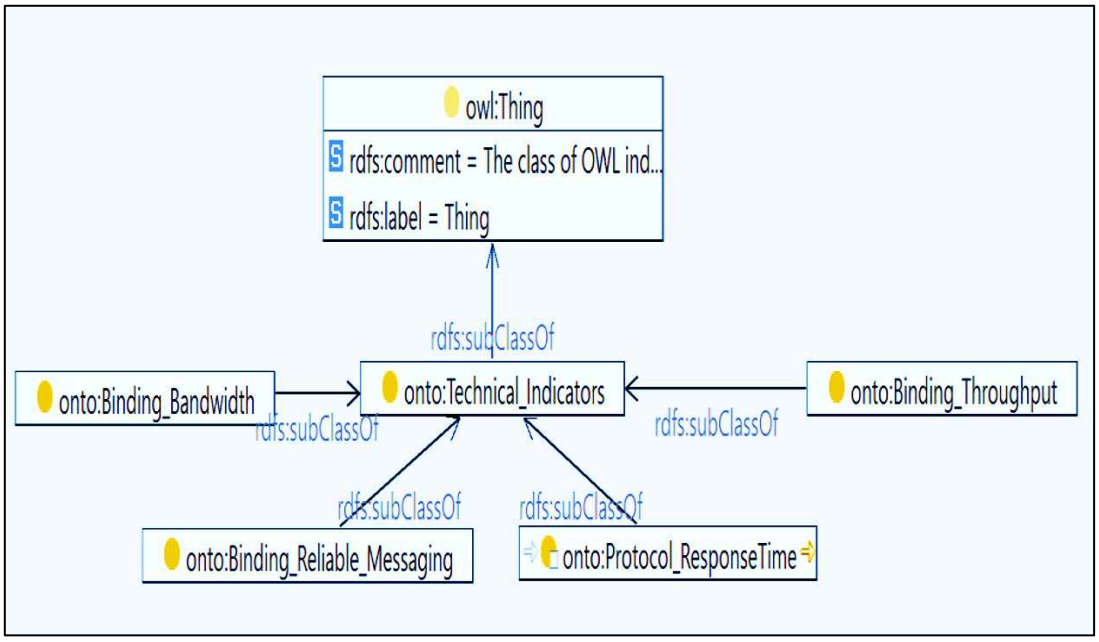


Figure 26: Ontology of Performance at Integration Layer.

The ontology of governance layer is shown in Figure 27. Governance ontology defines concepts as hasPolicy, hasSLA and hasServicePortfolio.

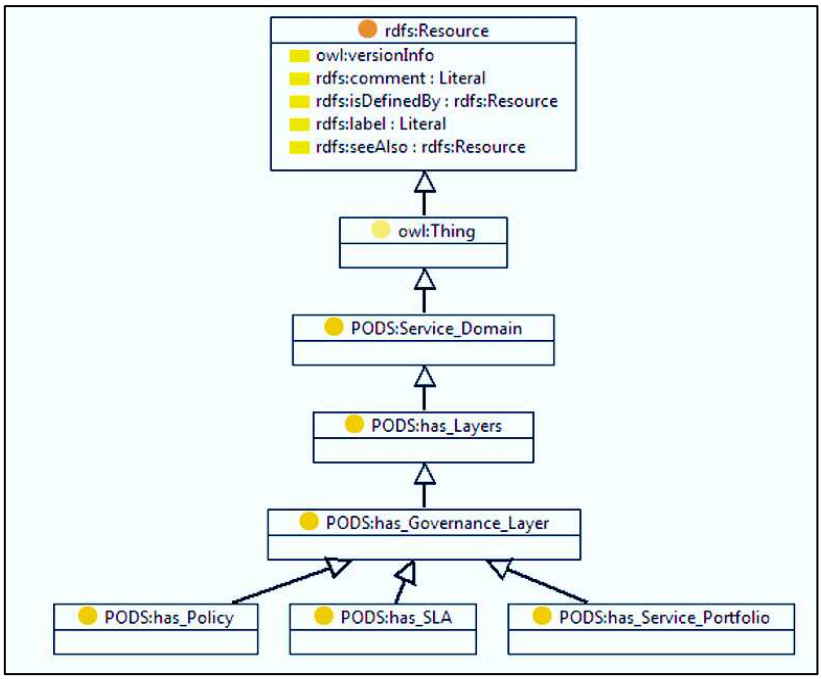


Figure 27: Ontology of Governance Layer.

- **Policy:** contains two types of policies that are global policy and specific service policy. Global policy is applied to all interfaces while specific service policy is for each service. Policy has elements such as service policy map and rules. Policy map contains ordered set of rules while rules are associated with class command.

- **SLA:** captures agreed quality, availability and responsibilities committed by service provider. It is measured by the monitored quality and availability as compared to one committed by service provider.

- **Service Portfolio:** is described by service design package. It has three elements that are service pipeline, service catalogue and retired services. Service pipeline contains references to services that are under development. Service Catalogue holds links to active services. Retired Services are services that are considered as obsolete.

B. Performance Oriented Decision Support Ontology PODSOnt

Now, we propose an ontology that integrates all performance ontologies at SOA layers of part B and extends ontologies at SOA layers and extends service layers in terms of performance and decision QoS. As a result, a composite and coherent global ontology in terms of performance and decision support for SBS is produced. This ontology is named as performance-oriented decision support ontology (PODSOnt) and is developed to maximize the service reuse capabilities. Figure 28 provides the structure of PODSOnt. PODSOnt begins with service domain concept. Service domain has performance layers. Layers are highlighted by pink colour. We have already explained the concepts of performance

at different layers. Now, we highlight performance and decision concepts in this ontology. Service layer has service 1 which has QoS concepts. QoS is defined by performance efficiency, maturity, risk and recoverability concepts. Performance efficiency is measured by time behaviour, resource utilization and capacity. Maturity is measured at five levels. Level 0 and level 1 are measured by service instance time while the other levels are measured by availability. Risk is measured by risk gravity, and risk gravity is measured by risk probability and severity. Recoverability has failure rate and MTTR. MTTR is measured by service down time and time intervals. Failure rate is measured by service instance response count and total service instance response count. Service has decision concept. Decision is measured by service instance ID and total service instance ID. Decision has result and result is defined by class OK, class MayBe, and class NO. If the result is OK then it means that service is recommended for reuse else if result is MayBe, service maybe recommended and finally, if result is NO then service is not recommended.

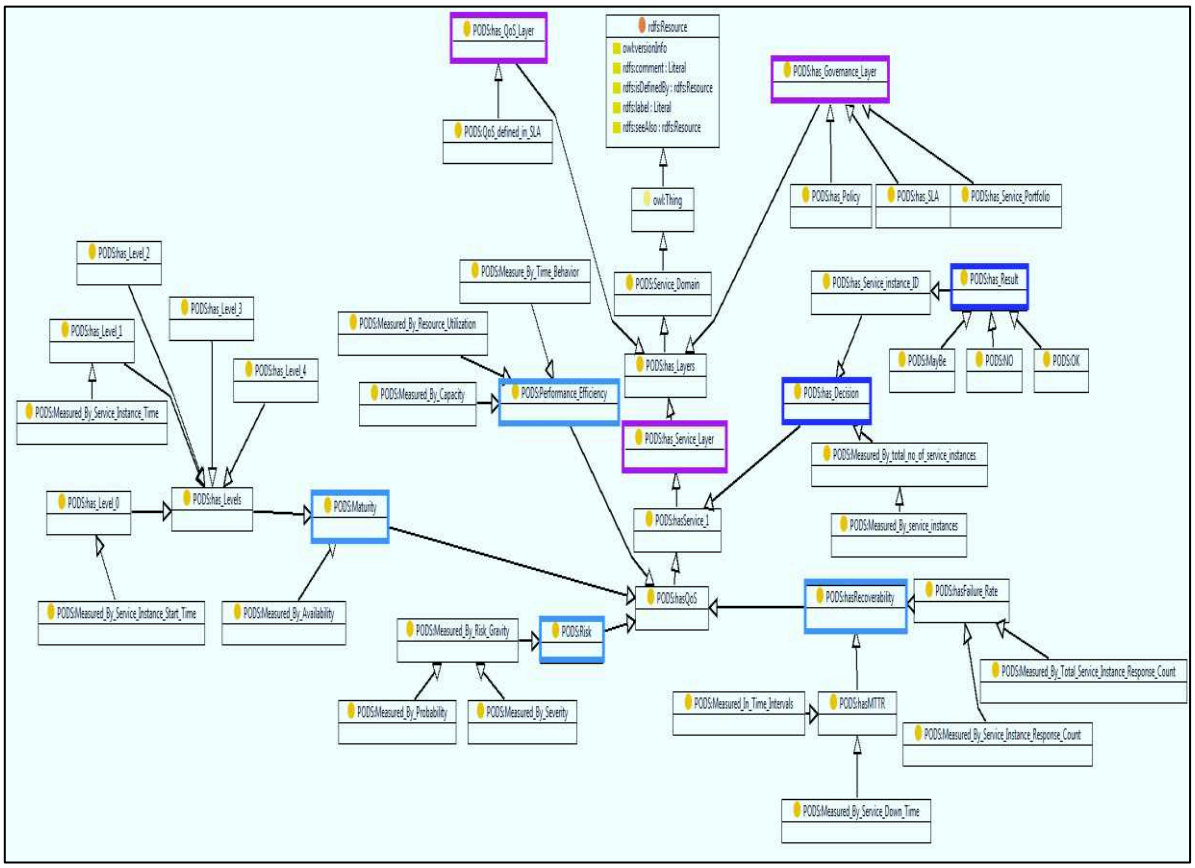


Figure 28: Performance Oriented Decision Support Ontology (PODSOnt)

A composite hierarchy of performance profile is shown in Figure 29. Performance profile is generated from the ontological concepts of PODSOnt.

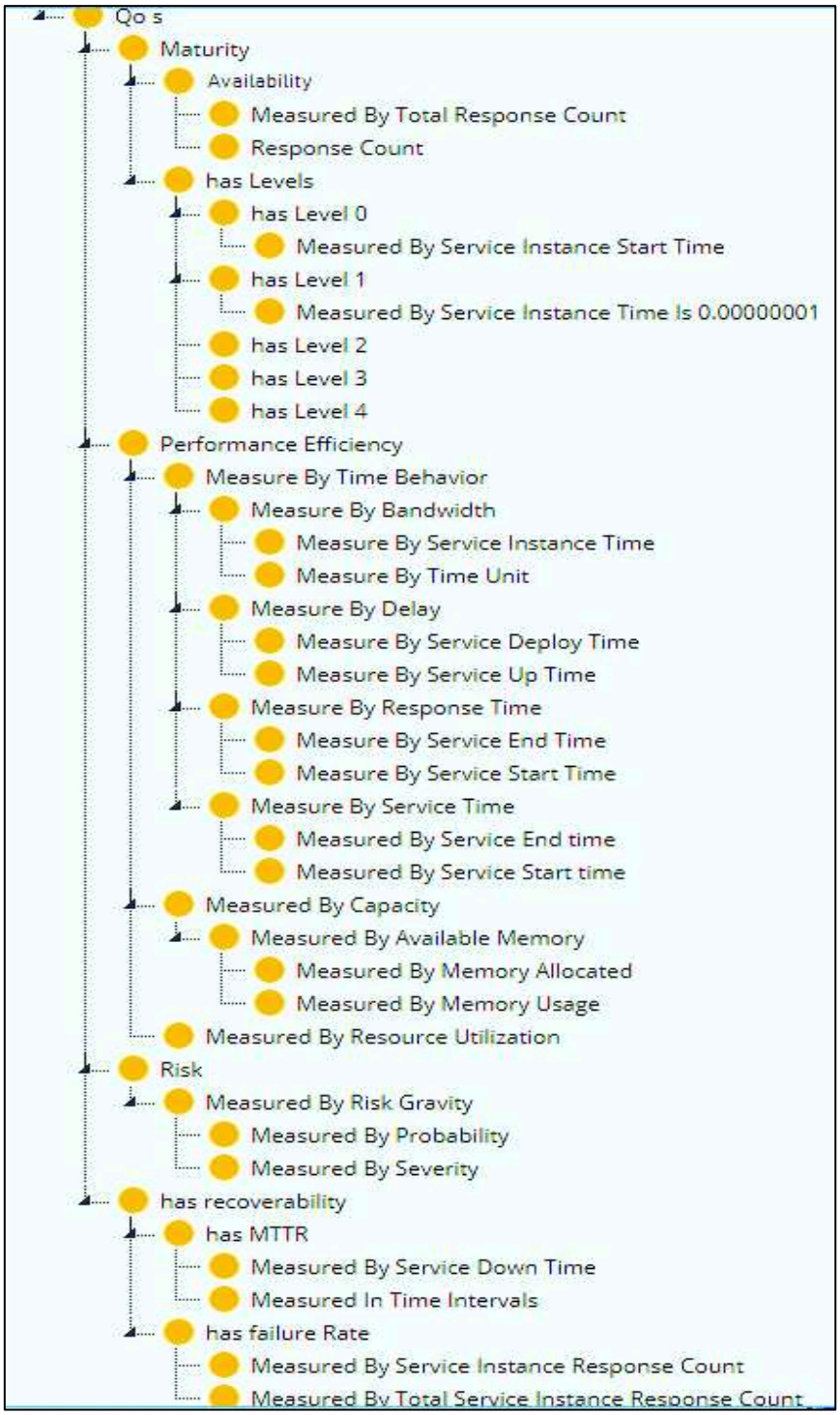


Figure 29: Performance Profile

By the implementation of ontological concepts, properties and relationships, new knowledge can be deduced by implementing semantic rules. The evolving or dynamic nature of ontology permits to assimilate or cumulate new concepts. As a result of this evolution, semantic rules evolve. The implementation of the semantic rules is presented in the following section.

4.1.2 PODS Reasoning

In order to solve problems and advise the End User in decision-making, the decision support system need to access a domain knowledge base. Therefore, the effectiveness of the system relies heavily on the way the knowledge is represented. Hence, the knowledge base needs to be implemented so that the knowledge becomes explicit and readable by a machine. This set of classes, relationships, instances and inference rules allows decision model to compose the domain knowledge base. We compose semantic inference rules to evaluate performance management. Some of these rules are already published in a conference [127] and described in section 3.3. Inference rules that compose the knowledge organization from the relationships between PODSOnt concepts are shown in Table 14 and Table 15. The right column contains the rules corresponding to the technical indicators listed in the left one. We define the rules using SWRL Protégé.

Table 14: Inference Rules (IR) (1/2)

Technical indicators	**SWRL Rules**
Availability	swrl:divide(Total_Response_Count,Total_Request_Count*100),?x) -> Availability(?x)
Throughput	swrl:divide(no_of_active_request,Average_Response_Time,?x)-> Throughput(?x)
Delay	swrl:difference(Service_Deployed_Time,Service_Up_Time,?x)->Delay(?x)
Response_Time	swrl:difference(Service_End_Time,Service_Start_Time,?x)-> Response_Time(?x)
Bandwidth	swrl:divide(service_instance,Time,?x)->Bandwidth(?x)
Failure_Rate	swrl:difference(Request_Count,Response_Count,?x)->Failure_Rate(?x)
MTTR	swrl:divide(Service_Down_Time,No_Intervals,?x)->MTTR(?x)
Service_Time	swrl:sum(Service_Start_Time,Service_End_Time,?x)->Service_Time (?x)
Available_Memory	swrl:difference(Memory_Allocated,Memory_Usage,?x) ->Available_Memory(?x)
Risk_Gravity	swrl:multiply(Probability,Severity,?x)->Risk_Gravity(?x)
Maturity_level_0	swrl:equal(Service_Instance_Time,0,?x)->Maturity_Level0(?x)
Maturity_level_1	swrl:greater(Service_Instance_Time,0.00000001,?x)-> Maturity_Level1(?x)
Maturity_level_2	swrl:greater(Availability, 98.999,?x)->Maturity_Level2(?x)
Maturity_level_3	swrl:greater(Availability, 99.99,?x)->Maturity_Level3(?x)
Maturity_level_4	swrl:equal(Availability, 100,?x)->Maturity_Level4(?x)
Total_No_of_Service_Instances	swrl:sum(Service_Instance_ID,?x)->Total_Service_Instances (?x)

Table 15: Inference Rules (IR) (2/2)

Decision Variables	SWRL
Count_Service_Instance_ID _Recommendation_OK	Service_Instance_ID (? p) ^ hasResult(?p, ?OK) -> swrl:count(?OK)
Count_Service_Instance_ID _Recommendation_MayBe	Service_Instance_ID(?p)^hasResult(?p,?MayBe)-> swrl:count(?MayBe)
Count_Service_Instance_ID _Recommendation_NO	Service_Instance_ID (?p) ^ hasResult(?p,? NO) -> swrl:count(?NO)
Set_ Service_Recommendation	swrl:greater(Count_Service_Recommendation_NO, (Count_Service_Recommendation_OK^ Count_Service_Recommendation_MayBe),?x)-> Set_Recommendation (?NO)
Set_ Service_Recommendation	swrl:greater(Count_Service_Recommendation_OK, (Count_Service_Recommendation_NO^ Count_Service_Recommendation_MayBe),?x)-> Set_Recommendation (?OK)
Set_ Service_Recommendation	swrl:greater(Count_Service_Recommendation_MayBe, (Count_Service_Recommendation_OK^ Count_Service_Recommendation_NO),?x)-> Set_Recommendation (?MayBe)

The implementation of inference rules using ontological concepts, properties and relationships provides a knowledge base. Since inference rules can evolve with time, they can help in providing enhanced analytics and manipulations in order to help decision makers to take right accurate on the fly. We explain the Implementation of PODS research prototype phase in the below part.

4.2. PODS Research Prototype

A research prototype is used to build a solution, release or model in order to test a process. It is also considered as the step between the formalization and the evaluation of an idea or a solution. PODS research prototype discusses the most important algorithms of PODS and explains their workings. It introduces our research prototype implementation and discusses various features and methods.

PODS research prototype is composed of two phases. These two phases are data management and decision support. Data management phase discusses the most important algorithms used to generate data related to the performance, risk and maturity of SBS. Decision support phase use these data to create training set and evaluate decisions by the help of algorithms.

4.2.1 Data Management

Data management is the preparation of shaping and sustaining data progressions to meet ongoing information lifecycle prerequisites. In order to provide better analytics and flexibility in the data manipulation, the data must be measured and managed statistically. By correctly managing and preparing the data for analytics, decision can be made efficiently. Therefore, the efficacy of decision support system relies heavily on the way the data is managed.

Data management is divided into three parts based on its implementation. The first part shows and describes the implementation of performance management. The second part displays and explains the implementation of maturity evaluation. The third part illustrates and explicates the implementation of risk management.

A. Performance Management

Performance management is performed based on the End User requirement and priority. The implementation of performance management is shown in Figure 30 through the Performance Management algorithm. The algorithm begins with inputs and outputs. Line 1, 2 and 3 takes inference rules IR, and concepts of PODSOnt as inputs. In this algorithm, we evaluate performance based on End User preferences. The End User can specify the quantitative technical indicator that he or she needs. He can also ask for a generic performance profile based on several technical indicators. All the technical indicators are calculated or measured as inference rules as shown in section 4.1 in Table 14 and Table15. Line 4, 5 and 6 are related to the final outputs as service ***s*** for reuse, composite service ***c*** for reuse and performance-profile.txt.

Line 7 begins with the Performance-Measurement function of the algorithm. This function measures the performance of services by inference rules. Line 8 initiates the Check-Availability function. This function evaluates the availability of services based on inference rules. In Line 9, the *for* loop starts to check the availability for all services. Line 10 activates the *for* loop to check all service invocations. Line 11 instigates the Check-Availability function to check the availability of all service invocations. Line 12, 13, 14, 15, 16, 17, 18 and 19 estimate response-counts, service_end_time, service_start_time, service_deployed_time, service_up_time, no_of_active_request service, service_down_time and delay. Line 20 performs services comparison in terms of delays. Line 21 stores service instance with minimum delay in variable m. Line 22 evaluates services based on response-time. Line 23 perform comparisons of services to evaluate best response-time. Line 24 stores service instance with minimum response time in ***mr*** variable. In Line 25, the *if* condition is initiated to check unavailability of atomic

service. Line 26 assess the availability of compose services and return the composite service if it is available in Line 27. Line 28, 29, 30 and 31 evaluate throughput, bandwidth, failure Rate and MTTR. All performance variables, values of service instance are reported in the Performance-profile.txt text file in Line 32. Finally, the algorithm ends by returning performance profile. By the help of this profile, we can evaluate atomic or composite service that is best in terms of performance based on the preferred choice of the End User. Moreover, global performance efficiency and reliability of services can also be assessed with the help of this algorithm.

```
Performance measurement:
Begin
Input:
1. Set of inference rules IR {ir1, ir2, ir3…irn}
2. Set of concepts of PODSOnt S {s1, s2…..sn}
Output:
4. Service s to reuse
5. Composite service c
6. Performance-profile.txt

7. Performance Measurement(s)
{
8.   Check-Availability
     {
9.        For each service s in S
10.           For each invocation j of service s
11.             Check-Availability of j in IR

 12.                  Check and store response-counts of j from IR
 13.                  Check and store Service_End_Time of j from IR
 14.                  Check and store Service_Start_Time of j from IR
 15.                  Check and store Service_Deployed_Time of j from IR
 16.                  Check and store Service_Up_Time of j from IR
 17.                  Check and store no_of_active_request service of j from IR
 18.                  Check and store service_down_time of j from IR
 19.                  Check-delay of j in IR
                    {
 20.                    Compare delay of j with i++
 21.                    Store service instance with minimum delay in m
 22.                    Check-response-time of m from IR
                       {
 23.                      Compare response time of m with j
 24.                       Store service instance with minimum response time in mr
                       }
                    }
 25.         if no service is available
             {
 26.              Compose services
 27.               Return composite service CS
             }
        }
28. Check Throughput from IR
29. Check Bandwidth from IR
30. Check Failure Rate from IR
31. Check MTTR from IR

32. Perform function of writing all performance variables, values of service instance in Performance-profile.txt
}
End
```

Figure 30: Performance Management Algorithm

Performance management part provided implementation of measurement method for quality characteristics of performance efficiency, recoverability and reliability in terms of availability. Now, we explain the implementation of maturity and its evolution along five levels of CMMI.

B. Maturity Evaluation

Maturity evaluation is also performed based on the End User requirement and priority. Maturity evaluation is performed based on availability, service end time and service start time. The maturity evaluation algorithm is presented in Figure 31. The algorithm begins with inputs and outputs. Line 1, 2 and 3 take as inputs inference rules IR and ontological concepts of PODSOnt in the form of sets. Line 4 indicates the output of the algorithm which is the Maturity-profile.txt text file. The Maturity-Evaluation function starts at Line 5. Line 6 starts with a *for* loop to evaluate each service while line 7 instantiates a second *for* loop for each service invocation.

```
Maturity Evaluation:
Begin
Input:
1. Set of inference rules IR {ir1, ir2, ir3…ir_n}
2. Set of concepts of PODSOnt S {s1, s2…..s_n}
Output:
4. Maturity-profile.txt

5. Maturity-Evaluation(s)
      {
6.        For each service s in S
7.        For each invocation invo of service s
8.        Check-Maturity-level of invo in IR
          {
9.          Check and store service-instance-start-time from IR
10.         Check and store availability from IR
11.             For (j=0; j<5; j++)
                {
12.               Check and store maturity level j value from IR;

                }
            }
       }
13.   Write maturity profile with values for all service instances in Maturity-profile.txt
End
```

Figure 31: Maturity Evaluation Algorithm

Line 8 begins with Check-Maturity-level function. In this function, maturity is evaluated based on the inference rules presented previously in Table 14 and Table 15, Section 4.1.2. The evolution of maturity along the five levels of CMMI is taken into account in these inference rules. They are represented by the five rules in Table 14. Maturity of the service or process is computed by providing a threshold percentage value as shown in Table 16. For each level, the appropriate ponderation is associated in

order to facilitate the calculation of service or process maturity. The CMMI is a basic foundational building block for achieving service or process improvement and ensuring the service or business process optimization. The service is considered disciplined when its functional specifications are identified. The business process is considered as managed when its business specifications are identified. Once the functional and business specifications are defined, the service or process is able to be executable and used. The knowledge of service or process performance tends to be more qualitative rather than quantitative up to Maturity Level 3 'defined'. In the third level of 'defined', we can obtain measures that provide information about the availability of service or process. In this level, where the service or process is deployed and used, several means have been set up in order to supervise the evolution of their maturity over the time. When the service or process runs, we are able to assess its performance in the fourth level named as 'quantitatively managed'. The real use of the service or process by its end End Users corresponds to the Maturity Level 4. It insists on managing service or process performance and addressing the main causes and sources of variation. These causes of variation can indicate a problem in service or process performance and may require correction and solution to maintain service or process performance during its utilization. At Maturity Level 5, organization emphasizes on reducing the common cause of variation and it improves the overall performance level. The service or process is considered optimized if the service or process is stable for a long time. There is no evolution of the means of control and performance. On the basis of the service or process history during a certain period, a deviation is detected. If the service or process is not able to answer, it should be return to the second or the third level of maturity to redefine the specifications or it has to finish.

Line 9 and 10 estimates and stores service-instance-start-time and availability along with values. Line 11 explains the third *for* loop to calculate maturity-level of services in Line 12. Line 13 writes maturity profile with values for all service instances in the Maturity-profile.txt text file and the algorithm ends.

Table 16: Maturity Evaluation based on CMMI Levels

CMMI Level	Course of Action	Percentage Threshold
Initial	No control, no reliable service or process	15 %
Managed	Modelled	30 %
Defined	Tasks and roles are defined	55 %
Quantitatively Managed	Systematic measurement	75 %
Optimizing	Repetitive improvement	100 %

Maturity evaluation part explained the implementation of maturity estimation at the five stages of CMMI. Now, we describe the implementation of risk management and computation of risk mitigation actions.

C. Risk Management Algorithm

Risk management is also performed based on the End User requirement and priority. We show risk types, corresponding assessment and mitigation action in Table 17. Risk types are loss of service, specification changes and unexpected behavior. We define an assessment for each risk type. The loss of a service is imputed to network problems, specification changes is resulted due to format change or loss of service and finally, an unexpected service behavior is caused due to particular service specification change. Specific mitigation actions are proposed for each type of risk. One of the way to mitigate the loss of a service is to suggest alternative services. If the specification of a service changes, the only way to solve this problem is to provide a new service. In order to identify the cause of unexpected service behaviour, we need to perform testing of services. In this way, this problem can be mitigated.

Table 17: Risk Assessment and Mitigation

RISK	Assessment	Mitigation Action
Loss of service	Network problems	Use of alternative service
Specification changes	Format change or loss of service	Discovery of new service
Unexpected service behavior	Analysis of published service specifications	Service testing

The Risk Management algorithm is shown in Figure 32. The algorithm begins with inputs and outputs. Line 1, 2 and 3 take inference rules IR and ontological concepts of PODSOnt as inputs, in the form of sets. Line 4 indicates the output of the algorithm, which is the Risk-profile.txt text file. The Risk-Evaluation function is initiated at Line 5. Line 6 starts with a *for* loop to evaluate each service while Line 7 instantiates the second *for* loop for each invocation of service. At Line 8 the Check-Risk-level function begins. This function evaluates the risk level of services based on the threshold values. At Line 9, a third *for* loop calculates risk-level of services based on the inference rules IR and store this value at line 10. Risk profile with values for all service instances are reported in the Risk-profile.txt text file at Line 11, and the algorithm ends.

In the Risk Management algorithm, risk is evaluated based on the inference rule presented previously in Table 14 and Table 15, Section 4.1.2. Risk level is computed based on risk gravity. Risk gravity is

measured by risk probability and risk severity. The steps for risk management process are identify risk, measure the level of risk, calculate the probability percentage, make assessment and mitigate action. The algorithm returns the risk text file for risk profile.

Risk Evaluation:
Begin
Input:
1. Set of inference rules IR {ir1, ir2, ir3…ir_n}
2. Set of concepts of PODSOnt S {s1, s2…..s_n}
Output:
4. Risk-profile.txt

```
5. Risk-Evaluation(s)
{
6.        For each service s in S
7.        For each invocation invo of service s
8.        Check-Risk-level of invo in IR
          {
9.            For (j=0; j<5; j++)
                {
10.               Check and store risk level j value from IR;
                }
           }
}
11.  Write risk profile with values for all service instances in Risk-profile.txt
```

End

Figure 32: Risk Management Algorithm

Risk management provided the implementation for the identification of risk level. This data management part shaped and managed data by computational analysis. Hence, data are now tuned in order to apply decision-oriented algorithms. We explain the implementation of the decision support system in the following part.

4.2.2 Decision Support

A decision support is a manner of scrutinizing business data and delivers it to the End User so that they can take business decisions more easily, efficiently and dynamically. A system that accumulates, categorizes and investigates business data to enable quality decision-making for management, operations and planning is categorized as decision support. A well-designed DSS supports decision makers in assembling data and forecasting based on manipulations of data.

The decision support is divided into three parts. The first part shows and describes performance evaluation for a specific decision. The second part displays and explains the algorithm for impact analysis of a new consumption. The third part illustrates and explicates decision evaluation.

A. Performance Evaluation for Specific Decision

Performance evaluation for specific decision provides decision based on performance, risk and maturity. The performance evaluation for specific decision algorithm is shown in Figure 33. The algorithm begins with inputs and outputs. Line 1, 2, 3 and 4 of this algorithm takes inputs as PODSOnt ontology, maturity-profile text file, risk profile text file and performance profile text file. Line 5 of this algorithm indicates the output in the form of text file Specific-Decision.txt. Line 6 starts with the function Apply-Classification-Algorithms. Line 7 initiates *for* loop for each service while line 8 instantiates *for* loop for each invocation of service. Classification algorithms are logistic regression, naïve Bayes, support vector machine and decision Trees. Line 9 stores results of text files. Line 10 begins the function of Apply-Logistic-Regression and stores results in line 11. Line 12 begins the function of Apply-Naïve-Bayes and stores results in line 13. Line 14 begins the function of Apply-Support-Vector-Machine and stores results in line 15. Line 16 begins the function of Apply-Decision-Trees and stores results in line 17. Line 18 compare results of all classification algorithms and choose the optimum. Line 19 write the specific decision results for each service and service instance in Specific-Decision.txt and ends algorithm.

```
Performance Evaluation for Specific Decision:
Begin
Input:
1. SOnt S {s1, s2.....sn}
2. Performance-profile.txt
3. Maturity-profile.txt
4. Risk-profile.txt
Output:
5. Specific-Decision.txt

6. Apply-Classification-Algorithms
{
7. For each service s in S
8.      For each invocation i of service s
        {
9.          Store results in p, m and r from input text files;

9.          Apply-Logistic-Regression (i, p, m, r)
          {
10.           Store results;
          }
 11.        Apply-Naïve-Bayes (i, p, m, r)
          {
 12.           Store results;
          }
 13.        Apply-Support-Vector-Machine (i, p, m, r)
          {
 14.           Store results;
          }
 15.        Apply-Decision-Trees (i, p, m, r)
          {
 16.           Store results;
          }
        }
}
 17. Compare results and choose the optimum;
 18.  Write specific decision results for each i in Specific-Decision.txt
End
```

Figure 33: Performance Evaluation for Specific Decision based Algorithm

In this part, we explained the implementation of performance evaluation for specific decision. Decisions evaluation is performed based on the priority of End User. This kind of decision is called specific decision. Specific Decision is computed based on classification algorithms. Result of the most optimum algorithm has been taken into consideration. Hence, decision is now tuned based on the priority of End User. After performance evaluation, we explain the implementation of impact analysis in the following part to evaluate a global decision.

B. Impact Analysis

Impact analysis evaluates global decision based on performance, risk and maturity by increasing the consumption of services. The impact analysis algorithm is shown in Figure 34. The algorithm begins with inputs and outputs. Line 1 and 2 take inputs as PODSOnt ontology and impact analysis text file. Line 3 of this algorithm indicates the output in the form of text file Impact-analysis.txt. Line 4 starts with the function Apply-Impact-Analysis. Line 5 initiates *for* loop for each service while line 6 instantiates *for* loop for each invocation of service. DSS apply the data sets containing several services and instances and evaluate it by machine learning model. Machine learning model that we choose is analytic hierarchy model in order to analyze global performance by increasing the consumption. Line 7 stores specific decisions in array a. Line 8 begins with the function Apply-Analytic-Hierarchy-Algorithm and store results in line 9. Line 10 write the global decision results for each service and service instance in Global-Decision.txt and ends algorithm.

```
Impact Analysis:
Begin
Input:
1. SOnt S {s1, s2.....sn}
2. Specific-Decision.txt
Output:
3. Global-Decision.txt

4. Apply-Impact-Analysis
{
5. For each service s in S
6.   For each invocation i of service s
     {
7.            store specific decision in array a;

8.        Apply-Analytic-Hierarchy-Algorithm (a, s)
          {
9.            Store results in array g;
          }
}
10. Write global decision results for each i in Global-Decision.txt
End
```

Figure 34: Impact Analysis Algorithm

In this part, we analyze decision evaluation based on all service instances. Now, we explain the evaluation of decision based on each service.

C. Decision Evaluation

The decision evaluation part creates a decision matrix based on both specific and global decisions. The decision evaluation algorithm is shown in Figure 35. This algorithm begins with inputs and outputs. Line 1 and 2 of this algorithm takes inputs as Specific-Decision.txt and Global-Decision.txt. Line 3 of this algorithm indicates the output in the form of Identity-matrix-chart. Line 4 initiates to create matrix. Line 5 writes specific decision in matrix while line 6 stores global decision in matrix. Line 7 evaluate results and store in identity matrix chart and ends algorithm. Results of this identity matrix chart to guide for atomic service reuse or composite service reuse are in the form of OK, MayBe and NO.

Decision Evaluation:
Begin
Input:
1. Specific-Decision.txt
2. Global-Decision.txt
Output:
3. Identity-Matrix-Chart

4. create matrix
{
5. Write specific decision from Specific-Decision.txt in matrix
6. Write global decision from Global-Decision.txt in matrix
}
7. Evaluate results and store in Identity-Matrix-Chart
End

Figure 35: Decision Evaluation Algorithm

In this part, we have explained PODS research prototype that provided the algorithms of data management and decision support. In the next part, we will explain the automation of this prototype.

4.3. PODS System

PODS system (PODSS) is the implementation of PODS research design and PODS research prototype. In this part, we discuss the implementation of all the major classes. PODSS is aided with the data management and decision support algorithms discussed above in section 4.2 for the generation of specific and global decisions.

4.3.1 PODSS High Level Architecture

PODSS takes rdf file as input. This rdf file contains the ontological concepts and inference rules. An XML specification has been generated from this rdf and stores in the object. After this, classification has been applied to generate specific decisions. Finally, generic decisions have been made for each service. The six components of the architecture are presented below.

SemanticXMLGenerator

PODSS creates semantic XML specification from the ontological concepts and inference rules presented in section 4.1. This generator generates XML from RDFconcepts, classes and properties of ontologies.

SemanticXML Reader:

The reader reads the SemanticXML specification and stores it in an object. The reader object stores the XML information from the SemanticXML specification in the form of vectors. The advantage of vectors in Java compared to arrays, lies in the fact that vectors expand automatically when new data are added to them.

Classifier:

The classifier takes SemanticXML object from the SemanticXML reader as input. It applies classification algorithms on the data gathered from SemanticXML object and evaluate performance for services instances and generates specific decisions. Specific decisions are also generated based on the priority of technical indicator specified by the End User.

Impact Analyzer:

The impact analyzer takes specific decisions from the classifier as input. It uses the impact analysis algorithm and generates global decisions. Global decisions are based on the overall evaluation of the performance with the increased consumption of services.

Decision Generator:

The decision generator takes the results of classifier as input and generates results for each service. Decisions are stored in the form of identitity matrix to recommend the most optimum service in terms of performance. It uses the decision evaluation algorithm and generates recommendations.

4.3.2 Classes of PODSS

The major classes implemented in PODSS are shown in the package diagram in Figure 36. The diagram contains five packages. These five packages are gui, java.io, java.util, PerformanceOrientedDecisionSupport and java.lang. Gui package includes the MainFrame which is the entry point for PODSS. This class is responsible for generating the GUI of the system, and this is the main controller class that calls other classes. Second package is PerformanceOrientedDecisionSupport that includes all functional classes. The remaining classes of third package named as java.io are helper classes for different functionalities in PODSS. Fourth package is java.util that uses vectors. The advantage of vectors in Java as compared to arrays, lies in the fact that vectors expand automatically when new data are added to them. Finally, fifth package of java.lang describes the data types used as integer and string. These classes are presented and explained below.

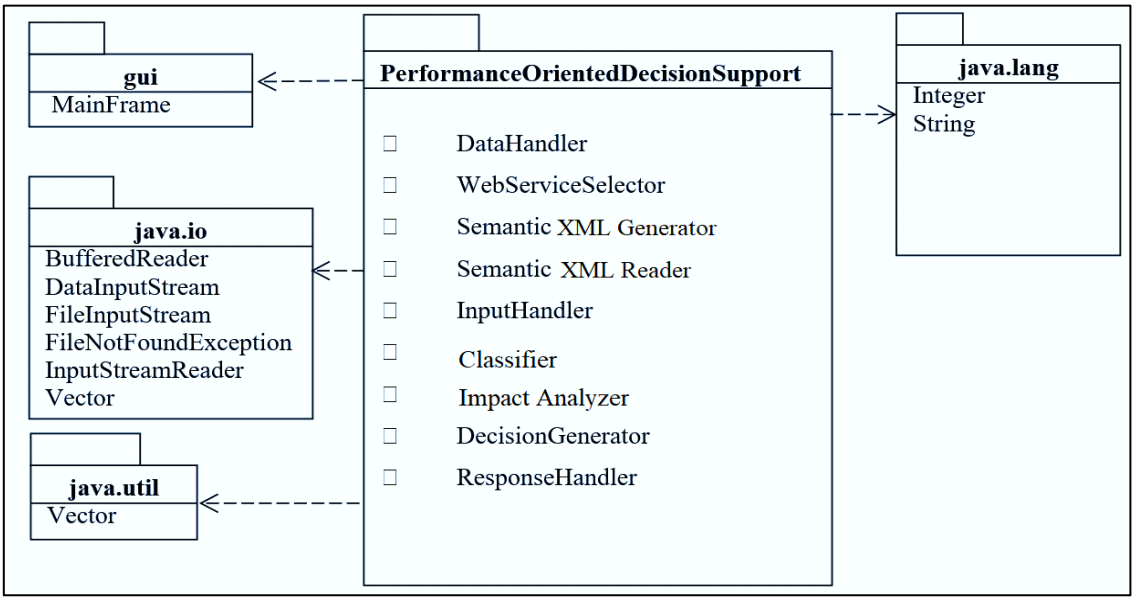


Figure 36: Package Diagram of PODSS

For each class, we present its general features followed by its signature and its functions.

1. Data Handler

This class gets the traces of technical indicators from the database. This class contains two functions.

Signature: public void DataHandler ()

Signature of Functions:

public void GetData ()

public void SetData ()

2. WebServiceSelector

This class selects the existing services from the repository of services. It implements existing Web service selection algorithm.

Signature : public void SelectWebService ()

3. SemanticXMLGenerator

This class generates semantic XML from the RDF concepts. This class has one function.

Signature: public class SemanticXML.

1. RdfToXML:

This function generates XML syntax from RDF. RDF is a collection of ontological concepts, classes, objects and relationships between them. As a result, a SemanticXML file has been generated.

Signature: public void RdfToSemanticXML ()

4. SemanticXMLReader

This class reads XML specification and it has one function named as ReadSemanticXML.

Signature: public class SemanticXMLReader.

1. **ReadSemanticXML:**

This function reads the SemanticXML specification and stores it in a reader object. It takes the SemanticXML file as input and output is the object.

Signature: public void ReadSemanticXML (String file) throws FileNotFoundException

5. InputHandler

This class handles all the inputs that the system used and it has fourteen functions.

Signature: public class InputHandler.

Functions:

Signature: Public String getServiceName ()

Signature: Public void setServiceName (String Name)

Signature: Public void public String getSLAName ()

Signature: Public void getSLAName (String Name)

Signature: Public void public String getServiceProviderName ()

Signature: Public void setServiceProviderName (String Name)

Signature: Public void public String getServiceConsumerName ()

Signature: Public void setServiceConsumerName (String Name)

Signature: Public void public String getServiceHostName ()

Signature: Public void setServiceHostName (String Name)

Signature: Public void public String getTechnicalIndicatorName ()

Signature: Public void setTechnicalIndicatorName (String Name)

Signature: Public String getValue ()

Signature: Public void setValue (Double Name)

6. Classifier

This class applies classification algorithms on the data. This class Classifier class includes four main functions.

Signature: public class Classifier

1. ApplyLogisticRegression

The ApplyLogisticRegression function reads technical indicators information of the SemanticWSDL objects from the SemanticWSDLParser class. This function applies logistic regression classification algorithm.

Signature: public void ApplyLogisticRegression ()

2. ApplyNaïveBayes

The ApplyNaïveBayes function reads technical indicators information of the SemanticWSDL objects from the SemanticWSDLParser class. This function applies naïve Bayes classification algorithm.

Signature: public void ApplyNaïveBayes ()

3. ApplySupportVectorMachine

The ApplySupportVectorMachine function reads technical indicators information of the SemanticWSDL objects from the SemanticWSDLParser class. This function applies support vector machine classification algorithm.

Signature: public void ApplySupportVectorMachine ()

4. ApplyDecisionTrees

The ApplyDecisionTrees function reads technical indicators information of the SemanticWSDL objects from the SemanticWSDLParser class. This function applies decision Trees classification algorithm.

Signature: public void ApplyDecisionTrees ()

7. ImpactAnalyzer

The ImpactAnalyzer class applies proposed impact analysis algorithm and generates global decision.

Signature: public class ImpactAnalyzer

8. DecisionGenerator

The DecisionGenerator class applies proposed decision evaluation algorithm and generates decision matrix. It has one main function.

Signature: public class DecisionGenerator

1. CreateMatrix

The CreateMatrix function generates the matrix which shows recommendations for most optimum service in terms of performance.

Signature: public void CreateMatrix ()

4.4. Discussion

In this chapter, we have presented the implementation and important algorithms of PODS. Implementation of PODS has been made on the basis ontologies, reasoning rules and PODS algorithms. We provide algorithms for performance, risk, maturity, analytical assessment, impact analysis and decision support. Performance based ontological graphs provide fast retrieval and exploitation. Inference rules are provided by using ontological concepts to generate knowledge base of performance profile and decision. This knowledge base is used to perform data discovery and analytical assessment by using classification algorithms to provide aggregation rules. We compare the result and select the optimum classification algorithms result which is decision tree in this case. From this analytical assessment, we get the trends of performance based technical indicators as a result. We analyze the performance of services with different time stamps and provide a decision chart. We also analyze the impact of consuming more services and a trend of overall performance under different consumption loads to make a generic decision for service reuse. We automate the whole process of performance oriented decision support in java language and developed a system that has used

proposed algorithms and inference rules. As a result, we obtain a dynamic efficient decision support system based on performance.

In the next chapter, we will evaluate our performance-oriented decision support system on an industrial case study and discuss some analytical results.

CHAPTER 5. Evaluation of PODSS

The PODS implementation is explained in detail with the help of its three phases of research design, research prototype and system. However, every system must need to be validated and evaluated. This chapter is dedicated to providing the evaluation of PODSS. PODSS is evaluated by fetching services from a public repository of shared services and usage scenarios.

Before explaining the different phases of this chapter, we first explain the high-level schema of the PODSS evaluation. High level schema of PODSS evaluation is shown in Figure 37.

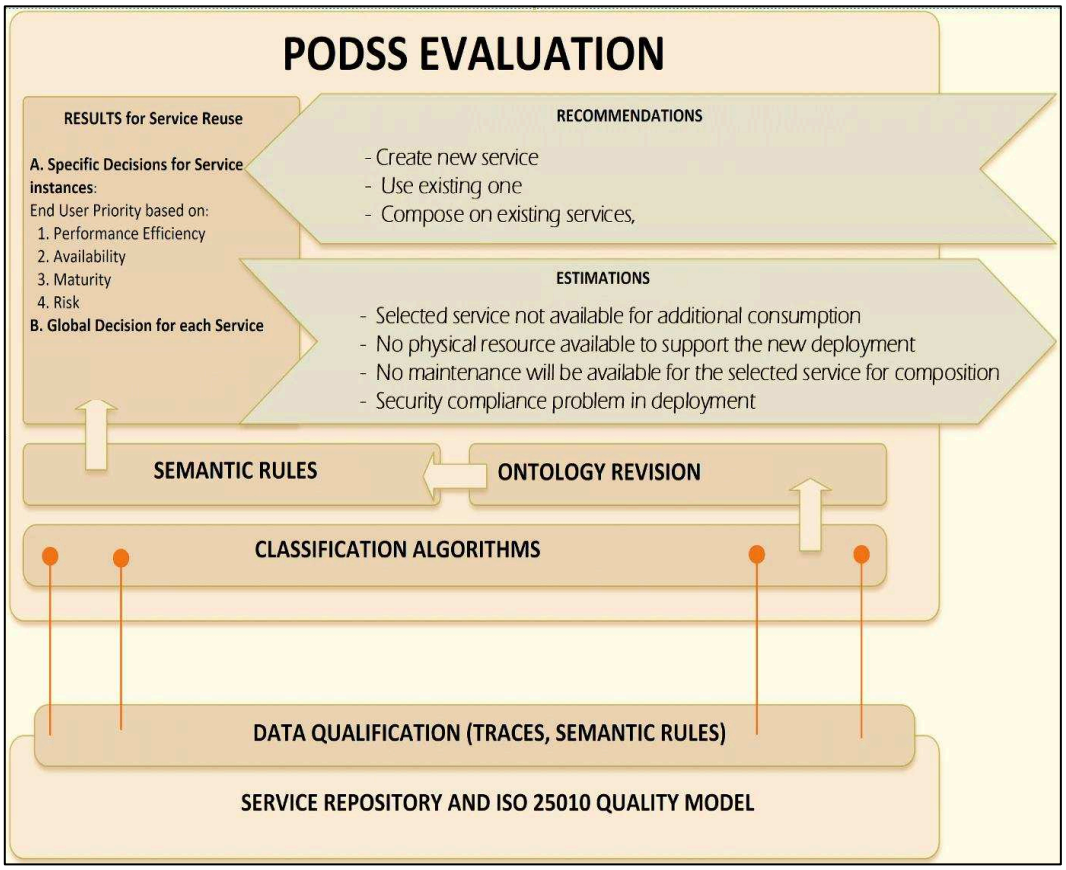


Figure 37: High level Schema of PODSS Evaluation

This chapter is composed of three phases. The three phases are PODSS data set, PODSS structure and PODSS evaluation. PODSS data set is supported with the analysis of public repository of shared services. PODSS structure is demonstrated with the help of use case. PODSS evaluation is carried out by formulating different scenarios or cases.

5.1 Service Reuse Use Case

The purpose of a use case diagram in UML is to demonstrate the different ways that a user might interact with a system. It helps to evaluate the system. We explain the use case by describing the notion of three different strategy levels defined for the business function of a company. These three levels are business level, functional level and applicative level. At the business level, a business process is defined based on the requirements of End User. At the functional level, processes are defined to provide the desired functionality of business process. Finally, at the applicative level, functionality of each service task is defined for every process. Service tasks are demarcated by defining service methods, inputs and outputs. The output of each service task is the input of the following service task.

We need to develop a business process in order to analyze collaboration of two business partners that are partner one and partner two. For this purpose, we explain a use case of the business process and the underlying services that we need to develop. This use case is used to evaluate the capacity of these two partners to industrialize products from common customer projects. The aim is to identify business opportunity and the objectives are to reduce project quotation costs, optimize the delay of customer quote treatment and reduce the delay of project acceptance without disturbing the process quality. We classify business opportunity in the form of Mandatory, High ROI, nice to have or not relevant. Customer project details and the data of two industrial partners are the expected results of this analysis or evaluation. The input is the project submitted by a customer for the evaluation. We name this business process as CollaborationAnalyzer. CollaborationAnalyzer schema is shown in Figure 38. This figure illustrates all the service tasks required to complete this business process. First row of the figure shows a user task in which a customer submits its project to inspect business opportunity. In the first step, we show the application and the services that we need to develop. First service is represented as Decomposeproject and defined as service task to decompose the project in “features” based on CAD file form. Feature analytic service will provide project classification and ranking based on existing BigData infrastructure under development by the partner 1 and partner 2. For this purpose, we define a second service task named as RankProject. Third service generates an internal ranking report and is defined as GenerateRankingReport service task. A notification is sent to the two partners to let them know that the ranking report is available. Partners receive the notification and they will accept or reject the quotation. The decisions from both partners is consolidated. If the project is accepted by both partners, then the project is sent to the quotation process and they update their respective QuotationProcess for this consolidated decision.

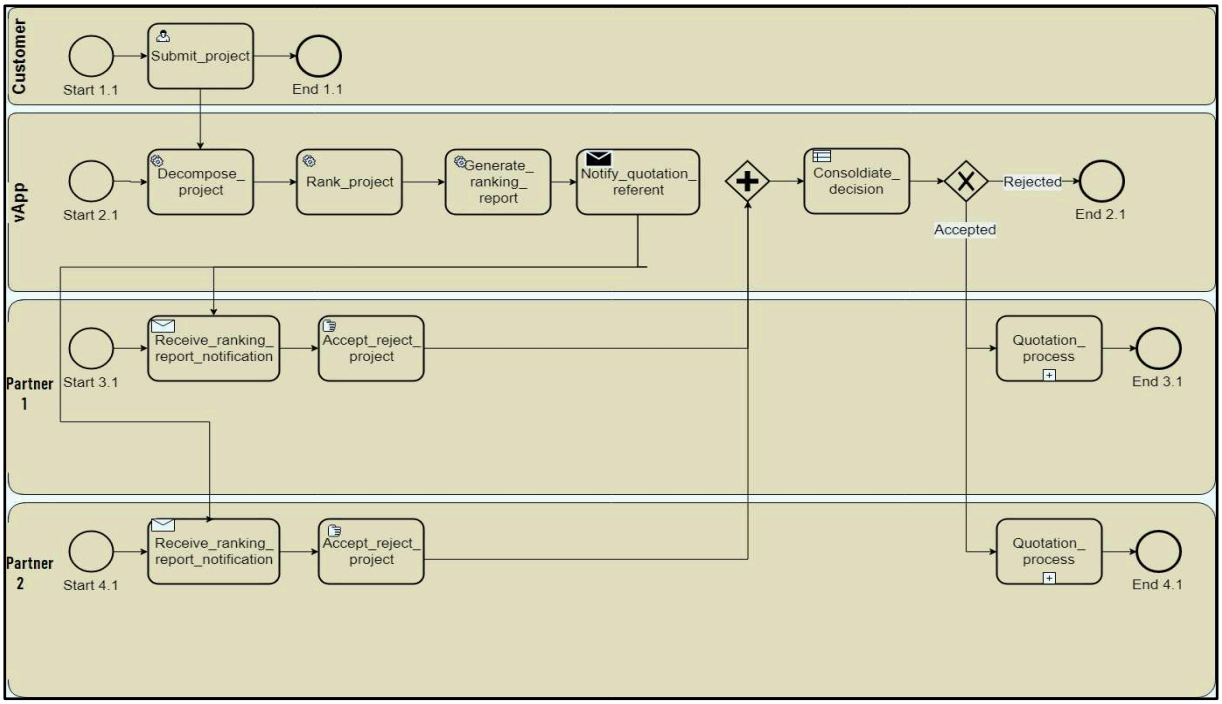


Figure 38: CollaborationAnalyzer Schema

Since the application is supported by different service tasks, it is very time consuming to follow the traditional process of development. It incurs cost, time, resources and trust problems. Therefore, in the perspective of maximizing service reuse, we identify in the FIWARE for Industry portal (http://www.fiware4industry.com/) some interesting APIs to be reused. For example, 3DScan offers comprehensive modules for 3D visualization of high density or high-resolution files consisting of millions of points in the format of point clouds (.txt) and mesh (.stl). Additionally, it provides the management interface for 3D file storage with relational database. There are two open source components provided by the specific enabler that are storage and visualization. The specific enabler aims at offering assistance for performing quality controls and an intuitive visualization decision support system to determine if the analyzed manufactured part must be accepted or rejected. This service is supported by 6 API that are developed in Java. SDScan application is shown in Figure 39. This figure demonstrates six services of 3DScan application. These services are “get name of stored object”, “get source code of file”, “upload file” “delete file” and “update file”. Get name of stored object gets the file path of the stored object. Get source code of file gets the Java script of the file. Upload file service creates new file. Delete file service deletes existing file. Finally update file service writes to the file.

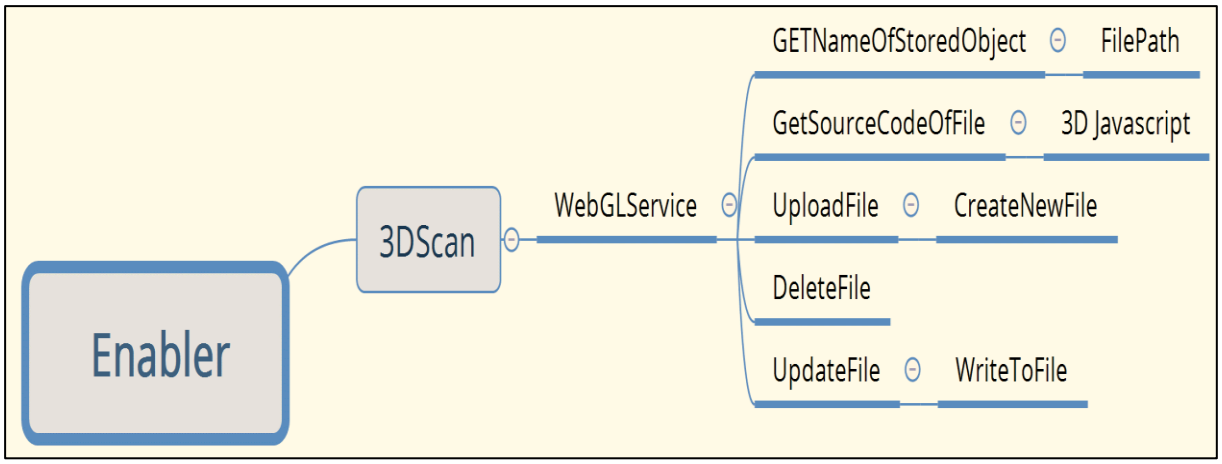


Figure 39: SDScan Application

The above defined use case of business process can reuse services of this 3DScan application based on the performance evaluation of services. We now explain the use case implementation for performance evaluation.

5.2. Use Case Implementation

Use case implementation includes the evaluation of the whole proposed approach. To validate the proposed concepts and implementation details described in the previous sections, we gather QoS web services inputs from two public repositories (https://github.com/wsdream/wsdream-dataset).

Repository 1: This dataset describes real-world QoS evaluation results from 339 users on 5,825 Web services.

Repository 2: This dataset describes real-world QoS evaluation results from 142 users on 4,500 Web services over 64 different time slices (at 15-minute interval).

The inputs provided from the both repositories are analyzed, aggregated and classified. Only 100 instances (with all necessary performance criteria) for about 10000 services are elected (based on the completeness of the QoS data) for the definition of our classification model. The recommendations are provided based on the rules sets provided in section 4. This rules set is related to performance efficiency, reliability, maturity and risk.

Use case implementation is shown in Figure 40. This figure shows the overall evaluation process. The proposed validation was conducted in an agile and incremental approach. On the one hand, the ontological models are instantiated for the raw data as well as for the traces results. On the other hand, data mining techniques have been applied to generate first level results. Final results involve the evolution of ontologies and semantic rules for service reuse.

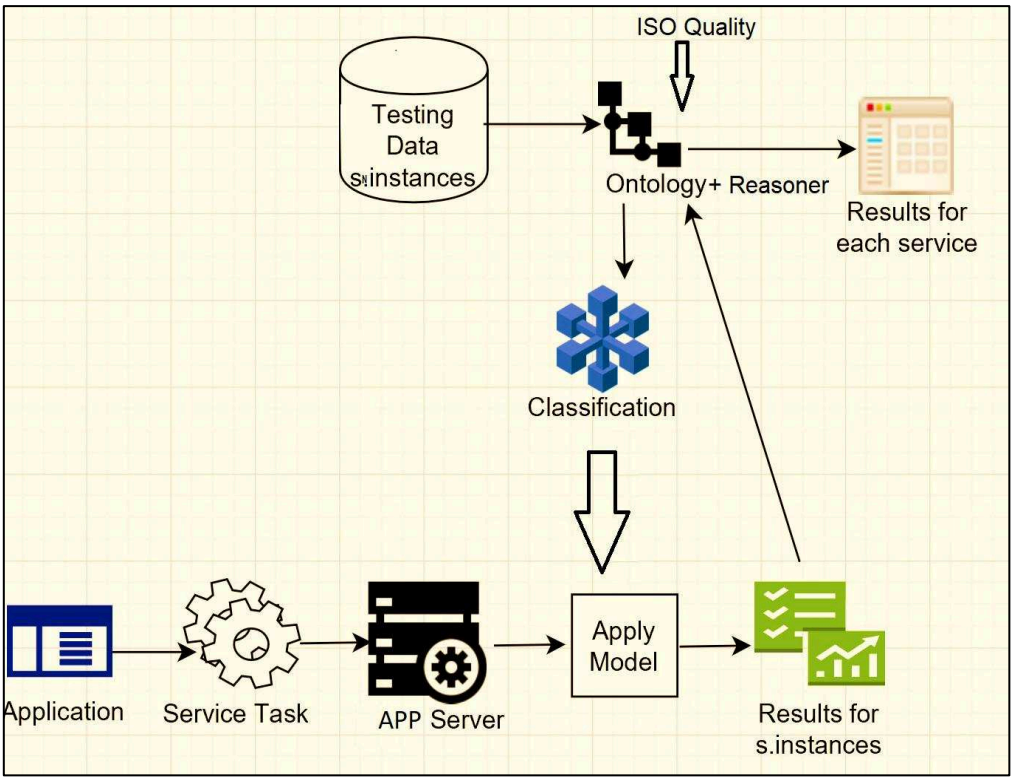


Figure 40: Use Case Implementation

First, raw data of service instances from the existing repository of shared services is instantiated in the proposed ontologies and semantic rules. After this, three classification models have been applied. From these results, we chose the model that generates the most optimum results. Secondly, we deploy application in the performance monitoring server, generates traces and applies the selected model. We consolidate the both results generated from the selected model in the first level as well in the second level. Final consolidated results are based on service instances are instantiated in the ontology in order to generate aggregation rules for each service. Finally, we get the results for each service based on which we provide decisions to the End User in terms of atomic or composite service reuse recommendation. We now explain the data analytics generated at each level of use case implementation.

5.2.1 Data Mining Analytics

Data mining analytics step involves the application of machine learning approach. We chose classification algorithms of machine learning approach in order to analyze the generated semantic rules and traces to get the performance evaluation results for services. Tests are performed in two iterations.

Iteration 1:

In this iteration, analytics are generated by applying three classification algorithms that are decision tree, naïve bayes and support vector machine. We designate 50 percent of data coming from the repository for this iteration. We classify this data by applying classification algorithms and get analytical results. We begin this iteration by illustration performance criteria in Figure 41. First column of the figure demonstrates list of all the technical indicators while second column shows data type of the corresponding technical indicator. Other columns represent average, minimum, maximum and median value for each technical indicator.

	ATTR	DATA_TYPE	AVG	MIN	MAX	MEDIAN_VAL
1	API_ID	VARCHAR2				
2	AVAILABILITY	FLOAT	0,9901	.9800000895535399786	.9999929311865179601	0,99
3	AVAILABLE_MEMORY	FLOAT	0,4905	.000009685254989749259466	.9976842760997554047	0,4905
4	BANDWIDTH	FLOAT	10	10	10	10
5	DELAY	FLOAT	0,5008	.00242592351801771075	.9999503843346091962	0,4987
6	FAILURE_RATE	FLOAT	0,05	.000108271497638269311	.09998056390402004146	0,0492
7	INSTANCE_ID	VARCHAR2				
8	MATURITY	FLOAT	2,0075	2	3	2
9	MTTR	FLOAT	0,0541	.00002688593952734551792	.1079784545367448647	0,0539
10	PROBABILITY	FLOAT	0,0502	.00003003176455564330698	.09999617075970687533	0,0492
11	RECOMMENDATION	VARCHAR2				
12	RESPONSETIME	FLOAT	0,5066	.001080719347649194527	.9998884902793459184	0,5105
13	RISK	FLOAT	0,0121	.000006006352911128661396	.09985639826451767525	0,0084
14	SERVICE_TIME	FLOAT	0,3273	.000043562703322015120 39	.9871540078952042627	0,2866
15	SEVERITY	FLOAT	0,2401	.090909090909090909	1	0,1667
16	THROUGHPUT_SEC	FLOAT	0,138	.000003494786802132468453	.2775581837582948073	0,1396

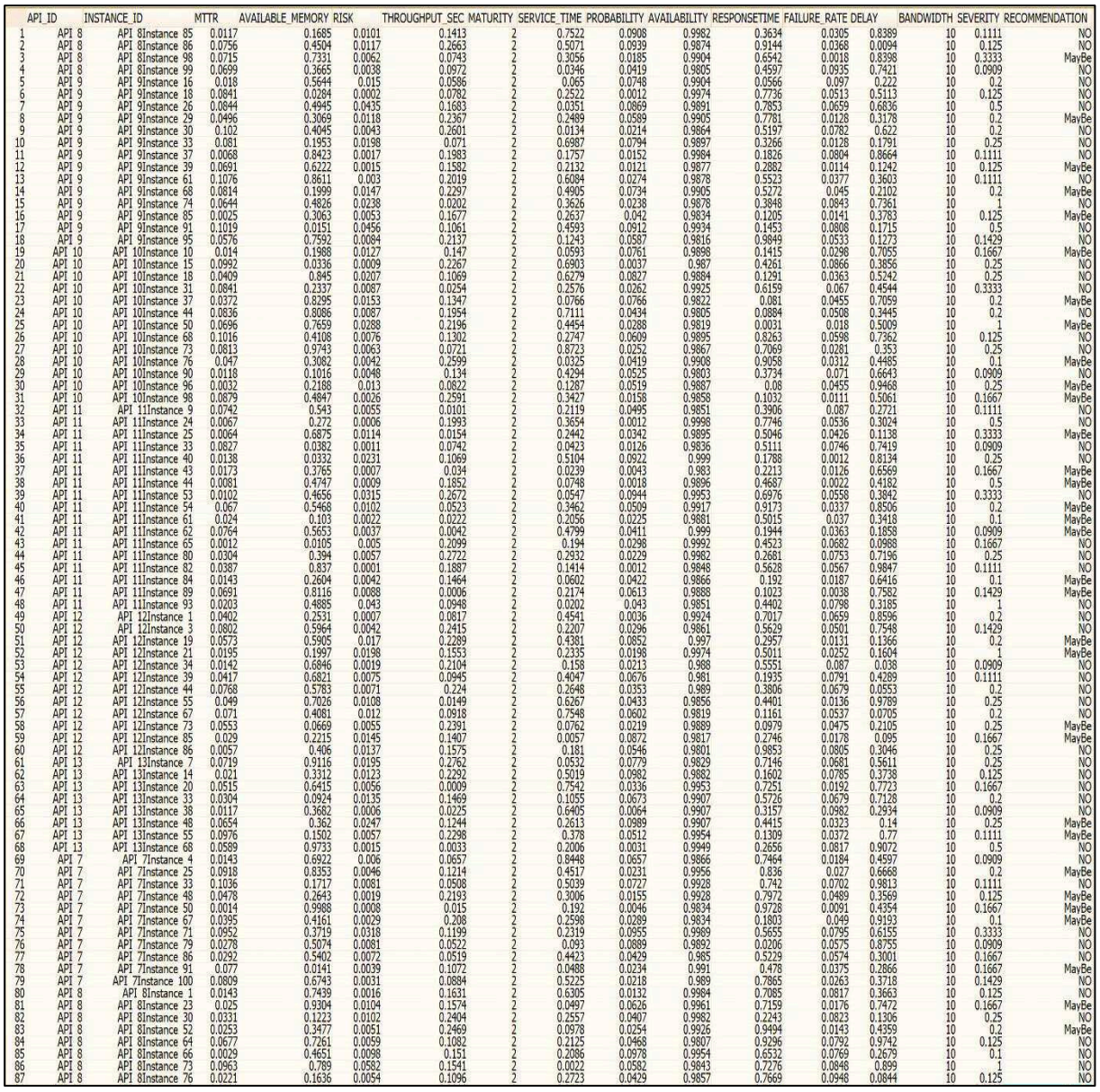
Figure 41: Performance Criteria

After the formulation of performance criteria, we construct a model based on classification algorithms. For this purpose, we define rules to evaluate the recommendation weights such as OK, NO and MayBe. Basic rules to generate recommendations weight are shown in Figure 42. We follow the pattern of SLA to define these rules. It starts by defining the recommendation NO. After the assignment of all weights, the remaining monitoring data is updated to the recommendation weight NO. The other recommendation weights are OK and MayBe. Recommendation weight is assigned to OK if the maturity is greater then and equal to 3 OR maturity is greater then and equal to 2 AND risk is less than or equal to 0.0001 AND available memory is greater then and equal to 0.8 AND failure rate is less then and equal to 0.03. Another predicate is used to assign the recommendation weight as OK that is if maturity is greater then and equal to 2 AND risk is less then and equal to 0.001 AND available memory is greater then and equal to 0.5 AND failure rate is less then and equal to 0.05. Recommendation weight is assigned as MayBe, if maturity is greater then and equal to 2 AND service time is less then and equal to 0.5 AND failure rate is less then and equal to 0.05.

```
Update MonitoringData
set Recommendation = 'NO'

Update MonitoringData
set Recommendation = 'OK'
where
Maturity >=3 OR (Maturity >=2 and Risk <=0.0001 and Available_Memory >=0.8 and
Failure_Rate <= 0.03)

Update MonitoringData
set Recommendation = 'OK'
where
Maturity >=2 and and Risk <=0.001 and Available_Memory >=0.5 and Failure_Rate <= 0.05

Update MonitoringData
set Recommendation = 'MayBe'
where
Maturity >=2 and Service_Time <= 0.5 and Failure_Rate <= 0.05
```

Figure 42: Basic Rules for the Assignment of Recommendation Weights

After applying these rules for all algorithms, we get the analytics as shown in Figure 43. This analytical result shows service ID, service instance ID, values of technical indicators and recommendation based. Technical indicators values are generated based on the semantic rules defined in section 4 while recommendations are generated based on the rules displayed above in Figure 42.



	API_ID	INSTANCE_ID	MTTR	AVAILABLE_MEMORY	RISK	THROUGHPUT_SEC	MATURITY	SERVICE_TIME	PROBABILITY	AVAILABILITY	RESPONSETIME	FAILURE_RATE	DELAY	BANDWIDTH	SEVERITY	RECOMMENDATION
1	API 8	API 8Instance 85	0.0117	0.1685	0.0101	0.1413	2	0.7522	0.0908	0.9982	0.3634	0.0305	0.8389	10	0.1111	NO
2	API 8	API 8Instance 86	0.0756	0.4504	0.0117	0.2663	2	0.5071	0.0939	0.9874	0.9144	0.0368	0.0094	10	0.125	NO
3	API 8	API 8Instance 98	0.0715	0.7331	0.0062	0.0743	2	0.3056	0.0185	0.9904	0.6542	0.0018	0.8398	10	0.3333	MayBe
4	API 8	API 8Instance 99	0.0699	0.3665	0.0038	0.0972	2	0.0346	0.0419	0.9805	0.4597	0.0935	0.7421	10	0.0909	NO
5	API 9	API 9Instance 16	0.018	0.5644	0.015	0.0586	2	0.065	0.0748	0.9904	0.0566	0.097	0.222	10	0.2	NO
6	API 9	API 9Instance 18	0.0841	0.0284	0.0002	0.0782	2	0.2522	0.0012	0.9974	0.7736	0.0513	0.5113	10	0.125	NO
7	API 9	API 9Instance 26	0.0844	0.4945	0.0435	0.1683	2	0.0351	0.0869	0.9891	0.7853	0.0659	0.6836	10	0.5	NO
8	API 9	API 9Instance 29	0.0496	0.3069	0.0118	0.2367	2	0.2489	0.0589	0.9905	0.7781	0.0128	0.3178	10	0.2	MayBe
9	API 9	API 9Instance 30	0.102	0.4045	0.0043	0.2601	2	0.0134	0.0214	0.9864	0.5197	0.0782	0.622	10	0.2	NO
10	API 9	API 9Instance 33	0.081	0.1953	0.0198	0.071	2	0.6987	0.0794	0.9897	0.3266	0.0128	0.1791	10	0.25	NO
11	API 9	API 9Instance 37	0.0068	0.8423	0.0017	0.1983	2	0.1757	0.0152	0.9984	0.1826	0.0804	0.8664	10	0.1111	NO
12	API 9	API 9Instance 39	0.0691	0.6222	0.0015	0.1582	2	0.2132	0.0121	0.9877	0.2882	0.0114	0.1242	10	0.125	MayBe
13	API 9	API 9Instance 61	0.1076	0.8611	0.003	0.2019	2	0.6084	0.0274	0.9878	0.5523	0.0377	0.3603	10	0.1111	NO
14	API 9	API 9Instance 68	0.0814	0.1999	0.0147	0.2297	2	0.4905	0.0734	0.9905	0.5272	0.045	0.2102	10	0.2	MayBe
15	API 9	API 9Instance 74	0.0644	0.4826	0.0238	0.0202	2	0.3626	0.0238	0.9878	0.3848	0.0843	0.7361	10	1	NO
16	API 9	API 9Instance 85	0.0025	0.3063	0.0053	0.1677	2	0.2637	0.042	0.9834	0.1205	0.0141	0.3783	10	0.125	MayBe
17	API 9	API 9Instance 91	0.1019	0.0151	0.0456	0.1061	2	0.4593	0.0912	0.9934	0.1453	0.0808	0.1715	10	0.5	NO
18	API 9	API 9Instance 95	0.0576	0.7592	0.0084	0.2137	2	0.1243	0.0587	0.9816	0.9849	0.0533	0.1273	10	0.1429	NO
19	API 10	API 10Instance 10	0.014	0.1988	0.0127	0.147	2	0.0593	0.0761	0.9898	0.1415	0.0298	0.7055	10	0.1667	MayBe
20	API 10	API 10Instance 15	0.0992	0.0336	0.0009	0.2267	2	0.6903	0.0037	0.987	0.4261	0.0866	0.3856	10	0.25	NO
21	API 10	API 10Instance 18	0.0409	0.845	0.0207	0.1069	2	0.6279	0.0827	0.9884	0.1291	0.0363	0.5242	10	0.25	NO
22	API 10	API 10Instance 31	0.0841	0.2337	0.0087	0.0254	2	0.2576	0.0262	0.9925	0.6159	0.067	0.4544	10	0.3333	NO
23	API 10	API 10Instance 37	0.0372	0.8295	0.0153	0.1347	2	0.0766	0.0766	0.9822	0.081	0.0455	0.7059	10	0.2	MayBe
24	API 10	API 10Instance 44	0.0836	0.8086	0.0087	0.1954	2	0.7111	0.0434	0.9805	0.0884	0.0508	0.3445	10	0.2	NO
25	API 10	API 10Instance 50	0.0696	0.7659	0.0288	0.2196	2	0.4454	0.0288	0.9819	0.0031	0.018	0.5009	10	1	MayBe
26	API 10	API 10Instance 68	0.1016	0.4108	0.0076	0.1302	2	0.2747	0.0609	0.9895	0.8263	0.0598	0.7362	10	0.125	NO
27	API 10	API 10Instance 73	0.0813	0.9743	0.0063	0.0721	2	0.8723	0.0252	0.9867	0.7069	0.0281	0.353	10	0.25	NO
28	API 10	API 10Instance 76	0.047	0.3082	0.0042	0.2599	2	0.0325	0.0419	0.9908	0.9058	0.0312	0.4485	10	0.1	MayBe
29	API 10	API 10Instance 90	0.0118	0.1016	0.0048	0.134	2	0.4294	0.0525	0.9803	0.3734	0.071	0.6643	10	0.0909	NO
30	API 10	API 10Instance 96	0.0032	0.2188	0.013	0.0822	2	0.1287	0.0519	0.9887	0.08	0.0455	0.9468	10	0.25	MayBe
31	API 10	API 10Instance 98	0.0879	0.4847	0.0026	0.2591	2	0.3427	0.0158	0.9858	0.1032	0.0111	0.5061	10	0.1667	MayBe
32	API 11	API 11Instance 9	0.0742	0.543	0.0055	0.0101	2	0.2119	0.0495	0.9851	0.3906	0.087	0.2721	10	0.1111	NO
33	API 11	API 11Instance 24	0.0067	0.272	0.0006	0.1993	2	0.3654	0.0012	0.9998	0.7746	0.0536	0.3024	10	0.5	NO
34	API 11	API 11Instance 25	0.0064	0.6875	0.0114	0.0154	2	0.2442	0.0342	0.9895	0.5046	0.0426	0.1138	10	0.3333	MayBe
35	API 11	API 11Instance 33	0.0827	0.0382	0.0011	0.0742	2	0.0423	0.0126	0.9836	0.5111	0.0746	0.7419	10	0.0909	NO
36	API 11	API 11Instance 40	0.0138	0.0332	0.0231	0.1069	2	0.5104	0.0922	0.999	0.1788	0.0012	0.8134	10	0.25	NO
37	API 11	API 11Instance 43	0.0173	0.3765	0.0007	0.034	2	0.0239	0.0043	0.983	0.2213	0.0126	0.6569	10	0.1667	MayBe
38	API 11	API 11Instance 44	0.0081	0.4747	0.0009	0.1852	2	0.0748	0.0018	0.9896	0.4687	0.0022	0.4182	10	0.5	MayBe
39	API 11	API 11Instance 53	0.0102	0.4656	0.0315	0.2672	2	0.0547	0.0944	0.9953	0.6976	0.0558	0.3842	10	0.3333	NO
40	API 11	API 11Instance 54	0.067	0.5468	0.0102	0.0523	2	0.3462	0.0509	0.9917	0.9173	0.0337	0.8506	10	0.2	MayBe
41	API 11	API 11Instance 61	0.024	0.103	0.0022	0.0222	2	0.2056	0.0225	0.9881	0.5015	0.037	0.3418	10	0.1	MayBe
42	API 11	API 11Instance 62	0.0764	0.5653	0.0037	0.0042	2	0.4799	0.0411	0.999	0.1944	0.0363	0.1858	10	0.0909	MayBe
43	API 11	API 11Instance 65	0.0012	0.0105	0.005	0.2099	2	0.194	0.0298	0.9992	0.4523	0.0682	0.0988	10	0.1667	NO
44	API 11	API 11Instance 80	0.0304	0.394	0.0057	0.2722	2	0.2932	0.0229	0.9982	0.2681	0.0753	0.7196	10	0.25	NO
45	API 11	API 11Instance 82	0.0387	0.837	0.0001	0.1887	2	0.1414	0.0012	0.9848	0.5628	0.0567	0.9847	10	0.1111	NO
46	API 11	API 11Instance 84	0.0143	0.2604	0.0042	0.1464	2	0.0602	0.0422	0.9866	0.192	0.0187	0.6416	10	0.1	MayBe
47	API 11	API 11Instance 89	0.0691	0.8116	0.0088	0.0006	2	0.2174	0.0613	0.9888	0.1023	0.0038	0.7582	10	0.1429	MayBe
48	API 11	API 11Instance 93	0.0203	0.4885	0.043	0.0948	2	0.0202	0.043	0.9851	0.4402	0.0798	0.3185	10	1	NO
49	API 12	API 12Instance 1	0.0402	0.2531	0.0007	0.0817	2	0.4541	0.0036	0.9924	0.7017	0.0659	0.8596	10	0.2	NO
50	API 12	API 12Instance 3	0.0802	0.5964	0.0042	0.2415	2	0.2207	0.0296	0.9861	0.5629	0.0501	0.7548	10	0.1429	NO
51	API 12	API 12Instance 19	0.0573	0.5905	0.017	0.2289	2	0.4381	0.0852	0.997	0.2957	0.0131	0.1366	10	0.2	MayBe
52	API 12	API 12Instance 21	0.0195	0.1997	0.0198	0.1553	2	0.2335	0.0198	0.9974	0.5011	0.0252	0.1604	10	1	MayBe
53	API 12	API 12Instance 34	0.0142	0.6846	0.0019	0.2104	2	0.158	0.0213	0.988	0.5551	0.087	0.038	10	0.0909	NO
54	API 12	API 12Instance 39	0.0417	0.6821	0.0075	0.0945	2	0.4047	0.0676	0.981	0.1935	0.0791	0.4289	10	0.1111	NO
55	API 12	API 12Instance 44	0.0768	0.5783	0.0071	0.224	2	0.2648	0.0353	0.989	0.3806	0.0679	0.0553	10	0.2	NO
56	API 12	API 12Instance 55	0.049	0.7026	0.0108	0.0149	2	0.6267	0.0433	0.9856	0.4401	0.0136	0.9789	10	0.25	NO
57	API 12	API 12Instance 67	0.071	0.4081	0.012	0.0918	2	0.7548	0.0602	0.9819	0.1161	0.0537	0.0705	10	0.2	NO
58	API 12	API 12Instance 73	0.0553	0.0669	0.0055	0.2391	2	0.0762	0.0219	0.9889	0.0979	0.0475	0.2105	10	0.25	MayBe
59	API 12	API 12Instance 85	0.029	0.2215	0.0145	0.1407	2	0.0057	0.0872	0.9817	0.2746	0.0178	0.095	10	0.1667	MayBe
60	API 12	API 12Instance 86	0.0057	0.406	0.0137	0.1575	2	0.181	0.0546	0.9801	0.9853	0.0805	0.3046	10	0.25	NO
61	API 13	API 13Instance 7	0.0719	0.9116	0.0195	0.2762	2	0.0532	0.0779	0.9829	0.7146	0.0681	0.5611	10	0.25	NO
62	API 13	API 13Instance 14	0.021	0.3312	0.0123	0.2292	2	0.5019	0.0982	0.9882	0.1602	0.0785	0.3738	10	0.125	NO
63	API 13	API 13Instance 20	0.0515	0.6415	0.0056	0.0009	2	0.7542	0.0336	0.9953	0.7251	0.0192	0.7723	10	0.1667	NO
64	API 13	API 13Instance 33	0.0304	0.0924	0.0135	0.1469	2	0.1055	0.0673	0.9907	0.5726	0.0679	0.7128	10	0.2	NO
65	API 13	API 13Instance 38	0.0117	0.3682	0.0006	0.0225	2	0.6405	0.0064	0.9907	0.3157	0.0982	0.2934	10	0.0909	NO
66	API 13	API 13Instance 48	0.0654	0.362	0.0247	0.1244	2	0.2613	0.0989	0.9907	0.4415	0.0323	0.14	10	0.25	MayBe
67	API 13	API 13Instance 55	0.0976	0.1502	0.0057	0.2298	2	0.378	0.0512	0.9954	0.1309	0.0372	0.77	10	0.1111	MayBe
68	API 13	API 13Instance 68	0.0589	0.9733	0.0015	0.0033	2	0.2006	0.0031	0.9949	0.2656	0.0817	0.9072	10	0.5	NO
69	API 7	API 7Instance 4	0.0143	0.6922	0.006	0.0657	2	0.8448	0.0657	0.9866	0.7464	0.0184	0.4597	10	0.0909	NO
70	API 7	API 7Instance 25	0.0918	0.8353	0.0046	0.1214	2	0.4517	0.0231	0.9956	0.836	0.027	0.6668	10	0.2	MayBe
71	API 7	API 7Instance 33	0.1036	0.1717	0.0081	0.0508	2	0.5039	0.0727	0.9928	0.742	0.0702	0.9813	10	0.1111	NO
72	API 7	API 7Instance 48	0.0478	0.2643	0.0019	0.2193	2	0.3006	0.0155	0.9928	0.7972	0.0489	0.3569	10	0.125	MayBe
73	API 7	API 7Instance 50	0.0014	0.9988	0.0008	0.015	2	0.192	0.0046	0.9834	0.9728	0.0091	0.4354	10	0.1667	MayBe
74	API 7	API 7Instance 67	0.0395	0.4161	0.0029	0.208	2	0.2598	0.0289	0.9834	0.1803	0.049	0.9193	10	0.1	MayBe
75	API 7	API 7Instance 71	0.0952	0.3719	0.0318	0.1199	2	0.2319	0.0955	0.9989	0.5655	0.0795	0.6155	10	0.3333	NO
76	API 7	API 7Instance 79	0.0278	0.5074	0.0081	0.0522	2	0.093	0.0889	0.9892	0.0206	0.0575	0.8755	10	0.0909	NO
77	API 7	API 7Instance 86	0.0292	0.5402	0.0072	0.0519	2	0.4423	0.0429	0.985	0.5229	0.0574	0.3001	10	0.1667	NO
78	API 7	API 7Instance 91	0.077	0.0141	0.0039	0.1072	2	0.0488	0.0234	0.991	0.478	0.0375	0.2866	10	0.1667	MayBe
79	API 7	API 7Instance 100	0.0809	0.6743	0.0031	0.0884	2	0.5225	0.0218	0.989	0.7865	0.0263	0.3718	10	0.1429	NO
80	API 8	API 8Instance 1	0.0143	0.7439	0.0016	0.1631	2	0.6305	0.0132	0.9984	0.7085	0.0817	0.3663	10	0.125	NO
81	API 8	API 8Instance 23	0.025	0.9304	0.0104	0.1574	2	0.0497	0.0626	0.9961	0.7159	0.0176	0.7472	10	0.1667	MayBe
82	API 8	API 8Instance 30	0.0331	0.1223	0.0102	0.2404	2	0.2557	0.0407	0.9982	0.2243	0.0823	0.1306	10	0.25	NO
83	API 8	API 8Instance 52	0.0253	0.3477	0.0051	0.2469	2	0.0978	0.0254	0.9926	0.9494	0.0143	0.4359	10	0.2	MayBe
84	API 8	API 8Instance 64	0.0677	0.7261	0.0059	0.1082	2	0.2125	0.0468	0.9807	0.9296	0.0792	0.9742	10	0.125	NO
85	API 8	API 8Instance 66	0.0029	0.4651	0.0098	0.151	2	0.2086	0.0978	0.9954	0.6532	0.0769	0.2679	10	0.1	NO
86	API 8	API 8Instance 73	0.0963	0.789	0.0582	0.1541	2	0.0022	0.0582	0.9843	0.7276	0.0848	0.899	10	1	NO
87	API 8	API 8Instance 76	0.0221	0.1636	0.0054	0.1096	2	0.2723	0.0429	0.9857	0.7669	0.0948	0.0844	10	0.125	NO

Figure 43: Analytics of Classification Algorithms

After the construction of analytical results from all models, we evaluate them based on some parameters. The parameters that we chose to evaluate classification algorithms are predictive confidence and precision. Parametric result of classification algorithm is shown in Figure 44. This result shows that decision tree algorithm is the most optimum with precision percentage of 99,8787 and predictive confidence of 99,7247.

Nom	% de confiance prédictive	% d'exactitude globale	% de précision moyenne	Coût	Algorithme	Date de création
CLAS_DT_1_7	99,7247	99,8787	99,8165	271,75378093	Arborescence de décision	27/08/18 15:32
CLAS_NB_1_7	94,7271	95,0156	96,4847		Naive Bayes	27/08/18 15:32
CLAS_SVM_1_7	84,4374	85,1296	89,6249		Machine à vecteurs de s...	27/08/18 15:35

Figure 44: Parametric Result of Classification Algorithms

Since it is proved from the results that decision tree is the most optimum therefore we select decision tree model to be applied in the next iteration. We now explain the working of next iteration two.

Iteration 2:

Iteration 2 involves the deployment of existing 3DScan application in performance monitoring server and gathered traces from it. After this, selected optimum decision tree algorithm is applied on the traces gathered from the server. Finally, it consolidates the both results generated from the model application on traces of server as well as the results generated in iteration 1. Traces management is already explained in chapter 3. We now explain the application of decision tree model on deployed 3DScan application traces.

After applying deploying 3DScan application, we first evaluate results based on technical indicators. For instance, we show 3DScan results based on risk level in Figure 45. This analytical result shows service ID, service instance ID, values of technical indicators and recommendation based on rules. Technical indicators values are produced based on the combined results of traces gathered from server as well as results generated in iteration 1 while recommendations are generated based on the rules displayed above in Figure 42.

	INSTANCE_ID	API_ID	MTTR	AVAILABLE_MEMORY	RISK	THROUGHPUT_SEC	MATURITY	RECOMMENDATION	SERVICE_TIME	PROBABILITY	AVAILABILITY	RESPONSETIME	FAILURE_RATE	DELAY	BANDWIDTH	SEVERITY
1	API 8Instance 85	API 8	0.0117	0.1685	0.0101	0.1413	2	NO	0.7522	0.0908	0.9982	0.3634	0.0305	0.8389	10	0.1111
2	API 8Instance 86	API 8	0.0756	0.4504	0.0117	0.2663	2	NO	0.5071	0.0939	0.9874	0.9144	0.0368	0.0094	10	0.125
3	API 8Instance 98	API 8	0.0715	0.7331	0.0062	0.0743	2	MaYBe	0.3056	0.0185	0.9904	0.6542	0.0018	0.8398	10	0.3333
4	API 8Instance 99	API 8	0.0699	0.3665	0.0038	0.0972	2	NO	0.0346	0.0419	0.9805	0.4597	0.0935	0.7421	10	0.0909
5	API 9Instance 16	API 9	0.018	0.5644	0.015	0.0586	2	NO	0.065	0.0748	0.9904	0.0566	0.097	0.222	10	0.2
6	API 9Instance 18	API 9	0.0841	0.0284	0.0002	0.0782	2	NO	0.2522	0.0012	0.9974	0.7736	0.0513	0.5113	10	0.125
7	API 9Instance 26	API 9	0.0844	0.4945	0.0435	0.1683	2	NO	0.0351	0.0869	0.9891	0.7853	0.0659	0.6836	10	0.5
8	API 9Instance 29	API 9	0.0496	0.3069	0.0118	0.2367	2	MaYBe	0.2489	0.0589	0.9905	0.7781	0.0128	0.3178	10	0.2
9	API 9Instance 30	API 9	0.102	0.4045	0.0043	0.2601	2	NO	0.0134	0.0214	0.9864	0.5197	0.0782	0.622	10	0.2
10	API 9Instance 33	API 9	0.081	0.1953	0.0198	0.071	2	NO	0.6987	0.0794	0.9897	0.3266	0.0128	0.1791	10	0.25
11	API 9Instance 37	API 9	0.0068	0.8423	0.0017	0.1983	2	NO	0.1757	0.0152	0.9984	0.1826	0.0804	0.8664	10	0.1111
12	API 9Instance 39	API 9	0.0691	0.6222	0.0015	0.1582	2	MaYBe	0.2132	0.0121	0.9877	0.2882	0.0114	0.1242	10	0.125
13	API 9Instance 61	API 9	0.1076	0.8611	0.003	0.2019	2	NO	0.6084	0.0274	0.9878	0.5523	0.0377	0.3603	10	0.1111
14	API 9Instance 68	API 9	0.0814	0.1999	0.0147	0.2297	2	MaYBe	0.4905	0.0734	0.9905	0.5272	0.045	0.2102	10	0.2
15	API 9Instance 74	API 9	0.0644	0.4826	0.0238	0.0202	2	NO	0.3626	0.0238	0.9878	0.3848	0.0843	0.7361	10	1
16	API 9Instance 85	API 9	0.0025	0.3063	0.0053	0.1677	2	MaYBe	0.2637	0.042	0.9834	0.1205	0.0141	0.3783	10	0.125
17	API 9Instance 91	API 9	0.1019	0.0151	0.0456	0.1061	2	NO	0.4593	0.0912	0.9934	0.1453	0.0808	0.1715	10	0.5
18	API 9Instance 95	API 9	0.0576	0.7592	0.0084	0.2137	2	NO	0.1243	0.0587	0.9816	0.9849	0.0533	0.1273	10	0.1429
19	API 10Instance 10	API 10	0.014	0.1988	0.0127	0.147	2	MaYBe	0.0593	0.0761	0.9898	0.1415	0.0298	0.7055	10	0.1667
20	API 10Instance 15	API 10	0.0992	0.0336	0.0009	0.2267	2	NO	0.6903	0.0037	0.987	0.4261	0.0866	0.3856	10	0.25
21	API 10Instance 18	API 10	0.0409	0.845	0.0207	0.1069	2	NO	0.6279	0.0827	0.9684	0.1291	0.0363	0.5242	10	0.25
22	API 10Instance 31	API 10	0.0841	0.2337	0.0087	0.0254	2	NO	0.2576	0.0262	0.9925	0.6159	0.067	0.4544	10	0.3333
23	API 10Instance 37	API 10	0.0372	0.8295	0.0153	0.1347	2	MaYBe	0.0766	0.0766	0.9822	0.081	0.0455	0.7059	10	0.2
24	API 10Instance 44	API 10	0.0836	0.8086	0.0087	0.1954	2	NO	0.7111	0.0434	0.9805	0.0884	0.0508	0.3445	10	0.2
25	API 10Instance 50	API 10	0.0696	0.7659	0.0288	0.2196	2	MaYBe	0.4454	0.0288	0.9819	0.0031	0.018	0.5009	10	1
26	API 10Instance 68	API 10	0.1016	0.4108	0.0076	0.1302	2	NO	0.2747	0.0609	0.9895	0.8263	0.0598	0.7362	10	0.125
27	API 10Instance 73	API 10	0.0813	0.9743	0.0063	0.0721	2	NO	0.8723	0.0252	0.9867	0.7069	0.0281	0.353	10	0.25
28	API 10Instance 76	API 10	0.047	0.3082	0.0042	0.2599	2	MaYBe	0.0325	0.0419	0.9908	0.9058	0.0312	0.4485	10	0.1
29	API 10Instance 90	API 10	0.0118	0.1016	0.0048	0.134	2	NO	0.4294	0.0525	0.9803	0.3734	0.071	0.6643	10	0.0909
30	API 10Instance 96	API 10	0.0032	0.2188	0.013	0.0822	2	MaYBe	0.1287	0.0519	0.9887	0.08	0.0455	0.9468	10	0.25
31	API 10Instance 98	API 10	0.0879	0.4847	0.0026	0.2591	2	MaYBe	0.3427	0.0158	0.9858	0.1032	0.0111	0.5061	10	0.1667
32	API 11Instance 9	API 11	0.0742	0.543	0.0055	0.0101	2	NO	0.2119	0.0495	0.9851	0.3906	0.087	0.2721	10	0.1111
33	API 11Instance 24	API 11	0.0067	0.272	0.0006	0.1993	2	NO	0.3654	0.0012	0.9998	0.7746	0.0536	0.3024	10	0.5
34	API 11Instance 25	API 11	0.0064	0.6875	0.0114	0.0154	2	MaYBe	0.2442	0.0342	0.9895	0.5046	0.0426	0.1138	10	0.3333
35	API 11Instance 33	API 11	0.0827	0.0382	0.0011	0.0742	2	NO	0.0423	0.0126	0.9836	0.5111	0.0746	0.7419	10	0.0909
36	API 11Instance 40	API 11	0.0138	0.0332	0.0231	0.1069	2	NO	0.5104	0.0922	0.999	0.1788	0.0012	0.8134	10	0.25
37	API 11Instance 43	API 11	0.0173	0.3765	0.0007	0.034	2	MaYBe	0.0239	0.0043	0.983	0.2213	0.0126	0.6569	10	0.1667
38	API 11Instance 44	API 11	0.0081	0.4747	0.0009	0.1852	2	MaYBe	0.0748	0.0018	0.9896	0.4687	0.0022	0.4182	10	0.5
39	API 11Instance 53	API 11	0.0102	0.4656	0.0315	0.2672	2	NO	0.0547	0.0944	0.9953	0.6976	0.0558	0.3842	10	0.3333
40	API 11Instance 54	API 11	0.067	0.5468	0.0102	0.0523	2	MaYBe	0.3462	0.0509	0.9917	0.9173	0.0337	0.8506	10	0.2
41	API 11Instance 61	API 11	0.024	0.103	0.0022	0.0222	2	MaYBe	0.2056	0.0225	0.9881	0.5015	0.037	0.3418	10	0.1
42	API 11Instance 62	API 11	0.0764	0.5653	0.0037	0.0042	2	MaYBe	0.4799	0.0411	0.999	0.1944	0.0363	0.1858	10	0.0909
43	API 11Instance 65	API 11	0.0012	0.0105	0.005	0.2099	2	NO	0.194	0.0298	0.9992	0.4523	0.0682	0.0988	10	0.1667
44	API 11Instance 80	API 11	0.0304	0.394	0.0057	0.2722	2	NO	0.2932	0.0229	0.9982	0.2681	0.0753	0.7196	10	0.25
45	API 11Instance 82	API 11	0.0387	0.837	0.0001	0.1887	2	NO	0.1414	0.0012	0.9848	0.5628	0.0567	0.9847	10	0.1111
46	API 11Instance 84	API 11	0.0143	0.2604	0.0042	0.1464	2	MaYBe	0.0602	0.0422	0.9866	0.192	0.0187	0.6416	10	0.1
47	API 11Instance 89	API 11	0.0691	0.8116	0.0088	0.0006	2	MaYBe	0.2174	0.0613	0.9888	0.1023	0.0038	0.7582	10	0.1429
48	API 11Instance 93	API 11	0.0203	0.4885	0.043	0.0948	2	NO	0.0202	0.043	0.9851	0.4402	0.0798	0.3185	10	1
49	API 12Instance 1	API 12	0.0402	0.2531	0.0007	0.0817	2	NO	0.4541	0.0036	0.9924	0.7017	0.0659	0.8596	10	0.2
50	API 12Instance 3	API 12	0.0802	0.5964	0.0042	0.2415	2	NO	0.2207	0.0296	0.9861	0.5629	0.0501	0.7548	10	0.1429
51	API 12Instance 19	API 12	0.0573	0.5905	0.017	0.2289	2	MaYBe	0.4381	0.0852	0.997	0.2957	0.0131	0.1366	10	0.2
52	API 12Instance 21	API 12	0.0195	0.1997	0.0198	0.1553	2	MaYBe	0.2335	0.0198	0.9974	0.5011	0.0252	0.1604	10	1
53	API 12Instance 34	API 12	0.0142	0.6846	0.0019	0.2104	2	NO	0.158	0.0213	0.988	0.5551	0.087	0.038	10	0.0909
54	API 12Instance 39	API 12	0.0417	0.6821	0.0075	0.0945	2	NO	0.4047	0.0676	0.981	0.1935	0.0791	0.4289	10	0.1111
55	API 12Instance 44	API 12	0.0768	0.5783	0.0071	0.224	2	NO	0.2648	0.0353	0.989	0.3806	0.0679	0.0553	10	0.2
56	API 12Instance 55	API 12	0.049	0.7026	0.0108	0.0149	2	NO	0.6267	0.0433	0.9856	0.4401	0.0136	0.9789	10	0.25
57	API 12Instance 67	API 12	0.071	0.4081	0.012	0.0918	2	NO	0.7548	0.0602	0.9819	0.1161	0.0537	0.0705	10	0.2
58	API 12Instance 73	API 12	0.0553	0.0669	0.0055	0.2391	2	MaYBe	0.0762	0.0219	0.9889	0.0979	0.0475	0.2105	10	0.25
59	API 12Instance 85	API 12	0.029	0.2215	0.0145	0.1407	2	MaYBe	0.0057	0.0872	0.9817	0.2746	0.0178	0.095	10	0.1667
60	API 12Instance 86	API 12	0.0057	0.406	0.0137	0.1575	2	NO	0.181	0.0546	0.9801	0.9853	0.0805	0.3046	10	0.25
61	API 13Instance 7	API 13	0.0719	0.9116	0.0195	0.2762	2	NO	0.0532	0.0779	0.9829	0.7146	0.0681	0.5611	10	0.25
62	API 13Instance 14	API 13	0.021	0.3312	0.0123	0.2292	2	NO	0.5019	0.0982	0.9882	0.1602	0.0785	0.3738	10	0.125
63	API 13Instance 20	API 13	0.0515	0.6415	0.0056	0.0009	2	NO	0.7542	0.0336	0.9953	0.7251	0.0192	0.7723	10	0.1667
64	API 13Instance 33	API 13	0.0304	0.0924	0.0135	0.1469	2	NO	0.1055	0.0673	0.9907	0.5726	0.0679	0.7128	10	0.2
65	API 13Instance 38	API 13	0.0117	0.3682	0.0006	0.0225	2	NO	0.6405	0.0064	0.9907	0.3157	0.0982	0.2934	10	0.0909
66	API 13Instance 48	API 13	0.0654	0.362	0.0247	0.1244	2	MaYBe	0.2613	0.0989	0.9907	0.4415	0.0323	0.14	10	0.25
67	API 13Instance 55	API 13	0.0976	0.1502	0.0057	0.2298	2	MaYBe	0.378	0.0512	0.9954	0.1309	0.0372	0.77	10	0.1111
68	API 13Instance 68	API 13	0.0589	0.9733	0.0015	0.0033	2	NO	0.2006	0.0031	0.9949	0.2656	0.0817	0.9072	10	0.5

Figure 45: Results from 3DScan Application

After getting results for each technical indicator, the resulted decision tree model is shown in Figure 46.

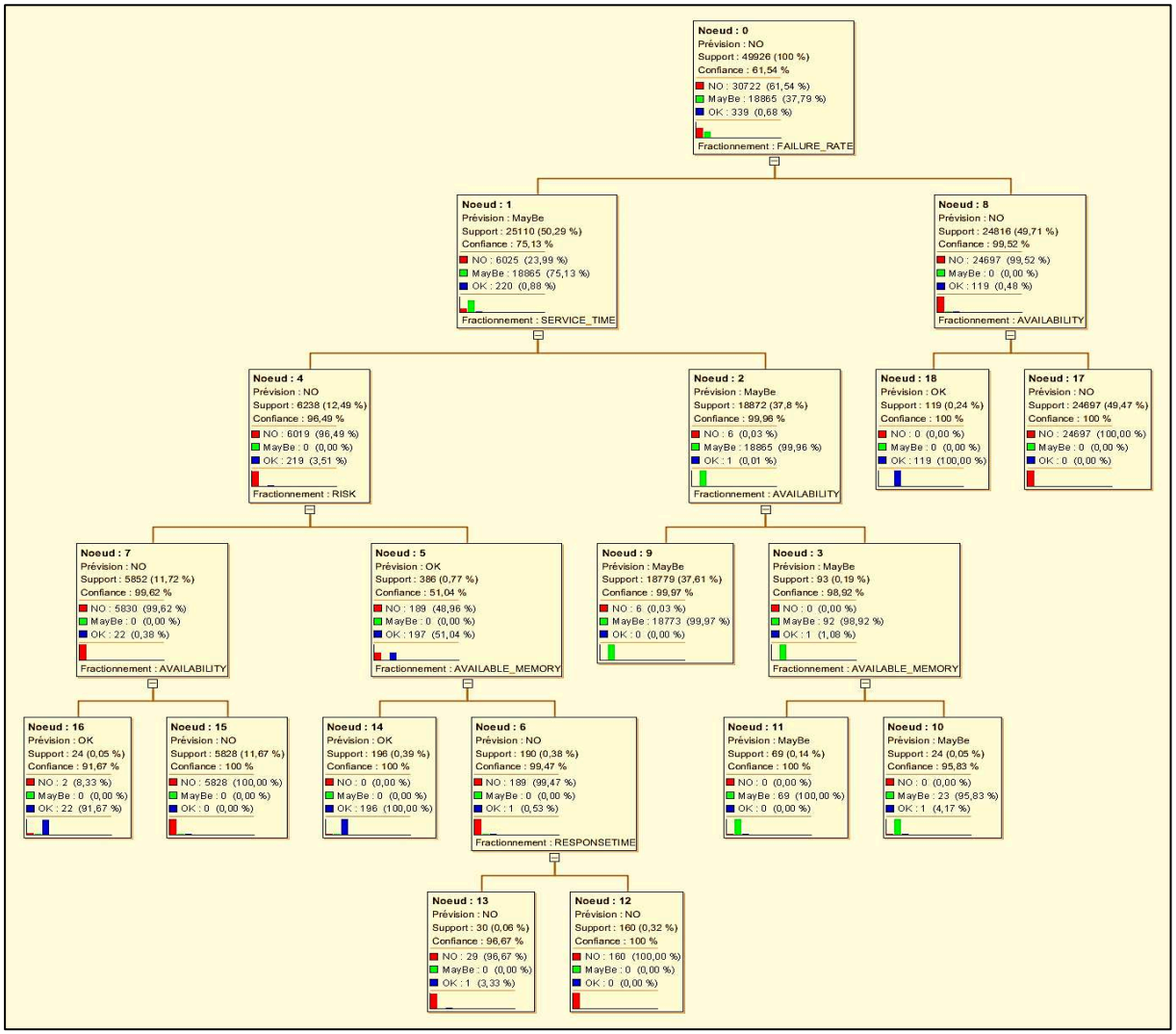


Figure 46: Decision Tree Model

Links between all nodes are generated based on implemented predicates. Node 0 is linked to node 1 based on the predicate FAILURE RATE is less than 0.05001628 else Node 8. Node 1 is connected to two nodes NODE 4, if SERVICE TIME is less then and equal to 0.4999 else Node 2. Similarly, Node 2 is linked to Node 9 on the basis of predicate AVAILABILITY is less than or equal to 0.9999 else Node 3. Same is the case for Node 4 which is connected to Node 7, if RISK greater than 0.000999 else Node 5. Further, Node 7 is connected to Node 16, if AVAILIBILITY is greater than 0.9999 else Node 15. If AVAILABLE MEMORY is greater than 0.5027 then Node 5 is linked to Node 14 else Node 6. Node 6 is further expanded to Node 13 and Node 12 on the basis of RESPONSE TIME. It is connected to Node if RESPONSE TIME is greater than 0.83439 else Node 12. Node 3 is linked to Node 10 If AVAILABLE MEMORY is less than or equal to 0.2611 else Node 11. Node 8 is connected to Node 17 If AVAILABILITY is less than or equal to 0.9998 else Node 18.

Each node in the above decision tree model demonstrates parametric results based on precision, support, prediction and confidence. Prediction is defined as percentage of recommendation results in the form of NO, MAYBE and OK. These results are shown in Figure 47 as well as displayed at each node of decision tree model. Support and confidence values are measured between 0 and 1 values. First column of the figure represents the ID while second column shows parent node ID. Third and fourth columns illustrates node ID and profile ID respectively. Prediction is displayed in column five. Column six, seven and eight constitutes record count, prediction count and total record count. Finally, it highlights confidence and support calculated for the respective nodes in column nine and ten.

	PARENT_NODE_ID	NODE_ID	PROFILE_ID	PREDICTION	RECORD_COUNT	PREDICTION_COUNT	TOTAL_RECORD_COUNT	CONFIDENCE	SUPPORT
1	3	10	2	MayBe	24	23	49,926	0.9583	0.0005
2	3	11	3	MayBe	69	69	49,926	1	0.0014
3	6	12	4	NO	160	160	49,926	1	0.0032
4	6	13	5	NO	30	29	49,926	0.9667	0.0006
5	5	14	6	OK	196	196	49,926	1	0.0039
6	7	15	7	NO	5,828	5,828	49,926	1	0.1167
7	7	16	8	OK	24	22	49,926	0.9167	0.0005
8	8	17	9	NO	24,697	24,697	49,926	1	0.4947
9	8	18	10	OK	119	119	49,926	1	0.0024
10	2	9	1	MayBe	18,779	18,773	49,926	0.9997	0.3761

Figure 47: Results from Decision Tree Model

Cost matrix for decision tree model is shown in Figure 48. First column of the figure shows the target weights. These target weights are MAYBE, NO and OK. The second column presents the MAYBE value for each target weight while third column shows NO value for each target weight. Finally, last column displays OK value for each target weight. From this result, we can see that there are 0 instances for the class of MayBe with MayBe. 2.6465 instances are resulted for the class of MayBe with NO and same number of instances justified the class of MayBe with OK. 1.6251 instances are resulted for the class of NO with MayBe and NO with OK while 0 instance justified for NO with NO class. Class OK with MayBe and class OK with NO justified 147.2743 instances while 0 instance fulfilled the class of OK with OK. From these results, we can see that 0 cost is benefited in case of the class MayBe with MayBe, OK with OK and NO with NO.

Weight	MayBe	NO	OK
MayBe	0	2.6465	2.6465
NO	1.6251	0	1.6251
OK	147.2743	147.2743	0

Figure 48: Cost Matrix for Decision Tree Model

The next part uncovers different decision scenarios developed for the capitalization of service reuse based on the above performance-oriented data mining analytics.

5.2.2 Decision scenarios for Service Reuse

Decision scenarios on service reuse are displayed through an interface manipulated by an IT manager who is interested to get performance analysis as well as recommendation report. The decision part allows the End User to accompany the evolution of the trajectory of the performance (performance efficiency, availability, maturity and risk) with time. First level decision interface displays information describing the specific decision based on priority of End User. Second level decision interface displays information describing the decision for the recommendation based on instances. Final level decision interface shows the global decision for each service.

In order to describe the application of classification algorithms, we develop scenarios. A scenario designates a goal for which an End User might use a software and all the features of the software that they would require in order to achieve the desired goal. We will provide in detail the unfolding of PODSS scenarios. We will also present the ability of the system to make a fast and reliable decision.

Scenario 1: **Decision Results based on priority of End User**

Specific decisions are made based on the priority of technical indicators specified by End User. These decision results can be provided in both graph and tabular form as desired by End User. We show a graph-based representation for the recommendation of service based on the priority of response_time as shown in Figure 49. This figure shows the response time fluctuations of service instances in different time intervals. From this graph, we recommend service who has less percentage recommendation of NO with minimum response time. Horizontal axis shows the response time values that starts with <0.0001 and ends at >=0.0009. Vertical axis shows the percentage of recommendation class NO that starts with 0 and goes until 14.

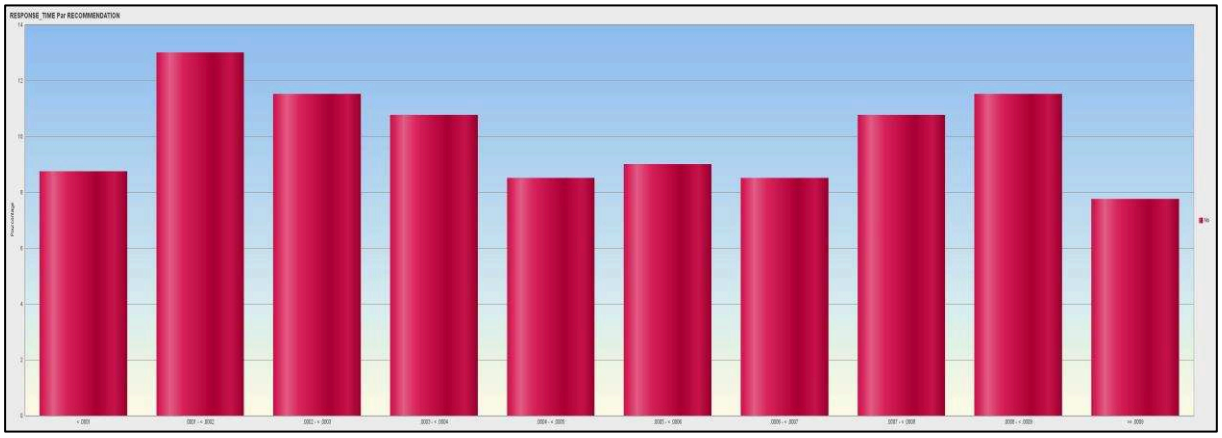


Figure 49: Recommendation of Service Based on Response_Time

A graph-based representation for the recommendation of service based on the priority of Service_instance is shown in Figure 50. This figure shows the service instance time deviations in

different time intervals. From this graph, we recommend service who has less percentage recommendation of NO with minimum service instance time. Horizontal axis shows the values of service instance time that starts with <0.10472 and ends with >0.89819. Vertical axis shows the percentage of recommendation class NO that starts with 0 and ends with 14.

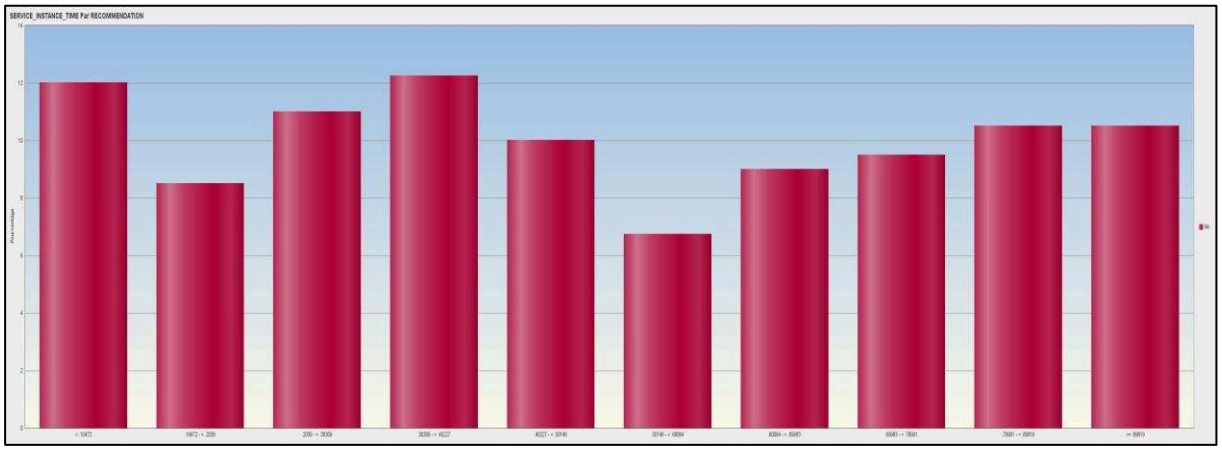


Figure 50: Recommendation of Service Based on Service_Instance_Time

A graph-based representation for the recommendation of service based on the priority of Throughput is shown in Figure 51. This figure shows the throughput variations in different time intervals. From this graph, we recommend service who has less percentage recommendation of class NO with less throughput percentage. Horizontal axis shows the values of throughput that starts with <1.05216 and ends with NULL. Vertical axis shows the percentage of recommendation class NO that starts with 0 and ends with 14.

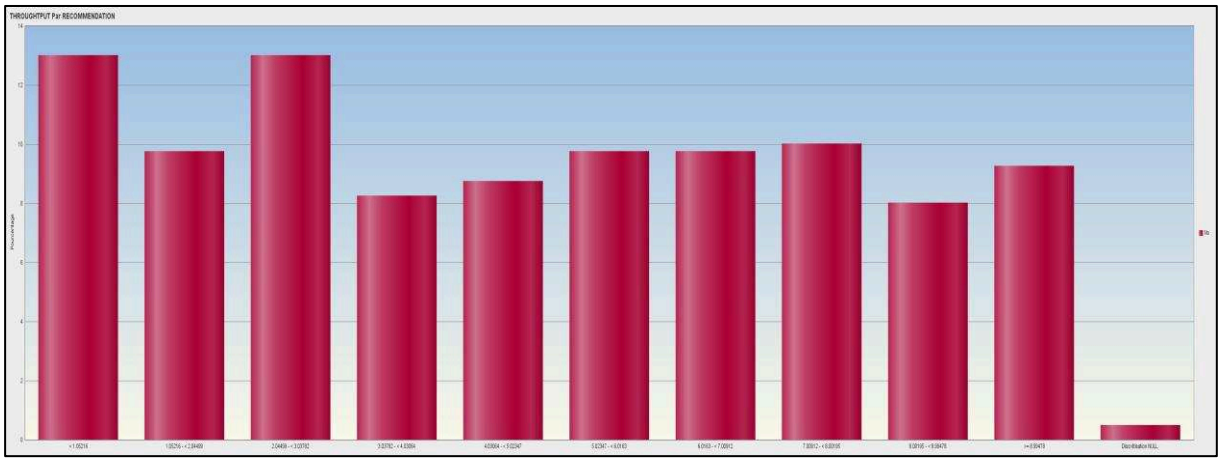
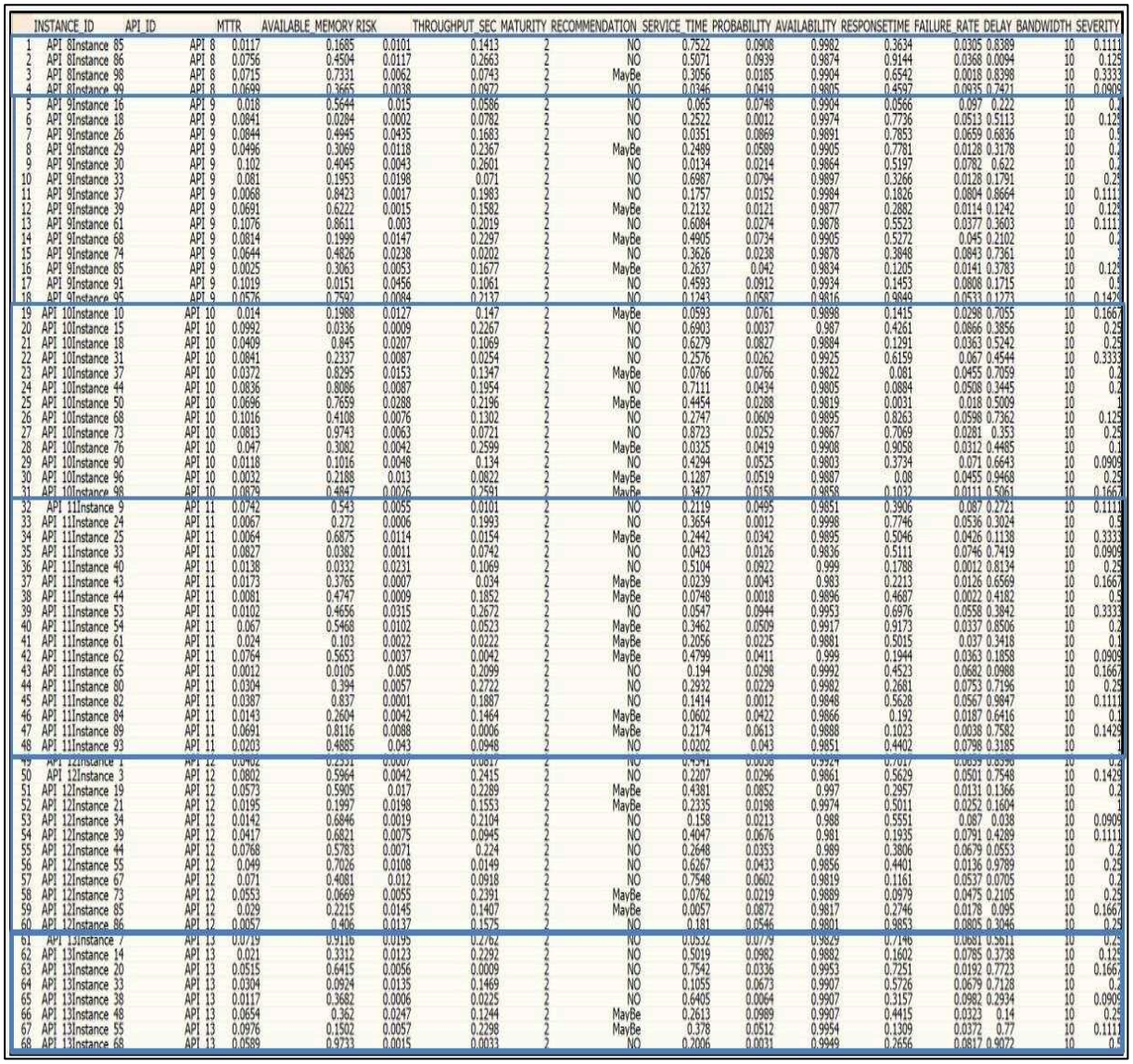


Figure 51: Recommendation of Service Based on Throughput

Scenario 2: **Decision Results based on Service Instances**

Second level decision interface provide a performance matrix based on the recommendation weights for all instances. A sample of decision results based on service instances is shown in Figure 52. For instance, we only show the results of six services separated by blue lines. First service involves four instances while second service involves thirteen instances. Third service is instantiated twelve times

and the fourth service is executed sixteen times. Fifth and sixth services are instantiated eleven and seven times respectively.



	INSTANCE_ID	API_ID	MTTR	AVAILABLE_MEMORY	RISK	THROUGHPUT_SEC	MATURITY	RECOMMENDATION	SERVICE_TIME	PROBABILITY	AVAILABILITY	RESPONSETIME	FAILURE_RATE	DELAY	BANDWIDTH	SEVERITY
1	API 8Instance 85	API 8	0.0117	0.1685	0.0101	0.1413	2	NO	0.7522	0.0908	0.9982	0.3634	0.0305	0.8389	10	0.1111
2	API 8Instance 86	API 8	0.0756	0.4504	0.0117	0.2663	2	NO	0.5071	0.0939	0.9874	0.9144	0.0368	0.0094	10	0.125
3	API 8Instance 98	API 8	0.0715	0.7331	0.0062	0.0743	2	MayBe	0.3056	0.0185	0.9904	0.6542	0.0018	0.8398	10	0.3333
4	API 8Instance 99	API 8	0.0699	0.3665	0.0038	0.0972	2	NO	0.0346	0.0419	0.9805	0.4597	0.0035	0.7421	10	0.0909
5	API 9Instance 16	API 9	0.018	0.5644	0.015	0.0586	2	NO	0.065	0.0748	0.9904	0.0566	0.097	0.222	10	0.2
6	API 9Instance 18	API 9	0.0841	0.0284	0.0002	0.0782	2	NO	0.2522	0.0012	0.9974	0.7736	0.0513	0.5113	10	0.125
7	API 9Instance 26	API 9	0.0844	0.4945	0.0435	0.1683	2	NO	0.0351	0.0869	0.9891	0.7853	0.0659	0.6836	10	0.5
8	API 9Instance 29	API 9	0.0496	0.3069	0.0118	0.2367	2	MayBe	0.2489	0.0589	0.9905	0.7781	0.0128	0.3178	10	0.2
9	API 9Instance 30	API 9	0.102	0.4045	0.0043	0.2601	2	NO	0.0134	0.0214	0.9864	0.5197	0.0782	0.622	10	0.2
10	API 9Instance 33	API 9	0.081	0.1953	0.0198	0.071	2	NO	0.6987	0.0794	0.9897	0.3266	0.0128	0.1791	10	0.25
11	API 9Instance 37	API 9	0.0068	0.8423	0.0017	0.1983	2	NO	0.1757	0.0152	0.9984	0.1826	0.0804	0.8664	10	0.1111
12	API 9Instance 39	API 9	0.0691	0.6222	0.0015	0.1582	2	MayBe	0.2132	0.0121	0.9877	0.2882	0.0114	0.1242	10	0.125
13	API 9Instance 61	API 9	0.1076	0.8611	0.003	0.2019	2	NO	0.6084	0.0274	0.9878	0.5523	0.0377	0.3603	10	0.1111
14	API 9Instance 68	API 9	0.0814	0.1999	0.0147	0.2297	2	MayBe	0.4905	0.0734	0.9905	0.5272	0.045	0.2102	10	0.2
15	API 9Instance 74	API 9	0.0644	0.4826	0.0238	0.0202	2	NO	0.3626	0.0238	0.9878	0.3848	0.0843	0.7361	10	1
16	API 9Instance 85	API 9	0.0025	0.3063	0.0053	0.1677	2	MayBe	0.2637	0.042	0.9834	0.1205	0.0141	0.3783	10	0.125
17	API 9Instance 91	API 9	0.1019	0.0151	0.0456	0.1061	2	NO	0.4593	0.0912	0.9934	0.1453	0.0808	0.1715	10	0.5
18	API 9Instance 95	API 9	0.0576	0.7592	0.0084	0.2137	2	NO	0.1243	0.0587	0.9816	0.9849	0.0533	0.1273	10	0.1429
19	API 10Instance 10	API 10	0.014	0.1988	0.0127	0.147	2	MayBe	0.0593	0.0761	0.9898	0.1415	0.0298	0.7055	10	0.1667
20	API 10Instance 15	API 10	0.0992	0.0336	0.0009	0.2267	2	NO	0.6903	0.0037	0.987	0.4261	0.0866	0.3856	10	0.25
21	API 10Instance 18	API 10	0.0409	0.845	0.0207	0.1069	2	NO	0.6279	0.0827	0.9884	0.1291	0.0363	0.5242	10	0.25
22	API 10Instance 31	API 10	0.0841	0.2337	0.0087	0.0254	2	NO	0.2576	0.0262	0.9925	0.6159	0.067	0.4544	10	0.3333
23	API 10Instance 37	API 10	0.0372	0.8295	0.0153	0.1347	2	MayBe	0.0766	0.0766	0.9822	0.081	0.0455	0.7059	10	0.2
24	API 10Instance 44	API 10	0.0836	0.8086	0.0087	0.1954	2	NO	0.7111	0.0434	0.9805	0.0884	0.0508	0.3445	10	0.2
25	API 10Instance 50	API 10	0.0696	0.7659	0.0288	0.2196	2	MayBe	0.4454	0.0288	0.9819	0.0031	0.018	0.5009	10	1
26	API 10Instance 68	API 10	0.1016	0.4108	0.0076	0.1302	2	NO	0.2747	0.0609	0.9895	0.8263	0.0598	0.7362	10	0.125
27	API 10Instance 73	API 10	0.0813	0.9743	0.0063	0.0721	2	NO	0.8723	0.0252	0.9867	0.7069	0.0281	0.353	10	0.25
28	API 10Instance 76	API 10	0.047	0.3082	0.0042	0.2599	2	MayBe	0.0325	0.0419	0.9908	0.9058	0.0312	0.4485	10	0.1
29	API 10Instance 90	API 10	0.0118	0.1016	0.0048	0.134	2	NO	0.4294	0.0525	0.9803	0.3734	0.071	0.6643	10	0.0909
30	API 10Instance 96	API 10	0.0032	0.2188	0.013	0.0822	2	MayBe	0.1287	0.0519	0.9887	0.08	0.0455	0.9468	10	0.25
31	API 10Instance 98	API 10	0.0879	0.4847	0.0026	0.2591	2	MayBe	0.3427	0.0158	0.9858	0.1032	0.0111	0.5061	10	0.1667
32	API 11Instance 9	API 11	0.0742	0.543	0.0055	0.0101	2	NO	0.2119	0.0495	0.9851	0.3906	0.087	0.2721	10	0.1111
33	API 11Instance 24	API 11	0.0067	0.272	0.0006	0.1993	2	NO	0.3654	0.0012	0.9998	0.7746	0.0536	0.3024	10	0.5
34	API 11Instance 25	API 11	0.0064	0.6875	0.0114	0.0154	2	MayBe	0.2442	0.0342	0.9895	0.5046	0.0426	0.1138	10	0.3333
35	API 11Instance 33	API 11	0.0827	0.0382	0.0011	0.0742	2	NO	0.0423	0.0126	0.9836	0.5111	0.0746	0.7419	10	0.0909
36	API 11Instance 40	API 11	0.0138	0.0332	0.0231	0.1069	2	NO	0.5104	0.0922	0.999	0.1788	0.0012	0.8134	10	0.25
37	API 11Instance 43	API 11	0.0173	0.3765	0.0007	0.034	2	MayBe	0.0239	0.0043	0.983	0.2213	0.0126	0.6569	10	0.1667
38	API 11Instance 44	API 11	0.0081	0.4747	0.0009	0.1852	2	MayBe	0.0748	0.0018	0.9896	0.4687	0.0022	0.4182	10	0.5
39	API 11Instance 53	API 11	0.0102	0.4656	0.0315	0.2672	2	NO	0.0547	0.0944	0.9953	0.6976	0.0558	0.3842	10	0.3333
40	API 11Instance 54	API 11	0.067	0.5468	0.0102	0.0523	2	MayBe	0.3462	0.0509	0.9917	0.9173	0.0337	0.8506	10	0.2
41	API 11Instance 61	API 11	0.024	0.103	0.0022	0.0222	2	MayBe	0.2056	0.0225	0.9881	0.5015	0.037	0.3418	10	0.1
42	API 11Instance 62	API 11	0.0764	0.5653	0.0037	0.0042	2	MayBe	0.4799	0.0411	0.999	0.1944	0.0363	0.1858	10	0.0909
43	API 11Instance 65	API 11	0.0012	0.0105	0.005	0.2099	2	NO	0.194	0.0298	0.9992	0.4523	0.0682	0.0988	10	0.1667
44	API 11Instance 80	API 11	0.0304	0.394	0.0057	0.2722	2	NO	0.2932	0.0229	0.9982	0.2681	0.0753	0.7196	10	0.25
45	API 11Instance 82	API 11	0.0387	0.837	0.0001	0.1887	2	NO	0.1414	0.0012	0.9848	0.5628	0.0567	0.9847	10	0.1111
46	API 11Instance 84	API 11	0.0143	0.2604	0.0042	0.1464	2	MayBe	0.0602	0.0422	0.9866	0.192	0.0187	0.6416	10	0.1
47	API 11Instance 89	API 11	0.0691	0.8116	0.0088	0.0006	2	MayBe	0.2174	0.0613	0.9888	0.1023	0.0038	0.7582	10	0.1429
48	API 11Instance 93	API 11	0.0203	0.4885	0.043	0.0948	2	NO	0.0202	0.043	0.9851	0.4402	0.0798	0.3185	10	1
49	API 12Instance 1	API 12	0.0402	0.2551	0.0007	0.0617	2	NO	0.4541	0.0038	0.9924	0.7017	0.0039	0.8398	10	0.2
50	API 12Instance 3	API 12	0.0802	0.5964	0.0042	0.2415	2	NO	0.2207	0.0296	0.9861	0.5629	0.0501	0.7548	10	0.1429
51	API 12Instance 19	API 12	0.0573	0.5905	0.017	0.2289	2	MayBe	0.4381	0.0852	0.997	0.2957	0.0131	0.1366	10	0.2
52	API 12Instance 21	API 12	0.0195	0.1997	0.0198	0.1553	2	MayBe	0.2335	0.0198	0.9974	0.5011	0.0252	0.1604	10	1
53	API 12Instance 34	API 12	0.0142	0.6846	0.0019	0.2104	2	NO	0.158	0.0213	0.988	0.5551	0.087	0.038	10	0.0909
54	API 12Instance 39	API 12	0.0417	0.6821	0.0075	0.0945	2	NO	0.4047	0.0676	0.981	0.1935	0.0791	0.4289	10	0.1111
55	API 12Instance 44	API 12	0.0768	0.5783	0.0071	0.224	2	NO	0.2648	0.0353	0.989	0.3806	0.0679	0.0553	10	0.2
56	API 12Instance 55	API 12	0.049	0.7026	0.0108	0.0149	2	NO	0.6267	0.0433	0.9856	0.4401	0.0136	0.9789	10	0.25
57	API 12Instance 67	API 12	0.071	0.4081	0.012	0.0918	2	NO	0.7548	0.0602	0.9819	0.1161	0.0537	0.0705	10	0.2
58	API 12Instance 73	API 12	0.0553	0.0669	0.0055	0.2391	2	MayBe	0.0762	0.0219	0.9889	0.0979	0.0475	0.2105	10	0.25
59	API 12Instance 85	API 12	0.029	0.2215	0.0145	0.1407	2	MayBe	0.0057	0.0872	0.9817	0.2746	0.0178	0.095	10	0.1667
60	API 12Instance 86	API 12	0.0057	0.406	0.0137	0.1575	2	NO	0.181	0.0546	0.9801	0.9853	0.0805	0.3046	10	0.25
61	API 13Instance 7	API 13	0.0719	0.9116	0.0195	0.2762	2	NO	0.0532	0.0779	0.9829	0.7146	0.0681	0.5611	10	0.25
62	API 13Instance 14	API 13	0.021	0.3312	0.0123	0.2292	2	NO	0.5019	0.0982	0.9882	0.1602	0.0785	0.3738	10	0.125
63	API 13Instance 20	API 13	0.0515	0.6415	0.0056	0.0009	2	NO	0.7542	0.0336	0.9953	0.7251	0.0192	0.7723	10	0.1667
64	API 13Instance 33	API 13	0.0304	0.0924	0.0135	0.1469	2	NO	0.1055	0.0673	0.9907	0.5726	0.0679	0.7128	10	0.2
65	API 13Instance 38	API 13	0.0117	0.3682	0.0006	0.0225	2	NO	0.6405	0.0064	0.9907	0.3157	0.0982	0.2934	10	0.0909
66	API 13Instance 48	API 13	0.0654	0.362	0.0247	0.1244	2	MayBe	0.2613	0.0989	0.9907	0.4415	0.0323	0.14	10	0.25
67	API 13Instance 55	API 13	0.0976	0.1502	0.0057	0.2298	2	MayBe	0.378	0.0512	0.9954	0.1309	0.0372	0.77	10	0.1111
68	API 13Instance 68	API 13	0.0589	0.9733	0.0015	0.0033	2	NO	0.2006	0.0031	0.9949	0.2656	0.0817	0.9072	10	0.5

Figure 52: Decision Results based on Service Instances

First level evaluation is made based on the maturity for all instances. We display a graph based representation for the maturity evaluation based on instanceID as shown in Figure 53. Horizontal axis of this graph shows the maturity levels that starts from 0 and goes until 3. Left vertical axis of this graph represents percentage of service ID that starts with 0 and goes until 120. Right vertical axis of the graph shows different service ID with instance ID in different colours. The light green colour service which is named as API157 has 100 percent instances that satisfies maturity level 2 while 1 percent instances of the same service lies at maturity level 3.

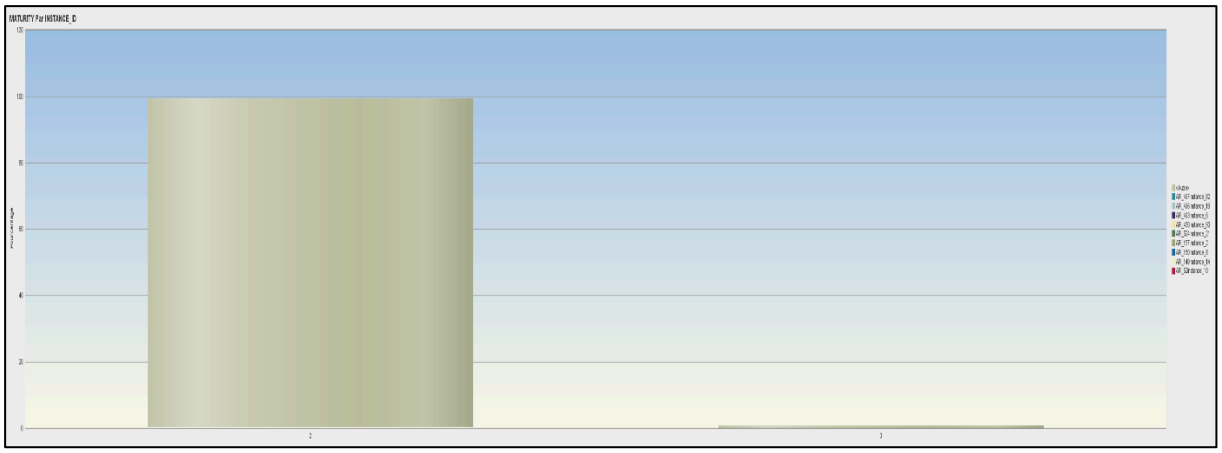


Figure 53: Maturity Evaluation based on instanceID

Second level evaluation is made based on the risk for all instances. We display a graph-based representation for the risk evaluation based on instanceID as shown in Figure 54. Horizontal axis of this graph shows the risk level that starts from <0.00999 and goes until NULL. Left vertical axis of this graph represents percentage of service ID that starts with 0 and goes until 70. Right vertical axis of the graph shows different service ID with instance ID in different colours. The light green colour service which is named as API157 has 60 % instances that resulted with <0.0099 risk level, 25 % instances that resulted with <0.01998 risk level, 7 % instances that resulted with <0.02996 risk level, 5 % instances that resulted with <0.03995 risk level.

Figure 54: Risk Evaluation based on instanceID

Third level evaluation is made based on the risk for all instances. We display a graph based representation for the risk evaluation based on instanceID as shown in Figure 55. Horizontal axis of this graph shows recommendation classes that are NO, MayBe and OK. Left vertical axis of this graph represents percentage of service ID that starts with 0 and goes until 70. Right vertical axis of the graph shows different service ID with instance ID in different colours. The light green colour service which is named as API157 has 60 % instances that resulted with NO recommendation, 39% instances that resulted with MayBe recommendation while 3% instances resulted with OK recommendation.

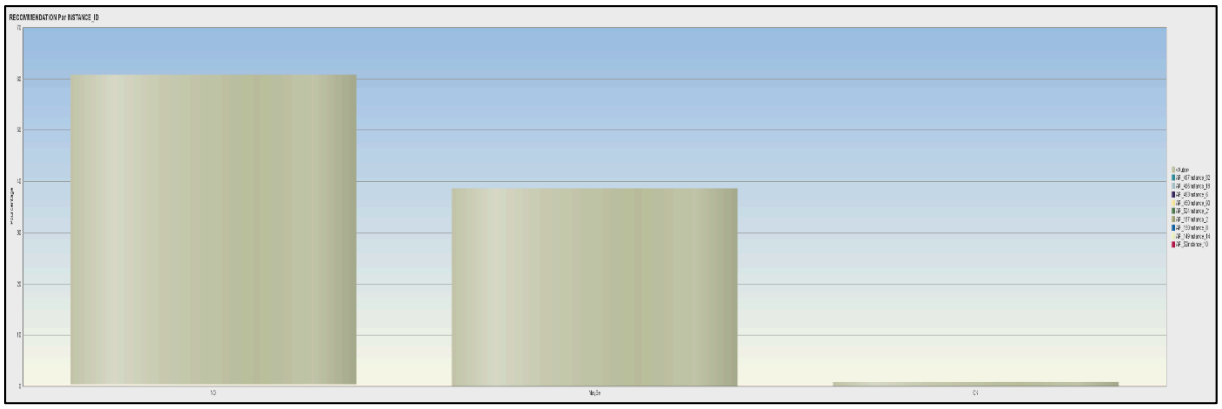


Figure 55: Service Time Evaluation based on instanceID

Scenario 3: **Decision Results for each service**

Decision results for each service are made based on the generated results for all instances. For this purpose, we instantiate these results in ontology and semantic rules already implemented in chapter 4. As a result of this, results are generated for each service. We now present the generated decision result in Table 18 for the 3DScan services.

Table 18: Decision Result for each service

Attributes	Recommendation
Service 1	NO
Service 2	NO
Service 3	May Be
Service 4	NO
Service 5	NO
Service 6	NO

Scenario 4: **Global Decision based on the overall Performance**

Global decision based on the overall performance has also been evaluated. The parameters that we consider for global evaluation are cost, accuracy, precision and confidence. Recommendation prediction and cost analysis matrix for all instances is shown in Figure 56. This matrix displays results in the form of recommendations to reuse service based on the instances. These recommendation results are based on the prediction cases of MAYBE, NO and OK. Other parameters that are illustrated in this figure are based on total instances, percentage of correct instances and cost of each prediction case. From this result, we can see that 18557 instances satisfied the case of MayBe with MayBe while 334 instances fulfilled the case of OK with OK. However, 30523 instances resulted for the case of NO with NO.

	MayBe	NO	OK
MayBe	18 557	2	26
NO	2	30 523	29
OK	0	1	334
Total	18 559	30 526	389
% correct	99,9892	99,9902	85,8612
Coût	3,2502	152,5673	115,9363

Figure 56: Recommendation Prediction and Cost Analysis Matrix

Parametric analysis matrix is shown in Figure 57. Parameters are statistics of average precision, global accuracy and cost for the target values of recommendation weights as well as in total for all instances. It also illustrates the total number of cases.

Précision moyenne: 99,8165
Exactitude globale: 99,8787
Coût total: 271,7538

Valeur cible	Nb total de cas	% de prévisions correctes	Coût	% de coût
NO	30 554	99,8985	50,37777488	18,5380
MayBe	18 585	99,8493	74,10166976	27,2679
OK	335	99,7015	147,27433628	54,1940

Figure 57: Parametric analysis matrix

A graph-based analysis of predictive confidence is shown in Figure 58. Horizontal axis of the graph shows the decision tree classification model while vertical axis of the graph represents percentage values of confidence predictive class. This graph shows that decision tree class resulted with 100 percent confidence with increased number of services and instances.

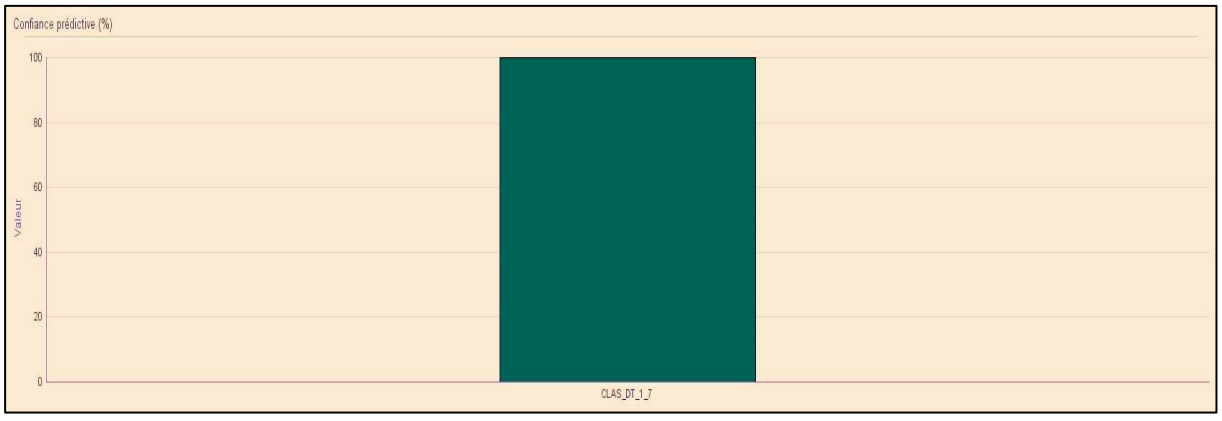


Figure 58: Predictive Confidence

5.2.3 Recommendations

Developing new services for new or changed requirements increases cost in terms of money and time. Therefore, proposed system generates different types of possible recommendations in the context of performance-based service reuse. These recommendations are explained below.

Recommendation 1. Atomic service

A service task is implemented for atomic service. Developing new services for new or changed requirements increases cost in terms of money and time. We recommend atomic services based on the needs of End User. Following are the possible recommendations of atomic services.

1. Atomic service that satisfies functional requirements with performance based on priority of technical indicator specified by End User.
2. Atomic service that satisfies functional requirements with increased performance based on all technical indicators.

Recommendation 2. Composite service

If atomic service is not available or not fulfilling the functional requirements of the user request for service, composite service will be recommended. A composite service is created by combining atomic services together. This involves combination of atomic services together in the form of process to provide requested functionality. Following are the possible recommendations of composite services.

1. Composite service that satisfies functional requirements with performance based on priority of technical indicator specified by End User.
2. Composite service that satisfies functional requirements with increased performance based on all technical indicators.

Recommendation 3. Resources

Resource allocation is recommended based on dynamic QoS provisioning. Recommendations have been made based on the analytical results generated for resources such as memory allocation. It involves the allocation and management of resources at run-time to satisfy certain application QoS requirements. Since, performance of services changes with time, there are fluctuations in resource availability and priority of resource requirements. This involves reallocation of resources based on the estimated results. Note that, our goal is to provide recommendations to the IT Admin. However, this approach will not provide any mechanism to allocate resources. It only provides suggestions based on the estimated results. As a result of this provisioning approach, following actions are suggested to be handled dynamically.

1. Actions to take when performance fluctuates with increased number of consumption of data.

2. Provisioning mechanisms in the infrastructure that need to be measured and controlled dynamically for the triggered event of resource allocation.

3. Provisioned resources restoration

4. Upgradation or migration

The results obtained from our performance system are encouraging and significant. Having interesting solutions shows, that this work has helped a lot in the convergence of performance.

Certainly, the interpretation of the information brought back to the End User interface (performance profile-based service portfolio, assessment report, decision matrix and recommendation of the solution) represents an interesting contribution for the company. An important aspect of our approach is therefore to put in place PODSS to provide an End User with useful information when he or she needs it, as quickly as possible, and in a usable way. However, the major advantage of the proposed system is its ability to quickly and dynamically display the information to the End User.

5.3 Discussion

Current approaches for performance evaluation rely primarily on workflow systems to ensure either monitoring or process monitoring or decision support based on execution traces and are not compliant with latest quality standards. Indeed, these traditional methods are based on the supervision and the monitoring of the behavior of the process, but do not integrate reasoning based on a semantic richness coming from an ontological model.

With the dynamic nature of services, SOA based companies also seeks a dynamic analytical decision support system based on the assessment resulting from performance evaluation to better meet its commitments. Accelerating decision-making is considered today as a strategic resource that can provide a decisive competitive advantage for the company. To do this, our proposal makes it possible to create a learning process from the statistical analysis of technical indicators and from the execution traces. This makes it possible to react fairly quickly to make decisions in a dynamic environment.

Usually, when the End User is involved in a decision support system, he is often confronted with a lot of information, which he has to analyze, synthesize and exploit. It is then necessary to automate certain tasks with high added value to optimize this decision support system. The PODSS is a dynamic system that analyzes the data and displays it to the End User.

For the validation of this research work, a case of application in the SOA industry has been treated in this chapter with a network of partners. On the other hand, the case study described, although simplified, allowed us to illustrate the use of the PODSS and the real interest of such a tool in the complement of the analysis of the SBS over time. We have seen examples of scenarios leading to problems and the contribution of our proposal in the responsiveness of the decision making in terms of the evolution of performance efficiency, availability, maturity and risk for service reuse.

CHAPTER 6. Conclusion and Future Works

6.1 Conclusion

Reuse of services in a SOA environment provides many benefits including improving agility of solutions and reduction of cost. Agility is guaranteed by quickly assembling new business processes from existing services to meet changing marketplace needs. Cost is reduced by not just avoiding duplication of code for enabling similar business functions, but also throughout the SOA life-cycle spanning of service deployment and management.

This research work provides a decision support system called PODSS for accelerating performance of SBS by recommending reuse of services to ensure sustainability. The proposed research work accelerates the analysis of existing service networks to validate service reuse capabilities. PODSS is supported by five models.

1. Performance based technical model

In this model, we exploit ISO/IEC 25010 quality characteristics and their sub characteristics that are relevant to ensure the performance of SBS. Sub characteristics that we exploit are time behavior, resource utilization, capacity, maturity, availability, fault tolerance, recoverability and reusability. We measure these sub characteristics based on qualitative and quantitative technical indicators.

2. Performance based ontological model

This model is composed of two blocks. First block is the creation of structure and the second block is the creation of reasoning. In the first block, first of all we create service domain ontology. We create all the classes, instances, attributes and their relationship in this ontology. In the second step, we create the ontologies of qualitative and quantitative technical indicators for service, process, integration and governance layers of SOA. Finally, we aggregate all proposed ontologies to create a composite performance-oriented decision support ontology that mainly highlights concepts of performance and decision. In the second block, we develop reasoning rules for the evaluation of performance, maturity, risk and recommendation results.

3. Performance based machine learning model

In this model, we apply machine learning models to evaluate knowledge base of performance. We create a training data set to generate recommendation rules in this model. We apply classification algorithms on the training data of services and technical indicators. Classification algorithms are logistic regression, naïve Bayes, support vector machine and decision Trees. After the generation of model, we get the algorithm that provides optimum results which is in our case is decision tree classification algorithm. Therefore, we apply decision tree algorithm on the data generated from the traces of application server. Finally, the results for service instances are aggregated and passed to the ontologies and inference rules to generate results for each service.

4. Performance based decision model

This model is composed of three parts. Three parts are performance management, maturity evaluation and risk management. We propose decision support algorithm for all of these parts. Decision model will generate recommendations as reuse service or process if it is best in terms of performance, maturity and risk. This decision model provides three kinds of decisions. First decision is based on the End User choice. End User give priority to each of these measures and decision model recommend service accordingly. This type of decision is called specific decision. Second decision is based on the performance evaluation of all service instances. Third decision is the accumulative decision based on the performance of service instances for each service. Final decision is called global decision that is based on cost, confidence and precision. Decision model recommend service based on accumulative decision.

5. Performance based validation model

Validation model uses QoS web services inputs from two public repositories. We discuss a business process use case for atomic and composite services recommendation. In this model, we perform several tests by increasing the number of services and instances. We deploy 1000 services and each service is invoked 100 times. We apply classification algorithms on the data in two iterations. In the first iteration, we apply classification algorithms and selected the algorithm that has most optimum result. In the second iteration, we apply the selected decision tree classification algorithm on the data of traces gathered from application server and generates results for service instances. Finally, these results are aggregated and processed in the ontologies and inference rules that generates results for each service. We provide recommendations such as reuse existing service, suggesting composite service and resource allocation.

Our approach may not perfectly represent reality. It would be wise to consider possible discrepancies when considering a point of analysis. It is possible to integrate other quality characteristics like security

and maintainability. PODSS can also be tested in some real time environments. It is also possible to expand PODSS by implementing new services with new SLA and new business process. These limitations will obviously be taken into consideration in future works.

6.2 Future Works

Due to time constraints, this research work still has some works to improve for further research.

1. Inclusion of performance-based Quality Characteristics

There are several quality sub characteristics proposed by ISO/IEC 25010 series as mentioned in the introduction section, we have only included that are important to evaluate performance. However, there remains some quality characteristics that can also be considered in terms of following perspectives:

   a. Security

      Information systems that demands high performance also requires complicated network and computer infrastructure to support distributed collaboration that should be provisioned on-demand. Dynamic performance-based service recommendation also requires consistent security target such security issues.

   b. Portability

      Portability is the usability of the same system in different environments. It is the ability of the system to be transportable to any device or hardware. Portability is the key quality attribute for cost reduction.

   c. Usability

      Proposed system is implemented for End User who is IT Admin. This system does not provide any mechanism to interact with the user who has requested for the service. It would be interesting to provide a mechanism of interaction between the system and the user who intend to use the service.

   d. Compatibility

      Compatibility is the ability of the system to be able to work in any environment regardless of the platform and other dependencies. There are different kind of compatibility tests that can be performed.

2. For the moment, we are recommending reusing existing service and process.
   a. However, there can be a scenario in which it is necessary to implement new service with new SLA.

b. Implementing new business process

3. Test performance on other real cases:

   Experiments will be conducted to improve our results in the future.

4. Calculate the added value from the economic point of view of the application.

REFERENCES

[1]. Arsanjani, A. (2004). "Service-Oriented Modeling and Architecture," https://www.ibm.com/ developer works/library/ws-soa-design1/.

[2]. Simple Object Access Protocol (SOAP) 1.2, Part 0, Primer: (2007). – World Wide Web Consortium, http://www.w3.org/TR/soap12-part0/.

[3]. Chinnici, R., Moreau, J.-J., Ryman, A., and Weerawarana, S. (2007). "Web services description language (wsdl) version 2.0 part 1: Core language," W3C Recomm., vol. 26, p. 19.

[4]. Arsanjani, A., Zhang, L.-J., M. Ellis, Allam, A., and Channabasavaiah, K. (2014). "Design an SOA solution using a reference architecture," IBM Dev.

[5]. Kyusakov, R., Eliasson, J., Delsing, J., van Deventer, J., & Gustafsson, J. (2013). "Integration of Wireless Sensor and Actuator Nodes with IT Infrastructure Using Service-Oriented Architecture," IEEE Transactions on Industrial Informatics, 9(1), 43–51.

[6]. Wang, X., Feng, Z., Huang, K., & Tan, W. (2017). "An Automatic Self-Adaptation Framework for Service-Based Process Based on Exception Handling," Concurrency and computation practice and experience, 29, (5).

[7]. Sachan, D., Dixit, S, K., & Kumar, S. (2014). "Qos Aware Formalized Model for Semantic Web Service Selection," International Journal of Web & Semantic Technology (IJWesT), 5(4).

[8]. http://iso25000.com/index.php/en/iso-25000-standards/iso-25010.

[9]. OMB. (1999). "Federal Enterprise Architecture Framework (FEAF)," Chief Information Officer Council, United States Office of Management and Budget.

[10]. CMMI Product Team. (2010). "CMMI for Development, Version 1.3," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-2010-TR-033.

[11]. Linthicum., and Dave. (2007). "Core Value of a SOA is the Ability to Reuse Services? Not a Chance," DOI: http://www.infoworld.com/archives/emailPrint.jsp?R=printThis&A=http://weblog.infoworld.com/realworldsoa/archives/2007 /10/core_value_of_a.html.

[12]. Chappell., David, A. (2006). "Service Reuse - Fact or Fiction?," O'Reilly XML.COM, DOI= http://www.oreillynet.com/xml/blog/2006/10/service_reuse_fa ct_or_fiction.html.

[13]. McKendrick, Joe. (2006). "Pouring cold water on SOA 'reuse' mantra," ZDNet, DOI= http://blogs.zdnet.com/serviceoriented/?p=699.

[14]. W3C. SWRL: Semantic Web Rules Language. (2004). http://www.w3.org/Submission/SWRL/.

[15]. Prud'Hommeaux, E., and Seaborne, A. (2008). "SPARQL query language for RDF," W3C Recomm., vol. 15.

[16]. Bucchiarone, A., Cappiello, C., Nitto, E. D., Kazhamiakin, R., Mazza, V., & Pistore, M. (2010). "Design for Adaptation of Service-Based Applications: Main Issues and Requirements," In Proceedings of the international conference on Service-oriented computing LNCS Series, 6275: 467–476.

[17]. An Oracle White Paper. (2007). "Using Services Oriented Architecture to Extend JD Edwards EnterpriseOne".

[18]. Java TM Management Extensions (JMX) Specification, (2006). Sun Microsystems, version 1.4, Final Release.

[19]. DeFee, J., and Harmon, P. (2005). “Business Activity Monitoring and Simulation,” in Workflow Handbook, L. Fischer, Ed. Lighthouse Point, FL, USA: Future Strategies, pp. 53– 74.
[20]. Lee, K., Jeon, J., Lee, W., Jeong, S.-H., and Park, S. (2003). “Qos for Web services: Requirements and possible approaches,” W3C Work. Group Note, vol. 25, pp. 1–9.
[21]. Oh, S.-C., Kil, H., Lee, D., and Kumara, S. R. (2006). “Algorithms for Web services Discovery and Composition Based on Syntactic and Semantic Service Descriptions,” in CEC/EEE, p. 66.
[22]. The Open Group. (2013). “TOGAF® Version 9.1,” The Open Group, [Online]. Available: http://www.opengroup.org/togaf/.
[23]. Laskey, K.B. (2006). “Decision Making and Decision Support,” http://ite.gmu.edu/ ˜klaskey/SYST542/DSS_Unit1.pdf
[24]. Han, J., Kamber, M., and Pei, J. (2012). “Data Mining Concepts and Techniques,” Third Edition, The Morgan Kaufmann Series in Data Management Systems, Elsevier.
[25]. Mirandola, R., Potena, P., & Scandurra, P. (2014). “Adaptation Space Exploration for Service-Oriented Applications,” Science of Computer Programming, 80, Part B:356 – 384.
[26]. Aversano, L., di-brino, M., and Tortorella, M. (2016). “Business Process Aware Identification of Reusable Software Components,” 11th International Conference on Software Engineering and Applications, vol. 1: ICSOFT-EA, pp. 59-68.
[27]. Feuerlicht, G., Lozina, J. (2007). “Understanding Service Reusability,” in the Proceedings of the 15th International Conference Systems Integration, June 10-12, Prague, Czech Republic, ISBN 978-80-245-1196-2, 144-150.
[28]. Perepletchikov, M., Ryan, C., Frampton, K. “Cohesion Metrics for Predicting Maintainability of Service-Oriented Software,” QSIC 2007: 328-335.
[29]. Xue, G., Liu, J., Wu, L., Yao, S. (2018). “A Graph Based Technique of Process Partitioning,” Journal of Web Engineering, Vol.17, No. 1&2, pp.121-140.
[30]. Khoshkbarforoushha, A., Jamshidi, P., and Shams, F. (2010). “A metric for composite service reusability analysis,” Proceedings of the 2010 ICSE Workshop on Emerging Trends in Software Metrics (WETSoM '10), ACM, New York, USA, pp.67-74.
[31]. Choi, Si W., and Kim, Soo D. (2008). “A quality model for evaluating reusability of services in SOA,” In 2008 10th IEEE Conference on ECommerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, pages 293–298. IEEE.
[32]. Doultsinou, A., Roy, R., Baxter, DI., and Gao, JX. (2009). “Developing a Service Knowledge Reuse Framework for Engineering Design,” Journal of Engineering Design 20(4):389–411.
[33]. Lee, D., Muthig, M., Naab, A. (2010). “Feature-oriented approach for developing reusable product line assets of service-based systems,” Journal of Systems and Software 83 (7), pp 1123–1136.
[34]. Ahmed-Kristensen, S., and Vianello, G. (2015). “A model for reusing service knowledge based on an empirical case,” Research in Engineering Design, 26, 57-76.
[35]. Allen, A. A., Costa, F. M., and Clarke, P. J. (2016). “A user-centric approach to dynamic adaptation of reusable communication services,” Personal and Ubiquitous Computing, vol. 20, no. 2, pp. 209–227.
[36]. The Open Group. (2009). “SOA Governance Framework”, [Online]. Available: https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?catalogno=c093.
[37]. Janiesch, C., Niemann, M., Repp, N. (2009). “Towards a service governance framework for the internet of services,” in 17th European Conference on Information Systems.

[38]. Filho, H M T., and Azevedo, L G. (2012). "CommonGOV: A consolidate approach for governance of service-oriented architecture," Fourth International Conference on Computational Aspects of Social Networks (CASoN).
[39]. Joachim, N., Beimborn, D. and Weitzel, T. (2013). "The influence of SOA governance mechanisms on IT flexibility and service reuse," The Journal of Strategic Information Systems 22 (1), 86-101.
[40]. Dan, A., Johnson, R.D., Carrato, T. (2008). "SOA Service Reuse by Design," In: SDSOA'08, Leipzig, Germany.
[41]. Kim, Y., Choi, J-S., and Shin, Y. (2014). "A Decision Model for Optimizing the Service Portfolio in SOA Governance," 4th World Congress on Information and Communication Technologies.
[42]. Oriol, M., Franch, X., & Marco J. (2010). "SALMon: A SOA System for Monitoring Service Level Agreements," Universitat Politècnica de Catalunya Technical Report, LSI-10-18-R.
[43]. Garcia-Valls, M., Lopez, I, R., & Villar, L, F. (2012). "iLAND: An Enhanced Middleware for Real-Time Reconfiguration of Service Oriented Distributed Real-Time Systems," IEEE Trans. Ind. Inf., 99:1–1.
[44]. Asadollah, S. A., & Chiew, T. K. (2011). "Web service Response Time Monitoring: Architecture and Vaidation," Q.Zhou (Ed.): ICTMF, CCIS 164, © Springer-Verlag Berlin Heidelberg.
[45]. Avila, S. D. G., & Djemame, K. (2013). "Fuzzy Logic Based QoS Optimization Mechanism for Service Composition," in 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering:182–191.
[46]. Garcia-Valls, M., & Basanta-Val, P. (2013). "A Real-Time Perspective of Service Composition: Key Concepts and Some Contributions," JSA.
[47]. Zheng, Z., Zhang, Y., & Lyu, M. R. (2014). "Investigating QoS of Real-World Web services," IEEE Transactions on Services Computing.
[48]. Kahlon, N. K., Kaur, K., & Narang, S. B. (2014). "Web services Monitoring: A Life Cycle process," IUP Journal of Information Technology X, (3):51-60.
[49]. Mckee, D. W., Webster, D., & Xu, J. (2015). "Enabling Decision Support for the Delivery of Real-Time Services," in IEEE 15th International Symposium on High-Assurance Systems Engineering.
[50]. Boumahdi, F., Chalal, R., Guendouz, A., & Gasmia, K. (2016). "SOA+D: A New Way to Design the Decision in SOA – Based On the New Standard Decision Model and Notation (DMN)," Service Oriented Computing and Applications 10(1): 35-53.
[51]. Aschoff, R., & A. Zisman. (2011). "QoS-driven Proactive Adaptation of Service Composition," In the proceedings of International Conference on Service Oriented Computing (ICSOC 2011).
[52]. Fan, X-Q. (2013). "A Decision-Making Method for Personalized Composite Service," Expert Systems with Applications:5804–5810.
[53]. Sheng, Q. Z., Qiao, X., Vasilakos, A. V., Szabo, C., Bourne, S., & Xu. X. (2014). "Web services Composition: A Decades Overview," Information Sciences,280: 218–238.
[54]. Tari, Z., Phan, A. K. A., Jayasinghe, M., & Abhaya, V. G. (2011). "Benchmarking Soap Binding. On the Performance of Web services," springer:35-58.
[55]. Tari, Z., Phan, A. K. A., Jayasinghe, M., & Abhaya, V. G. (2011). "The Use of Similarity & Multicast Protocols to Improve performance," On the Performance of Web services, Springer: 59-104.
[56]. Tari, Z., Phan, A. K. A., Jayasinghe, M., & Abhaya, V. G. (2011). "Network Traffic Optimization. On the Performance of Web services," springer: 105-138.

[57]. Yoon, G., Lee, S., & Choi, H. (2016). "QoS Optimizer," 2016 International Conference on Platform Technology and Service (Platcon).
[58]. Solli-Sæther, H., Gottschalk, P. (2010). "The modeling process for stage models," Journal of Organizational Computing & Electronic Commerce, 20:279–293.
[59]. Kohlegger, M., Maier, R., Thalmann, S. (2009). "Understanding maturity models," Results of a structured content analysis. In: Proceedings of the I-KNOW'09 9th international conference on knowledge management and knowledge technologies, Verlag der TU Graz.
[60]. Mettler, T., Rohner, P., Winter, R. (2010). "Towards a classification of maturity models in information systems," In: Management of the interconnected world. Springer, pp 333–340.
[61]. "Software Engineering- Process Assessment. Part 2: Performing an assessment," (2003). ISO/IEC 15504-2.
[62]. "Systems and Software Engineering – Software Life Cycle Processes," (2008). ISO/IEC 12207.
[63]. "Information Technology - Process Assessment. Part 1: Concepts and vocabulary," (2004). ISO/IEC 15504-1.
[64]. Röglinger, M., Pöppelbuß, J., Becker, J. (2012). "Maturity models in business process management," Bus. Process Manage. J, v18, pp. 328–346.
[65]. De-Bruin, Doebeli, T. (2010). "An organizational approach to BPM: the experience of an Australian Transport Provider," in: Handb. Bus. Process Manage. 2. Springer-Verlag, pp 559–577.
[66]. "Object Management Group-OMG, Business Process Maturity Model (BPMM)," (2008). Ver.1, Needham, MA, USA.
[67]. Lee, J., Lee, D., & Kang, S. (2009). "vPMM: A Value Based Process Maturity Model," Comput. Inform. Sci. 2009, SCI 208, pp. 193–202.
[68]. Rohloff, M. (2009). "PMMA Process Management Maturity Assessment," AMCIS 2009 Proc. San Francisco, California.
[69]. Krivograd, N., Fettke, P., & Loos, P. (2014). "Development of an Intelligent Maturity Model-Tool for Business Process Management," HICSS, pp. 3878-3887.
[70]. Stall, A., Forrester, E., (2012). Using CMMI-DEV and CMMI-SVC together where build stuff happens in CMMI-SVC. https:// resources.sei.cmu.edu/asset_files/Presentation/2012_017_001_ 22930.pdf.
[71]. CMMI Product Team. (2006). Cmmi for development, version 1.2. Technical report, CMU/SEI-2006-TR-008, ESC-TR-2006008, Software Engineering Institute.
[72]. SCAMPI Team. (2011). "Standard CMMI Appraisal Method for Process Improvement (SCAMPI)," Version 1.3, SEI, Carnegie Mellon Univ., PA, USA, Tech. Rep, CMU/SEI-2011-HB-001.
[73]. Hirschheim, R., Welke, R., Schwarz, A. (2010). "Service-oriented architecture: Myths, realities, and a maturity model," MIS Q Exec 9:37–48.
[74]. Arsanjani, A., Holley, K. (2006). "The service integration maturity model: achieving flexibility in the transformation to SOA," In: Proceedings of the IEEE international conferences services computing, p 515.
[75]. The Open Group (2011). OSIMM Version 2 Technical Standard. The Open Group.
[76]. Hensle, B., Deb, M. (2008). "SOA maturity model-guiding and accelerating SOA success," Oracle Corp, Redwood City.
[77]. Baghdadi, Y. (2014). "SOA maturity models: guidance to realize SOA," Int J Comput Commun Eng 3:372.
[78]. Pugsley, A. (2006). "Assessing your SOA program," HP White Pap, Hewlett Packard, Palo Alto.

[79]. Rathfelder, C., Groenda, H. (2008). "iSOAMM: an independent SOA maturity model," In: Meier R, Terzis S (eds) Distributed applications and interoperable systems. DAIS 2008. Lecture notes in computer science, vol 5053. Springer, Berlin, Heidelberg.
[80]. Welke, R., Hirschheim, R., Schwarz, A. (2011). "Service oriented architecture maturity," Computer 44:61–67.
[81]. Pulparambil, S., Baghdadi, Y. (2015). "A comparison framework for SOA maturity models," In: 2015 IEEE international conference smart city/SocialCom/SustainCom. pp 1102–1107.
[82]. Pulparambil, S., Baghdadi, Y., and Al-badawi, M. (2017). "Exploring the main building blocks of SOA method: SOA maturity model perspective," Serv. Oriented Comput. Appl., vol. 11, no. 2, pp. 217–232.
[83]. National Institute of Standards and Technology. Nist special publication 800-30, risk management guide for information technology systems, 2002.
[84]. BSI (German Federal Office for Information Security). IT-Grundschutz Manual (english version), 2004.
[85]. International Organization for Standardization. Iso/iec133351:2004, information technology - security techniques - management of information and communications technology security - part 1: Concepts and models for information and communications technology security management, 2004.
[86]. British Standard Institute (BSI). British standard bs25999-1:2006: Business continuity management - part 1: Code of practice, 2006.
[87]. British Standard Institute (BSI). British standard bs25999-2:2007: Business continuity management - part 2: Specification, 2007.
[88]. International Organization for Standardization. Iso/iec 24762:2008 information technology - security techniques - guidelines for information and communications technology disaster recovery services, 2008.
[89]. European Network and Information Security Agency (ENISA). Business and it continuity overview and implementation principles, 2008.
[90]. Braber, F., Hogganvik, I., Lund, M.S., and Vraalsen, F., and Stolen, K. (2017). "Model-based security analysis in seven steps- a guided tour to the coras method," BTTechnologyJournal, 25:101–117.
[91]. Jallow, A.K., Majeed, B., Vergidis, K., Tiwari, A., & Roy, R. (2007). "Operational risk analysis in business processes," BT Technology Journal, 25:168–177.
[92]. Sackmann, S. (2008). "A reference model for process-oriented it risk management," In 16th European Conference on Information Systems.
[93]. Sackmann, S., Lowis, L., and Kittel, K. (2009). "Selecting services in business process execution - a risk-based approach," In Business Services: Konzepte, Technologien, Anwendungen, Tagung Wirtschaftsinformatik (WI09).
[94]. Muehlen, M. Z., and Rosemann, M. (2005). "Integrating risks in business process models," In Australasian Conference on Information Systems (ACIS 2005).
[95]. Sadiq, S., Governatori, G., & Namiri, K. (2007). "Modelling control objectives for business process compliance," In 5th International Conference on Business Process Management (BPM2007), pages 149–164.
[96]. Weber, I., Governatori, G., & Hoffmann, J. (2008). "Approximate compliance checking for annotated process models," In 1st International Workshop on Governance, Risk and Compliance-Applications in Information Systems (GRCIS'08).
[97]. Jakoubi, S., Tjoa, S., Goluch, S., & Kitzler. G. (2010). "Risk-aware business process management–establishing the link between business and security," In Complex Intelligent Systems and Their Applications, volume 41of Springer Optimization and its Applications, pages109–135.Springer.

[98]. Tjoa, S., Jakoubi, S., Goluch, G., Kitzler, G., Goluch, S., & Quirchmayr. G. (2011). "A formal approach enabling risk-aware business process modeling and simulation," IEEE Transactions on Services Computing, 4(2):153–166.
[99]. Suriadi, S., Weiß, B., Winkelmann, A., ter Hofstede, A., Wynn, M., Ouyang, C., Adams, M.J., Conforti, R., Fidge, C., Rosa, M. La., & Pika, A. (2012). "Current research in risk-aware business process management - overview, comparison, and gap analysis," BPM Center Report BPM12-13, BPMcenter.org.
[100]. Milanovic, N., Milic, B., & Malek, M. (2008). "Modeling business process availability," In IEEE International Conference on Services Computing (SCC 2008), pages 315–321.
[101]. Antoniou, G., and Harmelen, F.V. (2004). "A Semantic Web Primer," MIT Press Cambridge, ISBN 0-262-01210-3.
[102]. Fahad, M., and Qadir, M. A. (2008). "A Framework for Ontology Evaluation," ICCS Suppl., vol. 354, pp. 149–158.
[103]. Gomez-Perez, A., Lopez, M.F, and Garcia, O.C. (2001). "Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web," Springer ISBN:1-85253-55j-3.
[104]. Giallonardo, E., and Zimeo, E. (2007). "More semantics in QoS matching," in Service Oriented Computing and Applications. SOCA'07. IEEE International Conference on, pp. 163–171.
[105]. Lin, L., Kai, S., and Sen, S. (2008). "Ontology-based QoS-aware support for semantic Web services," Technical Report at Beijing University of Posts and Telecommunications.
[106]. Tran, V, X., Tsuji, H., Masuda, R. (2009). "A new QoS ontology and its QoS based ranking algorithm for Web services," Journal on Simulation Modelling Practice and Theory, Science Direct, vol (17), Issue No: 8, pp. 1378-1398.
[107]. D'Mello, D. A., and Ananthanarayana, V. S. (2009). "Semantic Web Service Selection Based on Service Provider's Business Offerings," IJSSST, vol. 10, no. 2, pp. 25–37.
[108]. Benaboud, R., Maamri, R., and Sahnoun, Z. (2012). "Semantic Web service Discovery Based on Agents and Ontologies," International Journal of Innovation, Management and Technology, Vol. 3, No. 4, 467-472.
[109]. Moraes, P., Sampaio, L., Monteiro, J., Portnoi, M., (2008). "Mononto: A domain ontology for network monitoring and recommendation for advanced internet applications users," In: Network Operations and Management Symposium Workshops, IEEE NOMS pp 116–123.
[110]. Pakari, S., Kheirkhah, E., and Jalali, M. (2014). "A Novel Approach: A Hybrid Semantic Matchmaker for Service Discovery in Service Oriented Architecture," Int. J. Netw. Secur. Its Appl., vol. 6, no. 1, pp. 37–48.
[111]. Chhun, S., Moalla, N. and Ouzrout, Y. (2014). "QoS ontology for service selection and reuse," Journal of Intelligent Manufacturing, pp.1-13.
[112]. Saaty, T. L. (1980). "The Analytic Hierarchy Process," New York, NY, USA: McGraw, Hill International.
[113]. Ansah, H. R., Sorooshian, S., & Mustafa, S. B. (2015). "Analytic Hierarchy Process Decision Making Algorithm," Glob. J. Pure Appl. Math. 11(4), 2403–2410.
[114]. Cabola, P. (2010). "Using the Analytic Hierarchy Process in evaluating decision alternatives," Operation Research and Decision, 5–23.
[115]. Alam, M. N., Jebran, J. K., & Hossain, M. A. (2012). "Analytical Hierarchy Process (AHP) approach on consumer's preferences for selecting telecom operators in Bangladesh," Information and Knowledge Management, 7–19.

[116]. Wua, H-Y., Tzeng, G-H., Chen, Y-H. (2009). "A fuzzy MCDM approach for evaluating banking performance based on balanced scorecard," Expert Systems with Applications, 36(101), 35–47.
[117]. Buyukozkan, G., Cifci, G., & Guleryuz, S. (2011). "Strategic analysis of healthcare service quality using fuzzy AHP methodology," Expert Systems with Applications, 38(8), 9407–9424.
[118]. Büyüközkan, G., & Çifçi, G. (2012). "A combined fuzzy AHP and fuzzy TOPSIS based strategic analysis of electronic service quality in healthcare industry," Expert Systems with Applications, 39, 2341–2354.
[119]. Paradi, JC., Zhu, H. (2013). "A survey on bank branch efficiency and performance research with data envelopment analysis," Omega, 41(1):61–79.
[120]. Lee, H., Kim, Ch. (2014). "Benchmarking of service quality with data envelopment analysis," Expert System Applications, 41(8):3761–8.
[121]. Ho, W., Xu, X., & Dey, P. K. (2010). "Multi-criteria decision making approaches for supplier evaluation and selection: A literature review," European Journal of Operational Research, 202(1), 16–24.
[122]. http://wso2.com/products/application-server/.
[123]. Protégè. (2007). Stanford Medical Informatics. Available at: http://protege.stanford.edu/.
[124]. Masood, T., Cherifi, C.B., & Moalla, N. (2015). "Ontology Based Service Network Monitoring for Better Quality of Service," 5th International Conference on Information Society and Technology, In the proceedings of ICIST 2015:278-283, Serbia.
[125]. Chhun, S., Cherifi, C., Moalla, N., Ouzrout, Y. (2015). "A multi-criteria service selection algorithm for business process requirements," CoRR abs/1505.03998.
[126]. Masood, T., Cherifi, C.B., Moalla, N., & Fahad, M. (2016). "Performance Oriented Decision Making to Guide Web Service Lifecycle," 8th Int. Conf. on Interoperability for Enterprise Systems and Applications, In the proceedings of I-ESA 2016:113-122, Portugal.
[127]. Masood, T., Cherifi, C.B., & Moalla, N. (2016). "Performance Monitoring Framework for Service Oriented System Lifecycle," 4th Int. Conf. on Model-Driven Engineering and Software development, In the proceedings of MDE4SI 2016:800-806, Italy.
[128]. Masood, T., Cherifi, C.B., Moalla, N. (2017). "Identifying Performance Objectives to Guide Service Oriented Architecture Layers," Communications in Computer and Information Science, Springer.