QOS-AWARE WEB SERVICE DISCOVERY, SELECTION, COMPOSITION AND APPLICATION

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

Institute for Sustainable Industries and Liveable Cities

Victoria University

by

Sarathkumar Rangarajan

August 2020

ABSTRACT

Since the beginning of the 21st century, service-oriented architecture (SOA) has emerged as an advancement of distributed computing. SOA is a framework where software modules are developed using straightforward interfaces, and each module serves a specific array of functions. It delivers enterprise applications individually or integrated into a more significant composite Web services. However, SOA implementation faces several challenges, hindering its broader adaptation. This thesis aims to highlight three significant challenges in the implementation of SOA.

The abundance of functionally similar Web services and the lack of integrity with non-functional features such as Quality of Service (QoS) leads to the difficulties in the prediction of QoS. Thus, the first challenge to be addressed is to find an efficient scheme for the prediction of QoS. The use of software source code metrics is a widely accepted alternative solution. Source code metrics are measured at a micro level and aggregated at the macro level to represent the software adequately. However, the effect of aggregation schemes on QoS prediction using source code metrics remains unexplored. The inequality distribution model, the Theil index, is proposed in this research to aggregate micro level source code metrics for three different datasets and compare the quality of QoS prediction. The experiment results show that the Theil index is a practical solution for effective QoS prediction.

The second challenge is to search and compose suitable Web services without the need for expertise in composition tools. Currently, the existing approaches need system engineers with extensive knowledge of SOA techniques.

A keyword-based search is a common approach for information retrieval which does not require an understanding of a query language or the underlying data structure. The proposed framework uses a schema-based keyword search over the relational database for an efficient Web service search and composition. Experiments are conducted with the WS-Dream data set to evaluate Web service search and composition framework using adequate performance parameters. The results of a quality constraints experiments show that the schema-based keyword search can achieve a better success rate than the existing approaches.

Building an efficient data architecture for SOA applications is the third challenge as real-world SOA applications are required to process a vast quantity of data to produce a valuable service on demand. Contemporary SOA data processing systems such as the Enterprise Data Warehouse (EDW) lack scalability. A data lake, a productive data environment, is proposed to improve data ingestion for SOA systems. The data lake architecture stores both structured and unstructured data using the Hadoop Distributed File System (HDFS). Experiment results compare the data ingestion time of data lake and EDW. In the evaluation, the data lake-based architecture is implemented for personalized medication suggestion system. The data lake shows that it can generate patient clusters more concisely than the current EDW-based approaches.

In summary, this research can effectively address three significant challenges for the broader adaptation of SOAs. The Theil index-based data aggregation model helps QoS prediction without the dependence on the Web service registry. Service engineers with less knowledge of SOA techniques can exploit a schema-based keyword search for a Web service search and composition. The data lake shows its potential to act as a data architecture for SOA applications.

DOCTOR OF PHILOSOPHY DECLARATION

I, Sarathkumar Rangarajan, declare that the PhD thesis entitled *QOS-aware Web service discovery, selection, composition and application* is no more than 100,000 words in length including quotes and exclusive of tables, figures, appendices, bibliography, references and footnotes. This thesis contains no material that has been submitted previously, in whole or in part, for the award of any other academic degree or diploma. Except where otherwise indicated, this thesis is my own work.

Signature  Date 01/08/2020

ACKNOWLEDGEMENTS

I owe my gratitude to all those who have made this thesis possible. First and foremost, I sincerely thank my principal supervisor, Prof. Hua Wang, for allowing me to embark on a PhD under his guidance. Thank you for believing in me and for accepting me as your doctoral student. Thank you for sharing your knowledge, and for giving me your continuous support, kindness, understanding, and patience. Thank you for allowing me to explore my ideas and to choose the specific research topic that I wanted to pursue. Also, thank you for giving me the financial support such as the ARC discovery project stipend and the tuition fee waiver at crucial times. Without your help, I would not have been able to write and submit my research papers, and I could not have attended the conferences and published my research in the Q1 journal. Thank you for sharing your excellent professional and personal advice over these years, and for being there with me in times of sadness and happiness, and in failure and success. A big thank you from the bottom of my heart for making me what I am today.

I am particularly grateful to my associate supervisor Dr Huai Liu for your kind support, valuable advice and encouragement throughout my PhD candidature. Thank you for giving your valuable time to me during our regular weekly meetings and for correcting my work. Even though you left VU, you extended your help to improve the quality of my papers and thesis. Thank you for all the support and for being a very kind mentor. I could not have possibly asked for more. I am also grateful to Prof. Yanchun Zhang, head of our research group, for your support and enlightenment during our group meetings. I would also like to extend my thanks to Professor Yuan Miao, Head of Information Technology, for his encouragement and support. Thank you for the motivation to present my research at the VU Open Day event. Thank you,

Professors, as the guidance from both of you was invaluable to me. Thank you, Dr Rui Zhou and Dr Siuly, for your insightful thoughts that significantly contributed to my research. I would like to thank Dr Khandakar Ahmed for offering me the sessional position at the university, which I was delighted to accept.

A big thank you to my best friend, Dr Sudha Subramani, for your guidance and encouragement regarding the PhD program admission into VU. Without your help, I may not have pursued my dream here. Thanks for supporting me in my early stages of my PhD in terms of research as well as for assistance with settling in Melbourne. Thank you for being my friend for the past ten years since we started our master's together. I am pleased to thank Sudha's parents, Aunty Latha and Uncle Subramani, for considering me as a part of their family and for giving me their unconditional love. I would like to thank my friends Hassan, Shekha, Ujjwal, Rubina, Khalid, Ravinder and Dinesh for supporting me and being there for me whenever I needed help.

I don't think a simple thank you is enough for the things my lovely wife Phavithra Manoharan has done for me. I am very grateful that she accepts me for the way I am and for supporting me through the tough times. For everything I achieved throughout my PhD candidature, I owe this to you, and I am very grateful to you. Thank you for bringing our lovely beautiful angel Niralya into this world.

Last but not the least, I thank my family for their unconditional love and generous support. Thank you, my beloved sister, Janu, for your love and being there for Niralya all these years. I am forever grateful to my parents, who sacrificed their lives for me and believed the struggle pays off in the future. I would like to express my love and gratitude to my dear Appa who believed in me that I will achieve in my life. Thank you, Amma, for all your effort and dedication in

making me dream big. Thank you for taking care of Niralya in India right after her birth until now and providing her with a memorable childhood. Finally, thank you, my lovely beautiful daughter, Niralya Sarathkumar. Though I could not spend much time with you in India and I missed all your childhood fun as we were apart, our video calls were the fuel which kept me running throughout this time.

I dedicate this thesis to my beloved parents for living their whole life just for me.

I also dedicate this thesis to my dear wife and my sweet baby girl Niralya.

PUBLICATIONS

Based on this research work, the following articles, have been published or submitted in Internation Journals and conferences.

1. **Rangarajan, S.,** Liu, H., & Wang, H. (2020). Web service QoS prediction using improved software source code metrics. Plos one, 15(1), e0226867.
2. **Rangarajan S.,** Liu H., Wang H., Wang CL. (2018) Scalable Architecture for Personalized Healthcare Service Recommendation Using Big Data Lake. In: Beheshti A., Hashmi M., Dong H., Zhang W. (eds) Service Research and Innovation. ASSRI 2015, ASSRI 2017. Lecture Notes in Business Information Processing, vol 234. Springer, Cham. https://doi.org/10.1007/978-3-319-76587-7_5.
3. **Rangarajan, S.,** & Chandar, R. K. (2017). Qos-based architecture for discovery and selection of suitable Web services using non-functional properties. EAI Endorsed Transactions on Scalable Information Systems, 4(12).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

SOA Service-Oriented Architecture

QoS Quality of Service

EDW Enterprise Data Warehouse

HDFS Hadoop Distributed File System

SOAP Simple Object Access Protocol

WSDL Web Services Description Language

UDDI Universal Description Discovery and Integration

DAML Digital Asset Modeling Language

SLA Service Level Agreement

IoT Internet of Things

MTJNT Minimal Total Joint NeTwork

SBS Services Based Systems

IR Information Retrieval

KS3 Keyword Search for Servicebased Systems

ETL Extract-Transform-Load

CN Candidate Networks

tmodel Technical Model

CF Collaborative Filtering

SVD Single Value Decomposition

MF factorization matrix

LACF location-aware collaborative filtering

APC average parameter count

RFC response for the class

ATC abstract type count

CBO coupling between the objects

LOCM lack of cohesion among the methods

VTC void type count

CAM cohesion among the methods of class

LCOM3 lack of cohesion in methods Henderson-Sellers version

TPC total parameter count

EPM empty parameters methods

DIT depth of inheritance tree

NOC number of children

WMC weighted method per class

RFC response for class

IUC interface usage cohesion

SQL Structured Query Language

API Application Programming Interface

PSR Pre-Computing Solution in RDBMS

AI Artificial Intelligence

CSTE cost-sensitive temporally expressive

SCP supply chain planning

IP integer programming

ZB Zetta Bytes

EHR electronic health records

RFID radio frequency identification

CARE Collaborative Assessment and Recommendation Engine

CFDRA collaborative filtering-based disease risk assessment

PTaaS physical therapy-as-a-service

PLSF personal lake serialization format

PCA principal component analysis

MAE Mean absolute error

RMSE root mean squared error

RDBMS relational database management system

SEO search engine optimisation

MPFS Maximum Possible Future Score

CHAPTER 1

INTRODUCTION

Web services are scalable, self-descriptive, easily integrated software programmes that do not rely on any programming language platform. One can advertise, search, and consume Web services across the internet with the use of a series of protocols including Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description Discovery and Integration (UDDI) [1]. The Web service providers encapsulate features and information for the application and make them accessible via traditional programmatic interfaces. Service-oriented architecture (SOA) aims to allow business applications to be built using independently designed and distributed Web services. The benefit of SOAs is that they allow the dynamic discovery and incorporation of Web services at runtime, thereby understanding the autonomic characteristics of system versatility and adaptivity. However, the existing mainstream strategies only partly resolve the concept of SOA. These strategies focus on static user interface specifications, and other general non-functional service attributes to publish and identify Web services. This situation creates three problems. Firstly, the consumer does not have any assistance in selecting a target service from the multiple services that appear to perform the same function. Secondly, the development of a composite Web service is highly complicated and is a significant obstacle to the further and broader use of SOA. Third, the data architecture for SOA applications lacks the potential to unearth its capability.

1.1 Service-Oriented Architecture

SOA is a strategy that fills the gap between enterprise and IT and also aims to make processes stable. Web service is the foundation of SOA. Businesses can save time and resources by creating and reusing Online services. Web services also facilitate the outsourcing of certain parts of an application to an external provider. Web services can make it possible to utilise the remote vendors for outsourcing some aspects of an application. Analysis has shown that the introduction of SOA creates economic value within an organisation.

Web services are defined as a programmable components with standardized interface descriptions developed to support business-to-business inter-operable communications via standard communication protocols [2]. Web services are loosely coupled application programs designed to help business-to-business interaction over the web using standard protocols [3]. Web services are developed and made public in web registries by development companies. Users can discover, configure, and orchestrate the services according to their business needs.

The most widely used principles in the IT industry for defining and locating Web services are WSDL and UDDI. Mcllarith et al. suggests Web services markup using the Digital Asset Modeling Language (DAML) family of Semantic web markup languages [4]. This markup enables the creation, execution, design and interoperation of a wide range of agent technologies for automated Web services. Service properties and functions declarative advertising allow automated product discovery. Declarative APIs provide automated service execution for individual services. Besides, automatic composition and interoperation of the service use declarative prerequisites and consequences specifications.

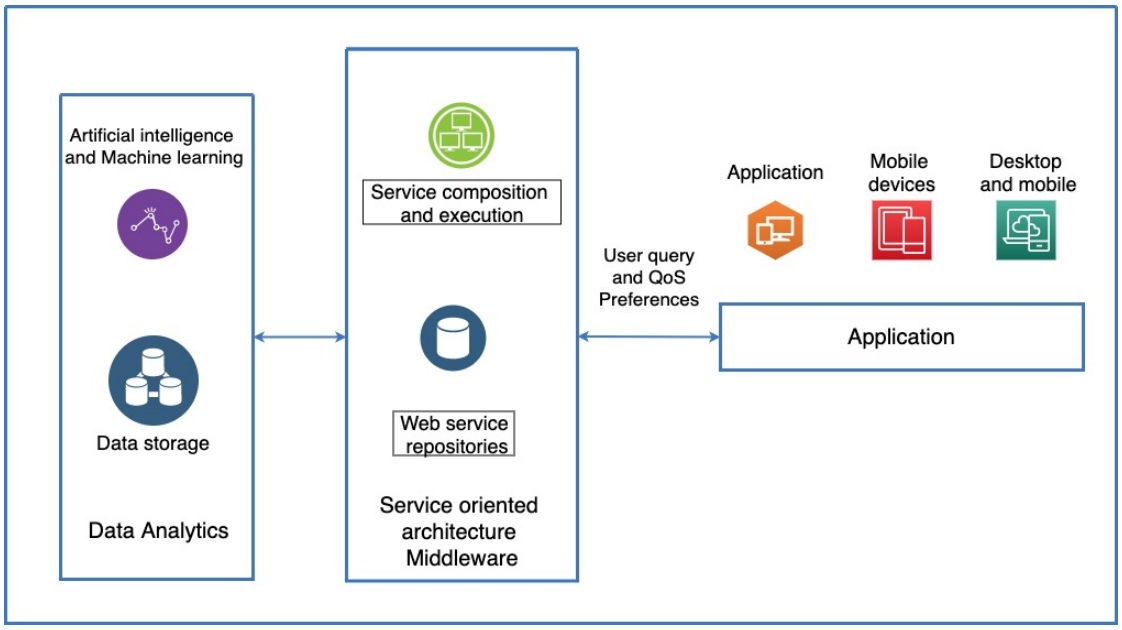


Figure 1.1: Service Oriented Architecture

SOA, as depicted in Figure 1.1, is an architecture paradigm which focuses on building systems through the integration of various Web services to construct the complete system. SOA has emerged as a medium for using Web services to process vast amounts of information and knowledge to provide essential services and data to users [5]. The basic unit for SOA is Web service. The user enters their search keywords and QoS preferences for their target SOA. The SOA middleware shown in the architecture is responsible for Web service composition and execution. The Service Composition & Execution layer provides automatic service composition approaches. This layer is also in charge of the execution control of these composite services. A critical aspect of this layer is its resilience to failure and its adaptation to changes in the system. The final user applications are built from services and executed in the Service Composition and Execution layer. In most cases, the composite SOA application acts as a data source for data analytics. Artificial intelligence and machine learning-based data analytic techniques can be used to extract data from these SOA ap-

plications to produce valuable insights.

The frequently discussed 'planning a vacation' scenario is used as a guide for a more precise understanding of the sections that follow. In this scenario, a proposal from a travel agent was envisaged. This application aims to take the place of a human travel agent, enabling the consumer to make more informed decisions and book a total package. This application has the functions to enable airlines, hotels, charters and taxis for airports to be queried and booked. Weather forecasts, hotel reviews and weather and currency orders may be included. All these will be achieved in a consumer application through the composition of relevant services.

It would probably be free to request the booking for currency services, as the benefit would come from actual reservations or transactions. If this was the case, the results could be aggregated from different providers (selecting as many services as possible to provide the full coverage). However, if these resources are not free, then the impact of search aggregations will be reduced. Note that QoS arrangements for publicly available services (and this is perhaps one reason for first of all rendering such services free) are unlikely to be provided by providers. When an agreement is reached, it will be of paramount interest to keep the search time to a minimum and to agree on acceptable standards of usability.

Reservation speed is another significant criterion of the customer. It is possible that the company will add a further clause to this in the arrangement of the Service Level Agreement (SLA), promising to have response times within those limitations but only if the server load is below a certain amount. In this case, QoS measurements and SLA conformance testing are important for the third

party, since in this situation consumers can not trust the vendor to correctly record their server load. Due to the combination of various resources to operate as an individual SOA, the collection of data from multiple sources is required in different formats. This situation illustrates the need for a framework for data management to manage and store data from multiple sources in one place for accelerated processing.

1.2 Motivations and problems

1.2.1 Web service discovery

Recent advances in the Internet of Things (IoT) have given significant opportunities for Web services with thin clients. As such, many functionally similar Web services are available, and their number is continuously increasing [6]. Since there are several functionally similar Web services, a search using functional criteria is no longer enough [7]. Thus, non-functional properties such as response time and throughput, have become vital criteria to discover, select, recommend and orchestrate Web services [8].

Issue

QoS data available at Web service repositories are not reliable for various reasons. Therefore, the prediction of QoS data is necessary to discover the most suitable Web service. However, most of the existing QoS prediction methods do not calculate the quality parameters from a macro level.

Proposed solution

As a response to these challenges, this research proposes an efficient methodology to calculate source code metrics using micro level software components. The inequality measure, the Theil index, is employed as an aggregation model for Web services at a micro level.

The success of cloud computing depends significantly on the QoS provided at wireless terminals with restricted computing and storage space, such as mobile phones [9, 10, 11]. Web services are of utmost importance amongst various types of cloud services [6]. Moreover, every day the number of Web services is increasing rapidly. According to the "programmableweb.com" open-source Web service repository, 22,367 Web services are available. There are several functionally similar services available due to the abundance of Web services [12, 13]. Therefore, searching for a Web service using a functional feature is no longer valid. Non-functional properties such as QoS have consequently become a pivotal criterion in the discovery, selection, recommendation and orchestration of Web services [8].

Several techniques for selecting QoS-driven Web services were proposed and successfully used in SOA. The developers made QoS values accessible on repositories for Web services. Due to the following real-world scenarios, it is difficult to use the QoS data available at service repositories:

1. Web service repositories turn into an elaborate hierarchy in various situations. Therefore, end-users take quite a lot of time to go through an enormous volume of QoS records [14].
2. Public service repositories such as Universal Description Discovery and

Integration (UDDI) may hold untrustworthy QoS information due to a lack of monitoring. Thus, they might list unavailable services and outdated QoS information in response to a user' query [5].

3. Commercial Web services require users to pay a subscription fee. Thus, if a user wants to test a Web services by themselves, they end up paying a considerable amount of money. It is not possible in practice for a user to monitor and collect QoS data for all the functionally similar Web services.
4. Some public repositories collect feedback from users to acquire QoS data. However, network and geological factors influence QoS information. As the Internet is dynamic and vulnerable, it is not possible to get the same QoS values for different users from diverse locations for a particular service [15, 16].

Even though an SLA contains the QoS parameters of a Web service, users are still unsure as to the quality it achieves. Indeed, a fundamental pre-request is to predict the QoS values instead of using the data available at repositories. Coscia et al. found a statistically significant and robust relationship among several conventional source code-level metrics and the catalogue of WSDL level service metrics [17]. Observing software quality metrics is a standard methodology to evaluate software maintainability. Each Web service comprises many micro level software components such as class, method and package. Therefore, source code metrics are calculated at the micro level and aggregated into the macro level to represent the entire software efficiently. Some of the standard practices in software development industries for aggregating source code metrics and its disadvantages are as follows [18]:

- Simple average: Calculating the mean of metric results for individual el-

ements of a system might not be efficient enough to represent it because it does not express the standard deviation and may mitigate the effects of unwanted values in the generalized result. In other words, the average function simply smooths the results but does not reflect reality.

- Weighted average: Weight factor used to differentiate less important components from critical components. However, defining the weight is very vital and may introduce problems of its own.
- Statistical aggregation methods: Central tendency measures such as mean, median or standard deviation cannot be trusted due to the highly skewed distribution nature of software.

The impact of aggregation schemes for source code metrics on Web service QoS prediction remains unexplored. A hypothesis framed as the metrics aggregation scheme plays a vital role in the high correlation between many parameters at the file-level. Furthermore, the performance of QoS prediction models may be negatively affected by the potential loss of information due to summation and aggregation.

This research investigates the impact of source code metrics aggregation on the correlation between QoS attributes and source code metrics. Our investigation will be based on three different sets of quality metrics, namely object-oriented quality metrics proposed by Chidamber et al. [19], complexity metrics proposed by Sneed [20] and the maintainability suite by Baski and Misra [21].

1.2.2 Selection and composition of Web service

The quality parameters of each Web service aggregate to calculate the quality of a Web service composite. If different execution paths are available from input service to output service, the one with the shortest execution time determines the response time of the framework.

Issue

The construction of a composite Web service is highly complicated and poses a significant obstacle to the SOA's further and broader use. To non-experts, relatively simple tools developed by SOA vendors like Oracle BPEL Process Manager and IBM System Designer are still too difficult to use.

Proposed solution

This research proposes a relational database-based keyword search to improve the selection and ranking of Web services. Service information can be stored in relational databases as tuples, where service composability information is stored as key references between tuples. A foreign key reference needs to be stored when the output of service used as input for another service. For a given set of keywords, given a relational database storing services and their relations, a set of connected tuples (Minimal Total Joint NeTwork (MTJNT) or candidate network) is required so that all keywords are covered, and the highly ranked MTJNT is the result.

In recent years, software development models have evolved from a

component-based to a service-based model, due to the widespread adoption of SOA and its new technologies [22, 23]. Many companies have increasingly adopted the service-oriented model to build their infrastructure. Not only does the SOA approach reduce operational costs, time and resources, it also fosters the reusability, stability and efficiency of the resulting systems [24]. A significant number of software systems were built via the discovery and composition of Web services provided by different vendors [25, 26]. Value-added services (composite or mash-up) and other such services- based systems (SBSs) developed using these technologies [27].The development and popularity of the e-business, e-commerce, and cloud business model helped the growth in the use of Web services. Online Web service databases such as ProgrammableWeb and Web services.seekda.co are showing a rapid increase in the number of published Web services in recent years [28, 29, 30].

Typically, the construction of an SBS takes three steps: 1) system planning to define the functions required for SBS implementation; 2) discovery of the service and identification of candidate services for each functionality, and 3) selecting a service from each pool of candidate services to attain the system quality requirement. This research concentrates on discovering a range of candidate services and a service registry that can perform a particular task and generate an efficient composite.

Building an SBS is very demanding and had been a significant barrier to broader applications of SOA. The simple tools built by SOA vendors, such as Oracle BPEL Process Manager and IBM Process Designer, are also too complicated, and the non-expert requires comprehensive training. The need to find solutions for systems engineers with no extensive knowledge of system design,

service discovery and selection activities has rapidly increased [31, 32].

Keyword search in Service discovery

A keyword-based search has become very popular over the past decade, namely the widely used information retrieval (IR) model for text and web databases. Service registries have recently begun to use the keyword search tool to find services with similar functionalities (e.g. keyword definition of a task) [33]. No existing keyword search techniques can be effectively used to find multiple Web services to build an SBS.

There are two approaches to keyword searches over relation databases [34]. The first method converts the database to a graph for the search. The edges of the graph refer to foreign key relations. The Keyword Search for Service-based Systems (KS3) is a graph-based keyword search for SOA proposed by He et al. [35]. The KS3 method combines and automates system planning, service discovery and service selection for SBSs. KS3 enables system engineers to lookup services by entering only a few keywords describing the required system tasks. This keyword query, i.e. a keyword query representing the system tasks needed, is modelled as a problem to optimise constraints and uses the integer programming technique to find systems solutions. However, when the Web service repository size is increasing, KS3 is not able to improve its performance. KS3 suffers from reduced performance in processing queries on repositories with an abundance of services. It takes up to 100s for queries on a list of 20,000 Web services [35]. He et al., therefore advanced KS3 and proposed KS3 + by incorporating dynamic programming principles to answer the queries [36].

A schema-based approach is the second category of keyword-based searches using a relational database. A schema-based keyword search approach creates a candidate network using a database schema. Since this approach involves using a database schema to create SQL queries to locate *l*-keyword query structures, it is called a schema-based approach. Relational databases store data in the form of columns, tables and the primary key to foreign key relations. There are two main steps in the schema-based approach. The first is how to generate a list of SQL queries that can extract all the structures within RDP tuples, and the second is how to evaluate the structures efficiently. The existing approaches for RDBMS querying are very inefficient because they obtain all the tuple trees with all the keywords.

1.2.3 Application of SOA

SOA defined as an open, agile, extensible, federated, composable architecture. It comprised of autonomous, Quality of Service (QoS)-capable Web services from a diverse vendor. Web service is an interoperable, discoverable, and potentially reusable services. Various enterprises increasingly adopt SOA as an enterprise information technology (IT) architecture. Although there is an increasing trend in the adoption of SOA in other sectors of the economy, its implementation in health care has been relatively slow. According to a survey of 2,165 companies conducted by Forrester Research, SOA adoption in healthcare and the public sector remains low, whereas utilities, financial institutions and insurance companies show a high rate of acceptance and implementation. In the last two decades, affordable smart and mobile devices, and digital services enriched our living and working environments. The interactions with digital services and de-

vices will generate a vast amount of data. On the other hand, it is of paramount importance to collect and analyse the data and then turn them into actionable knowledge to recommend suitable services to the client.

Issue

Contemporary IT infrastructure provides many data handling systems such as the enterprise data warehouse (EDW). Still, there is a lack of scalability because the EDW data management system is for well-known queries and defined policies. Traditional approaches, such as data warehouses, require the data set to be processed at the time of storage using extract-transform-load (ETL) processes. However, ETL processes are expensive. While these traditional ETL approaches provide structured and refined data, which is easier to analyze and query, this benefit is outweighed by high cost and time-to-market considerations.

Proposed solution

A data lake, an effective data environment, has been proposed as an alternative solution. Gartner.com formally defines a data lake as a collection of storage instances of various data assets additional to the originating data sources. These assets are stored in a near-exact, or even exact, copy of the source format. The purpose of a data lake is to present an unrefined view of data to only the most highly skilled analysts. It helps them explore their data refinement and analysis techniques independent of any of the system-of-record compromises that may exist in a traditional analytic data store (such as a data mart or data warehouse). The data lake is a schema-on-read data architecture. Each data entity in the

lake is associated with a unique identifier and a set of extended metadata. The consumers can use purpose-built schemas for query-relevant data, which will result in a smaller collection of data that can be analysed to help answer a consumer's question. A data lake can be connected to various target Web services to ingest data into the data architecture. An SOA for a clinical decision-making system composed using a certain number of health care Web services and data lake as a data architecture.

1.3 Research Questions

As discussed earlier, the current Web service technologies are insufficient in supporting dynamic SOA. Based on the motivations, the following research questions are formulated for this research.

1.3.1 Research Question 1

Can we improve the performance of Web service discovery through QoS-aware metrics?

This research tackle this question by formulating the following more specific questions for a detailed investigation in Chapter 3:

Sub Research Question 1.1: Will the proposed methodology improve the predictability of source code metrics?

A software system is not a single stand-alone system to provide a solution. Usually, it comprises many subordinate pieces of code such as class, method or function. Therefore, the software metrics must be calculated at the micro level and should be aggregated to the macro level to represent the source code metrics of a software system. Since software code metrics are highly skewed values, it is inappropriate to use simple statistical aggregation (mean, median, etc.) methods. The proposed inequality distribution models are very successful in economic data, which is as skewed as software source code data.

Sub Research Question 1.2: Can we predict the quality of service properties of Web services using source code metrics?

Given that Web service quality information in repositories is not stable and trustworthy, there is a need to predict the quality parameters. During the software development life cycle, developers extract source code metrics to evaluate maintainability to reduce future issues with the system. Thus, source code metrics and QoS properties are correlated. This research use the correlation between source code metrics and QoS to predict the QoS of Web service. Proposed framework uses linear regression-based Machine learning to create a prediction model.

1.3.2 Research Question 2

Can we search and composite QoS-aware service execution in the dynamic, unpredictable Web services?

To answer the research question, further sub- research questions are formulated as follows:

Sub Research Question 2.1: How effectively can a schema-based keyword search improve Web service composition quality?

Schema-based systems create several tuple sets for each database relation to include all the answers for each query with AND semantics. A tuple set generated separately with each combination of Q-keywords and each relation. This process generally results in an exponential number of CNs for queries containing more than four keywords.

By comparison, as previously stated, there is only one tuple set of R^Q for each generated R relation. A post processing step audits queries with AND semantics that returns the tuple trees comprising all the given query keywords. This feature of our framework leads to much faster execution, allowing us to handle larger queries and also increases the quality of the composition.

Sub Research Question 2.2: Can the candidate networks be optimzed to improve the search query?

The queried data assumed as stored in a relational database. Therefore, the system uses a Relational database with SQL query handling capability. A user

issues a keyword query to searches for interconnected tuples that include the specified keywords. The system considers a tuple when a text attribute of the tuple contains the keyword. The system searches for subsets of relations that include the keywords from the queries with the keywords in hand. It extracts all such subsets from the database, called tuple sets. Candidate Networks (CN) represent the tuples containing the requested keywords produce join-relations using relational algebra expressions. That is to say, each CN specifies how the entered keyword query will provide possible answers. In addition to the tuple sets created in the last step, CN generation needs information on the referential integrity limitations taken from the schema. The execution engine produces many different CNs as the tuples that contain the keywords joined in many different ways. A tuple set is considered a CN in our approach when it fulfils the following characteristics:

- The non-free tuple sets in a set do not surpass the number of keywords in query.
- The leaf of any tuple sets should not be free
- An associative construct form should not be available in tuple form

Chapter 4 discusses in detail on responses to the research questions.

1.3.3 Research Question 3

Can the proposed data architecture improve the efficiency of SOA applications?

This research question decomposed into further sub research questions as follows and Chapter 5 discusses the responses.

Sub Research Question 3.1: How efficiently does the proposed architecture reduce the time for data ingesting and crawling from various internal and external data stakeholders?

Compared with the traditional data warehouse, the data lake allows the storage of data as it comes without bounding with any schema. Also, it uses the HDFS file system which can connect to any remote application using Apache tools. The data ingestion time represents the time taken for the data architecture to load and store the data. The less the data ingestion time, the better the data architecture for healthcare analytics. Therefore, if the proposed data lake architecture can reduce the data analytics processing time, it will also improve healthcare recommendations.

Sub Research Question 3.2: Can the data lake architecture avoid data silos?

Due to the pre-defined data schema for data warehouse architecture, it is impossible to store different types of data in a centralised storage location which leads to the creation of numerous data silos for a dataset about each patient. The precision of clustering is dependent on the perimeter of the data on the patients. The more the data is from various data stakeholders, the better the results of the data analytics. Hence, if our proposed data lake architecture can handle multiple types of data in a centralised location without the data swamp threat, it can improve the precision of the clustering significantly.

1.4 Thesis Contributions

The following are the major contributions of this thesis:

- This research suggests an aggregation model at the micro level for Web service quality prediction using the Theil index, a measure of inequality. The Web service selection model using QoS is explained in detail.
- This research proposes a framework using a schema-based keyword search to integrate and automate the planning of the service-based system, discovery and the selection of services. It helps the system engineer without the extensive experience of SOA techniques to design the SBS.
- The widely used relational DB system is adopted to store the Web services library. Each database tuple represents the information on Web services and the key references used to demonstrate the composability information between tuples.
- The experiments are designed to evaluate Schema-based keyword search model using WS-dream dataset. The success rate and response time of the proposed model will be used for the performance analysis. The WS-dream dataset contains the URL of WSDL files for 5825 Web services.
- The data lake architecture is adapted as the data architecture in the SOA application for healthcare to crawl and ingest data from vendors without any pre-processing data delay.
- The proposed architecture demonstrates the ability to accumulate data with different formats and store it in the unified data lake to avoid delay in processing healthcare SOA data.

1.5 Structure of the Thesis

The research conducted is presented in the following chapters. *Chapter 2* presents a review of various existing approaches to support Web service discovery, selection and composition and its applications. *Chapter 3* introduces Web service selection using improved software source code metrics. This chapter discusses the significance of source code metrics aggregation and its impact on QoS-based Web service discovery. *Chapter 4* presents an implementation of our keyword-based Web service selection and composition framework. A schema-based keyword search is implemented for Web service selection and composition with and without QoS criteria. *Chapter 5* presents our proposed data architecture for an efficient SoA. Data lake technology is recommended as the data architecture for healthcare SOA. *Chapter 6* concludes this thesis with a summary, a discussion of the contributions, its limitations, and future research directions.

CHAPTER 2

LITERATURE REVIEW

This chapter outlines the background of service-oriented architecture (SOA), in particular, its engineering process and its importance to distributed computing. A Web service data structure and its message structures are discussed, and the various methodologies used for Web service discovery are presented. Furthermore, the need for quality of service (QoS) in Web service discovery and selection is discussed, and different methods of QoS prediction are presented. Later, various studies on keyword-based search approaches in the context of Web service selection and composition are explained. Lastly, the application of SOA and the multiple approaches used to achieve efficient data architecture are presented.

2.1 Service Oriented Architecture

SOA is an architecture paradigm that focuses on building systems using different Web services, integrating them to make up the complete system. Due to the rapid development in pervasive and distributed computing, SOA merged as a platform to utilize Web services for processing a large amount of information and knowledge to provide essential services and data to users [5]. Web services are the basic units for SOA. Recent developments in the Internet of Things (IoT) unearthed more opportunities to utilize Web services in thin clients. For this reason, there are many functionally similar Web services available, and the number continues to increase [6].

2.1.1 Web service request message

A Web service request must contain the functional and non-functional criteria for the target Web service in standardised XML messages. Harshavardhanan et al. proposed a standard to define the client request message using XML [37, 7]. For each quality measure, the client can give the weight value to prioritise the QoS attributes. For example, if a consumer needs a Web service with the best availability property and price is not a consideration, they need to give a high weight value (high-5 to low-1) for availability and a lower weight value of the price. Listing 2.1 shows a sample XML SOAP message request message for the Web service for credit card validation. The client has a QoS requirement as the maximum price can be 0.01 with a weight of 2, and response time can be 0.05 with a weight of 3. Furthermore, after choosing the best matching Web services, a customer can determine the number of Web services to be returned. In this example, the maximum number of Web services requested is 2.

Listing 2.1: Web service request message

```
<?xml version="1.0" encoding="UTF-8" ?>
<envelope xmlns="http://schemas.xmlsoap.org/soap/envelope">
<body>
        <find_service generic="1.0" xmlns="urn:uddi-org:api">
                <functional_Requirement>Credit/debit card validation</functional_Requirement>
                <quality_Requirement>
                        <property>price</property>
                        <value>0.01</value>
                        <weight>2</weight>
                </quality_Requirement>
                <quality_Requirement>
                        <property>Response time</property>
                        <value>0.05</value>
                        <weight>3</weight>
                </quality_Requirement>
                <Max_Service>2<Max_Service>
        </find_service>
</body>
</envelope>
```

2.1.2 Data structure for Web service information

Rajendran et al. proposed a data structure called the Technical Model (tmodel) to represent the functional and QoS properties of a Web service [38]. The role of a tModel is to register categorizations, which provides an extensible mechanism for adding property information to a Web service registry. The tModel provides the QoS details on binding templates. The name and value of the QoS domain described using the general name-value pair structure of each QoS parameter. An example of the QoS specifics of a stock quote service is shown in Listing 2.2. The bindingTemplate references the tModel with tModelKey "uddi: mycompany.com: StockQuoteService: PrimaryBinding: QoSInformation" which contains the QoS attribute categories. The location of a WSDL description stored in the keyword tModelKey "uddi: mycompany.com: StockQuoteService for fur-

ther information management. The purpose of a Web service is expressed in < function > and < Web serviceid > is the registry's unique Web service identifier.

Listing 2.2: tmodel for storing QoS Information

```
<tModel tModelKey="mycompany.com:StockQuoteService:PrimaryBinding:QoSInformation"">
<function>QoS Information for Stock Quote Service</function>
<ws_id>abdc12345<ws_id>
<overviewDoc>
<overviewURL>
http://<URL describing schema of QoS attributes>
</overviewURL>
</overviewDoc>
<categoryBag>
<keyedReference tModelKey="uddi:uddi.org:QoS:Availability"
keyName="Availability"
keyValue="99.9%" />
<keyedReference tModelKey="uddi:uddi.org:QoS:Throughput"
keyName="Average Throughput"
keyValue=">10Mbps" />
<keyedReference tModelKey="uddi:uddi.org:QoS:Reliability"
keyName="Average Reliability"
keyValue="99.9%" />
</categoryBag>
</tModel>
```

2.1.3 Web service discovery using tmodel

The client's service request uses functional criteria information to list the registry's functionally similar services. The registry publishes all WSDL service interfaces as tModels. The tModel is a data structure defined in a Universal Description, Discovery and Integration (UDDI) registry for an XML Web services type (generic representation of a registered service). They deliver a framework that facilitates software reuse and standardisation. The tModels can represent any unique construct or information to embody within the UDDI information

model. Therefore, the model reuses metadata information throughout.

Each tModel has the URL to identify as a description of the WSDL service. Rajendran et al. proposed a find_tModel method which lists all the matching details of the tModel_id interface [39]. Using the OverviewURL, the content of the WSDL service interface document can be identified once a tModel_id has been obtained. Additional keyedReference added to the categoryBag to restrict the set of tModels returned in response to this query. An example is displayed in Listing 2.3 for the find_tModel message to find all the stock quote services in the registry.

Listing 2.3: Sample Web service discovery XML

```
<?xml version="1.0"?>
<find_tModel generic="1.0" xmlns="urn:uddi-org:api">
<categoryBag>
<keyedReference tM_find_Key="UUID:DB77450D-9FA8-45D4-A7BC-D14E384"
keyName="Stock market trading services"
keylimit="50"/>
</categoryBag>
</find_tModel>
```

2.1.4 Web service Selection

For Web service selection, the min-max normalisation technique and weighted AND-OR tree is proposed by D'Mello et al. to rank the service based on the QoS and its weight [40]. A Weighted AND-OR tree is an AND-OR tree where every edge between the parent and child node is labelled with a non-negative real number in an interval (0, 1). Therefore, for any parent node, the sum of edge labels (weights) of all child nodes is equal to one, i.e. for any parent node

P with C (2≤C≤N) child nodes, the sum of edge weights WPCi (1≤i≤C) is equal to 1.

In "<"conditioned leaf node:

$$wss_n = \begin{cases} \frac{2(WS_{last} - ws_x)}{10} x = 2, 3, ...last - 1 \\ wss_{first} = 1 \\ wss_{last} = 0 \end{cases} \tag{2.1}$$

$$wss_n = \begin{cases} \frac{2(ws_{first} - ws_x)}{10} x = 2, 3, ...last - 1 \\ wss_{first} = 0 \\ wss_{last} = 1 \end{cases} \tag{2.2}$$

wss_n = QoS score of n^{th} Web service

wss_{first} = First Web service in descending order in the leaf node

wss_{last} = Last Web service in descending order in the leaf node

Equations 2.1& 2.2 are used to calculate the score of the target Web service. After constructing the tree, the root node will be in a descending ordered list of the Web services based on their QoS Score. The best Web service based on the client's quality requirement is available at the top of the list.

With the growing number of Web services with the same functionalities, the discovery and selection of the most suitable Web service have become a significant challenge on service-oriented systems [41]. With the availability of functionally similar Web services, a functionality-based search is no longer valid to discover suitable services. Therefore, non-functional properties or QoS such as response time and throughput have become pivotal criteria in Web service dis-

covery, selection, recommendation and orchestration [8].

2.1.5 Web service discovery using QoS

There are many QoS-driven Web service-based techniques proposed and successfully utilised in SOA [42, 43, 44]. The functional semantic method of describing Web Services to provide dynamic Web service discovery is described by Ye and Zhang [45]. Here, defined functional semantic format used to represent both the Web service and the functional request. Creating a domain-oriented functional ontology, and a matching algorithm used to match the annotated services and requests provides Heterogeneity. Ye et al. extend their work by describing FWSDL, a Web service description language to represent functional semantics and the structure of discovery system FunWDS [46].

Mello et al. propose a well-formed functional semantics to describe the operations of Web services by providing extendible functional knowledge to map the requested or published operation descriptions into an abstract operation [47]. The extended functional knowledge is implemented by maintaining a list of all the related operations and describing Web services based on both the functional semantics and extended knowledge to provide dynamic Web service discovery.

However, the focus lies on specifying the structure of the service interfaces and the exchanged messages. Thus, the previous works address the discovery problem relying on structural, keyword-based matching, which limits their search capabilities [48]. So, service consumers currently pay more attention to QoS instead of functionality than ever before. QoS mainly consists of non-functional attributes such as response time, throughput, availability, etc. [49].

Xu et al. proposed a QoS-based Web service discovery model by extending the data structure types to enhance the UDDI model with QoS properties [50]. However, this approach demands the human consumer to undertake service discovery and selection. This approach is not scalable if an enormous number of Web services are available as an option.

Another approach in [51] suggests a QoS-based Web service selection procedure which receives QoS requests with exact values and fuzzy values and returns matching offers in both categories: super-exact and partial matches. In [52], users' preferences are defined by a lexical ordering in accordance with their perceived importance. The authors proposed an algorithm which considers QoS to rank them. Based on that algorithm, if the first QoS attribute distinguishes Service A from Service B, then Service A will be given a higher rank. This algorithm is simple, but if the user indicates that two preferences are equally important, the algorithm will consider only the first preference in the lexical ordering and ignores the second preference.

2.2 Quality of Service prediction

In practice, it is challenging for an end-user to obtain QoS information. The user needs to spend a large amount of resource, time and cost to invoke and measure QoS for all available Web services. Different users will have different QoS experiences while using the same Web service due to the dynamic nature of the network environment and geographically distributed locations [53]. Therefore, predicting the QoS properties of a Web service has become an important step in service-oriented systems. Using available QoS values in invocation records

to calculate the unavailable or missing QoS parameters is called QoS prediction [54]. The collaborative filtering (CF) technique is widely adopted in Web service community due to the success of commercial recommender systems. CF predicts unknown QoS values based on historical user data [55]. Predicted QoS values can be used as additional criteria to rank the matching results during the service discovery and selection process. The top-ranked services hold importance among the other services [56]. In service orchestration, considering the QoS of services is as important as combining the functionalities of different services.

2.2.1 Collaborative Filtering based prediction

Using available QoS values in invocation records to calculate the unavailable or missing QoS parameters is called QoS prediction [54]. CF predicts unknown QoS values based on historical user data [55]. To satisfy the basic requirements to develop an SOA, a necessary pre-requisite is to predict the missing QoS values. At present, CF is a common technique to predict the QoS values among Web service consumers [57]. Briefly, the CF technique comprises of two steps:

1. Identifying the users or nodes with similarities and mining the likeliness.
2. Calculating the missing QoS values based on the available QoS data from a similar user cluster.

Predicted QoS values can be used as additional criteria to rank the matching results during the service discovery and selection process. The top-ranked service holds importance among the service results [56]. In service orchestration,

composing the QoS of services is as essential as combining the functionalities of different services. The reliability of the predicted QoS helps to improve end-to-end QoS output [58]. In 2007, Shao et al. introduced the QoS prediction of Web service using the CF method [59].

Web services can be defined as programmable components with standardized interface descriptions developed to support business-business interoperable communications via standard communication protocols [60]. SOA is an architecture paradigm that focuses on building systems using different Web services, integrating them to make up the complete system.

With the growing number of Web services with the same functionality, the discovery and selection of the most suitable Web service have become a significant challenge on service-oriented systems [41]. A QoS-based evaluation of services has become an ideal method for users to decide on selecting appropriate services.

In many situations, a stand-alone Web service cannot provide the functionality of a user's demand and often services will be composed together to achieve a specific functional system [61]. The quality of a composite service depends on the QoS of the services it comprises. So, the service composer must consider the QoS of all the candidate services to be able to yield the desired overall QoS of the service composition [42].

QoS refers to the non-functional properties of Web services, namely availability, throughput, response time, security and so on. A brief definition for some of the widely used QoS properties are:

- **Availability**: Availability is whether the Web service is accessible or ready

for immediate use.

- **Accessibility**: Accessibility means the degree to which a web service request can be met.
- **Integrity**: The integrity of the web service is the consistency element of how the interaction with the source remains correct.
- **Performance**: Web service efficiency is calculated in terms of throughput and latency. A reliable Web service performance should have higher throughput and lower latency.
- **Regulatory**: This is compliance with the rules, law and guidelines, and the agreed SLA.
- **Security**: Security is the quality component of the Web service that offers privacy and confidentiality, authentication, encryption of messages and access control for the parties concerned.

However, from a user's perspective, obtaining QoS information is very difficult in practice. It demands a significant amount of time, resources and cost for pay-before-use Web services. The dynamic nature of the network and the distribution of locations result in different users having different QoS experiences when invoking the same Web service [53]. Therefore, predicting the QoS properties of a Web service is an essential step in service-oriented systems.

Using the available QoS values in invocation records to calculate the unavailable or missing QoS parameters is called QoS prediction [54]. A user can provide QoS values for the services which he used and obtain the predicted QoS values of the services he has not used before. The following mathematical formulation simplifies the CF-based prediction model [55].

1. $S = \{s_1, s_2, ...s_m\}$ is a set of services with similar functionality where $s_i = (1 \leq i \leq m)$ denotes one service.
2. $U = \{u_1, u_2, ...u_l\}$ is the set of service consumers and $u_i = (1 \leq i \leq l)$ denotes a service consumer.
3. $Q_{i,j} =< p^1_{i,j}, p^2_{i,j}, ..., p^n_{i,j} >, p^k_{i,j} \in R \cup \{\emptyset, ?\}$ $Q_{i,j}$ is a vector representing the quality of s_j measured by consumer u_i. n denotes the number of regarded QoS properties. $p^k_{i,j}$ denotes the value of k-th QoS property of service s_j measured by consumer u_i. The value of $p^k_{i,j}$ could be a real number, $\emptyset$ or ?. When u_i has no QoS data on service s_j for the k-th property, $p^k_{i,j}$ denotes as $\emptyset$. When denoted as ?, $p^k_{i,j}$ denotes the QoS property to making prediction on.
4. $D_i =< Q_{i,1}, Q_{i,2}, ..., Q_{i,k} >$ is a vector, denoting the QoS data collected from consumer u_i.
5. $T =< D_i, D_2...., D_l >$ is a vector of all QoS data.

Let's say $Q_{1,3}$ is the unknown QoS value. Predicting the $Q_{1,3}$ from the available QoS data is the overall idea of CF. It comprises four general steps namey data preparation, normalization, similarity mining, and prediction making.

- **Data Preparation** ensures our collected QoS data consistent with the format of collaborative filtering.
- **Normalization** makes the values of the QoS property into a uniform scope based on the Gaussian approach.
- **Similarity Mining** calculates the similarity between two consumers based on their historical experiences using the Pearson correlation co-efficient

- **Prediction Making** makes linear prediction for each QoS property based on the similarity and combine the prediction values into a quality vector.

Zheng et al. suggested a hybrid model of CF that would combine CF algorithms dependent on users and items. The confidence weights have been used to match the values of both versions, respectively [62]. While CF-based methods are simple to implement and relatively efficient, they suffer from decreasing accuracy. The QoS values are scarce and are challenging to combine with any other variables in the model. The role of geographical data to enhancing the accuracy of the QoS prediction has been studied recently. A hierarchical clustering algorithm was developed for the detection, which was expected to take place in the same area, of our neighbours with a similar historic web service involving experience [63].

This approach is irrational, as users in Seoul and Tokyo can have identical QoS values in a certain period, infrastructure changes in Seoul cannot impact Tokyo's web-service calls. Lo et al. take into account in a true geographical context, the influence of the surrounding consumers. At the end of the objective function of the single value decomposition (SVD) factorization matrix was appended. (MF) [64, 65]. Although the critical aim of this method of use is to avoid the overfitting of the learning process, it is difficult to understand the QoS principles from a neighbour's point of view.

Collaborative filtering focused on neighbourhoods includes user-centred approaches, item-driven approaches and their mergers. Usage-based models estimate one user's missing values based on identical user values. Item-based models calculate missing unit values based on relative item values. Zheng et al. suggested a hybrid user-oriented and item-dependent CF algorithm for pre-

dicting QoS values, which performed a series of large-scale tests focused on a specific dataset of web services [62]. Neighbourhood-based approaches also use the PCC algorithm and VSS algorithm as the methods of computational similarities [66]. PCC-based collaborative filtering methods will typically achieve greater predictive precision than VSS-based algorithms because PCC acknowledges the user rating differences. Training data sets use hypothesis-based methods to learn a predefined hypothesis (e.g., clustering model, aspect model, and matrix factorisation model). Matrix factorisation approaches rely on fitting the user-item matrix with low-rank approximations and is engaged in making additional assumptions if only a limited number of variables impact the values in the user-item matrices.

The neighbourhood-based approaches use the values of related users or objects to predict value, while model-based approaches, including matrix factorization models, use all matrix knowledge to predict value. The model-based methods typically presume that Gaussian is the distribution of QoS values. Zhang et al. [67] proposed that it would be easier to incorporate consumer QoS interactions, climate variables, and user feedback variables to estimate QoS values for the Web site. Some techniques incorporate location-conscious prediction. For example, Tang et al. [68] propose LACF (location-aware collaborative filtering), and Xu et al. [69] propose WL-PMF (weighted location-aware PMF). Some methods introduce social relations. For example, Zheng et al. propose NIMF (neighbourhood integrated matrix factorization) [70].

Predicted QoS values can be used as additional criteria to rank the matching results during the service discovery and selection process. The top-ranked service holds importance among the service results [56]. In service orchestration,

composing the QoS of services is as essential as combining the functionalities of different services.

2.2.2 Software source code metrics

Mateos et al. discussed the methods available in code-first Web services to remove unnecessary anti-patterns [71]. The authors worked on the hypothesis that the occurrence of anti-patterns can be avoided using object-oriented source code metrics. To find the existence of an anti-pattern at the WSDL level, they considered eleven source code metrics, namely: average parameter count (APC), response for the class (RFC), abstract type count (ATC), coupling between the objects (CBO), lack of cohesion among the methods (LOCM), void type count (VTC), cohesion among the methods of class (CAM), lack of cohesion in methods Henderson-Sellers version (LCOM3), total parameter count (TPC), weighted methods per class (WMC), and empty parameters methods (EPM) [72, 73]. Mateos et al. used a real-time Web service dataset to identify the correlation between object-oriented metrics and the occurrence of anti-patterns using well- known statistical methods. They also measured the impact of simple metric-driven code refactoring on the existence of anti-patterns to some of the generated WSDLs from the dataset. As a summary, Mateos et al. observed that the complexity and maintainability of Web services can be predicted using object-oriented metrics and refactoring.

2.2.3 Correlation between source code metrics and QoS

Charrad et al. introduced a high correlation between traditional object-oriented source code-level metrics and WSDL-level service metrics [56]. They used the most comprehensive and thoroughly evaluated set of metrics to calculate the maintainability of Web services using WSDL interfaces. The findings of this research suggest that software developers can avoid developing non-maintainable services by applying simple early code refactoring. As Java is widely used as a programming language to build back-end services, the authors focused on Java-based Web services, but their findings did not depend on the programming language.

Romano et al. tried to identify the list of source code metrics to predict Java interfaces that are vulnerable to change [74]. The source code metrics such as lack of cohesion among methods (LCOM), coupling between objects (CBO), depth of inheritance tree (DIT), number of children (NOC), weighted method per class (WMC), response for class (RFC), and interface usage cohesion (IUC) were used along with fine-grained source code changes in interfaces of ten open-source Java-based systems [72, 73]. The correlation between the metrics of the source code and the fine-grained changes in the source code was tested empirically. Romano et al. concluded that the external interface cohesion source metrics have the most significant association with the number of changes in the source code.

2.2.4 Software metrics aggregation

Software metrics calculated at micro level artefacts and aggregated to macro level artefacts for the analysis. The popular aggregation technique used for source code metrics is mean [75], [76] even though there are increasing research works to demonstrate the inappropriateness of this technique [77], [78] due to the skewness of source code metrics distribution [79]. The sum is another popular aggregation technique. Chidamber et al. used the sum to aggregate the complexity of individual methods to the class level in their metrics suite [19]. Alexander et al. [80] used the Theil index, a widely used inequality measure in econometrics to identify the wealth distribution to aggregate the software metrics values as they both share the same kind of data distribution. The Theil index is not specific to a particular metric and can be used to aggregate a wide range of metrics.

2.3 Web service composition

The invoking process for services selected on runtime in service-oriented environments represented as complex application. In this scenario, an application described as a flexible process consisting of abstract Web services. Web services are chosen from a set of services which have similar functionality and different non-functional properties, i.e. QoS.

Recently, there has been considerable interest from the research community in the dynamic composition of Web services. The existing literature can be divided into two categories: composition by planning and optimization of busi-

ness processes [81]. The first method, suggested by the Semantic Web and AI groups, explores the question of synthesizing a complicated action from a specific goal and from a variety of applicant resources, which leads to a partial solution of the complex issue. The second category defines complex systems as BPEL procedures, and the best set of services are selected dynamically during operation by addressing an optimization problem [82, 83].

Varied approaches to semantics are Web and AI. The composite service process is built with a high-level functionality specification by automatically or semi-automatically. A framework is proposed in [84, 85], which incorporates the planning and execution of complex applications with XSRL and XSAL languages to specify its functional goal and QoS. Model checking is used to perform planning as in [86]. Similarly, in [87], any complex application is constructed from a high-level process design that is synthesized through the implementation of contingency plans. Planning is indeed very adaptable; however, it is intensively computational, and only sub-optimal solutions can be found from the QoS point of view [88]. The work in [89] proposes a trade-off between planning and optimization approaches. The goal is translated into a workflow-based specification that introduces abstract work in a first semi-automatic logical composition stage. The second physical composition stage brings abstract activities to concrete services, and the service designer supervises them. The optimization of business processes enables the specification of complex systems as abstract software BPEL processes that serve as the placeholders of Web service components called upon during runtime. In such instances, a dynamic/late binding method is applied to the best service collection, chosen to solve an optimization problem.

2.4 Keyword-based search

The most accepted method for making queries in a text document and over the internet is Keyword search. A community for the database research has accepted the benefits of keyword search and have already adopted the same in relational databases. A relational database, a term coined by E. F. Codd at IBM (International Business Machines) in 1970, refers to an organized set of data in tabular form, which can be accessed or restructured in various ways without interfering with the original tabular data. This model uses SQL (Structured Query Language) as API (Application Programming Interface) [90]. Relational operators are frequently used to manipulate the data in tabular form. Relational model uses a unique key for identification of each row, also termed as tuples or records. Columns, however, are referred to as attributes. Each table/relation has only one type of entity. Rows contain instances while columns contain values of that instance [91]. Keyword search is an alternate method of querying the relational databases just like the queries made on web search engine. Keyword search enables users to make queries without the knowledge of programming language or database management. Keyword search in relational databases find the tuples assigned with unique keys containing keywords [92]. These methods can be classified into three distinct types:

- Methods using Candidate Network (CN)
- Algorithms using Steiner tree
- Approaches based on Tuple-unit

The first two methods on-the-fly finds the connected tuples using unique keys (Primary/foreign). These methods are time consuming or slow when the

connections are many. These methods lack an indexing service, which can make the work search faster [93].

The primary purpose of using a search method with keywords is to find a set of interrelated tuples that fit the keywords together. Data modelling as a graph is one type of approach which results as sub-trees. A similar kind of approach using relational databases, in which structured data is placed. Researchers performed several early keyword search systems for relational databases. Using a *l*-keyword query, the creation of structural data among tuples in an RDB is surveyed Yu et al. The keyword search systems were addressed by contrasting a schema-based keyword search with graph-based RDB keyword search. The first method assessed the sets of results by defining all the minimal total joining networks of tuples between CNs. The second method demonstrated how weighted direct graph visualisation could obtain the keyword query results[94].

DBXplorer considers each tuple tree and generates an undirected graph [95]. The DBXplorer system accesses the symbol table and then computes the tuple tree as per the schema graph to obtain information about tuples. DISCOVER introduced an algorithm for the CN generator based on an extensive search space [92]. This algorithm expanded the CNs to include larger partial CNs until all CNs were produced. The exponentially large number of partial CNs implies that the algorithm can extend arbitrarily to an incredibly high cost of generating the CN collection and is retained for further expansion. S-KWS introduced an algorithm that decreases the number of partial results obtained by expanding components of the nodes in a partial tree and avoids isomorphic testing by designating a suitable expansion order. While the partial outcomes produced were reduced, there was an overhead for generating minimal CNs to the query

system accessed by a symbol tab to obtain tuple information.[96].

Zhou et al. introduced the exploration of Web services based on keyword clustering and expanded the idea. In Pareto 's principal-domain ontology, they calculate a word similarity matrix and use it to find matching services for semantic reasoning. Bipartite graphs can be used to identify matching degrees between the requirements for the service and the services available. They describe this using the Kuhn-Munkres algorithm to determine an optimal match of a bipartite graph [97].

Unlike plain text, the underlying data in relational databases has an essential structure to it, which indirectly defines the relationship between the data nodes that contain those keywords. The underlying structure needs to be taken into consideration while determining the answers to the keyword searches. Tuples are viewed as vertices in the data-graph. Connections between the tuples are primary-foreign key constraints. The results of the keyword searches are subgraphsf this data-graph. In RDBMS, it is a multi-query optimization problem to evaluate all the CNs to obtain all MTJNT. It faces the following two challenges:

1. How to share common generated subexpressions among CNs to reduce computational costs while evaluating.
2. Where to find a suitable join for the easy evaluation of all CNs. The number of CNs generated for a keyword query can be very large.

With several joints, an effective query processing strategy is exceedingly difficult to obtain, since one best strategy for a CN will slow the others down if other CNs share its sub-trees. The solution to the optimal implementation plan is an NP-complete problem, as stated by Vagelis et al. in DISCOVER. It suggests

an algorithm to evauate all CNs together based on the greedy algorithm using the following elements:

1. CNs sharing the subexpression should be evaluated at first
2. Subexpressions which may produce a low number of results must be evaluated first

Markowetz et al. developed an operator mesh in S-KWS to share the computation cost of all CNs[96]. There is n. $2^{(l-1)}$ clusters in a network, where the number of schema relations is denoted by n in graph G_sand, and the number of keywords is l. The cluster is made up of a group of operator trees, which have common expressions, (left-deep trees). A projected relationship with the smallest number of tuples is selected to continue and join when assessing all CNs in a mesh.

Zhai et al. motivated to provide a user friendly and accurate query mechanism so that by providing keywords the response can be obtained without any knowledge of programming. The major contribution of the authors was that they proposed a system with an architecture to produce more meaningful query response by semantic expansion. This was achieved by including the metadata and keyword matching elements using STAR algorithm for CN generation. Their architecture was proved by highly efficient. The future work of the scientists includes the CN reduction strategy for the decreasing the plan execution pressure and a more efficient ranking system for returned query response [98].

The authors of [99] proposed a system with an architecture to produce more meaningful query by the saved knowledge of using ranking strategies and tu-

ple tree methods with good results returned. Appropriate results are achieved by implementing ranking strategies and an index table where tuples of related knowledge are used for Information retrieval.

The authors of [100]proposed a tuple-unit-based method for effective keyword search over relational databases, which integrates multiple relevant database tuple units to effectively answer keyword queries. They also proposed two structure-aware indexes and stored the structural relationships between different tuple units into the indexes. They used the indexes to efficiently find the relevant answers. They developed effective ranking techniques to rank the tuple units by considering both the structural compactness of answers from the database point of view and the textual relevancy from the information retrieval viewpoint. Existing methods only identify a single tuple unit to answer keyword queries. However, the previous researches neglect the fact that in many cases, they need to integrate multiple related tuple units to answer a keyword query. To address this problem, in this paper, the authors proposed a structure aware-index-based method to integrate multiple related tuple units to effectively answer keyword queries.

2.4.1 Keyword-based search in Web services

Qiang et.al., proposed a keyword search for building service-based systems (KS3) [35]. They used a graph-based keyword search to extract combinations of Web services for a given keyword queries. KS3 does not consider the database schema but considers tuples and their primary-foreign key dependencies as the connections. Moreover, they did not use structured queries like SQL. Steiner

tree-based semantics are used to decide the structure of the tuple sub graphs to be returned. Finding optimal Steiner trees is an NP-complete problem. The KS3 system is efficient when the size of distinct keywords in the query and the size of the tuple sub graphs (constrained by the top-k scoring or weight function) is small. According to Qiang et al., it takes up to 100 seconds to answer queries on a repository with 20,000 Web services. A further development may be required to increase the efficiency to handle the ever-increasing databases and combining multiple Web services for a more complex query. Scientist needs to develop a system to keep the scalability high and better user handling.

Zeng et al. has proposed an unconsolidated storage method to make cloud computing flexible. This method allows to adapt to any new standard and also provide highly efficient algorithm for composition. An SMA, is also introduced, which recognizes the semantic similarities among different input and output parameters of WordNet. They also draw a comparison between service composition and precision ratio. The author proposes two algorithms SMA and Fast-EP for multiple input/output parameters. feasible in theory that search results are ranked based on semantic similarity degree. But the response time do not satisfy the search requirement in real time. Future work of the authors includes invention of high efficiency ranking technique of query results on semantic basis. In the next studies the author wished to provide the composition for more complex services[101].

Kwon et al. proposed a new system called PSR (Pre-Computing Solution in RDBMS). It provides a search based on service composition in any RDBMS. They were trying to tackle the problem of web service composition. Their work was also contributed towards storing the composition graph into a composition

table. The author used synthetic web services dataset with a simple structure, which consisted of 10,000 web services. They had run random queries over the synthetic dataset to find the web services composition search. The maximum number of queries is the value of 1,000. Earlier there was no way to use the output of one service as the input for new service. Their work was a first of its kind in web service composition in a RDBMS. Lower execution time and good scalability when handling large number of user queries is observed. A further development may be required to increase the efficiency to handle the ever-increasing databases and combining multiple web services for a more complex query. Scientist needs to develop a system to keep the scalability high and better user handling.[102]

2.4.2 Selection and composition of Web services

The SBS development cycle consists of three steps: system preparation, service discovery and the selection of services. In each of these steps, there has been a significant amount of work to solve the problems. Throughout the system planning process, the system engineer determines the configuration of the SBS to be performed and the order of execution of the tasks. During this step, most techniques used to identify tasks necessary to implement SBS are based on the techniques of Artificial Intelligence (AI) [103]. To illustrate with an example, the authors model a cost-sensitive temporally expressive (CSTE) as service composition and a supply chain planning (SCP) solver can be used.

The tasks of an SBS are decided upon completion of the system planning process. During the service discovery process, the system engineer selects a

collection of candidate services for each of the tasks based on the functional and semantic knowledge of the candidate services through service registries or service portals. Many semantic Web service languages have been proposed based on ontology techniques, e.g., DSD, to boost service matching accuracy [104]and OWLS-MX [105]. Such languages will enrich the service definition and SBS specification from a semantic perspective. The implementation of an ontology automates the service matching process defining the services that can perform the SBS tasks as defined in the planning phase of the programme. There are typically several operationally equivalent services for one task that can execute the function [83]. Such services differ in the quality properties of a multi-dimensional nature. The service discovery process has to be streamlined to identify a large number of these services. There are several proposed approaches to address this problem [106].

Such methods follow ontology techniques such as logical reasoning and temporal planning to simplify the process of service discovery, based on automated service matching. A group of functionally equivalent candidate services for each of the tasks needed for the SBS are selected at the end of this process. During the service selection process, the system engineer selects one service for the composition of the target SBS from each set of functionally equivalent candidate services. The chosen services must jointly satisfy the multidimensional quality constraints for the SBS in this process, e.g. reliability, performance, cost, etc., which would be an NP-complete problem. The critical technique adopted during this process is integer programming (IP) [107].

This was a significant impediment to further and broader SOA applications. It required a creative approach that can help system engineers easily find re-

sources to develop SBSs without needing to go through the challenging processes. A few solutions were proposed. Following the concept of tag-based searches offered in [108], the authors suggest a technique for preparing SBS approaches by searching for services that fit the SBS tags [31]. The engineer needs to type in a source tag and a destination tag for every query.

The method of helping system engineers navigate through numerous queries from the entry service to the exit service is proposed in [109]. This planning method has two main limitations. First, only two tags, one source tag and one destination tag are allowed for each query. Multiple tags can only be entered one by one in different queries and will be evaluated individually until a final answer is identified. Second, it cannot determine quality constraints. Therefore, the design methodology is not appropriate for building quality-constrained software systems. We are, without doubt, living in a world of smart devices that generate data for virtually everything. Reuters estimated that, in 2020, global data growth will be approximately 35 Zettabytes (one Zettabyte = one million petabytes) for all computerised organisations. 35 ZB is three times the amount of data we generated in 2015.

Use of search queries in a keyword-based RDBMS is the main component. A business environment usually facilitates the RDBMS to occupy the architectural layers. Indexing technology can replace the existing architectural layer. Queries made into SQL are become obsolete when keyword-based search engines are used. They can be used for any kind of structured or unstructured data. In this research the data is maintained in a multi-tier architecture and the information is retrieved by the keyword-based query search in SQL relational database.

2.5 Application of SOA

In 2012, 500 petabytes of data were generated by healthcare industries worldwide, and this is expected to grow 50-fold to 25,000 petabytes in 2020. So, it is clear that research on EHR and patient-related auxiliary data will illuminate the improvised healthcare at the point of care. The availability of rich data sources could make personalized / precision medication for each individual possible. Traditional IT infrastructure, such as relational database management systems (RDBMS), is no longer reliable to handle and manage healthcare big data. Although it can control and process structured data efficiently, it requires a tedious ETL process to ingest data without a proper structure [110].

2.5.1 Web service in Big data analytics

Due to this enormous usage of data all over the world, the term big data is widespread and is becoming a part of every business process. Collecting and storing this vast amount of information is worth nothing without retrieving useful knowledge through processing and analysing it [111]. Unlike traditional data generation models, big data contains a variety of data from diverse sources with high velocity in huge volumes.

The evolution of big data has resulted in the healthcare industry transitioning to electronic health records (EHRs) and the management of digital archives. EHRs may contain information about various patients attributes which could include demographic details, previous medication history, allergies, vaccination status, laboratory test results, EMR scan reports, payer information, insurer de-

tails, earlier visits to the hospital and so on [112].

The sources of the data also range from mobile devices, social media feeds, wearable devices, radio frequency identification (RFID) devices, sensors and monitoring devices attached to the patient and their bed [113]. Even though health IT has seen tremendous technological development, it is challenging to manage this complex data flood.

Therefore, it is obvious that the research on EHR and patient related auxiliary data will illuminate improvised healthcare at the point of care. Inevitably, big data analytics on healthcare will enhance clinical operation by providing more relevant, efficient, error-free and cost effective diagnosis and medication. Since the data are also shared with the patient, it will increase the transparency and reliability of medical institutions [114].

Proactive, patient-centric and tailored healthcare medication recommendation to the individual is based on the analysis of the corresponding person's EHRs, genomic profile, laboratory data and other related supporting data [115]. The Healthcare organisations create structured, semi-structured and unstructured data in the form of EHRs. Even more importantly, only 20% of data is in a structured format, which can easily be utilised by data scientists using analytics machines but the semi / unstructured production rate is 15 times higher than the structured one [114].

2.5.2 Healthcare applications using SOA

Systems like CARE (Collaborative Assessment and Recommendation Engine) and its iterative version called ICARE predict the future disease risk of the patient by analysing the patient's previous history [116]. However, the CARE system depends only on ICD-9-CM (International Classification of Diseases, 9th revision, Clinical Medication) rather than the patient's lab results and other available resources. The proposed architecture has the capability to connect with trusted third parties to collect data on patients.

HealthCare ND is also a system to predict the future disease risk of the patient [117]. However, it is an interactive system, which obtain health-related information from the patient and process the data with ICD-9-CM codes then return the results of the patient's future prediction. The HealthCare ND system does not take any laboratory data into account to avoid the cost and time needed to perform the prediction. It solely depends on the data given by the user alone.

Abbas et al. proposed a system called collaborative filtering-based disease risk assessment (CFDRA) [118], which is a cloud-computing-based risk assessment prediction system. The user requests an assessment and provides their details to the system. Then the system checks for a possible profile match with the request. Once it finds the matching profile, it will correlate both profiles and then calculate the disease risk assessment score for the requesting user. The CFDRA system has another module to find the expert's recommendation for the user. It analyses the vast number of tweets and identifies the experts for the health- care recommendation, then it ranks them based on reputation. The user receives their disease risk assessment score along with some expert's contacts.

The user can interact with the expert and seek their advice.

In a system proposed by Patel et al., genomics and clinical data from patients' EHRs are utilised to predict cancer risk [119]. They develop a knowledge base consisting of different types of cancers and their symptoms and possible drug suggestions for cancer. The knowledge base is a well-known graph-based database developed by processing the previous patient's datasets. A machine learning algorithm is used to make a support vector machine to learn the patterns of the genomic structures by analysing the omics data so a precise classification model can be developed. This classification model development needs the data to be pre-processed. If a new patient profile comes to the system, it considers the knowledge base, patient data, clinical information and classification model, and then the interface engine identifies whether the patient has cancer or not. If the system determines that the patient has cancer, it suggests the most suitable medication and treatment.

The physical therapy-as-a-service (PTaaS) application is a model designed to connect the sensors available in the patient's home and the therapist's office computer. They use layer 3 internet connection to connect both ends and transmit the real-time data from the patient's sensor and video cameras. All the data is stored in the therapist's institutional database for processing and the therapist will provide physical therapy through the Internet [120].

2.5.3 Datalake for Service oriented Architectures

The aforementioned healthcare systems using well-defined business rules and vocabularies. However, health IT systems need a different type of data manage-

ment architecture to address their particular challenges. Particularly, healthcare IT needs a late binding model, which is flexible, time efficient, scalable, and adaptable [121].

Blockchain data structures utilise data lakes to support data from a variety of sources such as patients' mobile applications, wearable sensors, EMR's, documents and images [122]. A personal data lake system proposed by Alrehamy and Walker lays the foundation for designing our proposed approach. They provide a centralized location for all the data on a single user from different social networks. They also argued that their system improved privacy and security by storing data in a single location, and the user is given the right to design the access rights to their data. They also proposed the personal lake serialization format (PLSF) approach to store meta-data. However, the personal data lake was restricted with structured and semi-structured data, and it was not able to manage unstructured data [123].

The proposed approach utilizes the efficient data lake system for healthcare data. This system can ingest unstructured, semi-structured and structured data and store it at a low cost using HDFS. Moreover, the data lake can be connected with trusted third-party data providers by adopting the Apache spring XD tool with our proposed architecture. This framework does not require any ETL process to pull the data to the HDFS. Therefore, it reduces time delay when making the data available to analytics processing. Hence, a data-lake-based healthcare recommendation system will overcome the drawbacks of traditional data architectures and provide additional capabilities for the future of data reusability.

2.5.4 Chapter summary

The software source metrics were usually calculated at the micro-level and aggregated to represent the quality of software at the macro level. There are many popular aggregation techniques proposed in the literature. However, significantly less focus was given to explore the aggregation model considering the nature of software source code data structure. This research addresses this gap and explores the econometric aggregation model named the Theil index to aggregate the software source code metrics. Web service composition is vital for a service-oriented architecture. However, choosing the most suitable candidate web service for the composite is still a challenging task. Recently, many researchers proposing a keyword-based search to list the best-suited web services for the composition. However, most of the literature's proposed approaches were did not efficiently utilise the nature of keyword-based search. This research explores the power of keyword-based search using a relational database. The data architecture for service-oriented architecture is a significant part. Although several frameworks are proposed in the literature, finding a suitable data architecture is still an open question. The conventional data architectures such as Enterprise data warehouse could not handle the unstructured, unprocessed data. This research addresses this gap and proposed data lake architecture as an alternative data architecture for service-oriented architecture. The data lake is capable of handling instructed, unaltered data from various sources in real-time.

CHAPTER 3

IMPROVED SOFTWARE SOURCE CODE METRICS TO PREDICT WEB SERVICE QOS

Even though a Web service can act as a single stand-alone system, it comprises many methods and classes. Thus, the aggregation of different micro level metric values for each set of code helps to obtain a single value for the global evaluation of a Web service. The arithmetic mean is the predominantly used aggregation function for most of the performance evaluation metric calculation models. However, most of the software quality metric values are very skewed. Consequently, the simple mean function is not reliable against this kind of distributions. A known approach to reduce this problem is to select a known family of distribution such as log-normal, exponential or negative binomial and aggregate the metric value by fitting its observed parameters. This method is not viable as it repeats the fitting process whenever a new metric is introduced.

This research proposes an aggregation model for Web services at the micro level by utilizing an inequality measure, the Theil index. The Theil index is widely used in econometrics to study the inequality of welfare or the income distribution among various groups of people. The distribution of data in econometrics is very much like data distribution in software engineering. As it has the potential to summarize a large amount of data, it has been proposed recently as an aggregation scheme for software source code metrics.

To achieve the goal of this study, the objectives are as follows:

- Extracting the source code metrics for micro level attributes from a Web service description file named the Web Service Description Language

(WSDL).

- Calculating source code metrics values using micro level software components.

- Identifying the correlation between source code metrics (example: weighted methods per class, lack of cohesion in methods) and QoS properties.

- Automating the prediction of QoS properties using the correlation via machine learning techniques.

The framework will extract the class files from the WSDL files obtained from the QWS-WSDL dataset using WSDL2Java tool. Our application will calculate source code metrics at the class level. Potential source code metrics will be obtained by feature selection and reduction for each quality of service. Finally, a Machine learning with various kernels will be trained to predict the QoS properties.

3.1 Proposed Work

3.1.1 Research Framework

Figure 3.1 demonstrates our proposed structure and the methodologies used for this research. The system comprises of many steps, and the first step is being to extract Java class files from WSDL files using WSDL2java software. The next

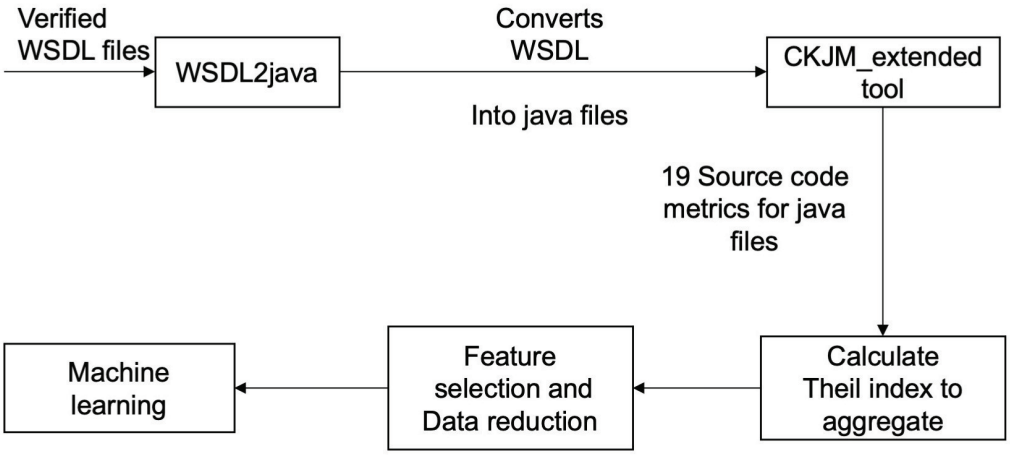


Figure 3.1: The proposed framework

step is to measure the various metrics of the source code. By following the steps described in Figure3.1, the CKJM_extended tool is used to measure the object-oriented source code metrics. For each Java class file, the application calculates 19 metrics of the source code. Then, the *Theil* index is used as an aggregation technique, to sum up, a single value for all the metrics of the source code. The principal component analysis is used to remove irrelevant features to achieve dimensionality reduction. As a final step, linear regression is applied to predict the quality metrics for the Web services. By using different combinations of available source code metric sets, seven sets of metrics are available to evaluate the performance of regression learner. The performance of the prediction model using different sets of metrics is evaluated by calculating the estimator evaluation metrics.

3.1.2 Source code metrics aggregation

The data on wealth distribution inequality from economics and source code metrics of software share a similar structure. The Gini coefficient, a widely applied economics inequality measure, attracted attention in the field of software metrics. It can be easily explained using the Lorenz curve. However, the Gini coefficient has a significant drawback as it cannot be decomposed [80]. Serebrenik et al. proposed another inequality measure named the Theil index to use instead of the Gini index as it is decomposable so it can be used not only to calculate the inequality but also to explain it. Moreover, it is not specific to any metrics so it can be used to aggregate a wide range of metrics [124]. Therefore, the Theil index is preferred to calculate the aggregation for source code metrics.

3.1.3 Feature reduction & selection

The framework uses the principal component analysis (PCA) as the data preprocessing method for feature extraction and selection. For each PC (principal component), eigenvalue, variance percentage, cumulative percentage and source code metrics are obtained. PCs with eigenvalue more than one are considered as potential source code metrics cohorts. Table 3.1shows the PCA results of the object-oriented metrics. Of the 19 source code metrics, 12 were identified as having the potential to predict QoS properties. Tables 3.2 and 3.3 show the PCA results for Baski & Misra metrics and Sneed's metrics.

Principal component	Eigenvalue	% Variance	% cumulative	Metric Interpreted
PC 1	5.744292522	38.29528348	38.29528348	WMC,CBO,RFC, Ca,LCOM,Ce,NPM, LOC,MOA,CAM,AMC
PC 2	3.553881589	23.69254393	61.98782741	DIT, CBO, LCOM, Ca, Ce, LOC, DAM, MOA, MFA, CAM, AMC
PC 3	1.92886401	12.8590934	74.8469208	WMC,CBO,RFC, Ca,LCOM,Ce,NPM,AMC
PC 4	1.104911736	7.36607824	82.21299904	WMC, RFC, LCOM,Ca,Ce,NPM, DAM, MOA, MFA,CAM,AMC

Table 3.1: PCA results for object-oriented metrics

Principal component	Eigenvalue	% Variance	% cumulative	Metric Interpreted
PC 1	2.908	48.465	48.465	OPS, DW,MRS,DMC
PC 2	1.489	24.809	73.275	DMC,DMR,ME

Table 3.2: PCA results for Baski & Misra metrics

Principal component	Eigenvalue	% Variance	% cumulative	Metric Interpreted
PC 1	2.359	29.490	29.490	Data flow complexity, Data access complexity, Interface complexity, Control flow complexity, Decisional complexity, Branching complexity, Language complexity
PC 2	1.946	24.327	53.817	Data complexity, Data flow complexity, Decisional complexity, Branching complexity, Language complexity
PC 3	1.320	16.501	70.318	Data flow complexity, interface complexity, Language complexity

Table 3.3: PCA results for Sneed's metrics

3.1.4 Machine learning

The aim of this research is to examine the impact of aggregation methods of source code metrics (e.g. Lack of cohesion in methods, coupling between object classes) on predicting QoS characteristics (e.g. reaction time, accessibility, throughput, testability, interoperability, etc.). Therefore, a simple regression model called the multiple linear regression model is preferred to employ the prediction. By fitting a linear equation to the measured data, multiple linear regression aims to model the relationship between two or more independent variables and a response variable. The training set developed according to the correlation standards is defined in [73]. Then the number of latent variables need to be defined for each QoS property. The next step is to build a model to generate the value of the QoS property using the training set knowledge base. The data set was submitted to a 10-fold cross-validated paired t-test analysis. In the 10-fold cross-validated paired t-test procedure the dataset is segmented into 10 equally sized parts, each of which is then used for analysis, while the remaining 10-1 parts (joined together) are used to train the regressor (i.e., generic k-fold cross-validation). To demonstrate the reliability of the regression learner, two different performance metrics (MAE, RMSE) are used.

3.2 Experiments & Analysis of results

3.2.1 Research Questions

Our experimental study is designed to answer the following two research questions:

RQ1.1: Will the proposed methodology improve the predictability of source code metrics?

A software system is not a single stand-alone system to provide the solution. Normally, it comprises many subordinate pieces of code such as class, method or function. Therefore, software metrics must calculated at the micro level and should be aggregated to the macro level to represent the source code metrics of a software system. Since software code metrics are highly skewed values, it is inappropriate to use simple statistical aggregation (mean, median, etc.,) methods. The proposed inequality distribution models are very successful in economics data, which is as skewed as software source code data.

RQ1.2: Can we predict the quality of service properties of Web services using source code metrics?

During the software development life cycle, developers extract source code metrics to evaluate maintainability to reduce future issues with the system. Thus, source code metrics and quality of service properties are correlated. The research uses the correlation between source code metrics and quality of service

to predict the QoS of Web services. This study use linear regression based Machine learnings to create a prediction model.

3.2.2 Variables and Objects

Independent Variable

A technique under investigation is defined as an independent variable. The Theil index is selected as the independent variable for this research. Dutch statistician Henri Theil presented an inequality measure named the Theil index [125]. Given a (continuous) univariate distribution function F with the support $X \subseteq \mathbb{R}$ and the mean $\mu(F)$ the first Theil index is defined as:

$$I_{Theil}(F) = \int \frac{x}{\mu(F)} log\left(\frac{x}{\mu(F)}\right) dF(x) \tag{3.1}$$

The Theil index was introduced for the field of unequal income distribution aggregation. Let $x_0 \in X$ be a particular value of income, F is a distribution of income in the population, and $F(x_0)$ is the proportion of the population with income x less than or equal to x_0. Source code metrics calculated at the micro level are aggregated to represent the macro level software using Theil index. Metrics that have negative values cannot be aggregated using the Theil index because of the logarithmic calculation in its formula. Since $logx$ for $x \leq 0$ can produce an undefined value, the Theil index may also be an undefined value if it contains non-positive values [80].

Dependent variable

Mean absolute error (MAE) and root mean squared error (RMSE) are the two well-known statistical precision measurements utilized to assess the prediction results. MAE is the average absolute deviation of predictions to the ground truth data. For all test services and test QoS properties, MAE is calculated as:

$$MAE = \frac{\left(\sum ij \left\| Q_{ij} - \hat{Q}_{ij} \right\|\right)}{N} \tag{3.2}$$

In Equation 3.2, Q_{ij} denotes the observed QoS value of Web service j obtained from data set entry i; $\hat{Q}_{ij}$ is the predicted QoS value; and N is the number of predicted values. A smaller value of MAE indicates a better prediction result. RMSE can be expressed as:

$$RMSE = \sqrt{\frac{\sum ij \| Q_{ij} - \hat{Q}_{ij} \|^2}{N}} \tag{3.3}$$

RMSE can be measured using Equation 3.3 to find out the differences between the actual and predicted values. Once the model yields more than a 90% accuracy level, the Machine learning can predict the five chosen QoS properties for the WSDL of a service.

Objects

The proposed framwork used the WS-Dream dataset, an open source dataset produced by a group of researchers from The Chinese University of Hong Kong. WS-Dream dataset contains two versions of the datasets and the experiments

used version 1. The dataset contains the URLs of 5825 Web services and the response time and throughput readings from 339 geographically distributed users. The dataset also has more details about both the Web services and users such as IP address, country, continent, longitude, latitude, region, and city.

The main goal of this research is to identify the implications of the aggregation of source code metrics on QoS prediction. Therefore, it is necessary to calculate the source code metrics for the target Web services using its Web Service Description Language (WSDL). wsimport is a Java function to process WSDL files and extract class files for the corresponding Web services.

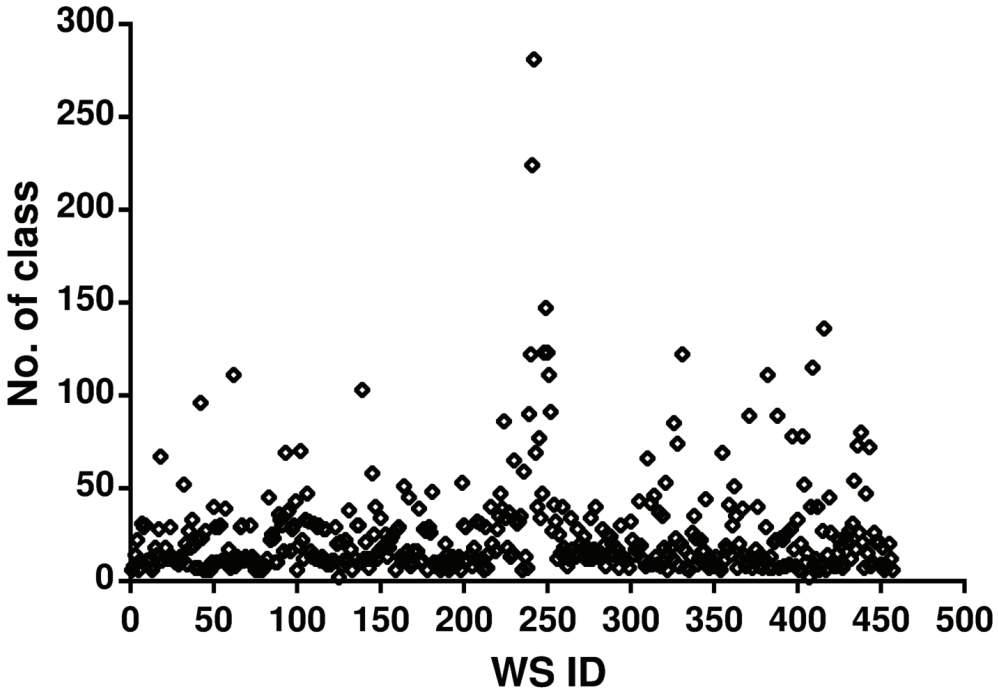
Since the dataset only contains the URL for the WSDL of Web services, a Java application is developed to crawl the WSDL files using the URLs from the dataset. Only 457 of the 5825 Web services have an active WSDL available on the Internet. As previously mentioned, wsimport was used to extract the Java class files. The number of class files per Web services ranges from one to 281. Figure3.2 shows the number of class files per Web service. The x-axis represents the Web service ID and the y-axis represents the number of Java files extracted from the Web service.

3.2.3 Empirical environment

Source code metrics

This research used three sets of metrics, namely the object oriented source code metrics proposed by Chidamber et al. [19], Baski and Misra metrics [21] and Sneed's metrics [20].

Figure 3.2: Number of Java files extracted from the available WSDL



Chidamber and Kemerer Metrics: The Ckjm_extended tool introduced by Chidamber et al. [19] can be utilized to calculate 19 size and structure software source code metrics from the generated Java class files. The metrics are as follows:

- **WMC - Weighted methods per class** Weighted methods of a class per WMC class metric are mostly the sum of the complexities of its methods. We can use the cyclomatic complexity as a measure of complexity, or we can assign each method an arbitrary complexity value of 1. The Ckjm software assigns each method a complexity value of 1, and thus, the WMC value is equal to the number of methods in the class.
- **DIT - Depth of Inheritance Tree** For each class the inheritance levels from

the top of the object hierarchy are determined by the depth of the inheritance tree (DIT) metric. The minimum value of DIT is 1 in Java, in which every class inherits object.

- **NOC - Number of Children** The number of children in the family (NOC) metric actually calculates the number of immediate descendants of the family.
- **CBO - Coupling between object classes** The coupling between the object classes (CBO) metric describes the number of classes coupled to the class (efferent couplings and afferent couplings). This coupling will take place via method calls, field accesses, inheritance, arguments, form returns, and exceptions.
- **RFC - Response for a Class** The metric called the response for a class (RFC) measures the number of different methods that can be executed when an object of that class receives a message (when a method is invoked for that object). Ideally, we would want to find for each method of the class, the methods that class will call, and repeat this for each called method, calculating what is called the transitive closure of the method's call graph. This process can however be both expensive and quite inaccurate. In ckjm, we calculate a rough approximation to the response set by simply inspecting method calls within the class's method bodies. The value of RFC is the sum of number of methods called within the class's method bodies and the number of class's methods. This simplification was also used in the 1994 Chidamber and Kemerer description of the metrics.
- **LCOM - Lack of cohesion in methods** The lack of cohesion of methods (LCOM) metric for a class counts the sets of methods in a class not con-

nected by exchanging any of the fields of the class. This metric's original definition (which is the one used in ckjm) considers all pairs of methods of a sort. Both methods reach at least one particular field of the class in some of these pairs, although the two techniques do not share any common field reach in other pairs. The lack of cohesion of methods is determined then by eliminating the number of method pairs which do not share a field from the number of method pairs. Note that the number of disjoint graph components of the class methods is used as a calculation reference for subsequent definitions of this metric. Others also changed the concept of communication to include calls between class methods. Chidamber and Kemerer's initial (1994) concept follows the method ckjm.

- **Ca - Afferent couplings** Afferent couplings in a class are an indicator of how many other classes utilise the particular class. In that context, the coupling has the same meaning as that used in CBO calculations.
- **Ce - Efferent couplings** Efferent couplings in a class are an indicator of how many other classes the individual class uses. In the scope of Ce, the coupling has the same meaning as that used for measuring CBO.
- **NPM - Number of Public Methods** The NPM metric simply counts all the methods in a class that are declared as public. It can be used to measure the size of an API provided by a package.
- **LCOM3 -Lack of cohesion in methods.** LCOM3 varies between 0 and 2.

$$LCOM3 = \frac{\left(\frac{1}{a}\sum_{a}^{j=1}\mu\left(A_j\right)\right) - m}{1 - m}$$

m - number of procedures (methods) in class

a - number of variables (attributes in class

$\mu(A)$- number of methods that access a variable (attribute)

The constructors and static initializations are taking into accounts as separately methods.

- **LOC - Lines of Code.** The lines are counted from java binary code and it is the sum of number of fields, number of methods and number of instructions in every method of given class.
- **DAM: Data Access Metric** This metric is the ratio of the number of private (protected) attributes to the total number of attributes declared in the class. A high value for DAM is desired. (Range 0 to 1)
- **MOA: Measure of Aggregation** This metric measures the extent of the part-whole relationship, realized by using attributes. The metric is a count of the number of data declarations (class fields) whose types are user defined classes.
- **MFA: Measure of Functional Abstraction** This metric is the ratio of the number of methods inherited by a class to the total number of methods accessible by member methods of the class. The constructors and the java.lang.Object (as parent) are ignored. (Range 0 to 1)
- **CAM: Cohesion Among Methods of Class** This metric computes the relatedness among methods of a class based upon the parameter list of the methods. The metric is computed using the summation of number of different types of method parameters in every method divided by a multiplication of number of different method parameter types in whole class and

number of methods. A metric value close to 1.0 is preferred. (Range 0 to 1).

- **IC: Inheritance Coupling** This metric provides the number of parent classes to which a given class is coupled. A class is coupled to its parent class if one of its inherited methods functionally dependent on the new or redefined methods in the class. A class is coupled to its parent class if one of the following conditions is satisfied: One of its inherited methods uses a variable (or data member) that is defined in a new/redefined method. One of its inherited methods calls a redefined method. One of its inherited methods is called by a redefined method and uses a parameter that is defined in the redefined method.
- **CBM: Coupling Between Methods** The metric measure the total number of new/redefined methods to which all the inherited methods are coupled. There is a coupling when one of the given in the IC metric definition conditions holds.

- **AMC: Average Method Complexity** This metric measures the average method size for each class. Size of a method is equal to the number of java binary codes in the method.
- **CC - The McCabe's cyclomatic complexity** It is equal to number of different paths in a method (function) plus one. The cyclomatic complexity is defined as:

  $CC = E - N + P$

  where

  E - the number of edges of the graph

  N - the number of nodes of the graph

P- the number of connected components

A Java-based system was developed to be integrated with the ckjm tool and calculate 19 metrics for all the 457 Web services. Listing shows an example calculated metrics for a single class of a passport validation webservice. For the next step, the R language used for the statistical calculation. The R library called *ineq* was used to calculate the Theil index I_{Theil} for the Web services.

```
<?xml version="1.0" ?>
<ckjm>
<metric>
<classname>passport.GetVersionResponse</classname>
                    <WMC> 14 </WMC>
                    <DIT> 1 </DIT>
                    <NOC> 0 </NOC>
                    <CBO> 7 </CBO>
                    <RFC> 15 </RFC>
                    <LCOM> 91 </LCOM>
                    <Ca> 4 </Ca>
                    <Ce> 3 </Ce>
                    <NPM> 6 </NPM>
                    <LCOM3> 1.0769 </LCOM3>
                    <LCO> 86 </LCO>
                    <DAM> 0.5000 </DAM>
                    <MOA> 0 </MOA>
                    <MFA> 0.0000 </MFA>
                    <CAM> 0.3776 </CAM>
```

```
                    <IC> 0 </IC>
                    <CBM> 0 </CBM>
                    <AMC> 5.0000 </AMC>
                    <CC> 0 </CC>
</metric>
</ckjm>
```

Sneed's tool: Sneed et al. [20] developed a tool named softAudit to measure the Web service interfaces. The suite consists of nine different source code metrics to measure the complexity of service interfaces: data complexity (Chapin metric), data flow complexity (Elshof metric), data access complexity (card metric), interface complexity (henry metric), control flow complexity (McCabe metric), decisional complexity (McClure metric), branching complexity (Sneed metric),language complexity (Halstead metric), weighted average program complexity. The Softaudit tool provided by Sneed et al. was used to calculate the 09 complexity metrics for all the Web services by processing micro level class files. Nine quality metrics using Sneed's tool were also calculated namely degree of modularity, degree of portability, degree of reusability, degree of testability, degree of convertibility, degree of flexibility, degree of conformity, degree of maintainability, weighted average program quality.

Baski and Misra Metrics: Baski and Misra [21] proposed a tool to compute six different complexity metrics of WSDL files. These metrics are based on analyzing the WSDL and XSD schema elements. The metrics are data weight of a WSDL (DW), distinct message ratio (DMR), distinct message count (DMC), message entropy (ME), message repetition scale (MRS) and operations per ser-

vice (OPS). Baski & Misra metrics are calculated by processing the WSDL file for each Web service rather than processing micro level class files. Baski & Misra's tool is used to calculate the six metrics for the 457 Web services WSDL files.

The framework uses the three sets as mentioned above of source metrics as independent variables and quality metrics calculated using Sneed's tool as dependent variables. Normalization is an essential process for data regression. Therefore, unsupervised normalization is applied with a range of 0.0 to 1.0 using the Weka tool. Then the metric sets are grouped in different combinations to populate more sets of metrics to compare our results. Consequently, the experiments conducted using the following metrics: 1. Chidamber and Kemerer metrics (CKM), 2. Baski and Misra metrics (BSM), 3. Sneed's metrics (SM), 4. CKM- BSM metrics, 5. CKM-SM metrics, 6. BSM-SM metrics, and 7. All metrics (AM). After grouping, there are seven different sets of metrics. Linear regression is applied to predict modularity and calculate the quality of service value. The results show that Sneed's metrics individually outperform the other sets of metrics. Object-oriented metrics also have the potential to predict the QoS value but not as efficiently as Sneed's metrics. Baski & Misra metrics have the lowest efficiency of the available groups of metrics. In summary, the metric values calculated at the micro level have better QoS prediction efficiency than those at the macro level.

3.2.4 Analysis of results

Four QoS metrics, namely modularity, testability, maintainability and, reusability for the available Web services were calculated using Sneed's tool. All three

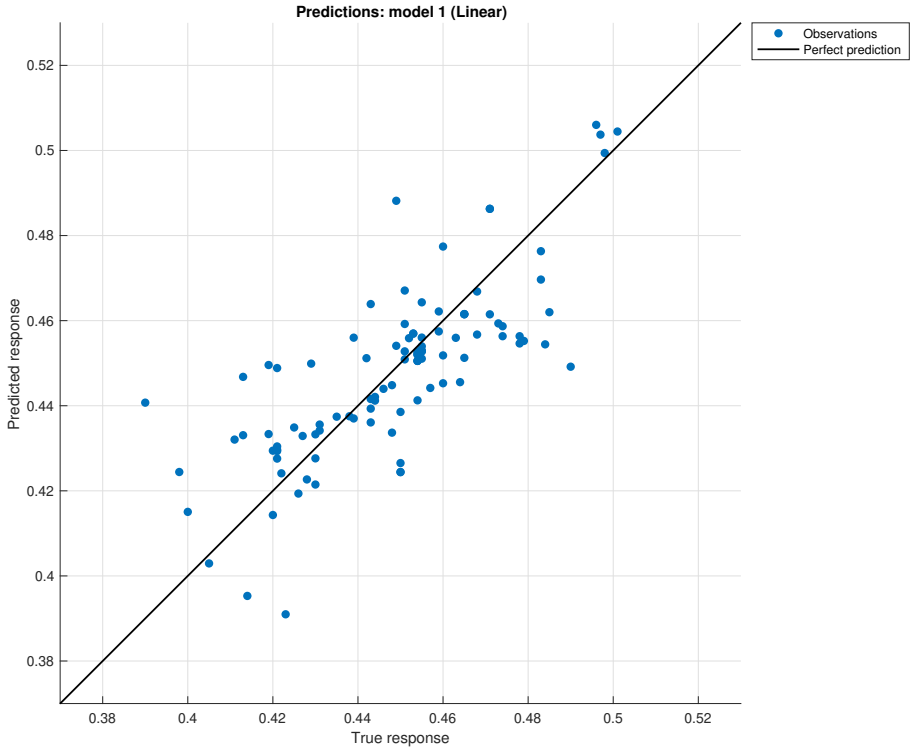


Figure 3.3: BM metrics vs Modularity prediction

sets of metrics and their combinations were used to predict the metrics values using robust linear regression. Figure 3.3 shows the graph of the actual modularity values and predicted modularity values. The RMSE and MAE values respectively are 0.284 and 0.172, which are not better values for a linear prediction model. Figure 3.4 compares the predicted modularity values using CKJM metrics and the actual modularity values. The RMSE and MAE values are 0.017 and 0.011. The prediction results are better than those obtained using Baski & Misra metrics. Figure 3.5 compares the results of the actual vs predicted modularity values using Sneed's metrics.

Figures 3.6, 3.7 and 3.8 show the prediction results with different combinations of the three sets of metrics. Neither combination produced better results

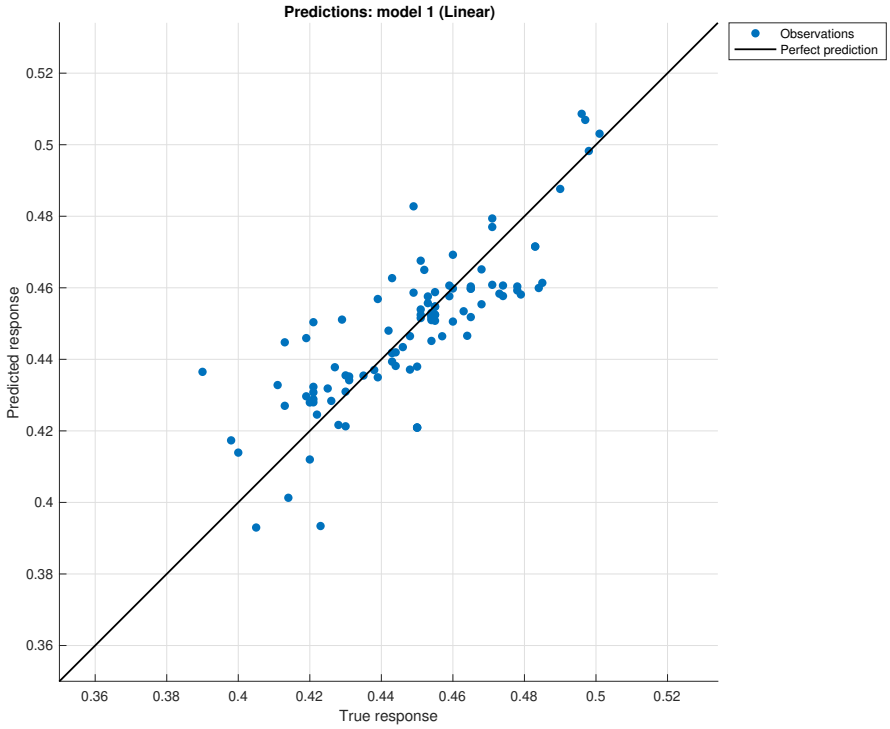


Figure 3.4: CKJM metrics vs Modularity Prediction

than Sneed's set of metrics. As shown in Figure 3.7., the BSM-CKM metrics have the least potential of producing better predictions of modularity. The prediction and actual modularity values using all the metrics are shown in Figure 3.9. It also is not able to produce better results than Sneed's metrics. Table 3.4 summarises the RMSE and AME values of different sets of metrics used for robust linear prediction. Sneed has set of metrics as stand-alone predictors to produce better results for modularity quality prediction. As shown in Figures 3.10, 3.11, 3.12, all the metrics combined produce slightly better results than BaSki and Misra (BSM) metrics for predicting testability. However, Sneed's metrics yields much better results than the all metrics and the BSM metrics.

Figures 3.13, 3.14, 3.15 show that using all the metrics combined and the

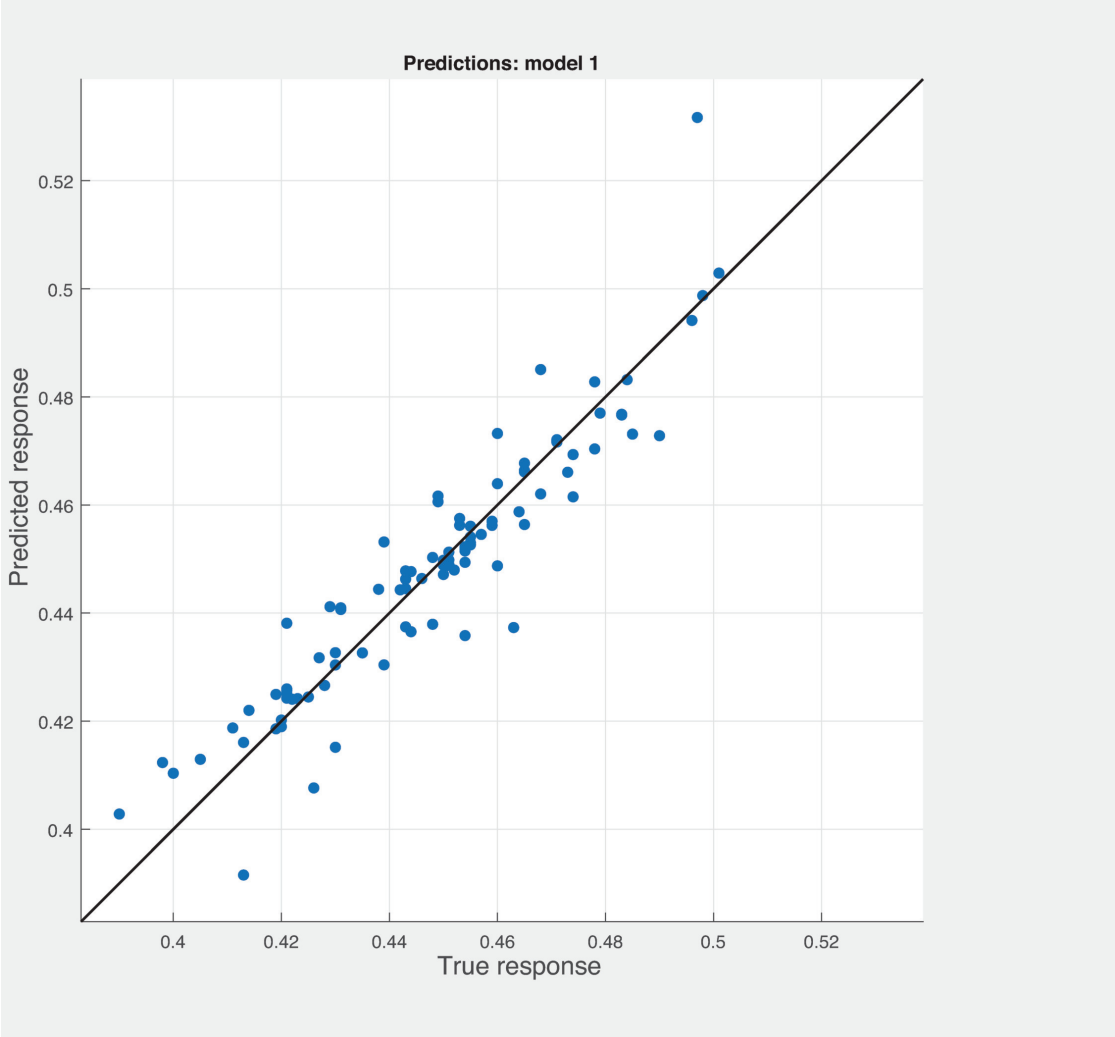


Figure 3.5: Prediction model for Modularity using SM metrics

BSM metrics did not achieve better results for reusability prediction. Sneed's metrics produced slightly better results compared to the other sets of metrics. The reusability of REST web services depends on the following characteristics:

- REST API endpoints should be simple and provide parameters to support a wide range of use cases.
- REST API endpoints should be consistently structured for SQL, NoSQL and file stores.

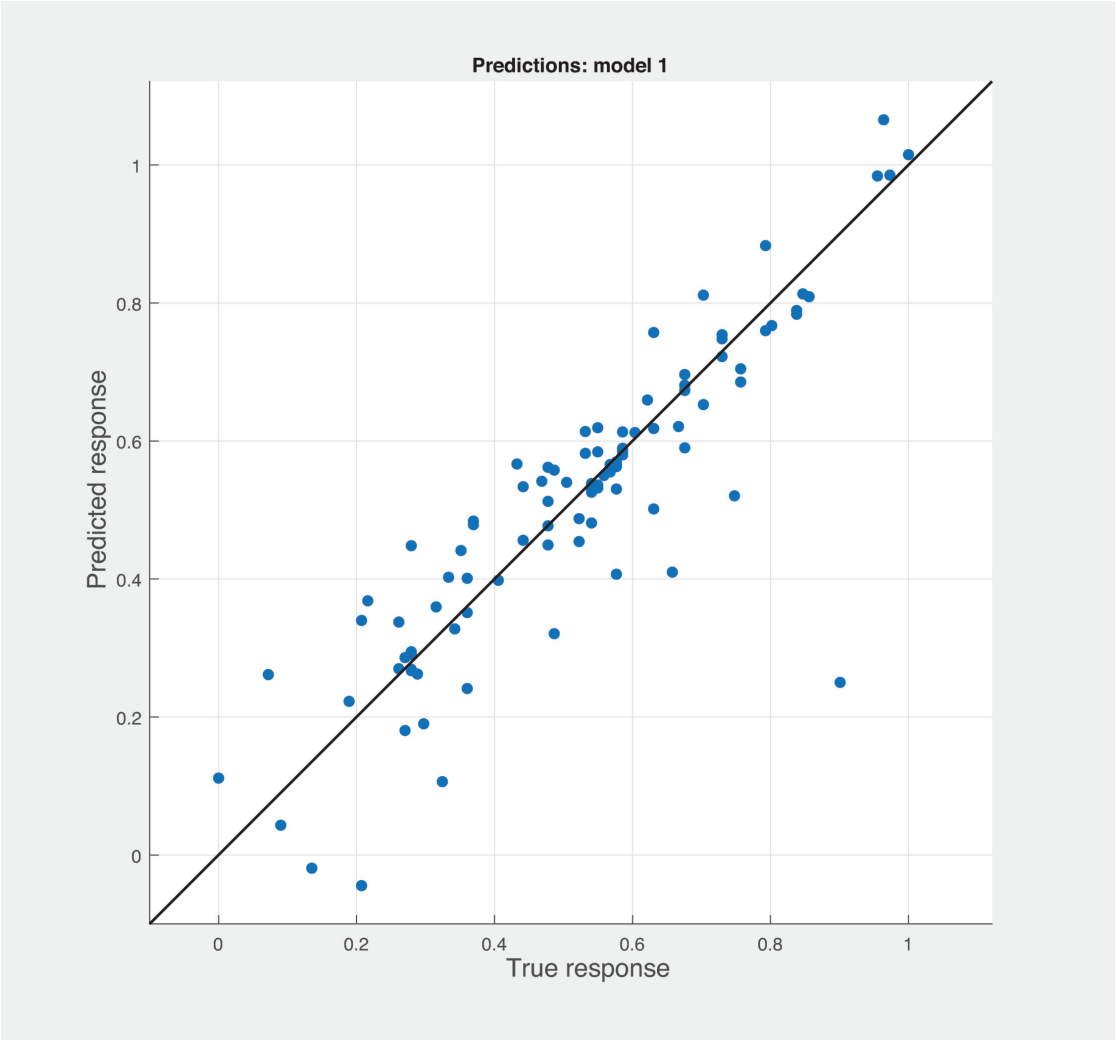


Figure 3.6: Modularity prediction results for SM-CKM metrics

- REST APIs must be designed for high transaction volume, hence simply designed.
- REST APIs should be client-agnostic and work interchangeably well for native mobile, HTML5 mobile and web applications.
- Noun-based endpoints and HTTP verbs are highly effective. Noun-based endpoints should be programmatically generated based on the database schema.
- Requests and responses should include JSON or XML with objects, arrays

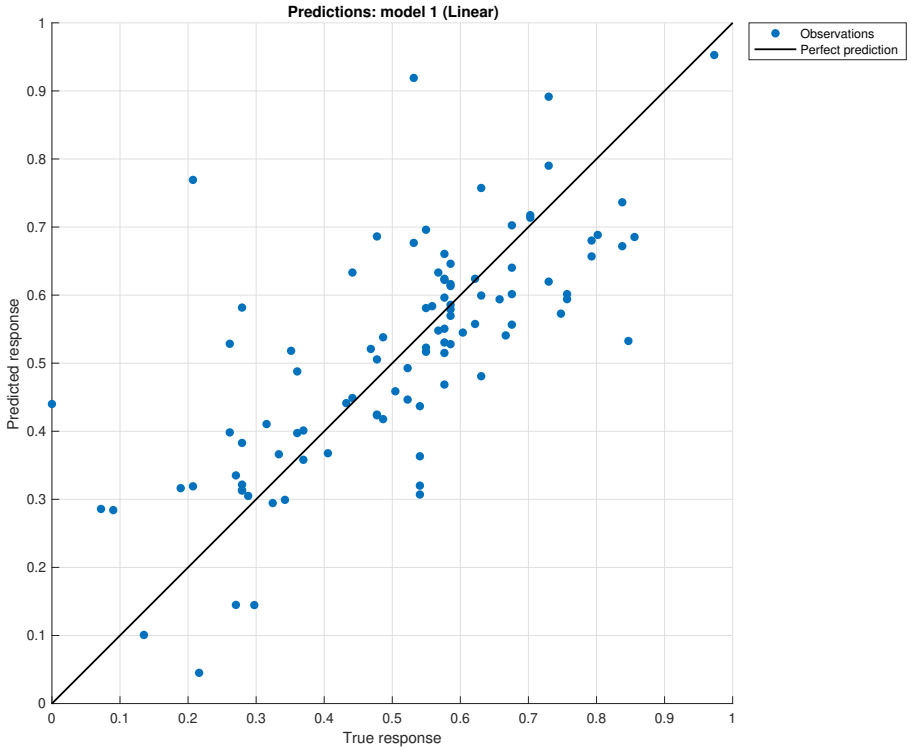


Figure 3.7: Modularity prediction results with BSM-CKM metrics

and sub-arrays.

- All HTTP verbs (GET, PUT, DELETE, etc.) need to be implemented for every use case.
- Support for web standards like OAuth, CORS, GZIP and SSL is also important.

Therefore, the reusability of a Web service depends on various external factors rather than software code. The experimental results depicted in Figures 3.13, 3.14, 3.15 support that the software source code metrics possess less potential to predict usability.

A comparison of Figures 3.16 and 3.17 shows that all the metrics together

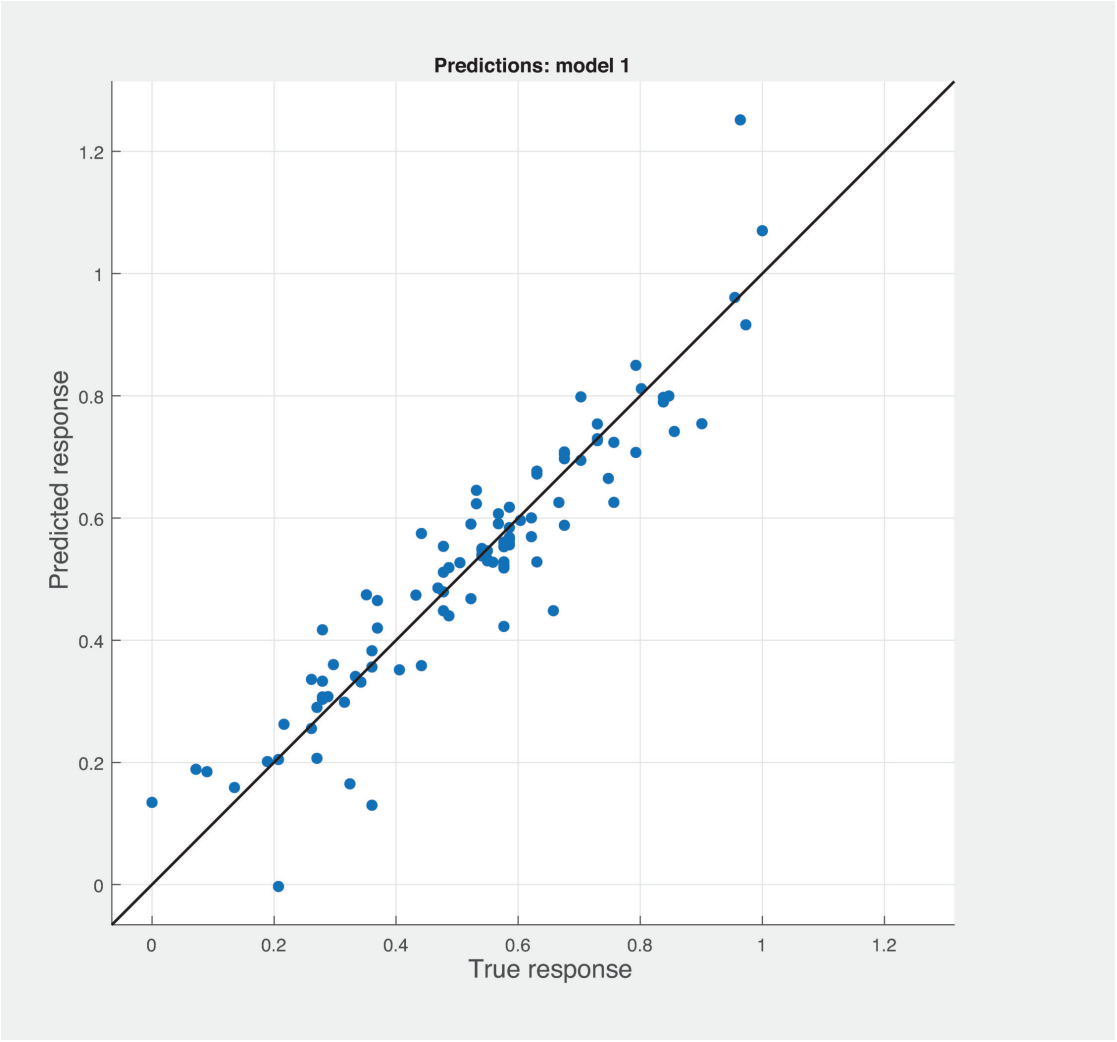


Figure 3.8: Modularity prediction results with BSM-SM metrics.
R-Squared = 0.9172.
Regression line equation: $y(x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14) = 2.7482 * x0 + 2.1856 * x1 + 2.3522 * x2 + 0.6562 * x3 + 4.6326 * x4 + -0.2187 * x5 + 1.8057 * x6 + 2.3450 * x7 + -19.0734 * x8 + -0.0930 * x9 + 0.1341 * x10 + 0.0335 * x11 + 0.1329 * x12 + 0.0456 * x13 + -0.0769 * x14 + 2.0273.$

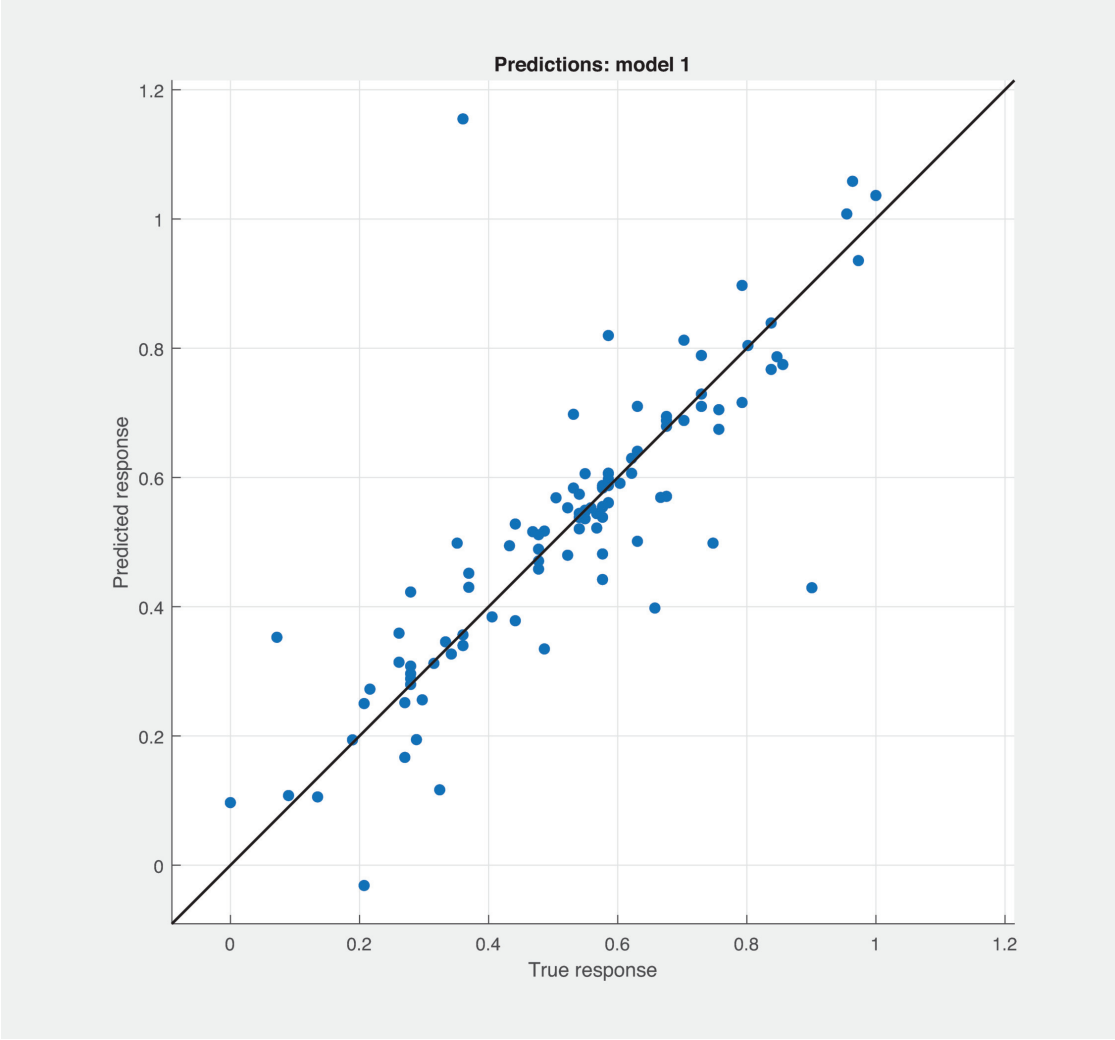


Figure 3.9: Modularity prediction vs actual for all metrics.
R-Squared = 0.9462.
Regression line equation:
$y(x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, x17, x18, x19, x20, x21, x22, x23, x24, x25, x26, x27, x28, x29) = 1.0537 * x0 + 0.7031 * x1 + 0.8778 * x2 + -0.9179 * x3 + 2.9656 * x4 + -1.7656 * x5 + 0.5277 * x6 + 0.7641 * x7 + -7.2421 * x8 + 0.7665 * x9 + -0.0321 * x10 + 0.1288 * x11 + -1.3603 * x12 + 0.2049 * x13 + 0.0841 * x14 + 0.0054 * x15 + -0.8873 * x16 + -0.1174 * x17 + 0.6724 * x18 + -0.0095 * x19 + -0.1247 * x20 + 0.0353 * x21 + 0.1189 * x22 + -0.1507 * x23 + -0.0185 * x24 + 0.1659 * x25 + 0.0422 * x26 + 0.1664 * x27 + 0.0883 * x28 + -0.1534 * x29 + 2.2353$

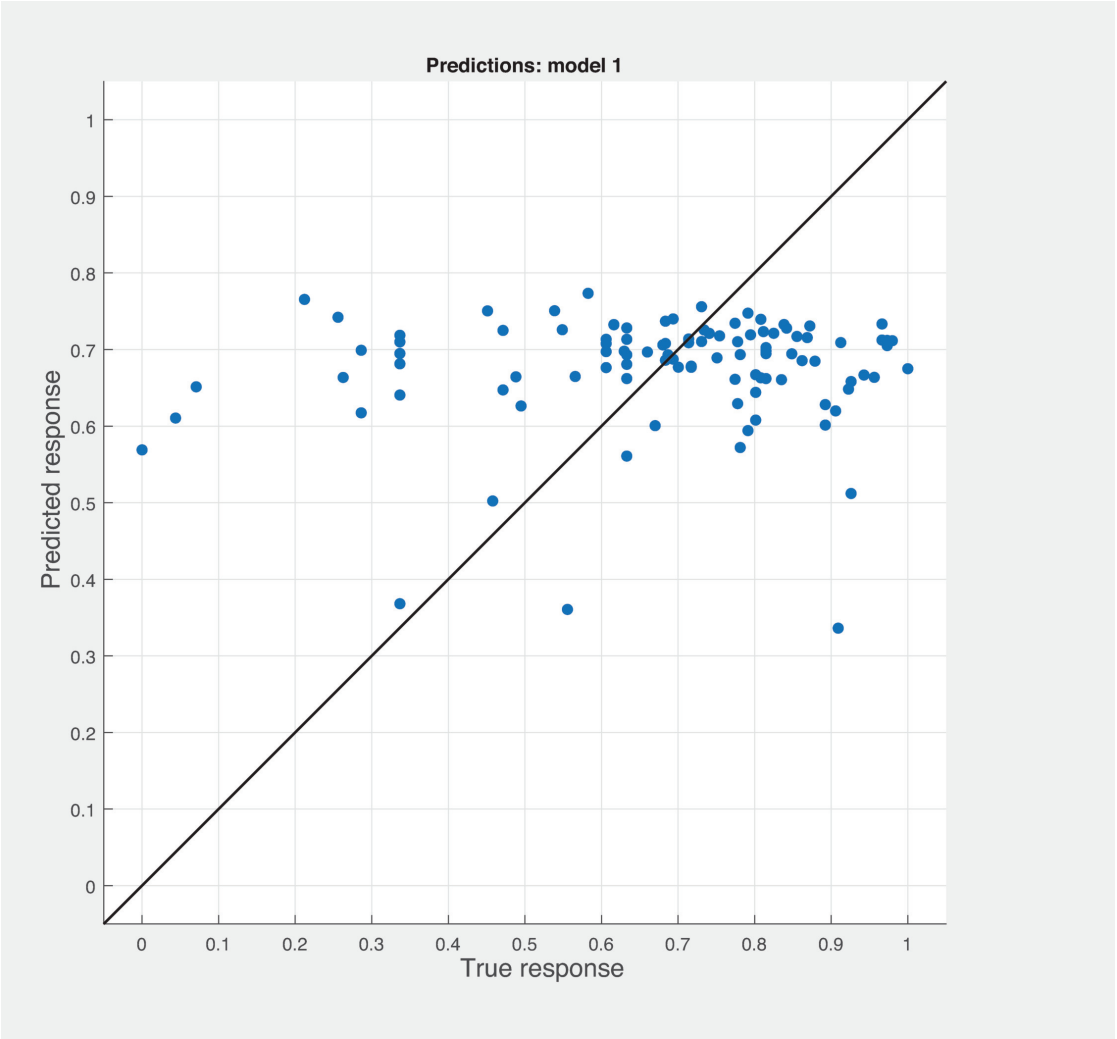


Figure 3.10: Testability prediction results with BSM metrics.
R-Squared = 0.0993.
Regression line equation:
$y(x0, x1, x2, x3, x4, x5) = 0.1893 * x0 + 0.3013 * x1 + -0.2701 * x2 + -0.4917 * x3 + -0.0315 * x4 + 0.2637 * x5 + 0.6984$

provide better prediction results than the BSM metrics. As shown in Figure 3.18, Sneed's metrics show better potential than all the metrics combined and the BSM metrics for maintainability prediction.

The comparison Table 3.4 shows that the CKM metrics and SM metrics individually produce better results. When CKM and SM are combined, the effi-

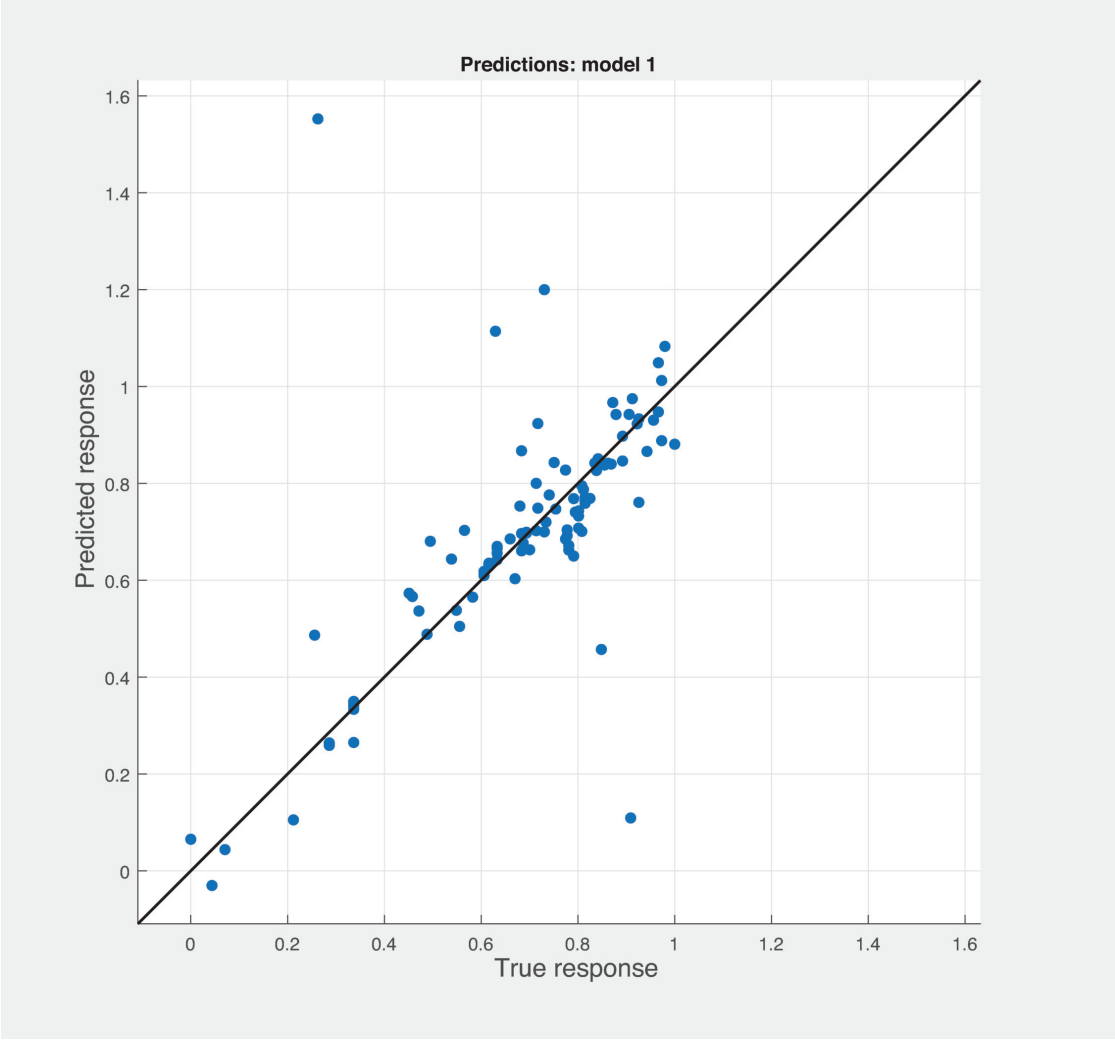


Figure 3.11: Testability prediction vs actual for all metrics
R-Squared = 0.943
Regression line equation:
$y(x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, x17, x18, x19, x20, x21, x22, x23, x24, x25, x26, x27, x28, x29) = 0.5717 * x0 + 4.2507 * x1 + 4.1199 * x2 + 3.4248 * x3 + 3.5362 * x4 + 5.3187 * x5 + 4.0641 * x6 + 3.7795 * x7 + -32.9526 * x8 + -9.1198 * x9 + -0.0838 * x10 + 0.0325 * x11 + 2.8808 * x12 + -0.3493 * x13 + -0.0582 * x14 + 0.0090 * x15 + 5.3039 * x16 + 0.1583 * x17 + 1.2257 * x18 + -0.5894 * x19 + 0.2328 * x20 + -0.0231 * x21 + -0.0966 * x22 + -0.2425 * x23 + -0.0048 * x24 + -0.0330 * x25 + -0.0385 * x26 + 0.0285 * x27 + -0.0115 * x28 + -0.0064 * x29 + 3.1057$

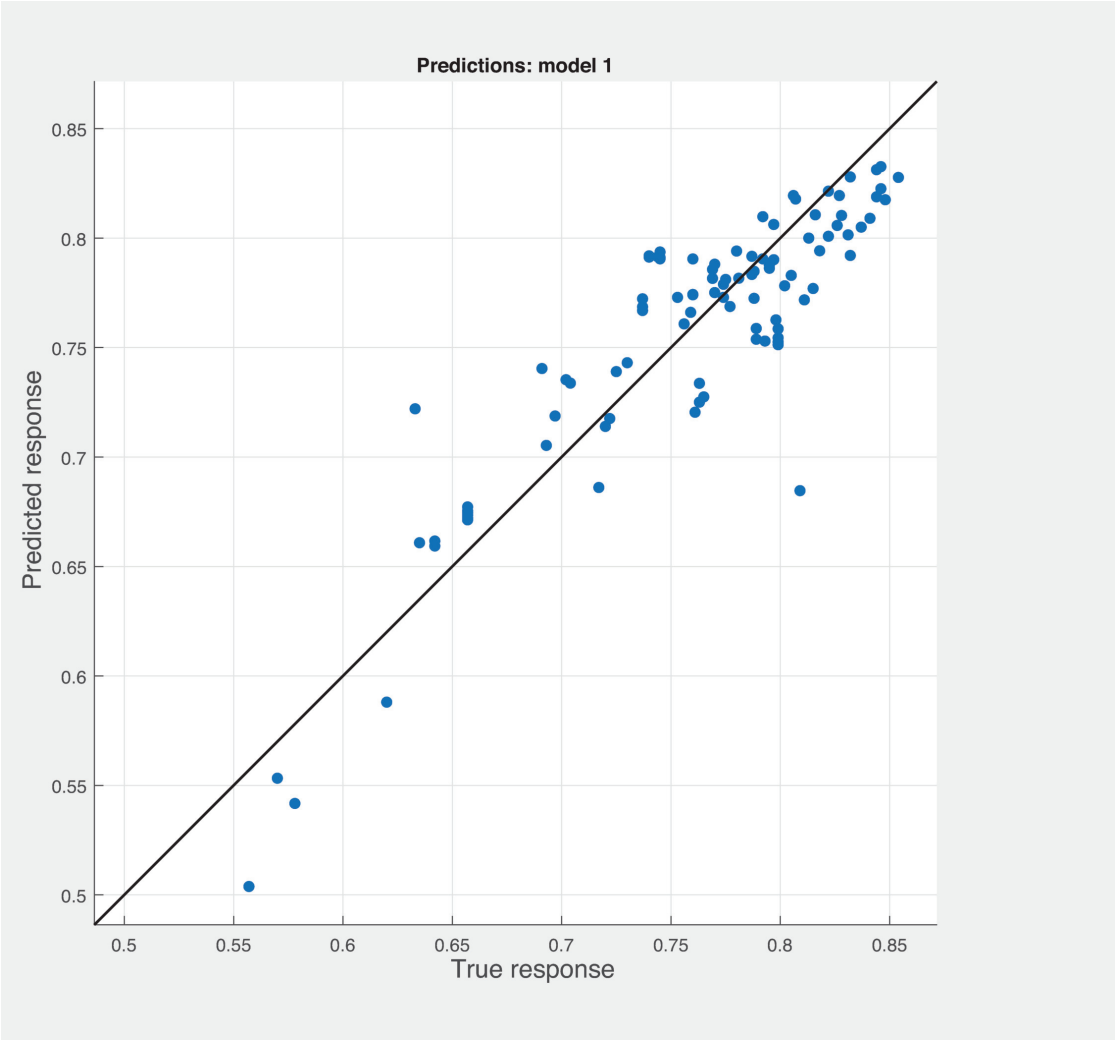


Figure 3.12: Testability prediction results with Sneed's metrics
R-squared = 0.8276
Regression line equation:
$y(x0, x1, x2, x3, x4, x5, x6, x7, x8, x9) = -2.4851 * x0 + 0.8176 * x1 + 0.7047 * x2 + 0.3849 * x3 + -0.1102 * x4 + 0.3296 * x5 + 1.2122 * x6 + 0.3881 * x7 + -5.4533 * x8 + -0.0820 * x9 + 2.4228$

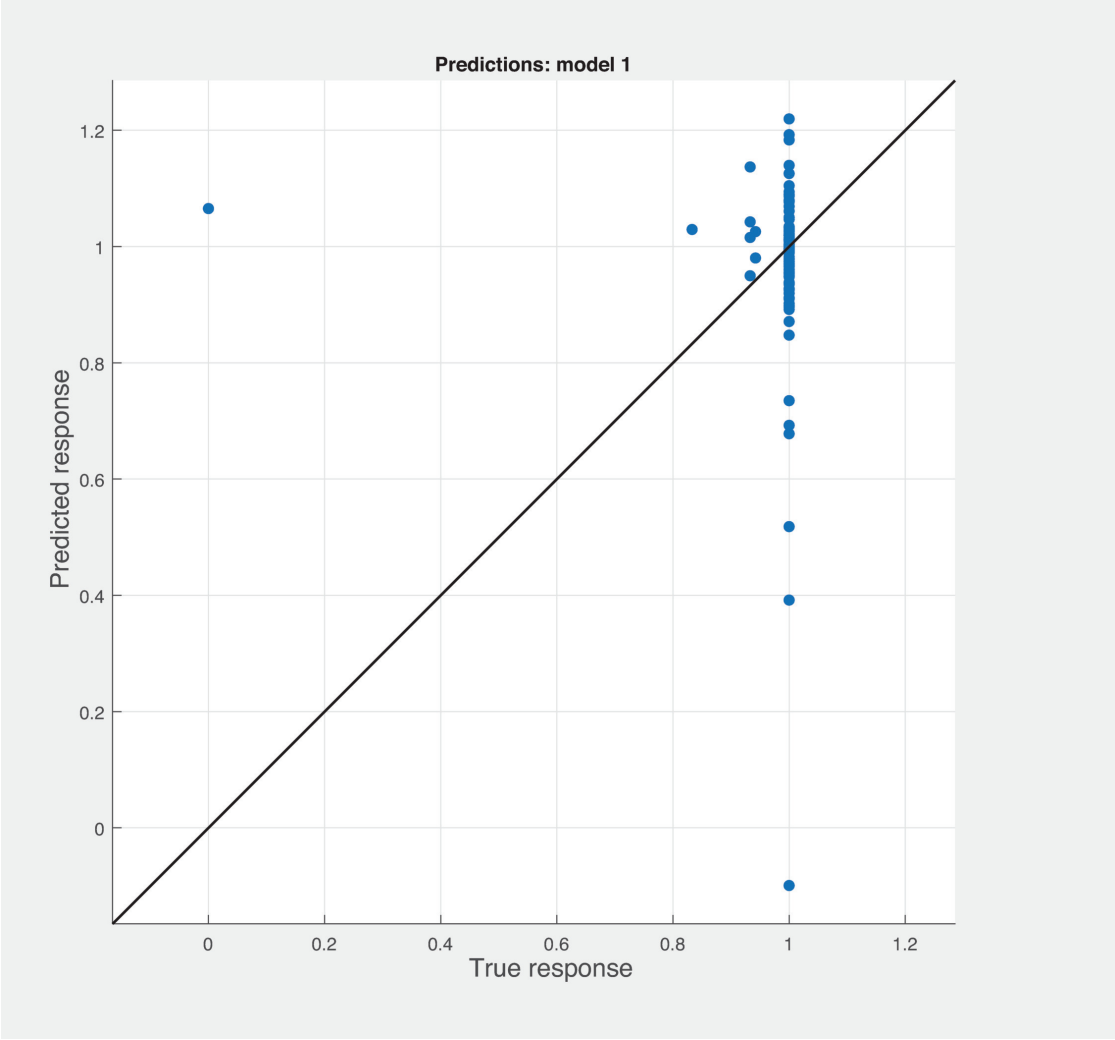


Figure 3.13: Reusability prediction results vs actual for all metrics
R-Squared = 0.5377
Regression line equation:
$y(x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, x17, x18, x19, x20, x21, x22, x23, x24, x25, x26, x27, x28, x29) = -0.9046 * x0 + -1.7537 * x1 + -1.5273 * x2 + -1.5990 * x3 + 0.3771 * x4 + -4.3778 * x5 + -1.6074 * x6 + -1.8729 * x7 + 11.7508 * x8 + -12.9236 * x9 + -0.4985 * x10 + 0.3619 * x11 + -1.8918 * x12 + 0.5193 * x13 + 0.0292 * x14 + -0.1872 * x15 + 12.3371 * x16 + -0.1079 * x17 + 1.7268 * x18 + -0.0756 * x19 + -0.1739 * x20 + 0.3085 * x21 + 0.0122 * x22 + -0.6268 * x23 + 0.0713 * x24 + 0.1877 * x25 + -0.1474 * x26 + 0.0253 * x27 + -0.0474 * x28 + -0.0364 * x29 + 1.4964$

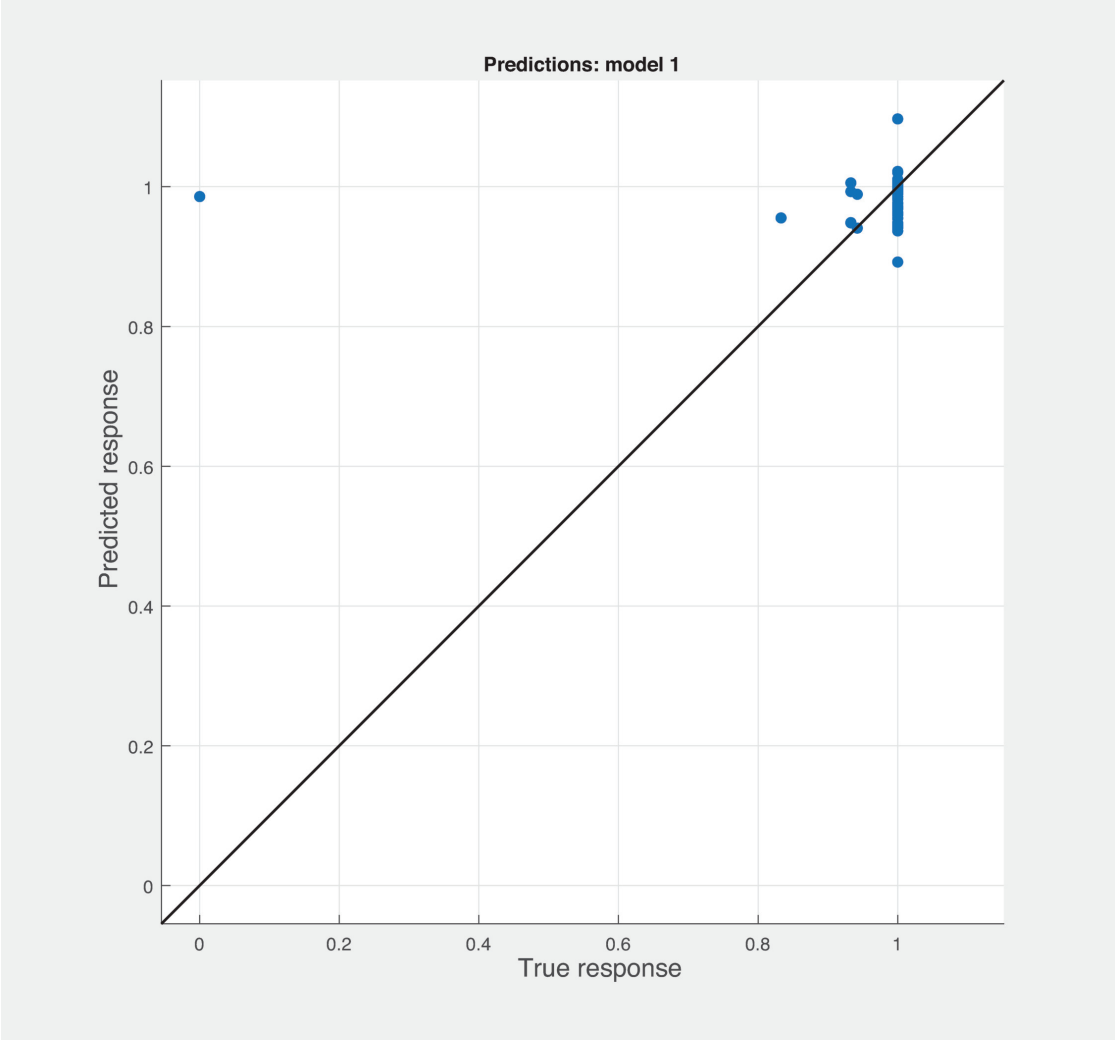


Figure 3.14: Reusability prediction results with BSM metrics.
R-Squared = 0.045
Regression line equation:
$y(x0, x1, x2, x3, x4, x5) = -0.0093 * x0 + 0.0011 * x1 + -0.0751 * x2 + 0.0473 * x3 + -0.0790 * x4 + -0.0379 * x5 + 1.0383$

S.No	Metric set name	RMSE	MAE
1	BSM	0.284	0.172
2	CKM	0.017	0.011
3	SM	0.0085	0.0057
4	BSM-CKM	0.1497	0.022
5	BSM-SM	0.0872	0.0581
6	SM-CKM	0.12114	0.085
7	AM	0.1556	0.0743

Table 3.4: RMSE & MAE comparison for different sets of metrics for modularity

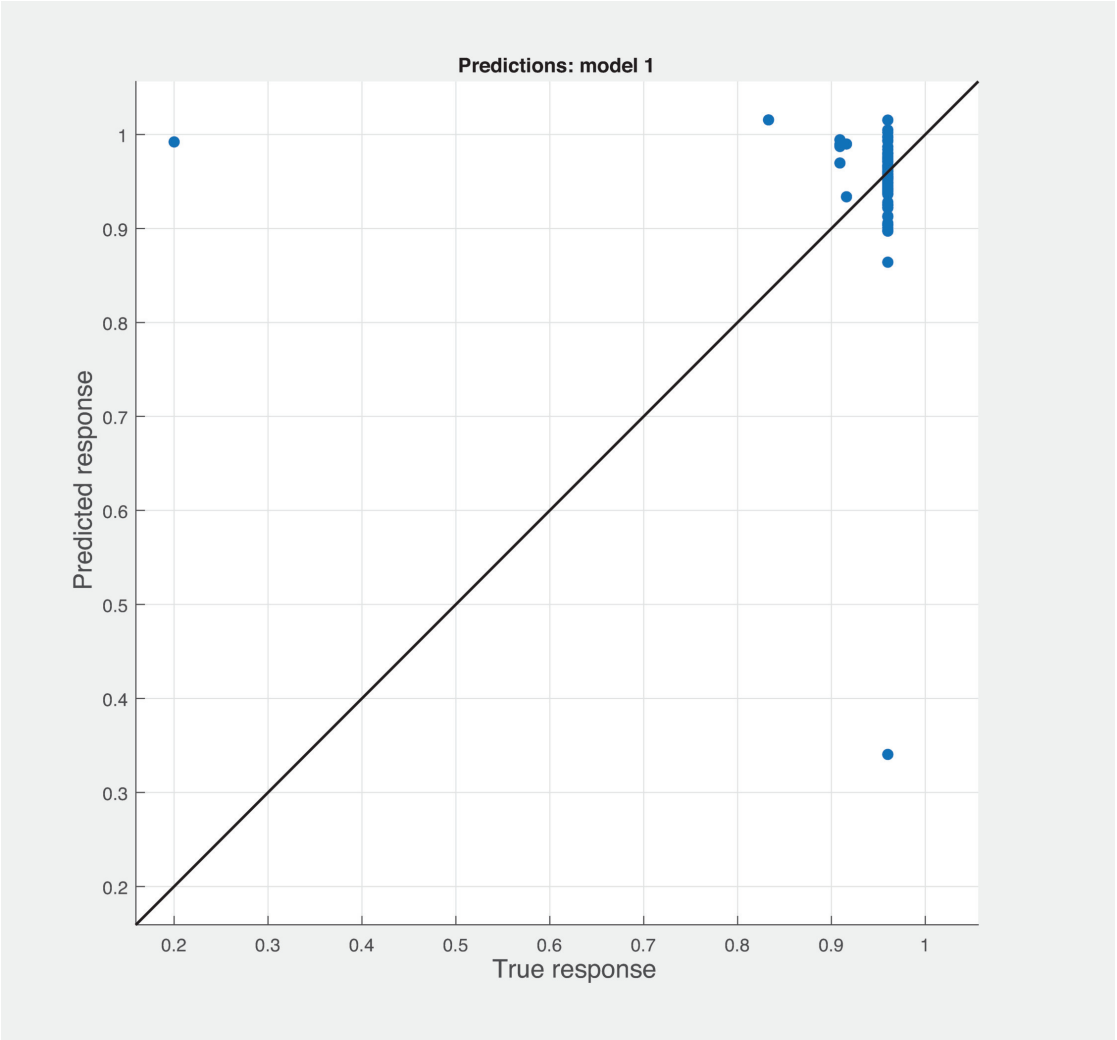


Figure 3.15: Reusability prediction results vs actual for Sneed's metrics
R-Squared = 0.2265
Regression line equation:
$y(x0, x1, x2, x3, x4, x5, x6, x7, x8, x9) = -2.3085 * x0 + -2.6429 * x1 + -2.7878 * x2 + -2.6285 * x3 + -2.4115 * x4 + -4.1766 * x5 + -2.9506 * x6 + -2.8893 * x7 + 21.6811 * x8 + -0.0100 * x9 + 0.9880$

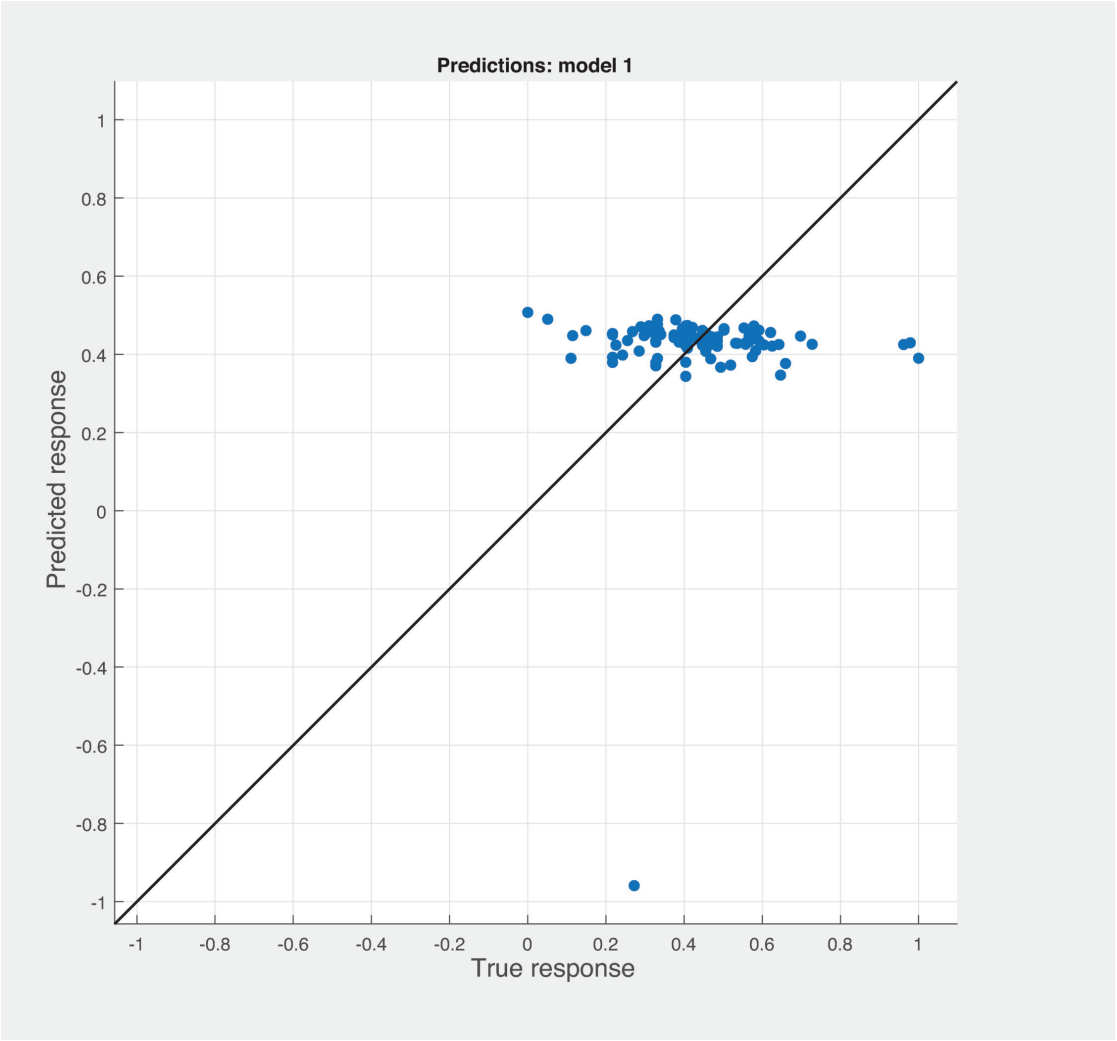


Figure 3.16: Maintainability prediction results with BSM metrics
R-Squared = 0.0359
Regression line equation:
$y(x0, x1, x2, x3, x4, x5) = 0.0664 * x0 + -0.1900 * x1 + 0.0152 * x2 + -0.1298 * x3 + 0.0340 * x4 + 0.1691 * x5 + 0.3629$

ciency of the prediction quality is reduced as indicated by the increasing MAE and RMSE values. BSM metrics have lower potential to predict the quality values individually. The results of BSM combined with SM are better than CKM. As a conclusion, this study demonstrates that the CKM and SM metrics that are calculated at a micro level have higher potential to predict the quality of a Web service.

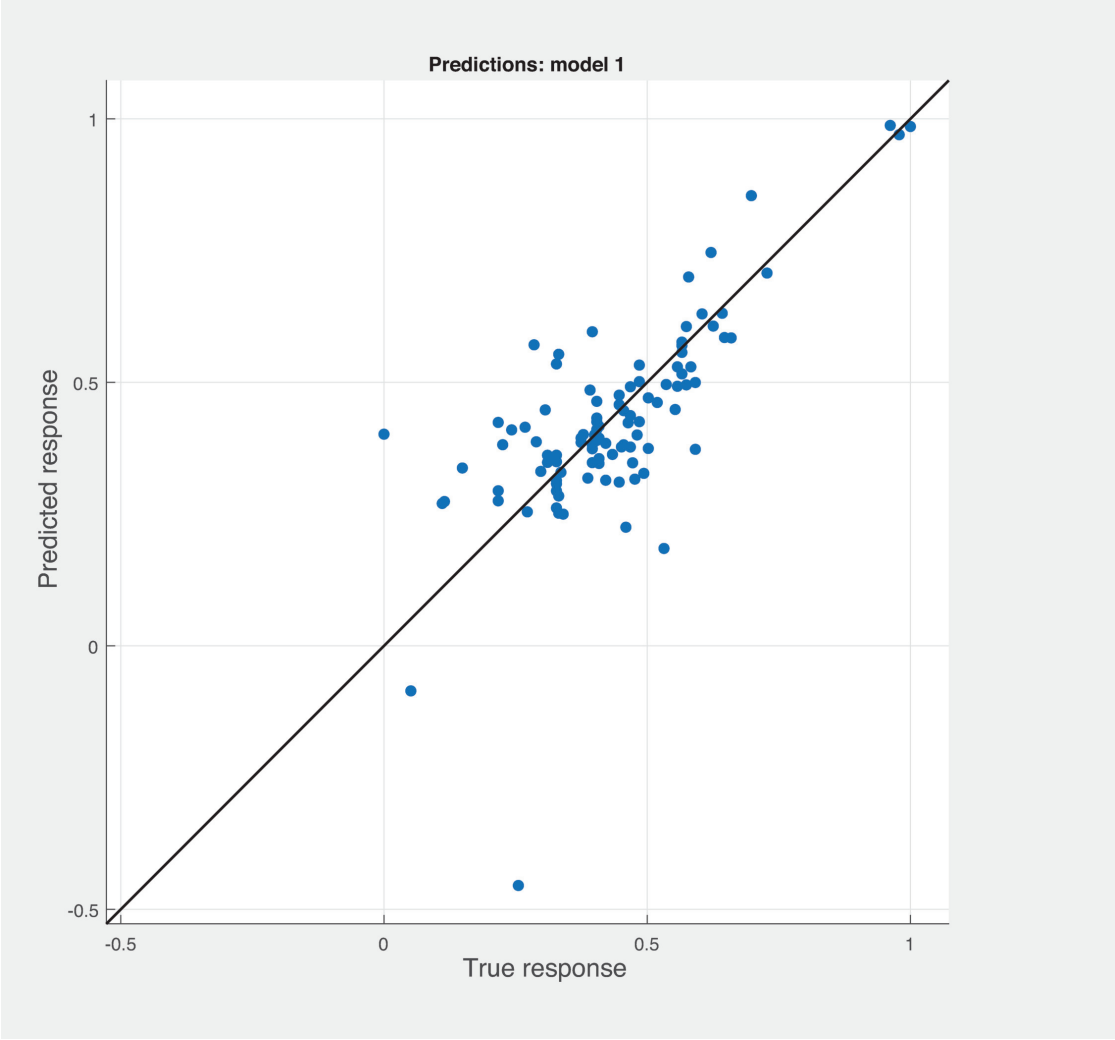


Figure 3.17: Maintainability predictioin results vs actual for all metrics.
R-Squared = 0.8431
Regression line equation:
$y(x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, x17, x18, x19, x20, x21, x22, x23, x24, x25, x26, x27, x28, x29) = -0.3151 * x0 + -0.7992 * x1 + -1.0216 * x2 + -0.8342 * x3 + -0.4060 * x4 + -1.7812 * x5 + -1.7800 * x6 + -0.9928 * x7 + 5.3500 * x8 + -2.0524 * x9 + -0.7867 * x10 + -0.2536 * x11 + -0.9415 * x12 + -0.2692 * x13 + -0.1068 * x14 + 0.3007 * x15 + 1.9078 * x16 + 0.1880 * x17 + 1.6854 * x18 + 0.7206 * x19 + -0.1059 * x20 + 0.0795 * x21 + -0.0664 * x22 + -0.5798 * x23 + 0.1732 * x24 + -0.1855 * x25 + -0.0317 * x26 + -0.0475 * x27 + 0.0675 * x28 + 0.0425 * x29 + 1.3905$

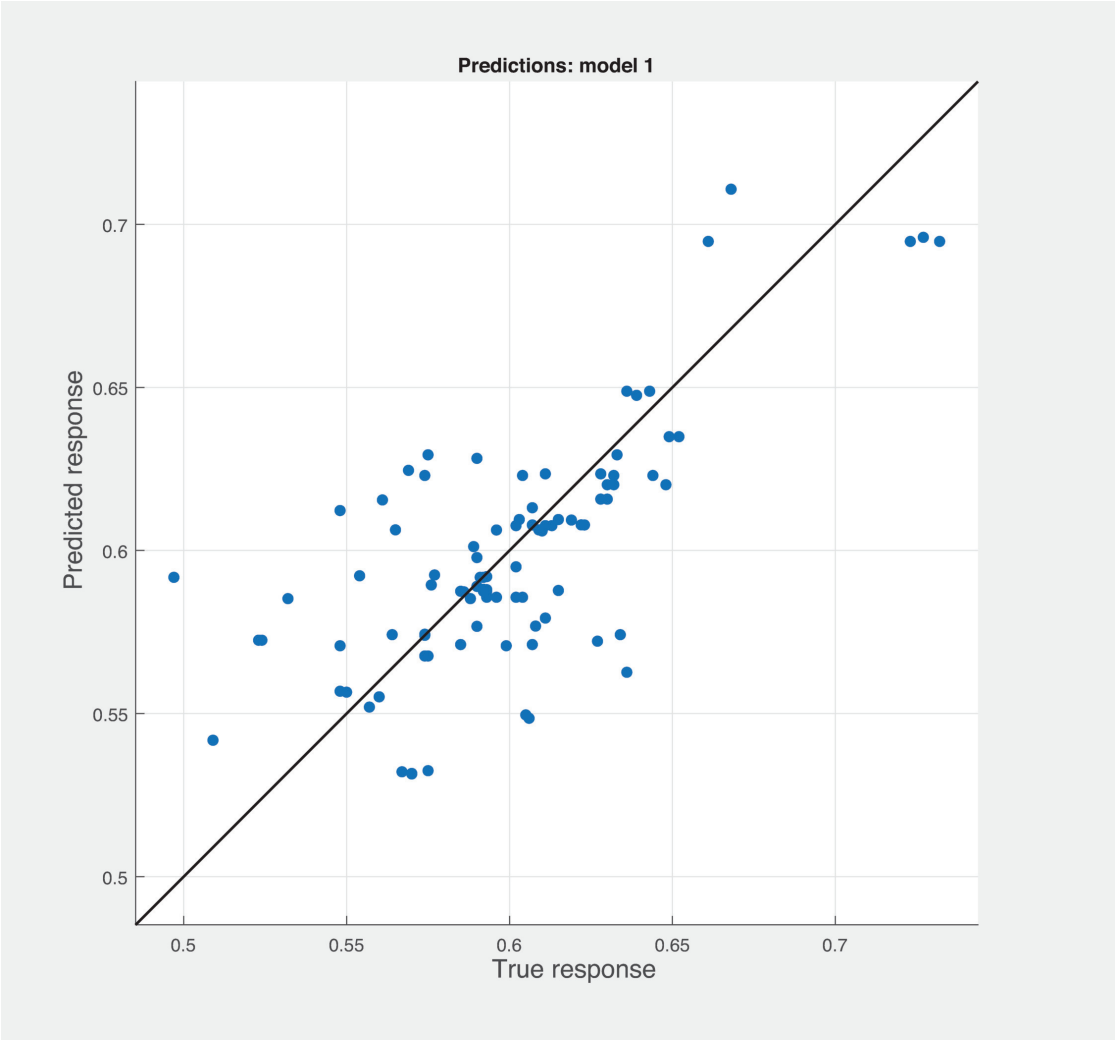


Figure 3.18: Maintainability prediction results with Sneed's metrics.
R-Squared = 0.8431
Regression line equation:
$y(x0, x1, x2, x3, x4, x5, x6, x7, x8, x9) = -2.4851 * x0 + 0.8176 * x1 + 0.7047 * x2 + 0.3849 * x3 + -0.1102 * x4 + 0.3296 * x5 + 1.2122 * x6 + 0.3881 * x7 + -5.4533 * x8 + -0.0820 * x9 + 2.4228$

3.2.5 Answers to the research questions

Answer to RQ1.1: Will the proposed methodology improve the predictability of source code metrics?

The experiments were conducted using seven different combinations of source code metrics, including CKM, BSM, SM, CKM-BSM, CKM-SM, BSM-SM and all metrics. The CKM metrics were not aggregated with the Theil index, and the other two had the Theil index applied. The results clearly illustrate that Sneed's, Baski & Misra metrics suites certainly have more potential than CKM metrics to predict the web services' QoS properties.

Answer to RQ1.2: Can we predict the quality of service properties of Web services using source code metrics?

The experimental results demonstrate that the source code metrics certainly pose the potential to predict a web service's quality of service properties. Other than reusability, the three source code metric suites produce better results for the multiple linear regression. The results show that the Sneed metrics have a higher potential to predict the quality metrics for Web services compared to the two other metrics. The RMSE and MAE values are 0.0085 and 0.0057, which shows that the prediction results are outstanding.

3.3 Chapter summary

Due to the popularity of Web-based applications, various developers have provided an abundance of Web services with similar functionality. Such similarity makes it challenging for users to discover, select, and recommend appropriate Web services for service-oriented systems. QoS has become a vital criterion for service discovery, selection, and recommendation. Unfortunately, service registries cannot ensure the validity of the available quality values of the Web services provided online. Consequently, predicting Web services' QoS values has become a vital way to find the most appropriate services. In this research, we proposed a novel methodology for predicting Web service QoS using source code metrics. The core component is aggregating software metrics using inequality distribution at the micro level of an individual class to the macro level of the entire Web service. The correlation between QoS and software metrics is used to train the Machine learning. Our approach is validated and evaluated using three sets of software quality metrics. Our results showed that the proposed methodology can improve the efficiency of the prediction of QoS properties using its source code metrics.

CHAPTER 4

KEYWORD-BASED WEB SERVICE SEARCH AND RANKING

4.1 Keyword search

The most critical and valuable data, such as business data, is stored in the relational databases. A relational database management system (RDBMS) saves data in tables and the relationships among the data. The data can be reassembled and accessed in many different ways without changing the table form. Most commercial relational database management systems use SQL to access the database. With an increasing trend of using relational databases to store data, it has become crucial for users to be able to search and browse the information stored in them. Keyword research is the process of finding and analysing search terms that people enter into search engines to use that data for a specific purpose, often for search engine optimisation (SEO) or general marketing. Keyword research can uncover queries to target, the popularity of theses queries, their ranking difficulty, and more. A keyword search on relational databases enables ordinary users. They do not understand database schema or SQL, to find the connected tuple sets among the tuples stored in the relations, with a given set of keywords. The existing methods of keyword searches on relational databases can be classified into two categories, namely, schema-based methods and graph-based methods.

A schema-based keyword search on relational databases is a standard method to generate a candidate network in schema graphs transformed from relations. The relational databases stores data in the form of columns, tables and primary key to foreign key relationships. Two schema graphs provided as

examples to understand the context. Figure 4.1 shows the schema graph of a publication database from the DBLP dataset. It consists of six relation schemas, namely Person, InProceeding, RelationPersonInProceeding, Proceeding, Publisher and Series. Each relation has a primary key except the RelationPersonInProceeding relation. The InProceeding relation has one foreign key that refers to the primary key defined in the Proceeding relation. The Proceeding relation has two foreign keys that refer to the primary key defined on both the Publisher and Series relations. The movies database schema graph of the IMDB dataset is shown in Figure 4.1. It consists of six relation schemas: Movies, Directors, Movies-Directors, Movies-Genres, Actors and Roles. Each relation has a primary key except the Movies-Directors relation. The Roles relation has one foreign key that refers to the primary key defined in the Actors relation.

4.1.1 Schema-based keyword search

The proposed approach uses a schema-based keyword search for Web service discovery using a relational database. It is a novel work to the best of our knowledge that uses a schema-based keyword search over the relational database for Web service discovery. Fitting the graph into the main memory is one of the significant challenges of the graph-based approach. Though many indexing strategies are proposed in the literature, transforming a relational database with millions of records into a graph consumes memory. Furthermore, identifying appropriate sub-graphs as query answers remains an open problem [126]. This research uses the Inverted index from the database schema in our proposed approach to quickly locate the candidate data and graph nodes summary.

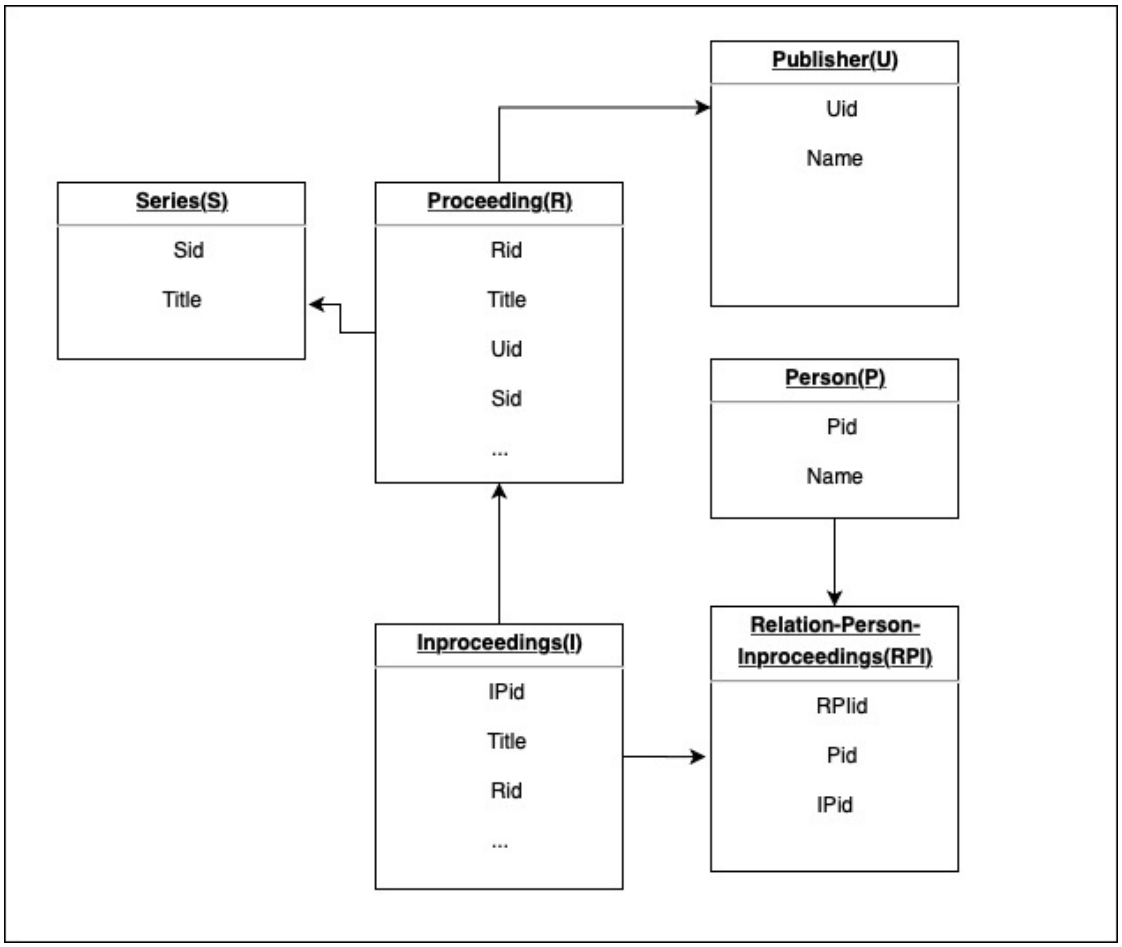


Figure 4.1: Schema graph for publication database

Web service composite quality can be measured using the appropriate aggregation functions on the quality parameters of individual Web services [12]. Where there are several execution paths from the entry service to the system's exit service, the one with highest processing time determines the system's response time. The experiments thoroughly demonstrate the efficiency and performance of our keyword-based search and selection approach. The computation time shows the efficiency, and the success rate illustrates the effectiveness of our proposed method. The success rate is the percentage of an instance where a group of keywords obtain a valid query result [127, 128]. The experimental setup calculates the computational time and success rate with different key-

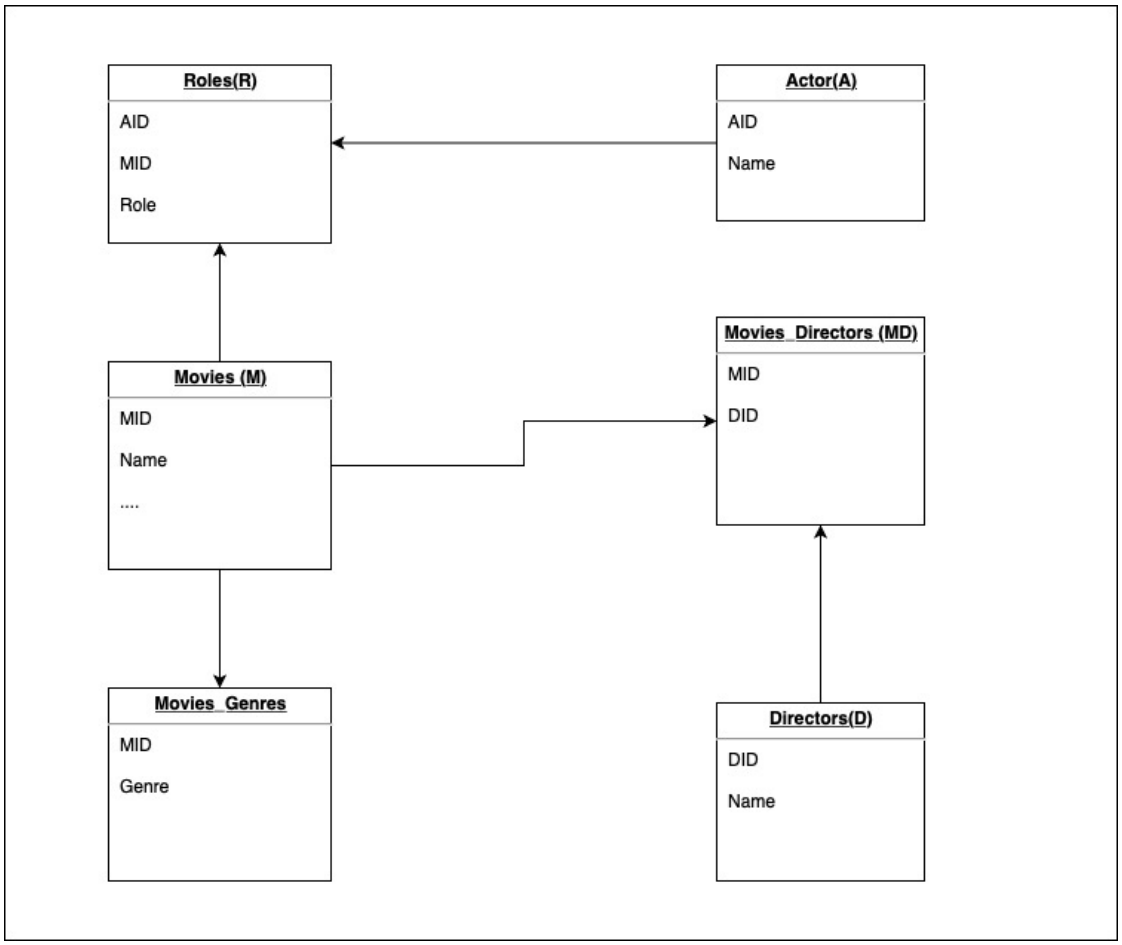


Figure 4.2: Schema graph for Movies database

word lengths, different tuple sets, several composite tasks and, several quality requirements. The key contributions of this research are:

- to develope a schema-based keyword search which offers a capable architecture through the integration and automation of system planning, service discovery and selection.
- to store Web services in the library, the existing relational database model is adopted, in which a tuple represents the Web service information and composability information stored as a key reference between tuples.
- to test the success rate and response time of the schema-based keyword

search model, comprehensive experiments were performed using the WS-Dream data set.

4.2 System Architecture

Relational databases can store service information as tuples and service composability information as key references between tuples. For instance, a foreign key reference placed if a service's output used as a service's input. A relational database which stores Web services and their relations generate a set of connected tuples (Minimal Total Joint NeTwork (MTJNT) or candidate network) for a specific number of keywords. A high ranking MTJNT which covers all keywords is the result. If more results needed, then more candidate networks are found to identify the top-k, where the user can define k. Figure 4.3 illustrates our proposed schema-based keyword search architecture for Web service discovery and selection. The proposed architecture consists of the following modules to achieve the keyword search.

4.2.1 Data Preprocessing

The WS-Dream dataset contains the URLs for 5825 WSDL (Web Service Description Link) files used for the experiments. A Java-based framework built to extract the WSDL files from the URLs in the current dataset, only 457 URLs have WSDL files for Web services. Sneed's tool used to calculate nine quality metrics for the available WSDL files. Keywords related to each Web service functionality are added to improve the search results. As a final step, the system used the

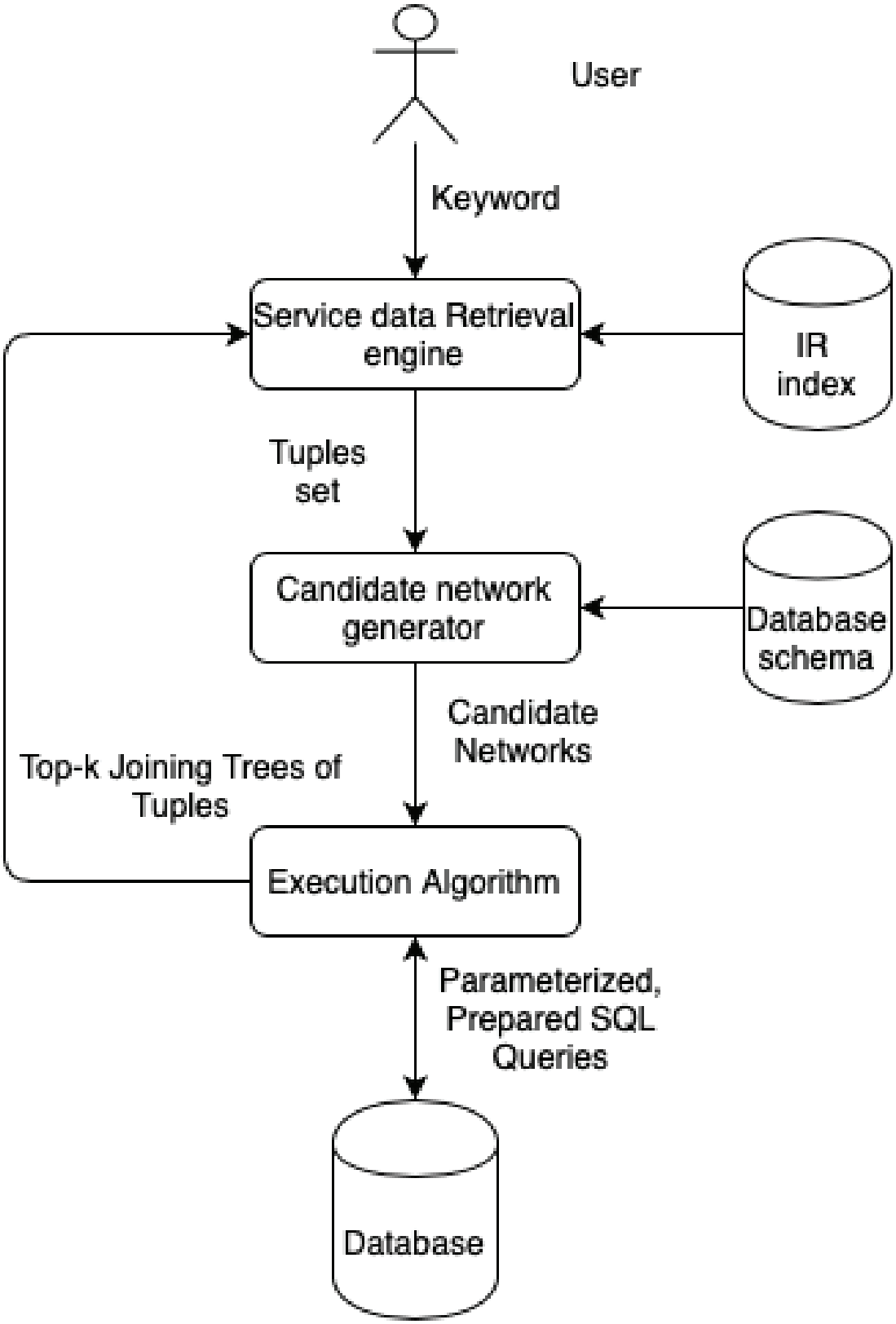


Figure 4.3: Architecture for Schema-based keyword search

SQL SELECT RANDOM in-built method to generate key references among the available tuples randomly.

4.2.2 Service data retrieval engine

For the service data retrieval (SDR) engine, the primary objective is to search for user queries and different related tuples for the keywords that are available. The Oracle Text tool was employed to test data and to create an index that records each occurrence of a keyword as an inverted index. Sophisticated RDBMSs provide an IR-style indexing functionality for the text attribute. The Service Data Retrieval Engine feature of our architecture uses IR-style indexing to classify all database tuples with a non-zero score for a given request. The SDR engine depends on the IR Index, an inverted index which links each keyword in the database with a set of keyword occurrences; every keyword instance is registered as a pair of tuple attributes. Oracle Text is used to establish a separate index of each relation attribute in our implementation. Such indexes are combined to create the IR index. When a query arrives, the IR Engine uses the IR Index to extract from each relation R the tuple set $R^Q = \{t \epsilon R | Score(t, Q) > 0\}$, which consists of the tuples of R with a non-zero score for Q. On the arrival of a Q query, the IR Engine uses the IR Index to retrieve the tuple set with non-zero score $R^Q = \{t \epsilon R | Score(t, Q) > 0\}$. As specified by the top-k query processing algorithm below, tuple t in the tuple sets are ranked in the order of $Score(t, Q)$.

4.2.3 Candidate network generator

The Candidate Network (CN) Generator is the next component in the system. CN receives the non-empty tuple sets from the SDR Engine, database scheme and a parameter M. This module plays a vital role in generating CNs, expressions that are essential to building joining tuple trees that are considered as potential answers.

a Java-based system developed to identify the CN from the pool of interconnected tuples provided by the IR engine. The Minimal Total Joining Networks (MTJN) concept needs to be followed to identify the CNs. The number of tuples is the size of the CN. An interconnected tuple should meet the following criteria to qualify as an MTJN:

1. The number of non-free tuple sets in the result set doesn't exceed the number of query keywords m: This constraint ensures the development of adequate CNs without compromising a result with all the keywords that are essential for Boolean-AND semantics. That is, for every result T that contains every keyword exactly once, a CN C exists such that $T \in C$.
2. The result set is not free of leaf tuple sets. This constraint guarantees CN "minimality."
3. The result set does not have an associative structure between tuples like $R \rightarrow S \leftarrow R$. If there were such a structure, each resulting tuple joining tree would contain multiple times the same tuple.

4.2.4 Execution Algorithm

The final module in this pipeline is the execution engine which receives many CNs with non-empty tuple sets. The Execution Engine interacts with the RDBMS query execution engine repeatedly to classify the top-k database results. The most challenging feature to incorporate is the Execution Engine module. Vagelis et al. [129] proposed an efficient hybrid algorithm to rank the CNs with the required number of results. The hybrid algorithm emerged from the Sparse algorithm and Global pipelined algorithms.

Sparse Algorithm

The Sparse algorithm aims to enhance the performance of query-processing by removing any CN which may not produce a top-k match for the given query. The steps followed by the Sparse algorithm are as follows:

1. Calculate the Maximum Possible Score (MPS) for tuple tree obtained from the CN.
2. Compare MPS with the already computed actual score of k for other tuple trees.
3. The corresponding CN can be ignored safely for further consideration if the already computed k value exceeds the MPS value.

Global pipelined Algorithm

The global pipelined algorithm performs well for queries with a larger number of results. The algorithm receives a set of candidate networks along with its non-

empty tuple set as input and produces a group of joining tress ranked based on their overall score for the query. The following steps explain the process of the single pipelined algorithm.

1. Calculate the Maximum Possible Future Score (MPFS) for the tuple tree obtained from the CN.
2. CN with a maximum MPFS value is evaluated and its priority is updated.
3. After that, the CN with the next highest MPFS value is selected and analysed.
4. A CN can be added to the output if its MPFS value is no lower than the Global MPFS value.

The most potent algorithm for queries with fewer results is Sparse. With a reasonably large number of results, the Global Pipelined performs best on queries. In contrast, the Hybrid algorithm estimates the number of results expected for a query and selects the best algorithm for query processing.

The Hybrid algorithm, as mentioned in Algorithm 1 is significantly dependent on the result-size estimator precision. The RDBMS result-size estimates for queries with OR semantics were found to be reliable. In contrast, this estimation is more challenging for queries with AND semantics. The RDBMS that we used for our implementation, Oracle 9i, ignores the text index when producing estimates. However, this estimation is even more challenging for queries with AND semantics. We obtain an estimate S of the number of tuples extracted from a CN from the RDBMS, but we need to refine this estimation so that we find only tuple trees that include all the keywords for queries. Consider a two-keyword question to demonstrate the simple adjustment $[w_1, w_2]$ with two non-

empty tuple sets TS_1 and TS_2. When we assume that the two keywords occur independently within tuples, we modify the estimation S by multiplying by $\frac{|TS_1^{w1}|\cdot|TS_2^{w2}|+|TS_1^{w2}|\cdot|TS_2^{w1}|}{|TS_1|\cdot|TS_2|}$, where TS_i^w is a subset of TS_i with keyword w. When calculating this adjustment variable, an implicit simplifying assumption is that a tuple does not have two keywords.

Algorithm 1 Hybrid algorithm

```
Hybrid Algorithm (CN,k,c,Query,Score())
c is a constant for tuning
E = GetEstimate(CN)
if (E > c . k ) then
    employ Global pipelined
else
    employ Sparse
end if
```

A Java system developed based on this hybrid algorithm to rank the CNs. The result size needs to be provided by the end-user. The hybrid algorithm uses the result size number to restrict the algorithm with the number of results. Once the top-k CNs obtained, the system extract the top sets of tuples and establish a service-oriented system. The experiments designed to comprehensively demonstrate the efficiency and effectiveness of our keyword-based Web service search and selection methodology. The Calculation of computation time shows the efficiency, and the success rate illustrates the effectiveness of our proposed approach. Various keyword lengths with different sets of a number of tuples used to calculate the computation time. The success rate is the percentage of an instance when a group of keywords obtain a valid query result.

4.2.5 Evaluation metrics

We planned to use two sets of Web service datasets, namely the QWS dataset and the WS-Dream dataset. The QWS dataset contains around 200 active Web service with 19 quality metrics, and the WS-Dream dataset contains about 700 active Web Service and two quality metrics. We calculate the computation time taken to respond to the query with variable lengths of keywords. We compare the computation time with different lengths of the keyword on both the Web service datasets.

4.3 Experiments & analysis of results

4.3.1 Research questions

The experiments conducted to answer the following research questions

RQ 2.1: How effectively can the schema-based keyword search improve Web service composition quality?

Graph-based systems generate multiple tuple sets for each database relationship to find all the answers to a query using this AND semantics. For each combination of the keywords in Q and each relationship, a separate tuple set is created. This process usually results in an exponential amount of CNs in the query size, which makes query execution expensive for queries with more keywords or an M value greater than 4. In comparison, for each *R* relationship,

we only create a single tuple set R^{Q},as specified above. A post processing step checks that we only returned tuple trees which comprise all query keywords for queries with AND semantics. This feature of our system results in considerably faster executions, which helps us to manage larger queries and also increases the consistency of the composition.

RQ 2.2: Can the candidate networks be optimzed to improve the search query?

RDBMS employed to process SQL queries as we planned to query relational database data. The keyword query given by a user retrieves interlinked tuples containing the keywords in the query. A tuple comprises a keyword if the keyword is available in the tuple text attribute. The system searches for subsets of relationships that include the keywords from the queries with the keywords in hand. These subsets retrieved from the database are called tuple sets. The tuples containing the required keywords are combined to generate a relational algebra expression, namely the CN. In other words, each CN describes potential answers for the entered keyword query. In addition to the tuple sets generated in the previous step, creation CN needs information on referential integrity constraints taken from the database schema. As there are many different ways to join relationships that store the keyword containing tuples, many different CNs may potentially be generated.

In our proposed method, if it satisfies the following properties, we find that tuple set is a CN:

- the number of non-free tuple sets is not more than the number of query

keywords

- there are no free leaf tuple sets
- the multiple set has no associative form construct $R \rightarrow S \leftarrow R$

4.3.2 Variables and objects

Most of the existing Web service ranking systems use the unreliable quality parameters available at Web service repositories. Due to a lack of maintenance, network connection and geographical disparity, repositories are not able to provide a valid quality parameter. However, Sneed's tool processes the source code of the Web service to generate quality parameters for Web services.Sneed proposed the following QoS properties as potential criteria to choose a more suitable web service. This research uses them as an additional search criterion to query web services:

$Modularity = [(No.Moduleinvocationsx2) + (No.ModulesxDesiremodule - Size)]/No.Statements$

$Portability = [No.Statements - (No.DatabaseAccessesx8) - (No. - TP - Operationsx6) - (No.FileAccessesx4) - (No.-Callsx2) - (No.Non-standardStatements)]/No.Statements$

$Testability = [l - (No.Edges/No. - Statements)]x[l - (Nopredicates/No.Data)]$

$Conformity = 1 - (No.Nonconforming\_Lines/No.Lines)$

$Readability = (CommentLinesxCommentWeight)/No.Lines$

Evaluation Variables

The experimental setup employed response time and success rate, two key parameters to validate our approach. The success rate is the percentage of circumstances when an answer to the keyword query for the Web service found. Finding a composite Web service solution that fulfils all quality constraints despite its various optimisation objectives determines the success rate. We use success rate to illustrate the efficiency of the proposed solution.

When there are several execution paths from the entry service to the system exit service, the system's response time is determined by the one with the longest execution time. Different scenarios such as keyword distance, number of search keywords, number of quality constraints and number of tasks for a composite are used to define response time.

Objects

The WS Dream dataset, an open-source dataset created by a group of researchers from the Chinese University of Hong Kong, was used in this research. The WS-Dream dataset has two dataset versions, and our experiments use version 1. The data consists of 5825 URLs and the response time and throughput data from 339 geographically distributed users. To improve the search results, we added keywords for each Web service related to its functionality. As a final step, we use the SQL SELECT RANDOM in-built method to generate key references among the available tuples randomly.

Empirical environment

The software development utilized different services using the Python language. The high levels of the Python language are created in the form of data structure, dynamic typing and binding. It is well suited as a scripting language and to connect different components. It is a simpler in terms of learning and syntax. It is a cost-effective language for programme maintenance. Python has in built support for modules and packages which provide the function of code reuse and modularity. It is freely distributed language. Sample Web services has been built in different python files for different components. The files are connected using a Rest API. For our purpose the code is accessed using a tool called POSTMAN for GET and POST methods. The web services are deployed onto a server (WSGI in our case), which listen to requests on a specific port. When a request is made the request is processed then a connection is made to the MySQL Database and then a SQL query is run using this DB connection which will give us a result set based on the query performed this result is then converted into a python relevant dictionary which will be the endpoints output.

The dataset was pulled into the MySQL database. The services were combined using the POSTMAN tool, which is a webservice testing tool. AngularJS and NodeJS are used for the development of the User Interface. In addition, the Web Description Language is used to define the features of a Web application using an XML-based interface description language.The user interface is developed using node.js and angular.js functions which helps in the good presentation of the developed work.

This research leveraged the functionality of the postman tool to test the APIs. A request made to each of these APIs individually to showcase the functionality

and then a request made to the POST endpoint which consumes the other three endpoints and provides a much more consistent output. The system extracts the results using POSTMAN, a web service testing tool. The architecture utilizes a combination of Python, Bottle (REST services library), Requests (simple web-service requester service), MySQL (database), and Postman (the web service testing tool). The database facilitators such as MySQL and Mongo, which uses an HTTP API used to provide access to the database files. A web service was developed to get the user inputs from the user interface. The organization of a web service composition that includes multiple web services is assumed. Retrieving data sets which are going to be an input for another service are taken. Node.js & Angular.js are used for creating the user interface platform in which the user can raise queries and can retrieve multiple results from which the higher prioritized ones are displayed first. The user gives queries in the form of common human language like "Find me a hotel in the city ***** with **requirements** and also find taxis around me", and then the system displays all the related list of results (List of hotels and taxi services). The result is displayed in the form of card layout by which the user can find website of the specific field and can access the taxi service directly without opening a third party application which helps all the users in a great manner as it makes the job easier of finding various services at one place rather than accessing them separately.

4.3.3 Analysis of results

The WS-Dream dataset used to create three groups of datasets containing Web services with three different functional capabilities. The three functionalities are Hotel, Flight and Taxi. SQL Random function used to populate the foreign

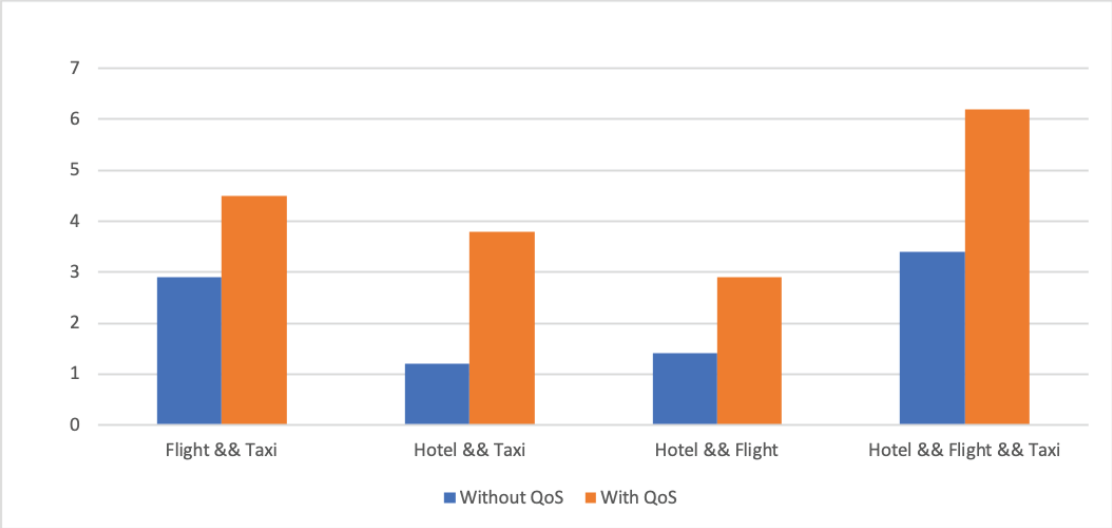


Figure 4.4: Response time(in seconds) for various keywords with and without QoS

key references between the three datasets to utilise them for the experiments. A Java applicatioin developed for retrieving the Minimum Total Joint Network Trees (MTJNT) for the given keywords of the user requirements. The system calculated the response time to get the MTJNT for each combination of keywords without QoS parameters and with QoS parameters. Figure 4.4shows the graph for both the experiments to calculate the response time with and without QoS. QoS parameters significantly affect the response time for retrieving MTJNT.

The results presented in Figure 4.4 illustrates that the response time of various combinations of queries with QoS requirement is comparatively higher than the queries without QoS requirement. However, the efficiency of web service composition depends on the aggregated quality of all the candidate services. With QoS requirements, the proposed methodology retrieved the best suitable web services for the query. Figure 4.5 can indicate that the success rate significantly increased while including QoS requirement in the search query. Therefore, the disadvantage caused by the QoS requirement in terms of response time

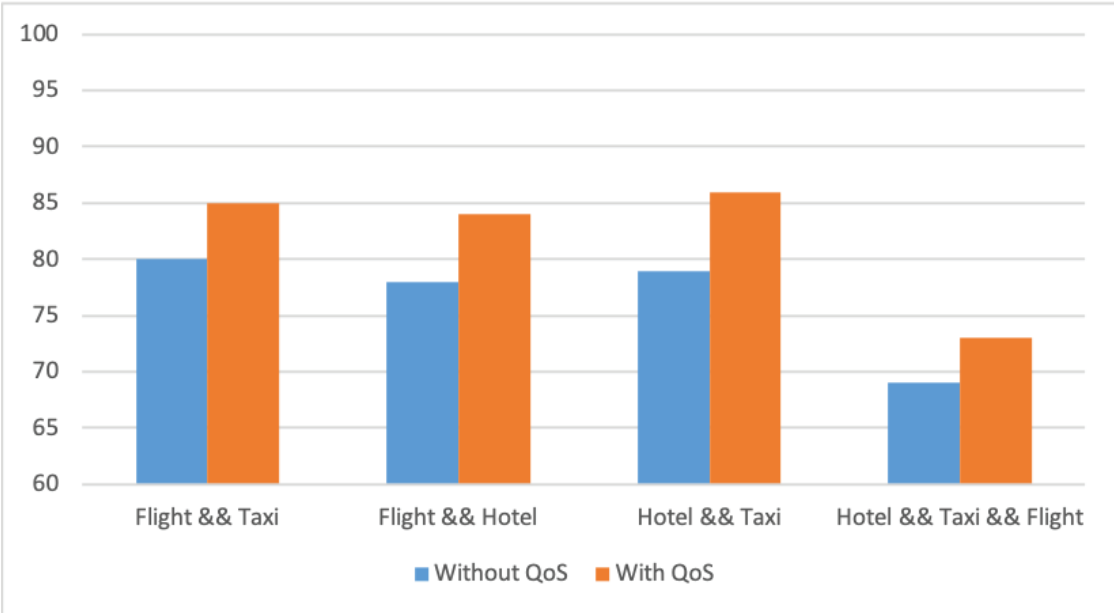


Figure 4.5: Success rate(%) for various keywords with 1000 services in repository

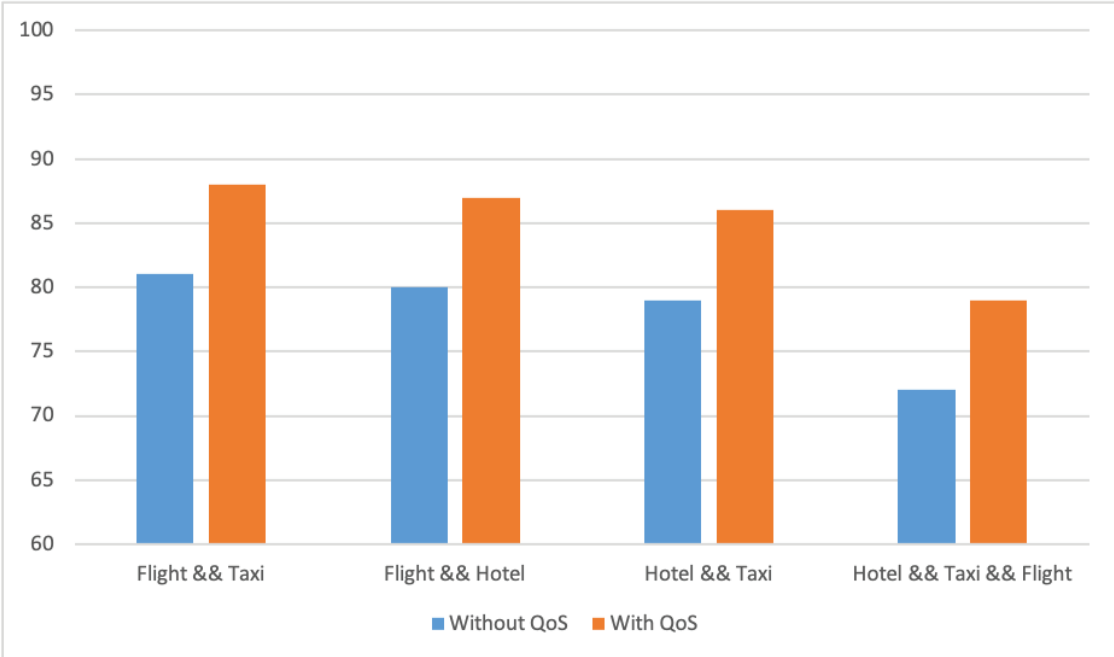


Figure 4.6: Success rate(%) for various keywords with 2000 services in repository

can be justified.

The success rate also calculated for the keyword results using the Web service testing tool called POSTMAN. Web services repositories are grouped with different numbers of services to calculate the success rate. Figure 4.5, 4.6, 4.7 and, 4.8 illustrate the results of the success rate with and without QoS property requirements. Figure 4.5 presents the results of the success rate for the experiment with 1000 services in the web service repository. When using only one keyword as a criterion, the success rate is always 100% as the repository only contains services with functionalities such as Flight, Hotel and Taxi. Therefore, we ignore the details of the success rate when using only one keyword in the graphs. With a various number of services in the repository, success rate improves while the number of services increases. However, queries without QoS demonstrate less potential than queries with QoS. This observation indicates that the success rate is maximising while considering the QoS as a requirement for Web service discovery.

Figures 4.9, 4.10, 4.11 and 4.12 illustrate the success rate for each combinations of queries with a different number of services in the repository. Based on Figure4.9, 4.10 ,4.10 and Figures 4.12, Flight and Hotel , Success rate is gradually increasing while the number of services increases. However, QoS plays a vital role in improving the success rate. Therefore, it is evident that the repository with more number of services yields better results while querying with QoS as a criterin.

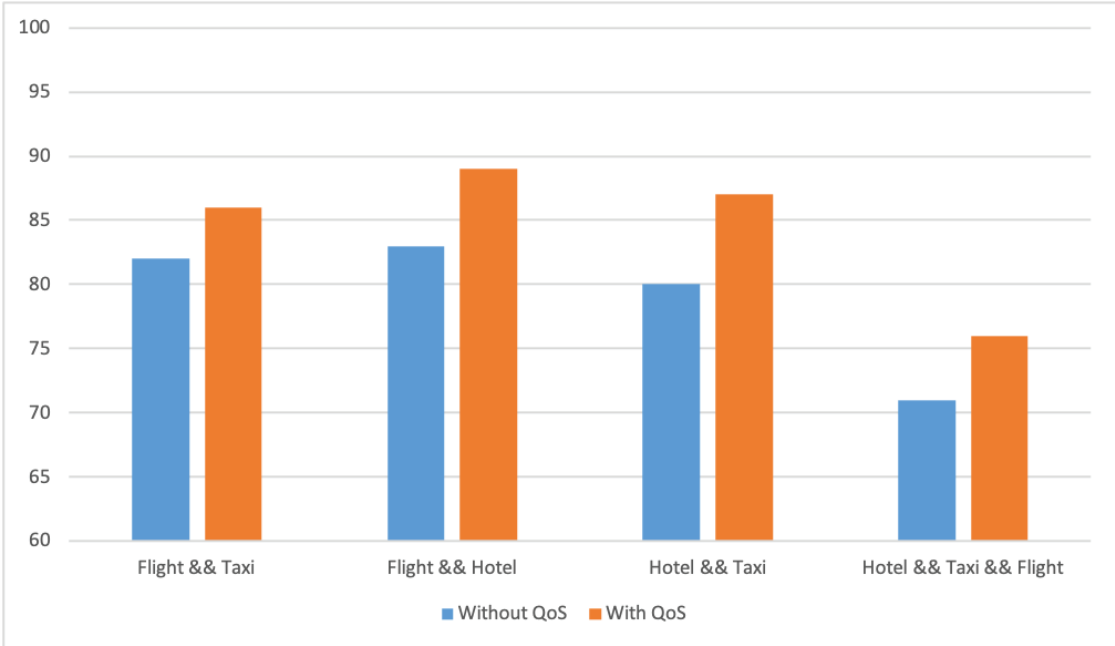


Figure 4.7: Success rate(%) for various keywords with 3000 services in repository

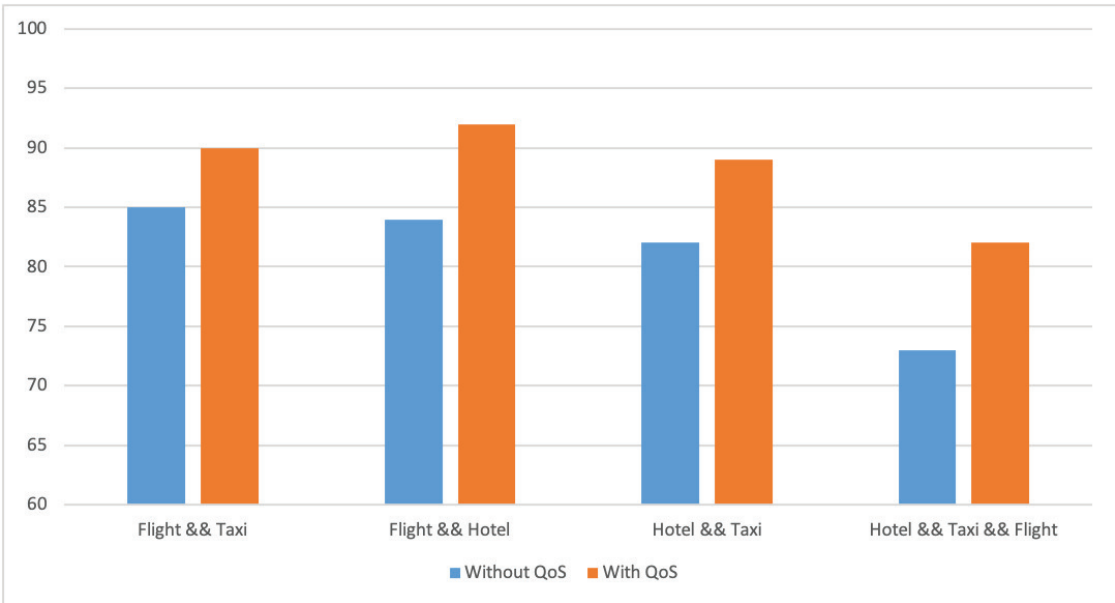


Figure 4.8: Success rate(%) for various keywords with 4000 services in repository

Figure 4.9: Success rate(%) for Flight AND Hotel keyword for different no. of services

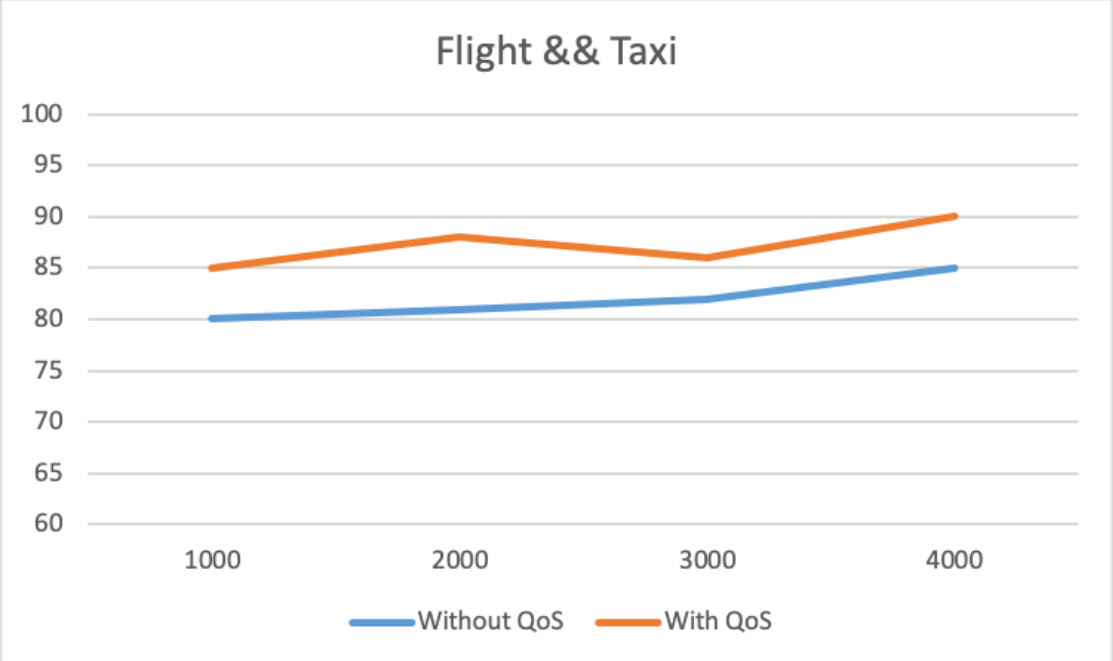


Figure 4.10: Success rate(%) for Flight AND Taxi keyword for different no. of services

Figure 4.11: Success rate(%) for Hotel AND Taxi keyword for different no. of services

Figure 4.12: Success rate(%) for Flight AND Taxi AND Hotel keyword for different no. of services

4.4 Chapter summary

Construction of SOAs through the composition of the existing Web services was reflected in the rapid growth of SOAs applications for software engineering. The selection of appropriate component services to compose is an essential step in the Web service composite engineering process. Currently, the available approaches require systems engineers with substantial knowledge of SOA techniques. A schema-based keyword search proposed for Web service discovery and composition to address the aforementioned issue. System engineers without a comprehensive understanding of SOA techniques can use schema-based keyword search. To search for SBS component services, they simply need to enter a few keywords that define the SBS functionalities. Experiments conducted using the WS-Dream dataset and the response time and success rate are estimated to validate our proposed system. The results of the experiment demonstrate that the scheme-based keyword search and quality constraints are feasible to achieve a higher success rate.

CHAPTER 5

SCALABLE ARCHITECTURE FOR PERSONALIZED HEALTHCARE SERVICE RECOMMENDATION USING BIG DATA LAKE

5.1 Enterprise Data Warehouse

Due to emerging technologies such as the Internet of Things (IoT), all the entities are electronified. Since the growth rate of data is 48% every year, it is estimated that in the year 2020, the world will have 25,000 petabytes of data. Data comprises not only structured data but also contains semi and unstructured data. Only 20% of data is in a structured format which can efficiently be utilized by data scientists. However, the production rate of semi and unstructured data is 15 times higher than structured data [114]. Essentially, researchers need to process all kinds of structured, semi and unstructured data to find valuable insights [110]. Inevitably, an improved data management system would help data scientists provide tailored insights. The contemporary IT infrastructure offers many data handling systems such as the Enterprise Data Warehouse (EDW); however, there is a lack of scalability because the EDW data management system is for well-known queries and clearly defined policies [130, 131].

To pull the data into the EDW for further processing, it should go through the procedure of data preprocessing, namely extract, transform, load (ETL) [132]. The ETL process predominantly consumes high cost and time. To provide a custom-made medical intervention, data from diverse sources need to be processed [133, 134]. However, the EDW system is not so capable of handling the various sourced data. Another contention in healthcare analytics in adapting EDW is the inability to coexist with contemporary programming-based query

languages. An EDW designed for processing specific business rules demands significant effort to redesign for future needs. Therefore, conventional IT architecture, such as the EDW, has several limitations in handling the complex nature of healthcare data [135]. This research addresses the issues faced by the existing IT architecture and proposes a Big Data Lake architecture for efficient data analytics and demonstrates its ability for healthcare recommendation.

The experiments were designed based on the following objectives:

- to diminish the data silos across healthcare organisations.
- to plan a system to store a variety of data on a huge scale without ETL delay.
- to model a language independent data architecture system to cope with big data initiatives.
- to acquire meta-data along with data to maximise the future reusability of the data.
- to develop an analytical system to predict personalised healthcare intervention.

5.2 Data lake

The data lake is an emerging data architecture. A data lake uses a flat architecture to store data in their raw format [136]. Each data entity in the lake is associated with a unique identifier and a set of extended metadata. The consumers can use purpose-built schemas for query-relevant data, which will result in a smaller set of data that can be analysed to help answer a consumer's question.

There are doubts and concerns about the possibility of data becoming incomprehensible due to a lack of schema or similar means of interpretation, which could cause the lake to turn into a "data swamp" [137]. Therefore, a metadata repository that registers high-level information about data entities (type, time, creator, etc.) is an essential component of a robust data lake structure. A data lake's flat structure stores data regardless of its format and places the responsibility for understanding the data elsewhere.

It motivates us to choose the data lake an alternative data architecture for the SOAs. A personal data lake system proposed by Walker and Alrehamy [123] gives a basis for designing our proposed approach.

The promising properties of personal data lake architecture are as follows:

- It can provide a unified location for all the data from different social networks about a single user.
- It can improve privacy and security by storing data in a single location and the user is given the rights to design how to access their data.
- It offers the Personal Lake Serialization Format (PLSF) approach for storing meta-data.

5.3 Personalized Healthcare

By adopting smart devices, the health care industry has become a predominant stakeholder of global data. Therefore, personalised healthcare is an emerging trend in healthcare data analytics. It provides tailored treatment recommenda-

tion for an individual by analysing their medical history, environmental factors, food habits, genomic structures and insurance details [118].

Based on the recent research observations, patients with the same diagnosis may respond to the same medication in different ways. A drug can be highly effective for one patient, but the same drug might not produce the expected results when given to another patient with the same diagnosis. Personalized medication means the prescription of precise treatments and therapeutics which are well suited to an individual, taking into consideration of all the data that influence responses to therapy [138]. Due to the enormous growth of the Internet of things, the healthcare industry is equipped with smart devices and applications. Consequently, digitalization creates valuable data about the patients and medications, namely electronic health records (EHR) [139]. The large-scale availability of EHRs allows researchers to unearth the possibilities of moving healthcare organizations towards personalized healthcare [140].

5.3.1 Role of the Data Lake in Healthcare

This study proposes to adopt a data lake architecture as a replacement for the traditional data management architecture for SOAs. A health care based SOA to predict the tailored medication is demonstrated. A data lake possesses the following capabilities to address the aims of this research:

- It can store data in its native format (structured/unstructured) as it arrived without any pre-processing delay.
- A data lake can connect with trusted external sources (clinical lab, genomic

centre, insurance payers, and social media) [141]. It will also reduce the data silo across health care institution [111].

- It can support new types of data processing and improve the adaptability of the analytics system. It can able to store a massive amount of data from a diverse source with less cost.

5.3.2 Research Questions

Our experimental study is designed to answer the following two research questions:

RQ3.1: How efficiently does the proposed architecture reduce the time for data ingesting and crawling from various internal and external data stakeholders?

Compared with a traditional data warehouse, the data lake allows the storage of data as it comes without bounding with any schema. Besides, it uses the HDFS file system, which can connect to any remote application using Apache tools. The time taken for the data architecture to load and store the data is represented as the data ingestion time. The lower the data ingestion time, the better the data architecture for healthcare analytics. Therefore, if the proposed data lake architecture can reduce the data analytics processing time, it also improves healthcare recommendations.

RQ3.2: Is the data lake architecture able to avoid Data silos?

Due to the pre-defined data schema for data warehouse architecture, it is impossible to store data of different types in a centralized storage location. It leads to the creation of numerous data silos for datasets about each patient. The precision of clustering is dependent on the perimeter of the data on the patients. The more data there is from various data stakeholders, the better the results from the data analytics. Hence, if our proposed data lake architecture can handle the multiple data types in a unified location without the data swamp threat, it can improve the precision of clustering significantly.

5.4 Contribution of the research

A data lake can store data in its raw format, which removes the burden of cleansing and transforming the data according to EDW rules. Hadoop Distributed File System (HDFS), open-source software used to implement the data lake. It enables the system to store a vast amount of data at a lower cost. EDW cannot be connected with external real-time streaming data sources because it cannot manage data which is changing quickly. But the proposed data lake can be connected with external data sources through tools such as Apache Spring XD and Apache Flume. On successful completion, the results of this research will serve as a prominent data management model in the healthcare industry. Motivated by the needs as mentioned above and possibilities, this research proposes a scalable architecture for personalized healthcare recommendation.

The main contributions of this study are:

- It introduces the data lake architecture in healthcare to crawl and ingest healthcare data from vendors without any data preprocessing delay.
- It enhances the data IT infrastructure in healthcare by accepting the connection from trusted third party data stakeholders.
- It accumulates data with different formats and stores this in the unified data lake to avoid data silos across healthcare organizations.

5.5 Proposed Data Lake Architecture

The proposed data lake architecture for this research is plotted in Figure 5.1. It has four layers, namely, data ingestion layer, data governance layer, security layer, analytics layer, which will be respectively discussed in the following four sections.

5.5.1 Data ingestion layer

Typically, data crawled from multitude sources in its raw format. More often, data is available as structured; still, sometimes it may also arrive as semi-structured or unstructured. A data lake can ingest all the available data without any ETL processing. However, it also needs good metadata management because a data lake without proper metadata management will turn it into a data swamp. Metadata contains information about how, when and by whom it was collected, created, accessed, modified and how it is formatted [113]. Metadata have three categories, such as technical, operational, and business.

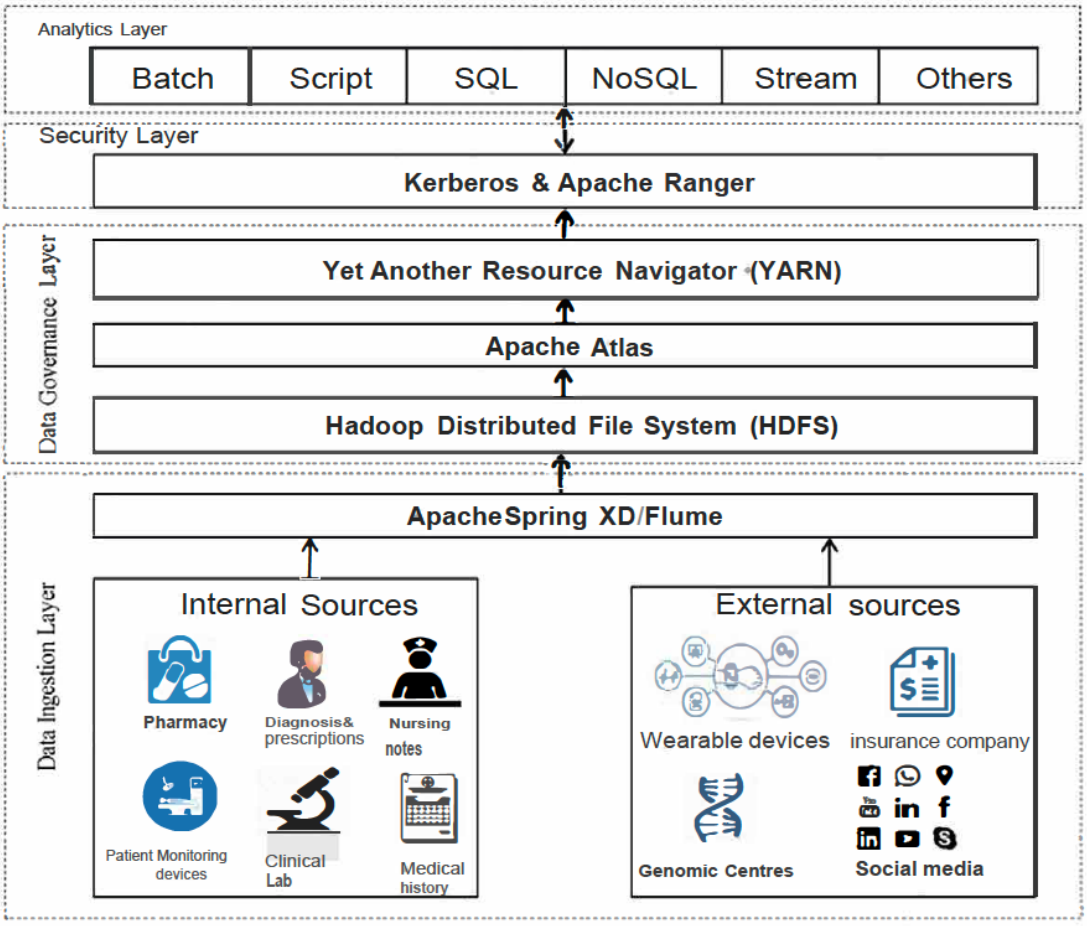


Figure 5.1: Data lake architecture

An open-source software named Apache Hadoop can implement a data lake can. It has a Hadoop Distributed File System (HDFS), which allows structured, semi-structured and unstructured types of files to be stored. HDFS can store petabytes of data and can act as a single storage location. It is fault-tolerant, scalable, and straightforward to expand [114]. There are many tools available to ingest data to the HDFS from various sources. In the healthcare industry, three basic types of data sources are available. The first type is the bulk size of data providers, such as genomic centres and biobanks. The next kind of data sources is an event-based data source, such as doctor's appointments, pharmacy purchases, and clinical notes. The last data source type is the trusted third party

streaming data [32] providers such as clinical labs, X-ray centres, social media, wearable devices.

Our proposed architecture has the following settings.

- Hadoop's Spring XD tool is used to transfer bulk data [142]. It can also create metadata information while the data ingested and loaded with the HDFS.
- Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data [143]. It has a simple and flexible architecture based on the data flows and it is robust and fault tolerant.
- To handle the stream of data and avoid data silos, the Hadoop Spring XD tool can be utilised. It can perform data ingestion along with metadata information creation from multiple input sources into HDFS with high throughput.

5.5.2 Data governance layer

The main purpose of this layer is to understand, organise, manage and provide access to all the data collected. This layer uses Apache Atlas, a tool for Hadoop, to handle the metadata framework and governance [143]. It has a set of basic governance services to meet the compatibility requirements for the HDFS. The Atlas tool has four important features, namely data classification, centralised auditing, search and lineage, and security and policy engine.

Data Classification

It imports or defines data-oriented metadata from the data source. It states, interprets and understands the relationships between data sets and core elements including source, target, and ingestion processes.

Centralized Auditing

It makes a log registry for interaction with the data stored in HDFS for reporting.

Search and Lineage

It develops a well-defined path for data exploration by recording the information about the creation of data and metadata.

Security and Policy Engine

It defines and justifies data access by role-based access and protects data from tampering [118].

The data governance layer is also responsible for resource management and job scheduling. Data available in the data lake are not in a similar structure. Therefore, efficient resource management is essential to make the negotiation of data analytics programs easier. The existing healthcare analytics system uses the Map Reduce methodology to schedule the CPU cycle and memory across the clusters in Hadoop to process the job [144]. However, MapReduce does not deal with the scheduling of data resources for waiting jobs. Therefore,

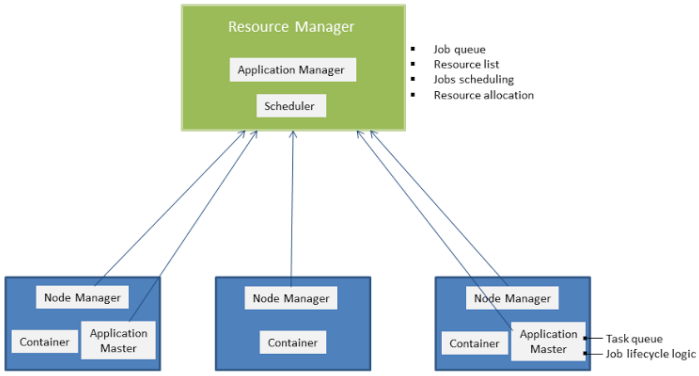


Figure 5.2: Workflow of YARN

Apache Hadoop YARN (Yet Another Resource Negotiator) is employed instead of MapReduce as a data operating system to enable the data processing systems to interact efficiently with the data [145]. Unlike MapReduce, YARN handles data processing efficiently by splitting resource management and job scheduling into separate process [119]. Figure 5.2 depict the workflow of YARN.

Application Master (AM), Node Manager (NM), Resource Manager (RM) are the major actors of YARN. These actors work as follows:

- Once a data processing engine request reaches YARN, it allocates the required resources to start AM. After start up, AM logs its entry in RM.
- If any additional resources are needed, AM can negotiate with RM. Once the resource are made available, AM makes the NM allocate the resources to the process.
- A vacuum created for the resources and the code to run within it. AM gets the execution status and it is reported to RM too.
- The client can communicate with either AM or RM to get the status update. Once the code is executed, AM for the respective job will remove itself from the RM and release all the allocated resources.

5.5.3 Security Layer

Healthcare data needs a highly secured environment because it contains information that is very sensitive [146, 147]. Therefore, a more efficient security system for HDFS needed. Authentication and authorization are the two crucial processes for providing controlled access in HDFS [148, 149].

- To provide authentication, the architecture uses the Kerberos authentication protocol. The Kerberos protocol creates a proxy server to receive a client request [150]. If the request is legitimate, it provides a ticket for the client with a timestamp.
- Apache Ranger is an efficient tool to provide authorization [151]. It is a unified authorization model for HDFS. It enables the data lake leaders to create security policies and role-based access control for the data.

Whenever a client with a legitimate ticket enters the system, the Apache Ranger validates the ticket using its security policies. It has a very flexible user interface, and it is easy to deploy security policies for the vast amount of data storage.

5.5.4 Analytics layer

Compared to a data warehouse, a data lake is very effective at utilizing the vast amount of data with data analytical algorithms to identify valuable insights that will improve real-time decision analytics. Since the HDFS can be connected with a large number of data analytics tools, a data lake can be adopted for the future.

Clusters of patients with similar health conditions and drug acceptance based on their details available in the EHR and other data sources created to evaluate our proposed architecture.

- the K-means clustering algorithm is used to perform the clustering. K-means clustering is most widely used clustering algorithm which is used in many areas such as information retrieval, computer vision and pattern recognition. K-means clustering assigns n data points into k clusters so that similar data points can be grouped together. It is an iterative method which assigns each point to the cluster whose centroid is the nearest. Then it again calculates the centroid of these groups by taking its average. The follow steps show the basic approach of K-means clustering:

  1: An initial clustering is created by choosing k random centroids from the dataset.
  2: For each data point, calculate the distance from all centroids, and assign its membership to the nearest centroid.
  3: Recalculate the new cluster centroids by the average of all data points that are assigned to the clusters.
  4: Repeat step 2 until convergence. The K-means algorithm implemented using the MATLAB programming environment.

- The next step is to find the best medication practice for each cluster using a support vector machine (SVM).

Support Vector Machine(SVM) has become an extremely popular algorithm. SVM is a supervised machine-learning model associated with a learning algorithm [152]. SVM is a supervised machine learning algorithm which can be

used for classification or regression problems. It uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs. Simply put, it does some extremely complex data transformations, then figures out how to separate your data based on the labels or outputs you've defined. Well SVM it capable of doing both classification and regression. In this research we focus on using SVM for classification. In particular we will be focusing on non-linear SVM, or SVM using a non-linear kernel. Non-linear SVM means that the boundary that the algorithm calculates doesn't have to be a straight line. The benefit is that one can capture much more complex relationships between your data points without having to perform difficult transformations on their own. The downside is that the training time is much longer as it's much more computationally intensive. Ideal medication training data is used to train the SVM. Then the cluster is processed by SVM, and the most efficient medication recommendation is identified. Some sample data will be given to the SVM to validate its accuracy level. Once the SVM reaches 90% accuracy, it acts as a recommender system for future medication recommendations.

5.6 Experiments

The architecture can be evaluated via experiments on sample EMRs data. In this section, a brief description of the research questions is given, and then the experimental setup to validate the usability of the proposed architecture is explained.

5.6.1 Variables and objects

Independent variables

The independent variable in the experiment is the technique under investigation. By nature, this research assumes the proposed data lake architecture as the independent variable. We focused on the patient clustering methodology in the architecture, as it is the vital process in identifying more suitable healthcare practices for the given patient pool. Besides, we selected the traditional data warehouse (DW) as the baseline technique to evaluate and compare clustering precision.

Dependent Variables

Data Ingestion time: We used data ingestion time as a metric to validate RQ 3.1. The ingestion time is the time taken for the data architecture to load the data to its storage. The Apache Spring XD updated the metadata log when a new data entered in the data architecture. The timer starts when the data reaches the data architecture, and it stops when the data entry is created in the meta-data log. Data ingestion time can be calculated by finding the difference between the meta log time and the arrival time. Equation 5.1 shows the calculation of the data ingestion time for data k by subtracting the data arrival time of k from the metadata log entry time of k.

$$IT_k = MLtime_k - DAtime_k, \tag{5.1}$$

where IT_k refers to the data ingestion time of kth data, $MLtime_k$ represents the data log entry time for data k, and $DAtime_k$ is the data arrival time of data k

Clustering precision: In pattern recognition studies, the importance is in finding the relevance between patterns falling in n-dimensional pattern space. It is crucial to examine the characteristic distance between them to find out the connection between the patterns. The characteristics distance between the patterns decides the unsupervised classification (clustering) criterion. As per the theory, we considered Euclidean distance to evaluate the precision of the clustering [153].

$$\text{Euclidian distance } d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{5.2}$$

We used Euclidean distance as a metric for RQ3.2. For the same collection of patient records, the system needs to create the patient clustering. The distance between two points can be calculated using Equation 5.2 where, x_i and y_i are the ith coordinates for points x and y, respectively, and d is the distance between x and y.

The clustering metric d was calculated for all the available clusters created by both DW architecture and proposed the data lake architecture. The lower value of d indicates the higher precision of clustering. Higher cluster precision implies a more effective data architecture for the healthcare recommendation system.

5.6.2 Objects

Healthcare research demands a set of data from various sources in public and private organisations. It may include organisational-level patient enrolment and payment details, medical records, drug and therapy prescriptions, and clinical notes from general practitioners and nurses. The patient medical records contain data about the ethnicity, age, family member's health status and so on. For this research, a de-identified data set is utilised, which is available online. Because the focus of the research is to handle data efficiently, sample data can be used to prove the effectiveness of the proposed data lake architecture.

We made use of anonymized EHR [154] and its supporting data to evaluate our data lake architecture. We made use of the UCI machine learning repository [155], which contains the data set of diabetes from 130 US hospitals during the years of 1999-2008. It has ten years of inpatient encounters from 130 US hospitals and integrated delivery networks. It contains 50 features representing patient and hospital outcomes. The data includes attributes such as patient number, race, gender, age, admission type, time in the hospital, medical specialty of admitting physician, number of lab test performed, HbA1c test result, diagnosis, number of medications, diabetic medications, as well as the number of outpatients, inpatients, and emergency visits in the year before the hospitalization, etc.

5.6.3 Empirical environment

The data set classified as internally sourced data and externally sourced data. The internal source data was connected with the HDFS system Apache Spring

XD and loaded into the data lake. The externally sourced data connected with the data lake system using Apache Flume. Apache Atlas identified the metadata available with data from the source. Each data is allocated a unique identifier for easy access. The K-means algorithm was performed on the available data to identify the available clusters using MATLAB. The identified cluster contained patients with similar health conditions. The training data from the data set used to train the SVM. The SVM runs on each cluster to find the most successful medication recommendation. When a new data arrives into the system after authorization, the SVM identifies its personalized recommendation.

5.7 Experimental Results

This section describes the performance of the proposed data lake architecture compared with the data warehouse.

5.7.1 Reduction of Data Ingestion Time

We collected the data arrival time, and metadata log entry time for all the data tuples from the dataset in both DW and our data lake architecture. We then calculated the data ingestion time based on Equation 5.1. Figure 5.3 shows the average values of the ingestion time for DW and the data lake architecture. It demonstrates that the data lake architecture has a much lower average value of ingestion time than the DW.

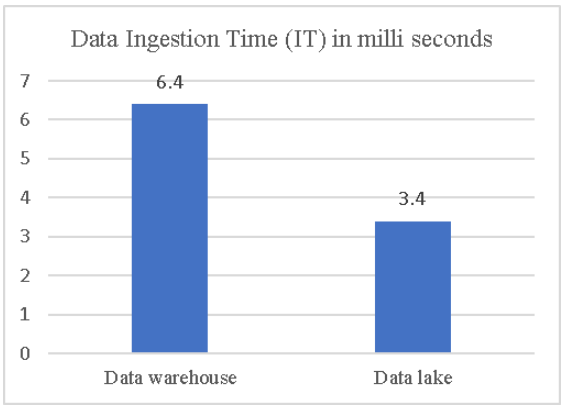


Figure 5.3: Comparison of data ingestion time of the DW and the data lake

Answer to RQ 3.1:

The experiment results clearly show the proposed approach's ability to store data, even in native form. The data ingestion time is improved significantly by the data lake architecture. Since the proposed architecture makes use of the HDFS file system, it does not require the data preprocessing stage. In contrast, the DW technique involves the ETL process, which takes much more time to ingest the data into the data warehouse.

The Apache Flume can pull the data from the remote data vendors and store it successfully in the data lake environment. The Kerberos engine creates an authentication ticket for each login and the Apache Ranger tool verifies the authentication tickets and provides the access rights to the remote login. These steps enable the third-party data stakeholders to connect with the data lake architecture with security.

5.7.2 Removal of Data Silos

We compared the clustering algorithms based on precision quality. We identified four clusters with maximum data points for each data architecture. The data points (namely, x and y) were further identified for the calculation of the precision value. In particular, the Euclidean distance d between x and y was calculated according to Equation 5.2. Table 5.1 summarises the values of d for each cluster. The results clearly show that the data lake architecture has smaller values of d for clustering than DW. In other words, the proposed data lake architecture has higher precision in clustering.

Answer to RQ 3.2:

The precision of clustering typically increases as the amount of data becomes more substantial. However, due to its schema and the organizational rules of DW, much vital information about patients will be lost during the ETL process. In contrast, the data lake architecture makes use of the schema-on-read method to load the data into the environment, which brings the ability to connect with the third-party and external data stakeholders. The high availability of data about patients helps improve the precision of clustering. In summary, the data lake architecture delivers the ability to store a variety of data within a unified location. Thus, clustering precision has been significantly improved.

Table 5.1: Comparison of precision value *d* for DW and the data lake

Euclidean distance	***d* value of data lake**	***d* value of DW**
Cluster 1	0.64	0.76
Cluster 2	0.71	0.87
Cluster 3	0.65	0.87
Cluster 4	0.88	0.99

5.8 Chapter summary

Personalized healthcare services utilise relational patient data and big data analytics to tailor medication recommendations. However, most healthcare data are in an unstructured form, and it consumes a lot of time and effort to turn them into a relational form. This study proposed a novel data lake architecture to reduce the data ingestion time and improve the precision of healthcare analytics. It also removed the data silos and enhanced the analytics by allowing connectivity to third-party data providers (such as clinical labs, chemists, insurance companies, etc.). The data lake architecture used the Hadoop Distributed File System (HDFS) to provide storage for both structured and unstructured data. This study used the K-means clustering algorithm to find patient clusters with similar health conditions. Subsequently, it employed a support vector machine to find the most successful healthcare recommendations for each cluster. Our experiment results demonstrated the ability of the data lake to reduce the time for ingesting data from various data vendors regardless of its format. Moreover, the data lake has the potential to generate clusters of patients more precisely than the existing approaches. It is evident that the data lake provides a unified storage location for data in its native format. It can also improve personalized healthcare medication recommendations by removing data silos.

CHAPTER 6

CONCLUSION & FUTURE DIRECTIONS

Service-oriented architecture (SOA) has become a modern computing infrastructure for most online-based services. The findings of this research will contribute to the development of an SOA framework to support Web service consumers in their discovery, selection, and composition of Web services. This research addressed three significant challenges for the broader adoption of SOA in various application domains. The first challenge is to improve the efficiency of QoS prediction and thus the Web service discovery. The second challenge for developing an efficient SOA is Web service selection and composition. The last problem is to identify a competent data architecture for SOA applications. On these grounds, the following objectives are framed for SOA broader adaptation:

- to develop an approach for improved Quality of Service (QoS) prediction for Web services.
- to introduce a framework for enhanced Web service selection and composition
- to propose a capable data architecture for the Web service composite to improve the performance of the SOA application.

To achieve these objectives, this thesis proposed a solution for each. QoS values have become an essential criterion for choosing a suitable service from an abundance of functionally similar Web services. On the other hand, service providers do not provide adequate QoS data, while an unequal computing and network environment make the QoS data supplied by the user is invalid. Therefore, predicting the QoS values of a Web service is an essential step in service-

oriented systems. One of the independent methods to predict the quality parameters of the Web service is to utilize software code metrics. A Web service is not just a single system; rather, it contains many classes and methods. Thus, source code metrics should be calculated from the micro level, such as classes and methods to a macro level system. However, most of the current systems either calculate code metrics at a macro level or use basic arithmetic average and mean to lift the metrics from the class level to the system level. Such methods may cover up the inefficient values and provide the values of a rounded-up metric for the predictor.

This thesis proposed the inequality coefficient for source code metric aggregation for the first problem stated and the limitation identified **(Chapter 3)**. Three sets of metrics, namely CKM, BSM and SM, were used to validate the proposed system. The system calculated source code metrics at the class level for CKM metrics. The framework used the Theil index, a method to aggregate source code metrics without compromising the distributed nature of the software source code. The system used Sneed's tool For SM metrics to calculate the complexity metrics from the Web service class files. Macro level WSDL files instead of class files were used to calculate the BSM metrics. The system applied Linear regression to predict modularity. The results showed that the SM metrics individually outperformed the other sets of metrics. CKM metrics also have the potential to predict the QoS value but not as efficiently as SM metrics. BSM metrics had the lowest efficiency among the available group of metrics. In summary, metric values calculated at the micro level have better QoS prediction efficiency than those at the macro level.

Most SOA applications demand that more than one Web service is combined

to do a particular task. Selecting the appropriate Web service and its composite is the second problem addressed in this research. Standard SOA tools require an extensive understanding of SOA techniques to implement, which is very demanding. As an alternative, the most commonly used information retrieval technique, named the keyword-based search, was applied in this research **(Chapter 4)**. A schema-based keyword search with a relational database is an innovative method for the incorporation and optimization of the SOA design, search and selection. Experiments used WS-Dream dataset to identify the appropriate Web service composition. The proposed framework used the success rate and the response as the validation parameters. According to the experiment results, the response time of Web services discovery without considering quality constraints performed well compared to the calculation of quality constraints. On the other hand, as the number of Web services and quality constraints increased, the success rate increased significantly. However, it is ignorable as the success rate is performed well where the search used the quality criteria. Despite the low performing response time, the keyword search with the quality requirement significantly improved the success rate for Web service discovery. The experiment results showed that schema-based keyword search technology has the potential to develop valid Web service selection and composition.

Insufficient support from the existing data architecture for the SOA is a significant obstacle for many mission-critical applications such as healthcare. The current data architecture available for SOA has less potential to handle dynamic data flow. The traditional data warehouse (DW) technique is no longer suitable for healthcare analytics due to its schema on write nature and its inability to handle data silos. This research proposed data lake as a valid data archi-

tecture for an SOA application to address the third problem **(Chapter 5).** A personalized healthcare recommendation application was used to demonstrate the proposed architecture. The data lake is a flat, schema-on-read data architecture. The experiments were conducted based on the UCI repository dataset to compare and evaluate both data lake and DW. The data lake architecture outperformed DW significantly in terms of data ingestion time. The time to load and store data in the data lake architecture was nearly 50% less than that for DW. Moreover, the data lake architecture had higher precision in clustering than DW, mainly because of its ability to connect with more data sources. In brief, the data lake architecture was demonstrated as an effective alternative to the existing health IT infrastructure. The proposed system can ingest all data in unstructured, semi-structured and structured formats. It can store data at a low cost, taking advantage of HDFS. Hence, the data lake-based healthcare recommendation system addresses the drawbacks of traditional data architectures and provides additional capabilities for the future calibre of data reusability.

The research conducted in this thesis can be extended in the several following promising directions:

Advanced machine learning on Improved software source code metrics to predict Web service QoS: The Theil index-based source metric aggregation framework used multiple linear regression with 10-fold cross-validation to demonstrate the potential to predict QoS using improved source code metrics. This framework's scope is to explore the ability of the Theil index as an alternative for source code metrics. However, the machine learning algorithm used for predicting the QoS is comparatively naïve. In the future, modern machine learning algorithms such as Artificial neural networks and deep learning can be

utilised to improve the prediction quality.

Semantic Web service: Web services can be categorised into two broader categories, namely syntactic-based and semantic-based. In this research, our focus was only on syntax Web services, and we ignore semantics Web services due to time and resource limitations. In future, keyword-based selection and composition can be extended to semantic Web services. In the domain of the Semantic Web, the Web Ontology Language for Services (OWL-S) and the Web service Modelling Ontology (WSMO) are two prominent techniques used for service composition. Semantic Web services are an extension of the existing Web services where the information is represented in a well-defined way.

Keyword recommendation: According to the keyword-based search experiment results, irrelevant keywords and the QoS requirement led to less relevant results for the user query. Thus, the system engineer needs to have a thorough understanding of the functional properties of the required SOA. In our future work, collaborative filtering-based prediction can be used to recommend keywords for system engineers.

Cloud computing: Cloud computing is the next-generation computing model which has a significant position in the field of scientific and business computing. Although the proposed Theil-index based approach can be directly used to predict the QoS values of cloud services, they do not consider the influence factors underlying the cloud architecture. We can consider more underlying hardware resource data and the relevance of other cloud service QoS attributes; also, we can increase the number of QoS attributes combined with software/hardware resources to predict cloud QoS, although complexity will increase.

Social media data lake: Social media and connected devices are twining with each other and unearthing an abundance of opportunities. Social media users produce a vast amount of data that can be processed and utilised to provide more optimised services. The nature of the data lake architecture and Service-Oriented Architecture's power can be used for sophisticated service offering by processing the social media and connected devices data. We are planning to explore this as a potential research direction.

Summary of contributions

An inequality distribution named the Theil index was proposed as an alternative aggregation model for source code metrics. Experiments were conducted using three different data sets, and the results showed that the Theil index can significantly improve QoS prediction. A schema-based keyword search supported by the relational database was adopted for Web service selection and composition. With a few keywords that explain composite functions, it could help system engineers without a thorough understanding of SOA technology to define SOA solutions. An emerging data architecture, the data lake, was employed to support SOA applications. An SOA-based application for personalised healthcare recommendation was implemented using both the data warehouse and the data lake, and the experiment results demonstrated that the data lake outperforms the DW.

BIBLIOGRAPHY

[1] K. Kritikos and D. Plexousakis, "Qos-based web service description and discovery," *ERCIM news*, 2008.

[2] L.-J. Zhang, J. Zhang, and H. Cai, "Services computing. 2007."

[3] J. Zhu, Y. Kang, Z. Zheng, and M. R. Lyu, "A clustering-based qos prediction approach for web service recommendation," in *Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), 2012 15th IEEE International Symposium on*. IEEE, 2012, pp. 93–98.

[4] S. A. McIlraith, T. C. Son, and H. Zeng, "Semantic web services," *IEEE intelligent systems*, vol. 16, no. 2, pp. 46–53, 2001.

[5] S.-Y. Lin, C.-H. Lai, C.-H. Wu, and C.-C. Lo, "A trustworthy qos-based collaborative filtering approach for web service discovery," *Journal of Systems and Software*, vol. 93, pp. 217–228, 2014.

[6] K. Su, B. Xiao, B. Liu, H. Zhang, and Z. Zhang, "Tap: a personalized trust-aware qos prediction approach for web service recommendation," *Knowledge-Based Systems*, vol. 115, pp. 55–65, 2017.

[7] P. Harshavardhanan, J. Akilandeswari, and R. Sarathkumar, "Dynamic web services discovery and selection using qos-broker architecture," in *Computer Communication and Informatics (ICCCI), 2012 International Conference on*. IEEE, 2012, pp. 1–5.

[8] Z. Chen, L. Shen, F. Li, and D. You, "Your neighbors alleviate cold-start: On geographical neighborhood influence to collaborative web service qos prediction," *Knowledge-Based Systems*, vol. 138, pp. 188–201, 2017.

[9] M. Peng, G. Zeng, Z. Sun, J. Huang, H. Wang, and G. Tian, "Personalized app recommendation based on app permissions," *World Wide Web*, vol. 21, no. 1, pp. 89–104, 2018.

[10] J. Li, C. Liu, and J. Xu, "Xbridge-mobile: efficient xml keyword search on mobile web data," *Computing*, vol. 96, no. 7, pp. 631–650, 2014.

[11] M. Li, X. Sun, H. Wang, Y. Zhang, and J. Zhang, "Privacy-aware access

control with trust management in web service," *World Wide Web*, vol. 14, no. 4, pp. 407–430, 2011.

[12] X. Huang, "Usageqos: Estimating the qos of web services through online user communities," *ACM Transactions on the Web (TWEB)*, vol. 8, no. 1, p. 1, 2013.

[13] X. Huang, W. Huang, and W. Lai, "Uip: Estimating true rating scores of services through online user communities," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2016, pp. 1–7.

[14] J. Xu, C. Zhu, and Q. Xie, "An online prediction framework for dynamic service-generated qos big data," in *International Conference on Database Systems for Advanced Applications*. Springer, 2017, pp. 60–74.

[15] Z. Chen, L. Shen, D. You, F. Li, and C. Ma, "Alleviating data sparsity in web service qos prediction by capturing region context influence," in *International Conference on Collaborative Computing: Networking, Applications and Worksharing*. Springer, 2016, pp. 540–556.

[16] H. Alexander, I. Khalil, C. Cameron, Z. Tari, and A. Zomaya, "Cooperative web caching using dynamic interest-tagged filtered bloom filters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 11, pp. 2956–2969, 2014.

[17] J. L. O. Coscia, M. Crasso, C. Mateos, A. Zunino, and S. Misra, "Predicting web service maintainability via object-oriented metrics: a statistics-based approach," in *International Conference on Computational Science and Its Applications*. Springer, 2012, pp. 29–39.

[18] K. Mordal, N. Anquetil, J. Laval, A. Serebrenik, B. Vasilescu, and S. Ducasse, "Software quality metrics aggregation in industry," *Journal of Software: Evolution and Process*, vol. 25, no. 10, pp. 1117–1135, 2013.

[19] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on software engineering*, vol. 20, no. 6, pp. 476–493, 1994.

[20] H. M. Sneed, "Measuring web service interfaces," in *Web Systems Evolution (WSE), 2010 12th IEEE International Symposium on*. IEEE, 2010, pp. 111–115.

[21] D. Baski and S. Misra, "Metrics suite for maintainability of extensible markup language web services," *IET software*, vol. 5, no. 3, pp. 320–341, 2011.

[22] K. N. Wang, J. S. Bell, E. Y. Chen, J. F. Gilmartin-Thomas, and J. Ilomäki, "Medications and prescribing patterns as factors associated with hospitalizations from long-term care facilities: a systematic review," *Drugs & aging*, vol. 35, no. 5, pp. 423–457, 2018.

[23] M. R. Islam, M. A. Kabir, A. Ahmed, A. R. M. Kamal, H. Wang, and A. Ulhaq, "Depression detection from social network data using machine learning techniques," *Health information science and systems*, vol. 6, no. 1, p. 8, 2018.

[24] M. A. H. Masud and X. Huang, "An e-learning system architecture based on cloud computing," *system*, vol. 10, no. 11, pp. 255–259, 2012.

[25] N. Zhang, J. Wang, Y. Ma, K. He, Z. Li, and X. F. Liu, "Web service discovery based on goal-oriented query expansion," *Journal of Systems and Software*, vol. 142, pp. 73–91, 2018.

[26] J. Huang, M. Peng, H. Wang, J. Cao, W. Gao, and X. Zhang, "A probabilistic method for emerging topic tracking in microblog stream," *World Wide Web*, vol. 20, no. 2, pp. 325–350, 2017.

[27] J. Du, S. Michalska, S. Subramani, H. Wang, and Y. Zhang, "Neural attention with character embeddings for hay fever detection from twitter," *Health information science and systems*, vol. 7, no. 1, p. 21, 2019.

[28] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," 2008.

[29] J. Zhang, X. Tao, and H. Wang, "Outlier detection from large distributed databases," *World Wide Web*, vol. 17, no. 4, pp. 539–568, 2014.

[30] D. Benslimane, S. Dustdar, and A. Sheth, "Services mashups: The new generation of web applications," *IEEE Internet Computing*, vol. 12, no. 5, pp. 13–15, 2008.

[31] X. Liu, Y. Ma, G. Huang, J. Zhao, H. Mei, and Y. Liu, "Data-driven composition for service-oriented situational web applications," *IEEE Transactions on Services Computing*, vol. 8, no. 1, pp. 2–16, 2014.

[32] H. Li, Y. Wang, H. Wang, and B. Zhou, "Multi-window based ensemble learning for classification of imbalanced streaming data," *World Wide Web*, pp. 1–19, 2017.

[33] L. Chen, Y. Wang, Q. Yu, Z. Zheng, and J. Wu, "Wt-lda: user tagging augmented lda for web service clustering," in *International Conference on Service-Oriented Computing*. Springer, 2013, pp. 162–176.

[34] J. X. Yu, L. Qin, and L. Chang, "Keyword search in databases," *Synthesis Lectures on Data Management*, vol. 1, no. 1, pp. 1–155, 2009.

[35] Q. He, R. Zhou, X. Zhang, Y. Wang, D. Ye, F. Chen, J. C. Grundy, and Y. Yang, "Keyword search for building service-based systems," *IEEE Transactions on Software Engineering*, vol. 43, no. 7, pp. 658–674, 2017.

[36] Q. He, R. Zhou, X. Zhang, Y. Wang, D. Ye, F. Chen, S. Chen, J. Grundy, and Y. Yang, "Efficient keyword search for building service-based systems based on dynamic programming," in *International Conference on Service-Oriented Computing*. Springer, 2017, pp. 462–470.

[37] J. Akilandeswari, K. A. Krishna *et al.*, "Semantic web service discovery with structural level matching of operations," in *International Conference on Advanced Computing, Networking and Security*. Springer, 2011, pp. 77–84.

[38] T. Rajendran and P. Balasubramanie, "Analysis on the study of qos-aware web services discovery," *arXiv preprint arXiv:0912.3965*, 2009.

[39] ——, "An optimal broker-based architecture for web service discovery with qos characteristics," *International Journal of Web Services Practices*, vol. 5, no. 1, pp. 32–40, 2010.

[40] D. A. D'Mello, V. Ananthanarayana, and S. Thilagam, "A qos broker based architecture for dynamic web service selection," in *Modeling & Simulation, 2008. AICMS 08. Second Asia International Conference on*. IEEE, 2008, pp. 101–106.

[41] X. Zhang, Z. Wang, W. Zhang, and F. Yang, "A time-aware qos prediction approach to web service recommendation," in *Proceedings of the 4th International Conference on Computer Engineering and Networks*. Springer, 2015, pp. 739–748.

[42] L. Bartoloni, A. Brogi, and A. Ibrahim, "Probabilistic prediction of the qos of service orchestrations: a truly compositional approach," in *International Conference on Service-Oriented Computing*. Springer, 2014, pp. 378–385.

[43] S. Li, J. Wen, F. Luo, M. Gao, J. Zeng, and Z. Y. Dong, "A new qos-aware web service recommendation system based on contextual feature recognition at server-side," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 332–342, 2017.

[44] Y. Zhang and M. R. Lyu, "Qos-aware web service searching," in *QoS Prediction in Cloud and Service Computing*. Springer, 2017, pp. 81–103.

[45] L. Y. L. Ye and B. Z. B. Zhang, "Web service discovery based on functional semantics," in *2006 Semantics, Knowledge and Grid, Second International Conference on*. IEEE, 2006, pp. 57–57.

[46] L. Ye and B. Zhang, "Discovering web services based on functional semantics," in *Services Computing, 2006. APSCC'06. IEEE Asia-Pacific Conference on*. IEEE, 2006, pp. 348–355.

[47] D. A. D'Mello and V. Ananthanarayana, "Effective web service discovery based on functional semantics," in *Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT'09. International Conference on*. IEEE, 2009, pp. 1–3.

[48] A. Averbakh, D. Krause, and D. Skoutas, "Exploiting user feedback to improve semantic web service discovery," *The Semantic Web-ISWC 2009*, pp. 33–48, 2009.

[49] M. Chen and Y. Ma, "A hybrid approach to web service recommendation based on qos-aware rating and ranking," *arXiv preprint arXiv:1501.04298*, 2015.

[50] Z. Xu, P. Martin, W. Powley, and F. Zulkernine, "Reputation-enhanced qos-based web services discovery," in *Web Services, 2007. ICWS 2007. IEEE International Conference on*. IEEE, 2007, pp. 249–256.

[51] D. Mobedpour and C. Ding, "User-centered design of a qos-based web service selection system," *Service Oriented Computing and Applications*, vol. 7, no. 2, pp. 117–127, 2013.

[52] R. Iordache and F. Moldoveanu, "Qos-aware web service semantic selec-

tion based on preferences," *Procedia Engineering*, vol. 69, pp. 1152–1161, 2014.

[53] Z. Chen, L. Shen, and F. Li, "Exploiting web service geographical neighborhood for collaborative qos prediction," *Future Generation Computer Systems*, vol. 68, pp. 248–259, 2017.

[54] P. He, J. Zhu, J. Xu, and M. R. Lyu, "A hierarchical matrix factorization approach for location-based web service qos prediction," in *Service Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on*. IEEE, 2014, pp. 290–295.

[55] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "A privacy-preserving qos prediction framework for web service recommendation," in *Web Services (ICWS), 2015 IEEE International Conference on*. IEEE, 2015, pp. 241–248.

[56] M. Charrad, N. Y. Ayadi, M. B. Ahmed *et al.*, "A semantic and qos-aware broker for service discovery," *Journal of Research and Practice in Information Technology*, vol. 44, no. 4, p. 387, 2012.

[57] Y. Ma, S. Wang, P. C. Hung, C.-H. Hsu, Q. Sun, and F. Yang, "A highly accurate prediction algorithm for unknown web service qos values," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 511–523, 2016.

[58] A. Kattepur, "Flexible Quality of Service Management of Web Services Orchestrations," Theses, Université Rennes 1, Nov. 2012. [Online]. Available: https://tel.archives-ouvertes.fr/tel-00756048

[59] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized qos prediction for web services via collaborative filtering," in *Web Services, 2007. ICWS 2007. IEEE International Conference on*. IEEE, 2007, pp. 439–446.

[60] L.-J. Zhang, H. Cai, and J. Zhang, *Services computing*. Springer, 2007.

[61] D. B. Claro, P. Albers, and J.-K. Hao, "Web services composition," in *Semantic Web Services, Processes and Applications*. Springer, 2006, pp. 195–225.

[62] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Qos-aware web service recommendation by collaborative filtering," *IEEE Transactions on services computing*, vol. 4, no. 2, pp. 140–152, 2010.

[63] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun, "Personalized qos-aware web service recommendation and visualization," *IEEE Transactions on Services Computing*, vol. 6, no. 1, pp. 35–47, 2011.

[64] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[65] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "Collaborative web service qos prediction with location-based regularization," in *Web services (ICWS), 2012 IEEE 19th international conference on*. IEEE, 2012, pp. 464–471.

[66] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.

[67] Z. Li, Z. Bin, L. Ying, G. Yan, and Z. Zhi-Liang, "A web service qos prediction approach based on collaborative filtering," in *Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific*. IEEE, 2010, pp. 725–731.

[68] M. Tang, Y. Jiang, J. Liu, and X. F. Liu, "Location-aware collaborative filtering for qos-based service recommendation," in *2012 IEEE 19th International Conference on Web Services*. IEEE, 2012, pp. 202–209.

[69] Y. Xu, J. Yin, W. Lo, and Z. Wu, "Personalized location-aware qos prediction for web services using probabilistic matrix factorization," in *International Conference on Web Information Systems Engineering*. Springer, 2013, pp. 229–242.

[70] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service qos prediction via neighborhood integrated matrix factorization," *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 289–299, 2013.

[71] C. Mateos, M. Crasso, A. Zunino, and J. L. O. Coscia, "Detecting wsdl bad practices in code–first web services," *International Journal of Web and Grid Services*, vol. 7, no. 4, pp. 357–387, 2011.

[72] L. Kumar, A. Krishna, and S. K. Rath, "The impact of feature selection on maintainability prediction of service-oriented applications," *Service Oriented Computing and Applications*, vol. 11, no. 2, pp. 137–161, 2017.

[73] L. Kumar, M. Kumar, and S. K. Rath, "Maintainability prediction of web

service using support vector machine with various kernel methods," *International Journal of System Assurance Engineering and Management*, vol. 8, no. 2, pp. 205–222, 2017.

[74] D. Romano and M. Pinzger, "Using source code metrics to predict change-prone java interfaces," in *2011 27th IEEE International Conference on Software Maintenance (ICSM)*. IEEE, 2011, pp. 303–312.

[75] M. Lanza and R. Marinescu, *Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-oriented systems*. Springer Science & Business Media, 2007.

[76] S. D. Suh and I. Neamtiu, "Studying software evolution for taming software complexity," in *2010 21st Australian Software Engineering Conference*. IEEE, 2010, pp. 3–12.

[77] M. Lumpe, S. Mahmud, and R. Vasa, "On the use of properties in java applications," in *2010 21st Australian Software Engineering Conference*. IEEE, 2010, pp. 235–244.

[78] R. Shatnawi, W. Li, J. Swain, and T. Newman, "Finding software metrics threshold values using roc curves," *Journal of software maintenance and evolution: Research and practice*, vol. 22, no. 1, pp. 1–16, 2010.

[79] H. Barkmann, R. Lincke, and W. Löwe, "Quantitative evaluation of software quality metrics in open-source projects," in *2009 International Conference on Advanced Information Networking and Applications Workshops*. IEEE, 2009, pp. 1067–1072.

[80] A. Serebrenik and M. van den Brand, "Theil index for aggregation of software metrics values," in *Software Maintenance (ICSM), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1–9.

[81] B. Srivastava and J. Koehler, "Web service composition-current solutions and open problems," in *ICAPS 2003 workshop on Planning for Web Services*, vol. 35, 2003, pp. 28–35.

[82] A. A. Patil, S. A. Oundhakar, A. P. Sheth, and K. Verma, "Meteor-s web service annotation framework," in *Proceedings of the 13th international conference on World Wide Web*, 2004, pp. 553–562.

[83] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and

H. Chang, "Qos-aware middleware for web services composition," *IEEE Transactions on software engineering*, vol. 30, no. 5, pp. 311–327, 2004.

[84] A. Lazovik, M. Aiello, and M. Papazoglou, "Associating assertions with business processes and monitoring their execution," in *Proceedings of the 2nd international conference on Service oriented computing*, 2004, pp. 94–104.

[85] ——, "Planning and monitoring the execution of web service requests," *International Journal on Digital Libraries*, vol. 6, no. 3, pp. 235–246, 2006.

[86] M. Pistore, A. Marconi, P. Bertoli, and P. Traverso, "Automated composition of web services by planning at the knowledge level," in *IJCAI*, vol. 19, 2005, pp. 1252–1259.

[87] L. A. da Costa, P. F. Pires, and M. Mattoso, "Automatic composition of web services with contingency plans," in *Proceedings. IEEE International Conference on Web Services, 2004.* IEEE, 2004, pp. 454–461.

[88] A. Lazovik, M. Aiello, and M. Papazoglou, "Planning and monitoring the execution of web service requests," in *International Conference on Service-Oriented Computing*. Springer, 2003, pp. 335–350.

[89] V. Agarwal, K. Dasgupta, N. Karnik, A. Kumar, A. Kundu, S. Mittal, and B. Srivastava, "A service creation environment based on end to end composition of web services," in *Proceedings of the 14th international conference on World Wide Web*, 2005, pp. 128–137.

[90] E. F. Codd, "A relational model of data for large shared data banks," in *Software pioneers*. Springer, 2002, pp. 263–294.

[91] R. Elmasri and S. Navathe, "Fundamentals of database systems, ch. 3," 2010.

[92] V. Hristidis and Y. Papakonstantinou, "Discover: Keyword search in relational databases," in *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*. Elsevier, 2002, pp. 670–681.

[93] G. Li, J. Feng, and L. Zhou, "Retune: retrieving and materializing tuple units for effective keyword search over relational databases," in *International Conference on Conceptual Modeling*. Springer, 2008, pp. 469–483.

[94] J. X. Yu, L. Qin, and L. Chang, "Keyword search in relational databases: A survey." *IEEE Data Eng. Bull.*, vol. 33, no. 1, pp. 67–78, 2010.

[95] S. Agrawal, S. Chaudhuri, and G. Das, "Dbxplorer: a system for keyword-based search over relational databases," *Proceedings 18th International Conference on Data Engineering*, pp. 5–16, 2002.

[96] A. Markowetz, Y. Yang, D. Papadias, and D. Papadias, "Keyword search on relational data streams," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM, 2007, pp. 605–616.

[97] J. Zhou, T. Zhang, H. Meng, L. Xiao, G. Chen, and D. Li, "Web service discovery based on keyword clustering and ontology," in *2008 IEEE International Conference on Granular Computing*. IEEE, 2008, pp. 844–848.

[98] J. Zhai, D. Shen, Y. Kou, and T. Nie, "Richly semantical keyword searching over relational databases," in *2012 Ninth Web Information Systems and Applications Conference*. IEEE, 2012, pp. 211–216.

[99] L. Zhu, S.-D. Ji, W.-Z. Yang, and C.-N. Liu, "Keyword search based on knowledge base in relational databases," in *2009 International Conference on Machine Learning and Cybernetics*, vol. 3. IEEE, 2009, pp. 1528–1533.

[100] J. Feng, G. Li, and J. Wang, "Finding top-k answers in keyword search over relational databases using tuple units," *IEEE transactions on knowledge and data engineering*, vol. 23, no. 12, pp. 1781–1794, 2011.

[101] C. Zeng, Y. Zheng, D. Han *et al.*, "Efficient web service composition and intelligent search based on relational database," in *2010 International Conference on Information Science and Applications*. IEEE, 2010, pp. 1–8.

[102] J. Kwon, K. Park, D. Lee, and S. Lee, "Psr: Pre-computing solutions in rdbms for fastweb services composition search," in *IEEE International Conference on Web Services (ICWS 2007)*. IEEE, 2007, pp. 808–815.

[103] J. Hoffmann, P. Bertoli, and M. Pistore, "Web service composition as planning, revisited: In between background theories and initial state uncertainty," in *AAAI*, vol. 7, 2007, pp. 1013–1018.

[104] U. Küster, B. König-Ries, M. Stern, and M. Klein, "Diane: an integrated approach to automated service discovery, matchmaking and composi-

tion," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 1033–1042.

[105] M. Klusch and F. Kaufer, "Wsmo-mx: A hybrid semantic web service matchmaker," *Web Intelligence and Agent Systems: An International Journal*, vol. 7, no. 1, pp. 23–42, 2009.

[106] A. Brogi, S. Corfini, and R. Popescu, "Semantics-based composition-oriented discovery of web services," *ACM Transactions on Internet Technology (TOIT)*, vol. 8, no. 4, p. 19, 2008.

[107] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Transactions on software engineering*, vol. 33, no. 6, pp. 369–384, 2007.

[108] A. V. Riabov, E. Boillet, M. D. Feblowitz, Z. Liu, and A. Ranganathan, "Wishful search: interactive composition of data mashups," in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 775–784.

[109] G. Huang, Y. Ma, X. Liu, Y. Luo, X. Lu, and M. B. Blake, "Model-based automated navigation and composition of complex service mashups," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 494–506, 2014.

[110] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.

[111] R. Henry and S. Venkatraman, "Big data analytics the next big learning opportunity," *Journal of Management Information and Decision Sciences*, vol. 18, no. 2, p. 17, 2015.

[112] D. G. Katehakis and M. Tsiknakis, "Electronic health record," *Wiley Encyclopedia of Biomedical Engineering*, 2006.

[113] P. S. Mathew and A. S. Pillai, "Big data challenges and solutions in healthcare: A survey," in *Innovations in Bio-Inspired Computing and Applications*. Springer, 2016, pp. 543–553.

[114] B. Feldman, E. M. Martin, and T. Skotnes, "Big data in healthcare hype and hope," *October 2012. Dr. Bonnie*, vol. 360, 2012.

[115] J. Yoon, C. Davtyan, and M. van der Schaar, "Discovery and clinical decision support for personalized healthcare," *IEEE Journal of Biomedical and Health Informatics*, 2017.

[116] D. A. Davis, N. V. Chawla, N. Blumm, N. Christakis, and A.-L. Barabási, "Predicting individual disease risk based on medical history," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. ACM, 2008, pp. 769–778.

[117] B. Dentino, D. Davis, and N. V. Chawla, "Healthcarend: leveraging ehr and care for prospective healthcare," in *Proceedings of the 1st ACM International Health Informatics Symposium*. ACM, 2010, pp. 841–844.

[118] A. Abbas, M. Ali, M. U. S. Khan, and S. U. Khan, "Personalized healthcare cloud services for disease risk assessment and wellness management using social media," *Pervasive and Mobile Computing*, vol. 28, pp. 81–99, 2016.

[119] V. Patel, M. Adhil, T. Bhardwaj, and A. K. Talukder, "Big data analytics of genomic and clinical data for diagnosis and prognosis of cancer," in *Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on*. IEEE, 2015, pp. 611–615.

[120] P. Calyam, A. Mishra, R. B. Antequera, D. Chemodanov, A. Berryman, K. Zhu, C. Abbott, and M. Skubic, "Synchronous big data analytics for personalized and remote physical therapy," *Pervasive and Mobile Computing*, vol. 28, pp. 3–20, 2016.

[121] S. Barlow, "Comparing the three major approaches to healthcare data warehousing," 2017.

[122] L. A. Linn and M. B. Koo, "Blockchain for health data and its potential use in health it and health care related research," in *ONC/NIST Use of Blockchain for Healthcare and Research Workshop. Gaithersburg, Maryland, United States: ONC/NIST*, 2016.

[123] C. Walker and H. Alrehamy, "Personal data lake with data gravity pull," in *Big Data and Cloud Computing (BDCloud), 2015 IEEE Fifth International Conference on*. IEEE, 2015, pp. 160–167.

[124] R. Vasa, M. Lumpe, P. Branch, and O. Nierstrasz, "Comparative analysis of evolving software systems using the gini coefficient," in *2009 IEEE International Conference on Software Maintenance*. IEEE, 2009, pp. 179–188.

[125] H. Theil, *Economics and information theory*, ser. Studies in mathematical and managerial economics. North-Holland Pub. Co., 1967. [Online]. Available: https://books.google.com.au/books?id=VVNVAAAAMAAJ

[126] M. Kargar and A. An, "Keyword search in graphs: Finding r-cliques," *Proceedings of the VLDB Endowment*, vol. 4, no. 10, pp. 681–692, 2011.

[127] F. Khalil, H. Wang, and J. Li, "Integrating markov model with clustering for predicting web page accesses," in *Proceeding of the 13th Australasian World Wide Web Conference (AusWeb07)*. AusWeb, 2007, pp. 63–74.

[128] F. Khalil, J. Li, and H. Wang, "An integrated model for next page access prediction," *IJ Knowledge and Web Intelligence*, vol. 1, no. 1/2, pp. 48–80, 2009.

[129] V. Hristidis, Y. Papakonstantinou, and L. Gravano, "-efficient ir-style keyword search over relational databases," in *Proceedings 2003 VLDB Conference*. Elsevier, 2003, pp. 850–861.

[130] W. H. Inmon, D. Strauss, and G. Neushloss, *DW 2.0: The architecture for the next generation of data warehousing*. Morgan Kaufmann, 2010.

[131] B. Devlin and L. D. Cote, *Data warehouse: from architecture to implementation*. Addison-Wesley Longman Publishing Co., Inc., 1996.

[132] A. Simitisis, P. Vassiliadis, S. Skiadopoulos, and T. Sellis, "Data warehouse refreshment," 2007.

[133] A. Amine, R. A. Daoud, and B. Bouikhalene, "Efficiency comparaison and evaluation between two etl extraction tools," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 3, no. 1, pp. 174–181, 2016.

[134] A. Simitsis, P. Vassiliadis, and T. K. Sellis, "Extraction-transformation-loading processes." 2005.

[135] H. Fang, "Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem," in *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE, 2015, pp. 820–824.

[136] B. Inmon, *Data Lake Architecture: Designing the Data Lake and Avoiding the Garbage Dump*. Technics Publications, 2016.

[137] R. Hai, S. Geisler, and C. Quix, "Constance: An intelligent data lake system," in *Proceedings of the 2016 International Conference on Management of Data*, ser. SIGMOD '16. New York, NY, USA: ACM, 2016, pp. 2097–2100. [Online]. Available: http://doi.acm.org/10.1145/2882903.2899389

[138] K. K. Jain *et al.*, *Textbook of personalized medicine*. Springer, 2009.

[139] Y. Zhang, M. Qiu, C.-W. Tsai, M. M. Hassan, and A. Alamri, "Health-cps: Healthcare cyber-physical system assisted by cloud and big data," *IEEE Systems Journal*, vol. 11, no. 1, pp. 88–95, 2017.

[140] H. Wang, Y. Zhang *et al.*, "Detection of motor imagery eeg signals employing naïve bayes based learning process," *Measurement*, vol. 86, pp. 148–158, 2016.

[141] M. M. Vernon, B. Ulicny, and D. Bennett, "An information provider's wish list for a next generation big data end-to-end information system." in *CIDR*, 2015.

[142] R. Kamal, M. A. Shah, A. Hanif, and J. Ahmad, "Real-time opinion mining of twitter data using spring xd and hadoop," in *Automation and Computing (ICAC), 2017 23rd International Conference on*. IEEE, 2017, pp. 1–4.

[143] N. Begum and A. A. Shankara, "Rectify and envision the server log data using apache flume," *International Journal For Technological Research In Engineering*, vol. 3, no. 9, 2016.

[144] J. Archenaa and E. M. Anita, "A survey of big data analytics in healthcare and government," *Procedia Computer Science*, vol. 50, pp. 408–413, 2015.

[145] S. Shaikh and D. Vora, "Yarn versus mapreduce—a comparative study," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2016, pp. 1294–1297.

[146] L. Sun, H. Wang, J. Soar, and C. Rong, "Purpose based access control for privacy protection in e-healthcare services," *Journal of Software*, vol. 7, no. 11, pp. 2443–2449, 2012.

[147] J. Li, H. Wang, H. Jin, and J. Yong, "Current developments of k-anonymous data releasing," *Electronic Journal of Health Informatics*, vol. 3, no. 1, p. 6, 2008.

[148] L. Sun, H. Wang, J. Yong, and G. Wu, "Semantic access control for cloud computing based on e-healthcare," in *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on*. IEEE, 2012, pp. 512–518.

[149] H. Wang, J. Cao, and Y. Zhang, "A flexible payment scheme and its role-based access control," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 425–436, 2005.

[150] V. Valliyappan and P. Singh, "Hap: Protecting the apache hadoop clusters with hadoop authentication process using kerberos," in *Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics*. Springer, 2016, pp. 151–161.

[151] S. Shaw, A. F. Vermeulen, A. Gupta, and D. Kjerrumgaard, "Hive security," in *Practical Hive*. Springer, 2016, pp. 233–243.

[152] J. Weston, "Support vector machine," *Tutorial, http://www. cs. columbia. edu/~ kathy/cs4701/documents/jason_svm_tutorial. pdf, accessed*, vol. 10, no. 0, pp. 0–5, 2014.

[153] S. Ghosh and S. K. Dubey, "Comparative analysis of k-means and fuzzy c-means algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 4, pp. 35–39, 2013.

[154] X. Sun, H. Wang, J. Li, and Y. Zhang, "Satisfying privacy requirements before data anonymization," *The Computer Journal*, vol. 55, no. 4, pp. 422–437, 2012.

[155] B. Strack, J. P. DeShazo, C. Gennings, J. L. Olmo, S. Ventura, K. J. Cios, and J. N. Clore, "Impact of HbA1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records," *BioMed research international*, vol. 2014, 2014.