UNIVERSITY OF CALGARY

Semantically Formalized Logging and Advanced Analytics for Enhanced Monitoring and Management of Large-scale Applications

by

M. Omair Shafiq

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE
DEGREE OF PHD IN COMPUTER SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA

APRIL, 2015

© M. Omair Shafiq 2015
Abstract

Monitoring and management of large scale applications has always been a complex task, especially because execution workflow and log (outcome from real-time execution) are modeled in a syntactic manner. This information is quite limited and requires manual interpretation, and hence makes the monitoring and management process slow, cumbersome and hard. We propose our solution by semantically (i.e., highly structured, formalized and expressive) modeling of the execution workflow and logs, and then we use Social Network Analysis, Classification, Clustering and Association Rule Mining based approaches to process the semantic information, to help in automating the monitoring and management process.

There have been several related efforts, but these solutions still could not achieve the goal effectively as described in this thesis. Two main reasons are: (1) they do not consider the correlation between the expressive modeling of execution workflow and logs, (2) the methods for processing (for monitoring) execution workflow and log methods are quite weak and limited.

To overcome the weaknesses of the approaches described in the literature, our proposed solution helps in automating the process of monitoring and management of large-scale distributed applications. We have designed and developed our unique hybrid approach of partially using formal semantics for logs description, as well as social network analysis and data mining tasks to be able to automatically interpret and process the highly structured information from the logs generated during the execution; this way our approach combines the best characteristics of both. Therefore, it helps in improving the automated monitoring and management of applications. Since the logs are generated
based on the execution workflow, our solution takes into account the correlation among both. Further the impact and usefulness of our solution have been demonstrated by applying it on real-life application scenario which was defined in consultation with our research collaborators from the industry. Our recent research publications and collaboration with industry have already shown promising results.
Research Publications


Advances in Social Networks Analysis and Mining (IEEE/ACM ASONAM 2014), 17-20 August 2014, Beijing, China.


Acknowledgements

I would like to acknowledge and dedicate special thanks to my supervisor Dr. Reda Alhajj for his continuous support, motivation and guidance throughout the PhD endeavor.

I would also like to thank my supervisory committee members of the PhD programme Dr. Jon G. Rokne and Dr. Yingxu Wang for their continuous feedback, support and guidance throughout the PhD endeavor.

I would further like to thank my mom, dad and all my family for being most gracious and generous to me throughout my PhD research. Thanks for all the love, kindness and all the support that you have given me to enable me to complete the PhD.

I would further like to thank several senior faculty members and colleagues for their help, support, brain-storming interesting ideas to make the research even more interesting.

I would further like to thank all the students I mentored to help them build their research projects and produce research publications. Thanks for being productive!

I would further like to thank all the faculty and staff in the Department of Computer Science and the University of Calgary for being kind and helpful to me during all the years I spent at University of Calgary.

I would further like to acknowledge Department of Computer Science at University of Calgary, Natural Sciences and Engineering Council of Canada (NSERC) Vanier CGS and Alberta Innovates - Technology Futures for funding my research.
Dedication

Dedicated to my Mom, Dad and all the family!!!
Table of Contents

Abstract .......................................................................................................................... ii
Research Publications .................................................................................................. iv
Acknowledgements ..................................................................................................... viii
Dedication ................................................................................................................... ix
Table of Contents ....................................................................................................... x
List of Tables ............................................................................................................... xii
List of Figures and Illustrations .................................................................................. xiv
List of Symbols, Abbreviations and Nomenclature ...................................................... xvi

CHAPTER 1: INTRODUCTION ....................................................................................... 1
  1.1 The Motivation ..................................................................................................... 2
  1.2 Problem Statement ............................................................................................. 6
  1.3 Overview of the Proposed Solution ................................................................... 9
  1.4 Contributions ..................................................................................................... 10
  1.5 The Research Methodology ............................................................................. 11
  1.6 Thesis Outline ................................................................................................... 15

CHAPTER 2: BACKGROUND AND RELATED WORK .................................................... 16
  2.1 Background and State-of-the-Art ......................................................................... 16
     2.1.1 Social Network Analysis and Mining ......................................................... 16
        2.1.1.1 Standard Calculation Techniques ......................................................... 17
        2.1.1.2 Social Network Analysis Softwares ...................................................... 18
     2.1.2 Web Services and Semantic Web Services .................................................. 21
        2.1.2.1 Central Concepts .............................................................................. 22
        2.1.2.2 Intended Scope ............................................................................... 24
        2.1.2.3 Approaches Overview ..................................................................... 26
  2.2 Related Work and Discussion ............................................................................. 35
     2.2.1 Survey of Related Work ............................................................................ 40
        2.2.1.1 Approaches using Data Mining .......................................................... 41
        2.2.1.2 Approaches using Semantics ............................................................ 42
        2.2.1.3 Approaches using mere structuring .................................................. 48
        2.2.1.4 Approaches focusing on combination of semantic formalism and data mining .................................................. 50
     2.2.2 Discussion on the Comparative Analysis ..................................................... 50

CHAPTER 3: SEMANTIC LOGGING ............................................................................... 56
  3.1 Semantic Model for Components and Logs ......................................................... 58
  3.2 A case-study application using Semantic Logging ............................................ 62

CHAPTER 4: FREQUENT PATTERN MINING USING SEMANTIC FP-GROWTH FOR EFFECTIVE WEB SERVICE RANKING .................................................. 86
  4.1 Introduction ........................................................................................................ 87
  4.2 Related Work ..................................................................................................... 90
  4.3 Proposed Solution .............................................................................................. 95
List of Tables

Table 1: Summarizing comparison of different approaches reviewed in related work .... 55
Table 2: Traditional logging as human readable logs .............................................. 66
Table 3: Semantically formalized logs for the initiated components ............................ 68
Table 4: Semantically formalized logs declaring components being started successfully ......................................................................................................................... 69
Table 5: Semantically formalized logs execution of user request ................................ 72
Table 6: Semantically formalized logs execution of user request ................................ 76
Table 7: Semantically formalized logs execution of user request .............................. 81
Table 8: Semantically formalized logs execution of user request .............................. 84
Table 9: Semantic FP-Tree Generation Algorithm .................................................. 102
Table 10: Semantic FP-Growth Algorithm ................................................................ 103
Table 11: Final Ranking Algorithm based on Association Rules ............................. 105
Table 12: Sample Semantically Formalized Log Events for a Discovery Request ...... 109
Table 13: Sample Association Rules Found .............................................................. 109
Table 14: Comparison based on Precision ............................................................... 115
Table 15: Comparison based on MAP ...................................................................... 115
Table 16: Formal description of Components and Log Events ............................. 135
Table 17: Outlook of the dataset used ................................................................. 141
Table 18: Accuracy in Classification Results ......................................................... 142
Table 19: A glimpse of Semantic Logs in use-case application ............................ 165
Table 20: Outlook of the dataset used ................................................................. 167
Table 21: Accuracy in Classifying Problem Types ............................................... 168
Table 22: Predicted Problem Types with accuracy .............................................. 169
Table 23: Stream Clustering Algorithm for Log Events ....................................... 180
Table 24: Outlook of the dataset used

Table 25: Comparison of Accuracy in Classification Results with or without Clustering
List of Figures and Illustrations

Figure 1: Effect of Formalization to Log Processing and Mining ...................................................... 5
Figure 2: Block diagram for the integrated framework to process semantic logs ............... 13
Figure 3: Top level elements of OWL-S ...................................................................................... 27
Figure 4: WSMO Components ...................................................................................................... 28
Figure 5: A layered structure of our proposed solution (semantically formalized logs for enhanced monitoring and management of software applications) .................. 57
Figure 6: Anatomy of component and its semantic description ................................................. 59
Figure 7: Semantic model for Log Events ...................................................................................... 62
Figure 8: Sample Bank Application Scenario ................................................................................ 63
Figure 9: Overall architecture for Ranking and Adaptation of Web Services using Association Rule Mining ........................................................................................................... 97
Figure 10: Model of Semantic Logs for Web Services .................................................................. 98
Figure 11: Semantic FP-Tree of items in Semantic Logs .............................................................. 104
Figure 12: Comparison of variance for number of Web Services .............................................. 112
Figure 13: Overall scenario for Enhanced Monitoring and Management of Large Scale Applications ......................................................................................................................... 127
Figure 14: Hierarchical representation in Semantic Logs, adapted from [80] .................... 128
Figure 15: Analysis of Problem Type classification ...................................................................... 143
Figure 16: Comparison of number of steps in fault detection ................................................... 145
Figure 17: Social Network Hexagon between Log Events, Components, Problem Space, Users, Functional Properties and Non Functional Properties ................ 159
Figure 18: Two dimensional Social Networks between Components, Functional and Non Functional Properties .................................................................................................................. 162
Figure 19: Overall architecture for the user-case application .................................................... 163
Figure 20: Comparison of number of steps in Problem detection .......................................... 170
Figure 21: Overall Integration of Classification, Association Rule Mining and Social Network Analysis over Clustering ........................................................................................................ 178

Figure 22: Analysis of frequency of recall versus value gained for different types of applications ........................................................................................................................................ 189
## List of Symbols, Abbreviations and Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ARM</td>
<td>Association Rule Mining</td>
</tr>
<tr>
<td>OWL</td>
<td>Ontology Web Language</td>
</tr>
<tr>
<td>OWL-S</td>
<td>Ontology Web Language for Services</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>SESA</td>
<td>Semantically Enabled Service Oriented Architecture</td>
</tr>
<tr>
<td>SLAB</td>
<td>Semantic Logging Application Block</td>
</tr>
<tr>
<td>SNA</td>
<td>Social Network Analysis</td>
</tr>
<tr>
<td>SOA</td>
<td>Service Oriented Architecture</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>SOS</td>
<td>Service Oriented Systems</td>
</tr>
<tr>
<td>SWS</td>
<td>Semantic Web Services</td>
</tr>
<tr>
<td>SWSF</td>
<td>Semantic Web Service Framework</td>
</tr>
<tr>
<td>SWSI</td>
<td>Semantic Web Service Initiative</td>
</tr>
<tr>
<td>SWSL</td>
<td>Semantic Web Service Language</td>
</tr>
<tr>
<td>SWSO</td>
<td>Semantic Web Service Ontology</td>
</tr>
<tr>
<td>UDDI</td>
<td>Universal Description Discovery and Integration</td>
</tr>
<tr>
<td>VO</td>
<td>Virtual Organization</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Service Description Language</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>WSML</td>
<td>Web Service Modeling Language</td>
</tr>
<tr>
<td>WSMO</td>
<td>Web Service Modeling Ontology</td>
</tr>
<tr>
<td>WSMX</td>
<td>Web Service Modeling eXecution environment</td>
</tr>
<tr>
<td>WSRF</td>
<td>Web Service Resource Framework</td>
</tr>
<tr>
<td>XML</td>
<td>The eXtensible Markup Language</td>
</tr>
</tbody>
</table>
Software applications are becoming increasingly complex and large with the increase in complexity of requirements. This makes the process of application monitoring and management a challenging task, especially when the requirement is to have automated monitoring and management of the application. Logging is a basic and fundamental part of an application design and development which allows an application to produce an execution log which is then used by software developers and administrators to monitor the execution and to debug as well as track any events during the application execution. It is to be noted that our work focuses on application level logs rather than system (operating system) level logs. Application level logs are generated by applications executing and contains application specific data only. Whereas, system level logs contains information specific to operating system during its execution of one or more applications.

The process of logging is often taken lightly and is not given the right attention as it deserves. A well-developed logging mechanism always helps in better monitoring and management of application execution. However, most of the logging mechanisms available today are quite limited. Thus, we argue the need for an effective and powerful technique capable of covering all the shortcomings in the existing methods. Such an approach is described in this thesis. The rest of this chapter further discusses the motivation, introduces problem statement, provides an overview of proposed solution, outlines contributions, and presents research methodology as well as thesis outline.
1.1 The Motivation

Some of the important limitations of the approaches described in the literature could be articulated as follows. The logs are syntactic, not well-structured and have very basic event correlation capability. Many solutions available so far require manual monitoring and management of applications, and hence make the monitoring and management process hard, cumbersome and inefficient. This applies especially to large and web-scale applications, where the process of monitoring and management of applications is even more difficult, complex and require maximum level of automation. The latest development in the area of web-scale applications is Service Oriented System (SOS) which has received considerable attention in the industry [1] as well as in the academia [2]. It is becoming increasingly important that SOS’s of the future should be able to flexibly adapt and deal with dynamic changes that may occur in distributed and large-scale environments like the Web. However, this is not possible with the use of traditional, syntactic and limited logging mechanisms and because of that the ability of monitoring and management mechanisms to sustain in a dynamically changing and open environment remains limited [3] [4]. Therefore, currently available middleware based solutions for Service-Oriented Systems, i.e., Enterprise Service Bus (ESB) solutions are limited to a closed environment and to a limited set of components with limited manual monitoring and management.

Our idea is to build a framework that allows applications, especially complex applications like middleware based solutions for services (often called Service Bus), to adapt to the dynamically changing environments and to automate the process of execution and monitoring. This calls for introducing highly structured, formalized
(semantic) descriptions [4] [5] [6] to the components, events and logs. Semantic descriptions for the components will help in precisely defining the descriptions of components; and the semantics will be modeled based on widely-accepted standards [3]. As a first step, we will build a model for semantically describing the components and logs. Secondly, we will build advanced log processing mechanism and engine to process semantically formalized logs as well as monitor the execution by applying different Social Network Analysis [7] and Data Mining techniques [8].

Social network based research requires expertise from anthropology, sociology, behavioral science, psychology, statistics, mathematics, computer science. Finding a balance between these domains of knowledge is by itself challenging and requires significant effort. We argue that the social network methodology is rich enough to successfully serve a variety of applications in software monitoring and management. The main theme is to analyze interactions in specific execution scenarios in order to discover key components, events and correlation among them, etc. The common trend applied in the literature is based on pair-wise links that reflect direct and explicitly expressed relationships between components. Though widely used, this approach reflects only a shallow utilization of known facts. The social network model constructed by our solution will be enriched by considering implicit links in addition to the explicit ones which could be properly achieved by employing data mining techniques to extract hidden relationships in the formalized and well-structured logs. The resultant model will be utilized for more effective monitoring and management. A detailed literature survey has been conducted as described in the related work section, and our successful research
publications and industry collaboration have already shown promising results which are described in the next chapters.

Web-scale applications are often composed of multiple components which may be hosted as self-contained services. It is also possible that an event at an application level may span across the execution of more than one component in sequential or parallel manner or a combination of both. In such a case, it is crucial to find out the right event and track it in all the application across multiple components or services, and hence bring the necessity that the logging information should be modeled precisely and with higher level of expressivity. Therefore, semantic annotations to components, execution workflow and logs have been proposed. Semantics can be utilized for finding, monitoring and managing the components required in the execution workflow. More precisely, highly structured, expressive and machine interpretable logs will be produced during the execution that will be used for monitoring and management of the application. Highly structured and expressive nature of the log information will also make the monitoring and management process automated. Once the logging is well-structured and formalized, it can be utilized by Social Network Analysis and Data Mining based techniques [9] [10] [11] to monitor the execution, track events and deduce interesting knowledge that can help in application monitoring and management. Some of our related work is available in [8] [12] [13].

There is a cost associated in incorporating highly structured and formalized logs into an application. However, this cost will be paid off when the process of monitoring and management of such an application will be simple, automated and effective. This is based on a simple formula that the more highly structured logs are, the easier it will be to
monitor and manage the application by processing the logs. Application designers and developers will be required to use the API (Application Programming Interface) that will be provided by our proposed solution, rather than using the traditional logging mechanisms. The more formalized the logs will be, the easier and the more effective it will be to process the logs and use it for application monitoring and management mechanisms. Today’s applications are mostly based on unstructured logs and hence require manual processing of logs by system administrators and developers. Since the logs in such applications are unstructured and based on syntactic standards, it is therefore harder to process such logs automatically and deduce new information.

Figure 1: Effect of Formalization to Log Processing and Mining
Our proposed solution brings the applications monitoring and management solution to a new level by allowing applications to have as much formalism in the logs as possible. Once the logs are formalized and well-structured, it becomes easier to process the logs automatically as well as more information can be deduced from the formalized logs by correlating, combining or splitting different application events in the logs. The ideal situation will be achieved by having the logs fully formalized and utilized up to maximum potential. In a real-life application using traditional logging mechanisms, it may not be possible to have all the logs fully structured and formalized. However, the more structured and formalized the logs are, the easier it will be for our solution to utilize it and perform effective monitoring of logs. Figure 1: Effect of Formalization to Log Processing and Mining depicts the correlation of formalism of logs with automated processing and mining of logs for application monitoring and management.

1.2 Problem Statement

In the process of monitoring and management of software applications, logging is a common practice in software applications. It enables applications to record execution foot-print in a serializable way such that it could be retrieved and analyzed anytime later. A log maintains application specific information about different steps in the execution of an application. A log is analyzed at a later stage to calculate any statistics or to debug, detect or track any possible problems, faults, exceptions or performance issues during application execution. When software applications used to be simple and straightforward, it was always easy to track an execution log to monitor such applications. However, today’s application monitoring and management tasks are based on manual
review of the execution log or on building basic parsing scripts that look for specific keywords about a particular event in the log to be monitored. Due to the lack of any standardization of building and processing execution logs, such log monitoring and mining approaches are quite limited. With the increase in the complexity of user requirements, software applications are also becoming increasingly complex and large. This makes the process of application monitoring through log analysis a difficult task. Traditional log processing procedures are manual and are not enough for efficient and effective application monitoring and management. There have been efforts, as discussed in the related work section, which try to automate the application monitoring and management procedure by building tools for parsing and analyzing application logs. However, due to the syntactic nature of log and the lack of any standardization in the process of building logs, such efforts cannot survive or stay limited with analysis.

Logging is one of the most important aspects that should be given considerable attention while designing and developing applications. A well-designed and developed logging mechanism will help in having an application monitoring process that can use such execution log to monitor application execution and to debug as well as track any events during application execution. The process of logging is often taken lightly and is not given the right attention as it deserves. A well-developed logging mechanism always helps in better monitoring and management of application execution. Most of the logging mechanisms available today are quite limited. Some of the important limitations, noted and discussed in the literature review and a survey section, could be articulated as follows. The logs produced are syntactic, not well-structured and have very basic event correlation capability. Because of such limitations in the log production mechanisms, the
monitoring solutions are also manual and hence make the process of monitoring and management of the applications a manual, hard, cumbersome as well as inefficient.

We tackle this problem by: (1) developing a semantic model for highly structured and formalized logs, and (2) employing data mining as well as social network analysis mechanisms to use such formalized and structured logs to carry out application monitoring and management in an effective manner. Such effective monitoring and management solutions are especially important for large and web-scale applications where applications are composed of multiple components and are often hosted as self-contained services [12]. In such systems, events at application level may also span from one to multiple components in a sequential or parallel manner which require tracking during the process of monitoring of applications. If the log is well-structured and formalized, it will be easier for the monitoring solutions to keep track of each of the events progressing across multiple components of the applications. Using semantics to formalize and structure logs will help in tracking and processing events in the logs and in finding further useful information, like determining failures for log events, i.e., which component or part of the application is causing failure. Semantics [2] help in producing highly structured, expressive and machine interpretable logs. It is produced during execution that later used for monitoring and management of applications. Such highly structured and expressive nature of the log information will make the monitoring and management process automated and will help in monitoring the application execution, in keeping track of events in the applications, and deduce interesting knowledge that can help in application monitoring and management.
Many solutions have been developed for monitoring and management of large-scale applications. However, issues of decoupling, dynamism and openness still form a challenge because existing solutions are limited due to the fact that the information is syntactically modeled. Fortunately, semantics have shown ability towards machine interpretable data. Thus, to contribute a novel framework capable of handling this emerging vital research area, our research questions or problem definition can be articulated as follows:

Question 1: How to formally model highly structured components, execution-workflow and logs?

Question 2: How to extract current and possibly new activities by mining event and activity logs?

Question 3: How to automate the monitoring and management of software applications using highly structured semantic workflow and logs?

1.3 Overview of the Proposed Solution

Our proposed solution includes building semantic models to formally describe components as well as events descriptions in the logs of application execution. This allows having more explicit information available with higher level of expressivity. The solution prescribes a well-defined model for semantically describing log events as well as a context in which the event being recorded has taken place. A semantic language has been used to formally write semantically formalized description of the components as well as events in the logs.
Advanced Social Network Analysis and Data Mining techniques are adapted and used to process highly structured information about components and logs. Once the information of event logs is available in a highly structured manner, it becomes easier for the analytical solutions to process the logs in order to use the information to have an enhanced and effective way to view the activities in the application execution.

Our proposed solution has been applied to a real-life application that shows how the currently available large scale applications may use our solution to formally describe its components as well as logs and use it for enhanced management and monitoring. We have also evaluated our work as demonstrated in our recent research publications.

1.4 Contributions

In this research, we have built solutions to tackle the research problems identified in the research questions enumerated above. The solution is strong and unique as it followed a hybrid approach to (1) make the information highly structured, formalized, and (2) use advanced data mining techniques to process the information, hence combine the best of both. Our proposed solution will solve the identified problem in a two-fold manner. First, it will provide semantic descriptions to the components and logs, so that information about components and logs will be available more explicitly and with higher level of expressivity. Second, it will use Social Network Analysis and Data Mining techniques to process the highly structured information about components and logs. It will allow the execution engine to manage the workflow of the Service Bus and to have more explicit information to precisely find out correlations in the process of monitoring and
management. Based on this, the proposed research will lead to the following contributions:

Contribution 1: Design models to formalize and describe events and other items in the logs.

Contribution 2: Use a formal language to semantically describe events and other items in the logs.

Contribution 3: Developing algorithms, techniques and hybrid analytical approaches to process and mine activities and events based on the semantically described information.

Contribution 4: Use the information from semantically formalizing logs and processing using advanced analytical solutions for enhanced monitoring and management mechanisms for software applications.

1.5 The Research Methodology

Our methodology has been developed after reviewing and analyzing existing and state-of-the-art solutions. We have reviewed several related work and related existing techniques about application monitoring and management using log processing and mining. We have categorized the related work techniques into three different categories: (1) approaches focusing on semantic formalism of logs, (2) approaches focusing on data mining based processing and analysis of logs, (3) approaches performing mere structuring of logs, and (4) approaches focusing on a combination of semantic formalism and data mining based processing and analysis of logs. We identified approaches related to all the four categories. After completing the literature survey and comprehensive analysis, we pointed out key deficiencies in the related work based on which we designed
our proposed solution. As per our literature survey and analysis of the related work, we have found out that most of the existing approaches are lacking the aspect of using semantic technologies along with data mining and analytics techniques. Our proposed solution takes into account this aspect and addresses the usage of semantically enriched logs by an integrated framework of data mining and analytics based solutions. Our proposed solution includes building semantic models to formally describe components as well as event descriptions in logs generated from application execution. This allows having more explicit information available with higher level of expressiveness.

A semantic language has been used to formally express semantically formalized description of components as well as events in the logs. The conceptual design of the proposed solution prescribes how semantics can be used to model component as well as log event descriptions semantically. A semantic language has been used to write semantic description of components as well as events based on the semantic model. Once the semantic descriptions of components and log events are available, an integrated framework consisting of data mining and analytics approaches is built and used to process such logs. Our proposed solution of semantic logs and the integrated framework to process such logs is generic. Therefore, we are able to customize and apply it to any application.

The integrated framework to process logs consists of different data mining and analytics techniques. We used Association Rule Mining and adapted it to process semantic logs as Semantic FP-tree (Frequent-Pattern-tree) and find out useful correlations between different elements of logs. Classification is used to classify, given a fault, error
or exception that may occur in the execution scenario, to find out what category or type this fault belongs to, in order to reduce and minimize the problem space.

Social Network Analysis based computation techniques are built to compute incomplete and missing information of logs. Clustering help in categorizing different types of events, being produced and recorded as logs during application execution, into clusters. These techniques, combined together, are used to process semantically formalized components and log events in execution log of software applications to find out and deduce important information regarding application monitoring and management during software execution. Figure 2: Block diagram for the integrated framework to process semantic logs depicts the block diagram which shows how the integrated framework processes events from logs.
As an example, we will apply our solution to a distributed and middleware based application. One possible option is Web Services Execution Environment (WSMX) [12]. In WSMX there are several components that are to be coordinated together in order to achieve Semantic Web Service discovery, selection, composition, and execution. WSMX has been designed based on Service Oriented Architecture (SOA), i.e., all the components of WSMX are deployed as services and then different services coordinate with each other. By applying our solution, different components in the WSMX system will be able to coordinate with each other based on semantically described components in a Service Bus. This will allow for automated and precise processing of work-flow, finding out exactly which components are to be used out of several available ones, i.e., to be invoked, as well as monitoring and management of events based on the highly structured logs. Other possibilities of applications may include large-scale and multi-component applications.

The evaluation criteria have also been planned both from quantitative and qualitative perspectives. We concentrate on how the semantic modeling of components descriptions and logs may improve the monitoring and management of the applications. The evaluation criteria may include factors like the level of ease in monitoring of components and event logs, flexibility in measuring latency, resource consumption, service availability, degree of preciseness of application execution as well as degree of flexibility in tracking and correlated events in different components to track the life cycle of an event during its execution in different components. The evaluation measures further include the amount of extra resources that may have to be invested in order to achieve the extended and automated monitoring and management of applications using the proposed
semantically formalized logging and processing mechanisms. The evaluation plans further include how the improvements can be achieved by using semantically-enabled coordination of components and services in terms of response time and how much complex execution workflows can be handled using semantics in user applications.

1.6 Thesis Outline

The rest of this thesis is organized as follows. Chapter 2 presents an overview to the relevant background information in the area of Social Network Analysis, Data Mining and Semantic Web as well as Semantic Web Services. It further discusses related approaches for monitoring and management of applications and categorizes them into different categories. For each of the approaches, it discusses advantages and disadvantages and identifies the gap in the currently available approaches. Chapter 3 introduces a detailed description of the proposed solution and presents a case study. Chapter 4 presents Frequent Pattern Mining using Semantic FP-Growth based on Semantic Logs with experimental results and evaluation. Chapter 5 presents adapted classification mechanism for Semantic Logs with experimental results and evaluation. Chapter 6 presents Social Network Analysis Hexagon based solution that helps in handling missing values and incomplete data with experimental results and evaluation. Chapter 7 presents the overall integration of the proposed solution with experimental results and evaluation. Chapter 8 discusses conclusions and future research directions followed by a list of published research publication and references.
CHAPTER 2: BACKGROUND AND RELATED WORK

Our proposed solution focuses on using the Social Network Analysis and Mining techniques to enable community-aware personalized Web search as well as Web Service discovery. Therefore, this chapter discusses the recent advancements in these areas. The subsections below describe the background and state-of-the-art in the areas of Social Network Analysis and Mining, Semantic Web and Web Services.

2.1 Background and State-of-the-Art

2.1.1 Social Network Analysis and Mining

Social Network Analysis allows modeling a real-world problem as a set of nodes (i.e., agents, organizations, or knowledge) and edges (relationships) from various types of input data (relational and non-relational), including mathematical models of social networks, and enable the analysis and visualization. A real-world problem is represented as a social network which is eventually a social structure that is made up of individuals (or organizations) called “nodes”. The nodes are connected to each other using “edges”. Semantics of the connections could be friendship, kinship, common interest, financial exchange, or any other kind of relationship.

Social network analysis views social relationships in terms of network theory consisting of nodes and ties. Nodes are the individual actors within the network, and ties are the relationships between the actors. The resulting graph-based structures are often very complex. There can be different kinds of ties between the nodes. The Social Network Analysis and Mining techniques view the problem as a graph, and involve
various calculation techniques in order to perform measurements from many different aspects. These techniques have been briefly described below:

2.1.1.1 Standard Calculation Techniques

Betweenness: is the extent to which a node lies between other nodes in the network. This measure takes into account the connectivity of the neighboring nodes, giving a higher value for nodes which bridge clusters. The measure reflects the number of persons who a person is connecting indirectly through their direct links.

Bridge: an edge is said to be a bridge if deleting it would cause its endpoints to lie in different components of a graph.

Centrality: This measure gives an indication of the social power of a node based on how well they “connect” to the overall network. “Betweenness”, “Closeness”, and “Degree” are all measures of centrality.

Centralization: is the difference between the numbers of links for each node divided by maximum possible sum of differences. A centralized network will have many of its links dispersed around one or a few nodes, while a decentralized network is one in which there is little variation between the numbers of links each node possesses.

Closeness: is the degree to which an individual is near all other individuals in a network, whether direct or indirect. It reflects the ability to access the information through the “grapevine” of the network members. Thus, closeness is the inverse of the sum of the shortest distances between each individual and every other person in the network.

Clustering coefficient: is a measure of the likelihood that two associates of a node are associates of each other. A higher clustering coefficient indicates a greater
“cliquishness”, and hence is a measure of the degree to which nodes in a graph tend to cluster together.

Degree: is the count of the number of ties to other actors in the network. It is defined as the number of ties that a node has. Degree is often interpreted in terms of the immediate risk of nodes for catching whatever is flowing through the network. If the network is directed (meaning that ties have direction), then there are two separate measures of degree centrality, i.e., indegree and outdegree. Indegree is a count of the number of ties directed to the node and Outdegree is the number of ties directed out of the node.

2.1.1.2 Social Network Analysis Softwares

A couple of software tools and libraries have been developed by the research community, that are used to mine, model, design, represent, analyze as well as visualize information in the form of a social network. A couple of related interesting tools are briefly described below:

Financial Network Analyzer (FNA) [http://www.financialnetworkanalysis.com/fna/] is an application for statistically analyzing financial networks using methods developed in network science and social network analysis. It differs from the other tools because of the fact that it builds networks from message (payments, trades, etc.) data and it is geared towards the analysis of network as a time series.

JUNG API: [http://jung.sourceforge.net/] is a comprehensive Java API and library that provides a common and extensible language for the modeling, analysis, and visualization of relational data. It supports a variety of graph types, graph elements of any type and with any properties, enables customizable visualizations, and includes algorithms from
graph theory, data mining, and social network analysis (e.g., clustering, decomposition, optimization, random graph generation, statistical analysis, distances, flows, and centrality). It has been used to analyze the networks in excess of 1 million nodes (although visualizations are currently more limited), and it is obviously limited only by the amount of memory allocated to Java.

ORA: [http://www.casos.cs.cmu.edu/projects/ora/](http://www.casos.cs.cmu.edu/projects/ora/) is a dynamic meta-network assessment and analysis tool developed by CASOS at Carnegie Mellon University. It is a dynamic meta-network assessment and analysis tool containing hundreds of social network, dynamic network metrics, trail metrics, procedures for grouping nodes, identifying local patterns, comparing and contrasting networks, groups, and individuals from a dynamic meta-network perspective. ORA has been used to examine how networks change through space and time, contains procedures for moving back and forth between trail data (e.g., who was where when) and network data (who is connected to whom, who is connected to where?), and has a variety of geo-spatial network metrics, and change detection techniques. It can handle multi-mode, multiplex, multi-level networks. It can identify key players, groups and vulnerabilities, model network changes over time, and can perform COA analysis. It has been tested with large networks. Distance based, algorithmic, and statistical procedures for comparing and contrasting networks are part of this toolkit.

Pajek: [http://pajek.imfm.si/doku.php](http://pajek.imfm.si/doku.php) is a widely used software for drawing networks. It has significant analytical capabilities, and can be used to calculate most centrality measures, identify structural holes, block-model, and so on. Macros can be recorded to perform repetitive tasks. Data can be sent directly to the tool in order to calculate the additional statistics.
SocNetV (Social Networks Visualizer) [http://socnetv.sourceforge.net/](http://socnetv.sourceforge.net/) is an open-source graphical application, developed in C++ and the cross-platform Qt toolkit. The user interface is friendly and simple, allowing the researcher to draw social networks or plain graphs by clicking on a canvas. SocNetV computes basic network properties (i.e. density, diameter, shortest path lengths), as well as more advanced statistics, such as centralities (i.e. closeness, betweenness, degree) and clustering coefficient, etc. Various layout algorithms are supported. For instance, nodes can be automatically positioned on circles or levels according to their betweenness centralities. Random networks and small world creation is also supported. SocNetV can handle any number of nodes, although with a speed penalty when nodes are more than 3000 nodes or the graph is quite dense (with many edges).

NetMiner: [http://www.netminer.com/](http://www.netminer.com/) is a software tool for exploratory analysis and visualization of network data. Its main focus is the analysis of large networks, comprehensive network measures and models, both exploratory as well as confirmatory analysis, interactive visual analytics, what-if network analysis, built-in statistical procedures and charts, full documentation, expressive network data model, facilities for data and workflow management, as well as user-friendliness.

Network Genie: [https://secure.networkgenie.com/](https://secure.networkgenie.com/) is used to: (1) design complete, egocentric, and hybrid social network surveys using a wide variety of survey question formats; (2) manage social network projects, including manage a collaborative team who has privileges defined by a project coordinator; (3) collect social network data using online forms; and (4) download and export data to the social network analysis program of your choice.
2.1.2 Web Services and Semantic Web Services

**Web Services** have added a new level of functionality to the current Web, by initiating the first step towards achieving seamless integration of distributed components. Nevertheless, current Web Service technologies only describe the syntactical aspects of a Web Service and, therefore, only provide a set of rigid services that cannot adapt to a changing environment without human intervention.

Web Services rely on three major technologies: SOAP [14], WSDL [15] and UDDI [16]. SOAP is a XML-based message format to exchange arbitrary XML data. WSDL is a XML-based description language for Web Services covering the interface description of web services with regard to the operations the service offers and the messages exchanged, i.e., defining how one can interact with the service. Finally, UDDI is a standard defining a data model and API for a web service repository, to enable discovery of services based on a classification, keywords in a human-readable description, and the respective WSDL interface of the service.

**Semantic Web Services** are building on to the Web services technology by describing various aspects of services using explicit, machine-understandable semantics that enable a certain degree of automation for various service-related tasks. In a nutshell, the work in the area of Semantic Web is being applied to Web Services in order to keep the intervention of the human user to a minimum. Semantic mark-up can be exploited to automate the tasks of discovering services, executing them, composing them and to enable seamless interoperation between them, thus providing what are also called intelligent Web Services.
The description of Web Services in a machine-understandable fashion is expected to have a great impact in the areas of e-Commerce and Enterprise Application Integration, as it can enable dynamic, scalable and reusable cooperation between different systems and organizations. These great potential benefits have led to the establishment of an important research area, both in the industry and the academia, to realize Semantic Web Services.

2.1.2.1 Central Concepts

The term service is a fundamental notion in both Web service and Semantic Web services areas. Moreover, it is becoming an important notion in everyday life and is shaping our society. Therefore, it is not surprising that this notion has become overloaded, i.e., having different meanings for different communities [17]. For example, in the business community, a service is seen as a business activity that often results in intangible outcomes or benefits [18] while in computer science the terms service and Web service are often regarded as interchangeable to describe a software entity accessible over the Internet. In our understanding, the notions related to Web services and Semantic Web services, namely service, Web service and Web service description are defined as in [19].

Service: A service is defined in [19] as a provision of value to a client in some domain. For example, if we consider a user who wants to book a ticket for an exhibition hall in Vienna on a given date, the service in this case will be the provision of such a ticket with the specified constraints. Such provision is independent of how the supplier and the provider interact, i.e., it does not matter whether the requester goes directly to the exhibition office or uses a Web site to book his ticket.
Web service: A Web service is defined in [19] as *a computational entity accessible over the Internet (using Web service standards and protocols)*. If we consider again the previous example, a Web service in this case will be a software component accessible via Web service standards, i.e., a Web service to request ticket booking. Thus, a Web service is an electronic means by which a client is able to request a specific service from a provider, but not the service itself. Therefore, the term *Web service* is to be understood as a means to request a service over the Internet, described using agreed upon standards.

Web service descriptions: Web service descriptions provide explicit, formal representation of different Web service aspects, including functional, behavioral and non-functional aspects. There are different levels of abstractions when it comes to Web service descriptions. A complete description of all the possible services a Web service can deliver seems rather unpractical and unrealistic, especially if we consider the big information volume and the dynamism of such service instances [20]. Therefore, Web service descriptions are an abstraction of the set of services that can be requested, i.e., a simpler, static characterization of the kind of services that can be accessed via the Web service.

A further analysis of the service domain shows that there are three important aspects when talking about services:

- **functional** – what a service can do
- **behavioral** – how to interact with the service in order to consume its functionality, or how other services are composed in order to provide the requested functionality
- **non-functional** – other aspects that are neither functional, nor behavioral and which often specify constraints over the first two

All of them can be seen as central concepts/notions usually associated with the term service. Additionally a set of tasks is usually associated with the term service. These include but are not limited to: service discovery, composition, mediation, negotiation, selection, execution and monitoring. They are part of the overall service lifecycle.

2.1.2.2 Intended Scope

Integrating different technologies requires first a survey of existing approaches and furthermore an analysis of benefits, boundaries and limitations of each technology. In the previous section we have provided a short description of current state of the art in SWS. This section provides an analysis of SWS technology in terms of benefits, boundaries and limitations.

Semantic Web services emerged as a promising technology for realizing distributed applications. They extend existing Web services by adding machine processable semantics to services, thus reducing the human intervention to a minimum. Among the many **benefits** of SWS, the three most prominent are: (1) *solution for integration* (2) *reusability* and (3) *automation*. By integration, we refer to Enterprise Application Integration (EAI) and Business to Business integration (B2B integration). The key factor in achieving effective integration of various applications (different interfaces, different implementations, and different behaviors) is to use a common set of standards. Semantic Web services and Web services provide such standards for description, communication and management in a uniform fashion. Furthermore, applications exposed as services through Semantic Web services and Web services are reusable pieces of functionality
which can be reused in any other scenarios. Finally, one of the greatest benefits of SWS is the increased degree of automation with respect to various service-related tasks (e.g., discovery, selection, composition, etc.) keeping human intervention at a minimum. As described in the previous section, this is achieved by making use of Semantic Web technologies which provide explicit, formal semantics for services.

The **boundaries** of the SWS technology are apriori shaped by the problems this technology addresses. They focus on solving business integration problems and knowledge representations in the context of service usage. However, there is a big overlap with the other two technologies we are trying to integrate. All these technologies have a common ground, more precisely semantic representation (e.g., RDF can be seen as a less expressive language, but uniformly shared between all these technologies). Moreover, with the adoption of Web Service Resource Framework (WSRF), the overlap between SWS and (Semantic) Grid actually increased.

Just like other technologies, SWS also has a set of **limitations**. Paradoxically these limitations are very much related to the aspects from which the benefits of SWS stem. For example, a basic requirement for realizing SWS vision is the provision of semantic descriptions for Web services. However, the semantic descriptions can be formalized in different ways by different people. Therefore, a new problem pops up, namely semantic mediation. Another limitation is the complexity. The problem is at what level of completeness the semantic descriptions are to be provided. A complete description of a service, even one capturing all the aspects related to the service, cannot be provided. There is as well an adaptation risk. This is due to the fact that providing semantic descriptions is not a trivial task. It requires people trained for such a task. It is not
surprising that currently the number of ontologies and semantic web service descriptions remains limited.

Based on this analysis, the scope of the SWS technology in GRISINO can be defined as a business infrastructure which exposes and manages functionalities as services. Using SWS technology, the application development of GRISINO applications becomes compliant with the SOA paradigm. Services are expected to use the computational and storage power provided by the Grid in order to provide their business value. Additionally, the information they are going to exchange will be semantically annotated, thereby having a meaningful content.

2.1.2.3 Approaches Overview

This section provides an overview of some of the most significant approaches in the Semantic Web services area, namely: OWL-S [21] WSMO [22], WSDL-S [23] and SWSF [24].

OWL-S

OWL-S is the Semantic Web Services effort of the DAML-program\(^1\), which is the major US-American Semantic Web research effort. OWL-S was the first approach towards an overall framework for describing Semantic Web Services, starting in 2001. OWL-S defines an ontology system for describing Web Services, using OWL as the description language. The top level elements of OWL-S are depicted in Figure 1: Effect of Formalization to Log Processing and Mining.

The OWL-S upper level ontology comprises three top-level concepts:

1. The *Service Profile* holds information for ‘service advertisement’ which is used
for Web Service Discovery. This is the name of the service, its provider and a 
natural language description of the service, as well as a black-box description of 
the Service (specifying the input, output, preconditions and effects (short: IOPE)).

2. The **Service Model** contains descriptive information about the functionality of a 
   service and its composition out of other services, described as a process. The 
   model defines three types of processes (atomic, simple, and composite processes), 
   where each construct is described by IOPEs, as in the Service Profile, with 
   optional conditions over these.

3. The **Service Grounding** gives details of how to access the service, mapping from 
   an abstract to a concrete specification for service usage. Although not restricted to 
   one grounding technology, WSDL is preferred for this.

---

Figure 3: Top level elements of OWL-S

---

1 www.daml.org
WSMO

The Web Service Modelling Ontology (WSMO) [22] aims to develop an overall framework for Semantic Web Services in order to support automated Web Service discovery, selection, composition, mediation, execution, monitoring, etc.

Figure 4: WSMO Components

WSMO defines four top-level notions related to Semantic Web Services, shown in Figure 4. Every WSMO component description may include non-functional properties, based on the Dublin Core Metadata Set [25] that is defined as a generic description model for information items. Two major design principles, inherited from WSMF [26] are applied in WSMO:

1. The principle of maximal de-coupling: all WSMO components are specified autonomously, independent of connection or interoperability with other components.

2. The principle of strong mediation: the connection and interplay between different
components is realized by Mediators that resolve possible occurring heterogeneities between the connected components.

WSMO specifies the following description elements and components:

1. **Ontologies**: are the key to link conceptual real world semantics defined and agreed upon by communities of users. Ontologies define a common agreed upon terminology by providing concepts and relationships among the set of concepts.

2. **Goals**: are descriptions of users’ desires. They represent the information space and state of the world after the execution of the service that would potentially satisfy the users’ desires.

3. **Web Services**: are descriptions of services that are provided, requested or agreed upon by service providers and requesters. The main elements of a service description are: a Capability describing the value the service can provide and one or more Interfaces in which the Choreography and the Orchestration of the service are described. The Choreography specifies how the service achieves its capability by interacting with its user - i.e., the communication with the user of the service; the Orchestration specifies how the service achieves its capability by making use of other services - i.e., the coordination of other services.

4. **Mediators**: are the components that realize the underlying principles of strong decoupling and mediation. Whenever WSMO components are to be connected, a Mediator connects these components and provides mediation in order to resolve possibly occurring heterogeneities. WSMO defines four types of Mediators: OO Mediators connect ontologies and import them as terminology definitions into
other components, GG Mediators for connecting Goals, WG Mediators connect Goals and Web Services, and WW Mediators connect Web Services.

The WSMO conceptual model is complemented by the Web Service Modeling Language (WSML) [27], a family of languages for describing various aspects of Semantic Web services based on WSMO conceptual model and its associated execution environment WSMX [28].

**WSDL-S**

WSDL-S [23] is another framework for Semantic Web services that follows a bottom-up approach for describing services. It proposes a mechanism to enhance the Web service functional descriptions represented in WSDL with semantics. For this purpose, the extensibility elements of WSDL are used. A set of annotations can be created to semantically describe the inputs, outputs and the operation of a Web service. The WSDL-S approach follows a set of principles: (1) it is based on existing Web services standards, more precisely WSDL, (2) annotations should be agnostic to the semantics representation language; WSDL-S does not specify what ontology language should be used, (3) Support annotation of XML Schema data type: because XML Schema is an important platform independent data definition format and it is desirable to reuse the existing interfaces described in XML, WSDL-S supports the annotation of XML Schemas. Finally, WSDL-S proposes five extensibility elements to be used when annotating the inputs, outputs and operations of Web services:

- *modelReference*: extension element that denotes a one-to-one mapping between schema elements and concepts from the ontology;
• **schemaMapping**: extension attribute that can be added to XSD elements or complex types to associate them with an ontology (used for one-to-many and many-to-one mappings);

• **precondition**: extension element (child of the *operation* element) used to point to a combination of complex expressions and conditions in the ontology, that have to hold before the execution of the Web service’s operation;

• **effect**: similar to precondition, with the difference that the conditions in the ontology have to hold after the execution of the Web service’s operation.

• **category**: extension attribute of the *interface* element which points to categorization information that can be used for instance when publishing the Web service.

Using these extensions one can create annotations for inputs, outputs and operation elements.

For **input/output elements annotation**, the *modelReference* and *schemaMapping* extensions are used. *ModelReference* is used when the input or output are simple types. *SchemaMapping* is used for complex types. In the latter case, two annotation approaches can be followed: (a) *bottom level annotation* which uses the *modelReference* attribute to annotate leaves of a tree base structure representing the complex type and (b) *top level annotation* which allows complex mappings to be specified between the XML element and the domain ontology.

For **operation elements annotation**, the *precondition* and *effect* extensions are used. A *precondition* represents a set of assertions that must hold before the execution of the operation. The precondition element might have different attributes: (1) *name*, which
uniquely identifies the precondition, (2) modelReference, which points to the semantic model part that defines the precondition, and (3) extension, which contains the precondition associated to the parent operation. The effect element is defined in a similar way as the precondition element. An effect defines the result of invoking a particular operation.

Finally, WSDL-S defines another extensible element called category. Using this element one can attach a category to a Web service. Many category elements can be used and, therefore, many categories can be attached to a service. This element is especially important in the context of service discovery.

**SWSF**

Semantic Web Services Framework (SWSF) [24] is one of the newest approaches for Semantic Web Services, being proposed and promoted by the Semantic Web Services Language Committee\(^2\) of the Semantic Web Services Initiative\(^3\). It is based on two major components: an ontology and the corresponding conceptual model with which Web services can be described, called *Semantic Web Services Ontology (SWSO)* and a language used to specify formal characterizations of Web services concepts and descriptions called *Semantic Web Services Language (SWSL)*.

Semantic Web Services Ontology (SWSO) presents a conceptual model for semantically describing Web services and an axiomatization, formal characterization of this model given in one of the two variants of SWSL: *SWSL-FOL* based on First Order

\(^2\) [http://www.daml.org/services/swsl/](http://www.daml.org/services/swsl/)

\(^3\) [http://www.swsi.org/](http://www.swsi.org/)
Logic or *SWSL-Rules* based on Logic programming. The resulting ontologies are called: *FLOWS* – *First-Order Logic Ontology for Web Services*, which relies on First Order Logic semantics, and *ROWS - Rule Ontology for Web Services*, which relies on Logic Programming semantics. Since both representations shared the same conceptual model we will focus our overview on FLOWS, the derivation of ROWS from FLOWS being straightforward.

The development of FLOWS ontology was influenced by the OWL-S ontology and the lessons learned from developing this ontology. Another fundamental aspect in the development of FLOWS is the provision of a rich behavioral process model based on Process Specification Language (PSL) [29]. FLOWS can be seen as an extension/refinement of OWL-S ontology with a special focus on providing interoperability or semantics to existing standards in the Web services area (e.g., BPEL, WSDL, etc.).

The FLOWS ontology consists of three major components:

1. **Service Descriptors:** These are used to provide basic descriptive information about the service in terms of non-functional meta-information and/or provenance information. This includes information like: name, textual description, version, etc. which are properties inherited from the OWL-S Profile. A Service Descriptor may also include the following full set of individual properties such as: *Service Name, Service Author, Service Contact Information, Service Contributor, Service Description, Service URL, Service Identifier, Service Version, Service Release Date, Service Language, Service Trust, Service Subject, Service Reliability* and *Service Cost*. 
2. **Process Model:** The Process Model is used to describe how the service works.

In accordance with the Web Services requirements, it extends the generic ontology for processes provided by the Process Specification Language (PSL) approach by adding two fundamental elements: (1) the structured notion of atomic process as found in OWL-S and (2) the infrastructure for specifying various forms of data flow. The core part of the PSL extended by FLOWS is called PSL Outer Core and the resulting FLOWS sub-ontology is called FLOWS-Core. Based on these extensions the FLOWS Process Model ontology can be regarded as a combination of six ontology modules namely: *FLOWS-Core, Control Constraints, Ordering Constraints, Occurrence Constraint, State Constraints* and *Exception Constraints*. They provide the terminology needed to specify activities in various ways, including sequential or nondeterministic order.

3. **Grounding:** The Grounding is used to link the semantic, abstract descriptions of the service provided in SWSO to detailed specifications of messages, protocols and so forth used by Web services. The grounding in SWSF is based on the OWL-S grounding.

In SWSF the Semantic Web Services Language (SWSL) is introduced to formally describe Web services concepts and descriptions of individual services. SWSL comes in two variants which are based on two well-known formalisms: First-Order Logic and Logic Programming. The two sub-languages are: *SWSL-FOL* and *SWSL-Rules*. Both languages were designed in compliance with Web principles such as: usage of URIs, integration with XML built-in types and XML-compatible namespaces and import
mechanisms. Both languages are layered languages where every layer includes a number of new concepts that enhance the modeling power of the language.

2.2 Related Work and Discussion

There is a lot of related work in the area of application monitoring and management. Most of the solutions attempt to utilize log information of software applications. Logs contain information about application execution. As requirements of users are increasing more and more, software applications are also becoming more and more complex. The days are gone when logging was used to record mere status of execution which used to be tracked by software developers and administrators in maintaining, monitoring and managing software applications. Software applications used to be simpler and straightforward in the old days and hence it was easier to track, analyze and use execution logs manually. The analysis of log used to be a mere parsing of logs to look for any specific keywords. Once the complexity of software applications started increasing with the increase in user requirements over the last few years, it became harder to track logs. Solutions based on small scripts started to emerge by parsing logs for detecting different keywords and patterns in the log data. Due to lack of any standardization of logging practice as well as any available best logging practices, monitoring and management of applications became a challenging task. Any time there is a change in the application, the scripts for parsing log files have to be changed. Moreover, for every application, different log structures are followed due to lack of standardization and hence different scripts have to be written to parse logs for such applications in order to perform application monitoring and management. We have noticed several efforts that have been made to
attempt to make the process of application monitoring and management automated and effective. However, due to lack of standardization and due to the syntactic and unstructured nature of logs, the process of application monitoring and management becomes manual, ineffective and cumbersome.

We argue that in addition to applying best practices and standardized guidelines during application development, these efforts should also be made for post application development and deployment, i.e., after such an application is developed and deployed for execution and is operational. Therefore, we envision a well-designed and developed logging mechanism to help in having application monitoring process that can use such execution log to monitor the application execution and to debug as well as track any events during the application execution. Having the process of logging not taken lightly and given the right attention that it should be given will help in advancing the process of application monitoring and management. We have carried out a detailed survey of the related work and also we have carried out a comparative analysis. We found out that most of the related works and methods for application monitoring and management focus on logs that are syntactic, not well-structured and have very basic event correlation capability. Due to these limitations in log production mechanisms of application, the related works stay limited and hence make the process of monitoring and management of applications manual, hard, cumbersome as well as inefficient.

After completing out our analysis of the related work, we identified the above mentioned gaps in the related work and devised our proposed solution accordingly. Our proposed solution tackles the identified gaps and lacking in the related work by employing a hybrid approach. First, it develops a formal semantic model for logs in order
to have highly structured and formalized logs, and second it uses the adapted analytical solutions, including classification, association rule mining, social network analysis and clustering to process such highly structured and formalized logs in order to carry out application monitoring and management in an effective manner. We believe that such effective monitoring and management solutions are required for large and web-scale applications where applications are composed of multiple components and are often hosted as self-contained services. In such type of applications, events may span from one to multiple components in a sequential or parallel manner which require tracking during the process of monitoring the applications.

We use semantics to formalize and structure logs. This helps in tracking and processing events in the logs and in finding further useful information, like determining failures for log events, which component or part of the application is causing failure. Semantics help in producing formalized, highly structured and machine interpretable logs. It is produced during the process of application execution which can be used at a later stage for monitoring and management of applications. Advanced and adapted analytical solutions make use of highly structured and expressive logs to extract and deduce maximum information which helps in automated, effective and enhanced application monitoring and management.

We have proposed a model for semantically describing logs as well as components of the applications. Semantic description of components is also an important part of modeling log events because it helps in keeping track of log events across different components. Our proposed solution is unique because it uniquely integrates formal semantics with data mining techniques to effectively process such information; hence
combines the best of both. Data mining and social network analysis techniques require data to be structured and formalized [8], and therefore our proposed solution of formalizing and structuring the application execution information will help in boosting the performance of such processing techniques.

In order to gain such advantage in application monitoring and management, there is cost associated to bring highly structured and formalized logs into applications. However, such cost is paid once data mining and social network analysis based techniques are applied to such structured and formalized logs during the process of application monitoring and management. Semantically formalized logs make the process of application monitoring and management simpler, easier and effective. We describe this cost and benefit of using formalized logs by a simple formula that the higher the formalized and structured logs are, the easier it is to monitoring such applications. Software designers and developers have to use our Application Programming Interface (API) for enabling formalized logs, similar to the usage practice of currently available traditional logging APIs like Log4J (http://logging.apache.org/).

Most of today’s applications are hardly using any formalized and structured logging mechanisms. Such logs are not well-structured and therefore, it is not possible to automatically process such logs to keep track of events during the application execution process. Also, it is not possible to deduce any further interesting information about events in applications. Traditional logging methods produce logs that are human-readable and not machine readable. Therefore, it often requires manual efforts to keep track of events by going through the logs. We argue that application execution logs should not only be human-readable; they should be also machine-readable. Therefore, the usage of semantics
in our proposed solution to allow formalized and well-structured logs will turn application monitoring and management into an automated and effective process. Currently, the available logging machines are not formalized and are rather unstructured. We have discussed a use-case in this thesis that describes how the unstructured logs normally look like and how the lack of standardization in logging practices makes the process of monitoring and managing applications a hard and cumbersome task.

Recently, there have been a few approaches, discussed in the related work section, that realize such problem of unstructured logs and the lack of any standardization in the logging practices and the attempt to provide structured logging practices. Such approaches do improve the process of application monitoring and management to a limited extent. However, we have found that such approaches are still lacking many important aspects that we have addressed in our proposed solution. Such aspects include building models for logs to be better used by data mining and social network analysis tasks, keeping the models for logs and components inline to correlate and track log events across multiple components as well as using formalized and structured logs.

Ideally, having formalized and highly structured logs makes the process of monitoring and management highly automated as well as effective. However, in real-life it may not be the case. It may not always be possible for all applications to have highly structured and formalized logs. Therefore, we have kept our proposed solution simple and flexible in a way that it is up to software designers and developers to decide on the level of information that should be kept in the execution logs. Depending upon application nature, circumstances and monitoring requirements, our proposed solution allows adjusting the level of information to be stored in logs during application execution. The
level of formalism in logs and the level of depth of the information stored in logs depend upon the nature of the application to be monitored. Basic applications and utilities which are barely used do not require extensive monitoring and management, and hence a basic level of formalism and keeping only important and key information in the logs would be enough to enable monitoring such applications. However, in such case, the monitoring and management of such applications would still be basic and may still require manual effort.

Applications that are frequently used and are still not important may use a higher level for formalism and a deeper level of information in logs which may be used by monitoring solutions to perform at least semi-automated monitoring and management of such applications. For critical applications that require extensive monitoring and management, an even higher level of formalism and even deeper level of information in logs have to be maintained so that such information could be used for highly automated and effective monitoring and management of such applications.

2.2.1 Survey of Related Work

In this section, we discuss related work in the area of application monitoring and management using mining of logs and other related execution data. The area of enhanced and automated monitoring and management of large-scale applications received considerable attention. Such work spans from monitoring of stand-alone applications to the monitoring of Web-scale applications, middleware solutions as well as Web Services [30] [31] [32]. We describe each of the related approaches and perform a comparison based on two aspects, i.e., structuring and formalization of logs, and the usage of any data
mining or analytics techniques to process the structured and formalized logs during application monitoring and management. We categorized the current available approaches into four different sub-categories: (1) approaches focusing on semantic formalism of logs, (2) approaches focusing on data mining based processing and analysis of logs, (3) approaches performing mere structuring of logs, and (4) approaches focusing on the combination of semantic formalism as well data mining based processing and analysis of logs. Below we discuss each sub-category and we conduct a comparison within the subcategory.

2.2.1.1 Approaches using Data Mining

This section presents related approaches that provide data mining based analytical solutions to process logs. However, the structuring and formalism of logs is a very crucial step towards using them in processing logs using data mining based analytics; such data mining based techniques are dependent on using concrete and precise information.

In [33], the authors propose to extract semantic relationships using logs of queries. The authors studied a large query log from millions of queries that were executed; they extracted semantic relations that were implicitly captured in the actions of the users submitting certain queries and then clicking on answers out of the given options. The authors define a cover graph for the queries and the answers that are clicked by the users and proposed an approach to analyze the graph in order to find out semantic relations from the queries and the answers that were given as output. The main benefit obtained from the approach proposed in [33] is to compute answers more efficiently by using information from similar queries. We have realized that the authors do not attempt to
provide any formalism to the logs. The approach only attempts to provide a mere structuring of logs. Such structuring of logs is then used to create cover graphs that are then used to find out semantic relations among queries and answers. It helps in computing answers faster based on historical search results. This approach is limited to query answering only.

In [34], the authors attempt to discover the related queries using the association rule mining approach. Similar to the Apriori algorithm, the log of query execution is viewed as a set of transactions. Each of the transactions represents a session in which a single user submits a sequence of related queries in a time interval. This proposed approach shows good results, but it is unable to handle two issues, i.e., it is not possible to find out sessions of queries that belong to the same search process. Secondly, the most interesting related queries cannot be discovered, submitted by different users, since the support of a rule increases only if its queries appear in the same query sessions (the queries that are submitted together by the same users). In this approach, the authors did not attempt to structure or formalize the logs, but only applied an algorithm similar to Apriori. The execution log is taken as a set of transactions and related queries are discovered to try computing their answers faster by using answers computed for similar set of queries. Again, this approach is also limited to query answering only.

2.2.1.2 Approaches using Semantics

This section presents approaches that attempt to bring formalism to logs by using semantic modeling and annotation. Semantically enabled enrichment of logs is an
important step that allows for inference of new as well as non-obvious patterns in the logs that could be helpful in the process of application monitoring and management.

In [35], the authors present a framework for semantic logging. The target is to enable structured information logging in an agent-based distributed system for chemical incident response purpose. The logging framework is called semantic because it allows having semantic interpretation of logs according to the relationships defined between different but related logging events. The authors use this approach to help in reconstructing the order of events that occurred during the response to a particular incident. It further helps in giving a detailed overview of the system execution trace, as well as of decisions taken by agents at various decision points during the incident management workflow. The semantic logs are used to help experts in analyzing and explaining system actions and hence improving system response to future possible incidents. This also helps in training stakeholders by setting the system to run replay-like simulations of any past incident management workflows. In this approach, authors attempt to provide semantic annotations to logs in order to represent semantic links between log events. The approach does not employ any data mining or analytics techniques to mine the semantic logs. This approach has been built in the context of distributed multi agent systems.

Another interesting proposal enables semantic logging using Resource Description Framework (RDF) [36]. The authors proposed to use log files as a data source for the purpose of evaluating as well as diagnosing performance and characteristics of systems that are distributed in nature. They argue that logs with various types of formats complicate the process of developing tools for the overall analysis of the system. They propose to use RDF in order to provide an infrastructure which can be used as a
repository for different types of logs formats. Such logs could then be searched and analyzed for gaining further understanding of the system of interest. The authors provide a vocabulary based on which a common log format will be achieved. This approach provides a basic formalism to logs using Resource Description Framework (RDF). RDF is based on a simple but useful data model which is used to model the resources over the Web like a subject-predicate-object expressions. However, we have found that log events need more levels of expressivity to accommodate detailed application specific information about the logs. Moreover, this approach also does not employ any data mining or analytics approaches to process the logs, but only tries to provide a common log format.

In [37], the authors describe a product called smartFIX which is a product portfolio for knowledge based extraction of data from any type of document format. The proposed approach determines document type as well as extracts all the relevant data for respective business process in an automated manner. It helps users to interpret document data. The proposed solution is based on using semantic technologies that enable to log the execution in a semantically formalized manner. The log contains all process relevant information enabling the explanation facility to generate customized and understandable explanations. This approach provides a basic level of semantic logging facility and also provides basic data mining and analysis approaches to process and mine the logs. However, this approach is limited to document analysis only.

In addition to these solutions, several semantics based solutions have been proposed to enable automated Web Service execution, including discovery, selection, composition as well as invocation. These approaches do not focus on semantic logging in particular; they
are useful for us to review as the nature of the problem is very similar, i.e., these approaches semantically formalize descriptions of Web Services in order to enable automated discovery, selection, composition and execution. Similarly we are seeking to semantically formalize the logging in order to enable extensive analysis of logs and hence enhanced and automated monitoring of applications.

Web Ontology Language for Services (OWL-S) [21] [38] [39], as a part of the DAML Program [40], proposes a set of ontologies based on OWL in order to describe different possible aspects of a Semantic Web Service [4]. There are three different core ontologies, namely: (1) service profile, (2) service model and (3) grounding. Service profile prescribes what a service does. The service model describes how a service works. Whereas, service grounding prescribes how to access the service using detailed specifications of message formats, protocols and so forth (normally expressed in WSDL). All of these core ontologies are linked to the top-level concept Service, which serves as organization point of reference for declaration of Web services. This approach provides a foundation to semantically model Web Services descriptions, however, it does not provide any data mining or analysis approaches to process or mine the semantically enabled Web Services descriptions.

Web Service Modeling Framework (WSMF) [26] was proposed as a fully-fledged framework to model Semantic Web Services [4]. It aims to attain full potential of the Web. From the collection of information into the distributed device of computation, this framework prescribes two complementary principles (maximal de-coupling and scalable mediation [41] [42]) and four key elements (i.e., Ontology, Goal, Web service and Mediator) in order to model any aspects related to the services’ definition and usage. To
finally realize the framework, a set of corresponding technologies have been
developed, namely, the modeling ontology called Web Service Modeling Ontology
(WSMO) [3] [5] [43] [44], the description language called Web Service Modeling
Language (WSML) [5] [45], and the execution environment called Web Service
Execution Environment (WSMX) [4] [46] [47]. It includes a basic micro-kernel [48] and
grounding support [49] with existing Web Service standards. This approach provides a
comprehensive model and semantic language to semantically describe Web Services as
well as user queries as Goals. This approach also provides a comprehensive framework
(WSMX) to process the semantic description of Web Services. No particular data mining
and analysis approach has been addressed in this framework, however, the framework is
flexible to accommodate data mining or analysis techniques to be used to process the
semantic descriptions that may help in dynamically discovering, selecting, composing or
invoking Web Services.

Web Service Description Language - Semantics (WSDL-S) has been proposed and
developed at LSDIS Lab with a mechanism to enrich WSDL with semantics, in particular
focusing on the functional descriptions of services. Based on WSDL, WSDL-S has the
advantage of carrying semantics built based on existing Web services, while it does not
have to dictate a specific language for semantic description [12]. This approach attempts
to provide light-weight semantic annotations to Web Services descriptions only and also
provides a high level and abstract framework for processing semantic descriptions for
Web Services, but does not take into account any data mining or analysis approaches to
process semantic annotations to Web Services.
Semantic Web Services Framework (SWSF) [4] is a specification proposed by the SWSL Committee as a part of the Semantic Web Service Initiative (SWSI) [50]. SWSF has proposed a conceptual model which is called Semantic Web Service Ontology (SWSO) and a relevant Semantic Web Service Language (SWSL). SWSO [50] has been influenced by OWL-S and adopted its three core ontologies, namely service profile, model and grounding. The rich behavioral process model based on PSL is the key contribution of SWSO. With these extensions, more powerful descriptions and reasoning on Web services can be supported by SWSO [50]. SWSL has two subsets, SWSL-FOL [50] and SWSL-Rules [52] that support first-order logic and logic programming, respectively. This approach tries to provide an umbrella framework for all the different approaches that exist to provide semantic descriptions to Web Services. The framework is flexible and accommodates different approaches to semantically model semantic descriptions of Web Services. The approach also provides a high level and abstract framework for processing semantic descriptions but does not discuss any data mining or analysis related approaches to actually process semantic descriptions of Web Services.

In [53], the authors present a design of a personalized presentation layer architecture for a Web-based information system. It is based on a set of interconnected software components that are implemented as autonomous software tools for personalization, presentation, and user modeling to support features like navigation support and different views on the presented data, data acquisition and evaluation of user characteristics, user adaptation as well as personalization. The authors create domain ontology. Content of the ontology and the characteristics of individual users are created as well as updated by analyzing logs of users using the application. After analyzing the logs, events in the logs
are processed asynchronously and the user model is updated with newly identified characteristics of user. This approach does not focus on structuring or formalizing the logs. It only uses OWL based ontology to capture user characteristics which are used in modeling and storing the logs. These recorded log events are then analyzed and processed using aggregation and soft-clustering techniques based on the semantic description of log events.

2.2.1.3 Approaches using mere structuring

This sub-section presents related approaches that were found to provide mere structuring of logs only. Structuring of logs is an important step towards application monitoring and management but still preliminary as such structuring may ease the process of executing and processing the logs while semantic formalism to logs brings us a step ahead, i.e., to deduce and correlate information about different events in logs.

Semantic Logging Application Block (SLAB) [54] is a recent approach that is based on the Windows operating system to perform Event Tracing for Windows. It stores information about events during the execution of the operating system like timestamp, event id, keywords, event source, task, etc. It controls the process of application monitoring and management. This control is based on patterns that are commonly encountered during application execution and practices that are taken in order to handle and monitor the commonly occurring patterns. This approach provides structuring to logs by storing timestamps, events related identifiers as well as a set of keywords. During the process of monitoring after the execution occurs and the log is produced, the approach provides an underlying infrastructure to extract events related information and uses it in
the analysis. It does not attempt to semantically formalize events in logs and therefore it is limited in terms of the level of expressivity of information about events in the logs. On the other hand, this approach does not provide any concrete analytical solution to analyze data about events obtained from logs and hence analysis of logs is left weak and on the discretion of users to interpret logs.

Approaches like Adiscon LogAnalyzer [55] and WebLog Expert [56] provide practical tools to analyze log data. However, these approaches do not make any attempt in structuring the logs. Also the data mining and analysis techniques employed to mine the log data are also naïve and only provide basic performance reports about software execution. SysLog Monitor [57] provides a rule based method to access and read the logs, but still does not make any attempt to structure or formalize logs. Also, it applies basic rule based monitoring techniques to generate reports like host system performance analysis, identifying faults in execution and identifying different types of events in application execution.

GitHub Log-analyzer [58], Retrospective Log Viewer Software [59] and XpoLog Log Analysis Platform [61] are similar tools that do not attempt to provide structuring or formalism to logs. These tools make use of basic statistical techniques to summarize and produce execution summary reports or reports related to any specific event in application execution. There are many other tools available in the market that provide similar functionality, but we mentioned the ones that are widely used.

CrazyEgg [60] is a very recent tool that provides a toolset which helps in identifying user patterns using websites. This approach does not make any attempt to provide structuring or formalism to logs. It only uses some basic visualization techniques with
statistical techniques to visualize user clicks on websites in order to identify the most popular areas of a given page, to see which parts of web pages are working and which ones are not.

2.2.1.4 Approaches focusing on combination of semantic formalism and data mining

Splunk (www.splunk.com) is another comprehensive framework for semantically logging and mining information from application execution. It performs enhanced monitoring and management of applications. The authors argue that logs (especially unorganized logs) can be a hassle to deal with as there is no real structure, nor any standardized format. Such logs may become useful once stored with proper structure. Analyzing such logs may help in finding problems, getting more insight information about IT infrastructure for an enterprise, behavior of users, and identifying potential problems. Splunk uses the terminology of semantic logging, but it only provides a way to structure the logs using basic structuring techniques only. On the other hand, it provides several data mining and analytics approaches to process structured log data. It uses the terminology of semantic logging for logging the semantic level of application events only, which may be web clicks, financial trades, transaction failures, etc.

2.2.2 Discussion on the Comparative Analysis

Given below is a table that summarizes our comparison and review of the available approaches for providing semantic modeling to logs, components and Web Services descriptions as well as for providing data mining or similar data analysis related techniques to process the semantic description.
All these approaches have made significant efforts towards automated execution and monitoring but are limited in various aspects. For example, all the Semantic Web Service solutions that have been discussed, e.g., [40] [5] [23] [12], are too focused on Web Service descriptions and user goal descriptions; hence do not specify issues related to execution monitoring. Other approaches also have been focused on specific logging issues like document analysis [37] or query answering [33] [34], and hence are limited. Other approaches like Splunk.com and semantic logging using RDF [36] have been too basic and limited in terms of the formal semantics used to semantically model logs. This limits the expressivity of log events, relationships among log events and constraints in the execution logs. Some of the approaches only focus on providing semantic modeling to logs [35] [36], and do not provide any data mining or related analysis solutions. Some approaches like Splunk.com focus only on mining the logs and do not focus on the semantic modeling of logs at a deeper level so that implicit information in the log events could be deduced and used for application monitoring and management. Similarly, approaches like those described in [55] [56] [57] [58] [59] [60] [61] do not provide any solution for structuring or formalizing logs, but rather focus only on using basic statistical approaches to generate summary and performance reports for application execution. SysLog Monitor [55] uses rules for logs, not for structuring, but only for accessing the logs and builds a rule based engine to generate performance reports and events summarization in application execution. The approach in [60] is a latest toolset that is offered to track the behavior of users using websites. This toolset uses extensive visualization techniques to identify different areas of web pages visited by users. It uses its own way to store logs, but does not make any attempt to specify structuring and
formalism in logs and to use such structuring or formalism in processing logs. XpoLog Log Analysis Platform [61] also does not make any attempt to provide structuring or formalism to logs. However, it offers a platform that provides extensive visual facilities for monitoring application execution by producing application execution reports summarizing performance and any faults.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Level of Formalism for Logs</th>
<th>Data Mining or similar approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueryLog [33]</td>
<td>Does not provide any level of formalism</td>
<td>Usage of cover graphs for mining</td>
</tr>
<tr>
<td>Mining of related Queries [34]</td>
<td>Does not provide any level of formalism</td>
<td>Usage of Association Rule Mining techniques</td>
</tr>
<tr>
<td>Distributed MAS Logging [35]</td>
<td>Proposes a semantic logging framework to represent semantic links between log events</td>
<td>Does not apply any Data Mining techniques on semantic logs. Only interprets semantic links between log events</td>
</tr>
<tr>
<td>RDF based Logging [36]</td>
<td>Basic formalism to logs provided using RDF only.</td>
<td>Does not provide Data Mining techniques. Only provides a common log format.</td>
</tr>
<tr>
<td>Splunk [86]</td>
<td>Structures logs using basic structuring techniques only.</td>
<td>Applies data mining and analytics techniques to structured log data</td>
</tr>
<tr>
<td>smartFIX [37]</td>
<td>Provides semantic logging</td>
<td>Basic data mining and analysis techniques limited to document analysis only</td>
</tr>
<tr>
<td><strong>OWL-S [21]</strong></td>
<td>Focuses on providing semantic annotations to Web Services only</td>
<td>No data mining or analysis techniques prescribed</td>
</tr>
<tr>
<td><strong>WSMO [3], WSMF [26], WSML [5], WSMX [28]</strong></td>
<td>Very comprehensive model and semantic language for describing Web Services and user queries (Goals)</td>
<td>A comprehensive and complete framework for analyzing semantically enabled Web Services but does not focus on log events</td>
</tr>
<tr>
<td><strong>SWSF [4]</strong></td>
<td>A generic framework for providing semantic descriptions to Web Services only</td>
<td>Provides a high level abstract framework for processing semantic descriptions for Web Services, but does not take into account log events information</td>
</tr>
<tr>
<td><strong>WSDL-S [23]</strong></td>
<td>Provides light-weight semantic annotations to Web Services descriptions only</td>
<td>Provides a high level and abstract framework for processing semantic annotations and Web Services descriptions only</td>
</tr>
<tr>
<td><strong>SemanticLog [53]</strong></td>
<td>Uses an OWL Ontology to provide semantic annotations to logs</td>
<td>Uses aggregation and soft-clustering techniques to semantic description to process log events</td>
</tr>
<tr>
<td><strong>Adiscon LogAnalyzer [55]</strong></td>
<td>Does not provide any structuring and formalism to logs</td>
<td>Uses basic statistical techniques to capture events, generates status reports and other related performance statistics</td>
</tr>
<tr>
<td><strong>WebLog Expert [56]</strong></td>
<td>Does not provide any structuring and formalism to logs</td>
<td>Uses basic statistical techniques to capture events and user activity as well as access statistics</td>
</tr>
<tr>
<td>Tool</td>
<td>Features</td>
<td>Techniques Used</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>Syslog Monitor [57]</td>
<td>Provides rule-based method to access and read logs, but does not structure or formalize logs</td>
<td>Applies basic rule based mining techniques for generating reports based on host, data, severity, group and by event type</td>
</tr>
<tr>
<td>GitHub Log-analyzer [58]</td>
<td>Does not provide any structuring and formalism to logs</td>
<td>Uses basic statistical techniques to summarize and produce execution performance reports</td>
</tr>
<tr>
<td>Retrospective Log Viewer [59]</td>
<td>Does not provide any structuring and formalism to logs</td>
<td>Uses basic statistical techniques to find specific events and data in log data</td>
</tr>
<tr>
<td>CrazyEgg Log Analyzer [60]</td>
<td>Uses its own basic format for logs. Does not provide any structuring and formalism to logs</td>
<td>Uses visualization techniques with statistical techniques to visualize user clicks on websites to identify the most popular areas of a given page, to see which parts of web pages work and which ones do not</td>
</tr>
<tr>
<td>XpoLog Log Analysis Platform [61]</td>
<td>Does not provide any structuring and formalism to logs</td>
<td>Provides a platform for application monitoring using log analysis. Uses basic statistical techniques for generating performance reports, fault reports (whether pre-defined or not)</td>
</tr>
<tr>
<td>SLAB [54]</td>
<td>Stores specific characteristics for Events in the log but mainly based on Keywords only. Does not provide semantically</td>
<td>Does not provide any specific analytical solution for analyzing events in logs</td>
</tr>
<tr>
<td>formalized logs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 1: Summarizing comparison of different approaches reviewed in related work**

Our proposed solution takes into account higher formal semantics used in Semantic Web Services and uses it in a generic way to enable semantically formalized logging that helps in enhanced monitoring and management of large-scale and complex applications. Once semantically formalized and structured logs are enabled, our solution further provides tailored data mining and social network analysis based approaches which are essential to process the semantic logs. This uses explicit information to deduce implicit information that empowers our vision of effective application monitoring and management using our hybrid approach.
CHAPTER 3: SEMANTIC LOGGING

Our proposed solution of semantically formalized logging for enhanced monitoring and management of software applications is based on building semantic models to formally describe components as well as events descriptions in execution logs of software applications. This allows having more explicit information available with higher level of expressiveness. The solution prescribes well-defined vocabularies for modeling event status as well as the context in which the event being recorded has taken place. A semantic language has been used to formally express the semantically formalized description of components as well as the events in the logs. Advanced Social Network Analysis and Data Mining techniques have been used and even developed further in order to process highly structured information about components and logs. With the information of event logs being available in a highly structured manner, it becomes easier for the monitoring solutions to process such logs in order to have an enhanced and effective way to view the activities in the application execution.

Figure 5: A layered structure of our proposed solution (semantically formalized logs for enhanced monitoring and management of software applications) depicts a layered structure of our proposed solution. It shows that for a software application to be monitored should be bundled up with semantically formalized layer on the top of the application layer, as well as log processing mechanisms at the bottom of the application layer. Semantic logging API will be the interaction point for all the layers, i.e., semantic formalism layers, application layer as well as the layers involving log processing mechanisms. The formal meta-model of logs prescribes an overall template for the semantic descriptions of events and components involved in the execution of the software
applications to be monitored. Semantic descriptions for events and components in the log are then written using semantic models and semantic languages.

Figure 5: A layered structure of our proposed solution (semantically formalized logs for enhanced monitoring and management of software applications)

Layers at the bottom of the application process the semantically formalized and enriched logs to monitor the application. Based on the lacking found while conducting the literature review, our solution is unique as it follows a hybrid approach to: (1) make the information highly structured and formalized, and (2) use advanced data mining and social network analysis techniques to process the information, hence combines the best of both. Semantic descriptions to components and events in logs are used by Social Network
Analysis and Data Mining based techniques to process the logs. This allows the monitoring and management of software applications to have more explicit information to precisely find out correlations during the process of monitoring and management of such applications. The process of semantic log generation and processing is carried out and handled through semantic logging API as shown vertical in the layered architecture.

Semantic formalism of logs has been carried out using our proposed semantic meta-model as well as a well-known semantic language based on Web standards [2]. It is to be noted that semantics is one of the ways to formalize logs. However, we have chosen semantics to be the approach to formalize the logs as it is based on widely adapted standards [3].

3.1 Semantic Model for Components and Logs

This section presents our model for semantically describing component descriptions and log events. A component is a part of an application that encapsulates a functionality based on implementation and an interface that is used to provide input to the component to get the functionality. An implementation neutral description to this component is provided in the application which is used by the execution engine to find out the component and to communicate with it. On the other hand, logs are produced by applications that contain footprint of the application execution. We propose semantic annotations to the component description, as well as the logs that are produced by the applications. Figure 6: Anatomy of component and its semantic description provides a glance about how the component descriptions and logs can be modeled using semantics.
The proposed model for semantic description of components includes obvious information about inputs and outputs. Moreover, it also precisely contains the information related to the functionality provided by the component in the context of a particular domain. Furthermore, the model for semantic descriptions of components also allows to precisely specify the conditions under which the component should be used (i.e., if some particular event occurs). It gives a notion of event-driven management of the components within the application. It also allows having precise information about the component, i.e., the action that the component should perform if a particular event occurs.

Therefore, the events during the execution in an application are handled based on the semantic information provided for the components. The events are modeled and processed from the logs of the application being executed. Figure 7: Semantic model for Log Events depicts our proposed semantic model for logs.

![Diagram](attachment:image-6.png)

**Figure 6: Anatomy of component and its semantic description**
The description of the Log Event is also connected to the description of the components in the application to be monitored. It contains the information about components that originate the event. It also includes the method within the component that originated the event. Furthermore, it includes the context in which the event took place. We have formally defined context vocabulary. Every event is distinguished by a unique identifier. Events also have names, date/time of events as well as event status as compulsory fields to be filled-in. Status of an event is derived from our formally defined vocabulary. An event may have n number of key-value pairs to enclose any application specific information. Both the semantic models for component description and event description are correlated with each other in order to have a global view of events of their execution across different components. Events are produced and recorded in a structured way and they are modeled with semantic descriptions. This highly structured and formalized way of modeling logs facilitates to have execution and monitoring mechanism to perform automated and enhanced monitoring of events during application execution.

We present definitions of Components, Log Events and Functionality that are three key elements to model and keep track of during the process of monitoring of applications. These key elements are required together in order to find out Components offering in link to processing Functionalities and Log Events requiring such Functionalities.

Given below are formal definitions for the proposed meta-models of Components and Log Events:
**Definition 1 (Component - C):** Let C be a component in an application that may be involved in the execution of an event. It prescribes meta-model for any component to contain necessary information. It can be represented as a tuple:

\[(\text{Binding}, \text{Type}, \text{Inputs}(h), \text{Outputs}(j), \text{Events}(l), \text{EnvironmentVariables}(p))\]

where Binding contains information about protocol binding and protocol information for invocation of the component. Type contains information about the different possible kinds of components an application may have which could be defined and implemented by application developers. Inputs(h) represents h key-value pairs that a component may accept as input. Outputs(j) represents j key-value pairs that a component may accept as output. Events(l) contains l events that a component might be involved in executing. EnvironmentVariables(p) contains p possible variables that may contain information about the computing and storage environment that a component may encounter during the execution.

**Definition 2 (Log Event - LE):** Let LE be a Log Event that prescribes meta-model for any event in the log to contain necessary information. It can be represented as a tuple:

\[(\text{EventID}, \text{EventName}, \text{TimeStamp}, \text{EventStatus}, \text{InboundComponents}(k), \text{OutboundComponents}(m), \text{Context}, \text{KeyValuePairs}(n))\]

where EventID is a unique identifier for any event defined for a software execution; EventName is a human readable name of an Event with a unique identifier. TimeStamp contains exact date and time of any update that may take place for an event. InboundComponents(k) represents k inbound components that may affect an event during the execution. OutboundComponents(m) represents m outbound components that may get
effected by an event during the execution. Context represents the application execution context out of many possible contexts an application execution may have and defined by the application developer. KeyValuePairs(n) represents n Key Value pairs that may contain application specific data and variables to be logged.

Figure 7: Semantic model for Log Events

3.2 A case-study application using Semantic Logging

This section presents a use-case application with a technical and step by step walk-through using our proposed solution of semantically formalized logging for enhanced monitoring and management of software applications. We present and compare two scenarios, one where we have traditional form of logging and a scenario where we have
semantically formalized logging for enhanced monitoring and management of the application. This use-case application has multiple components that execute concurrently in order to process user requests. In the banking application, users initiated transactions from a foreign or local banking machine have to go through multiple components in order to be processed before the response could be issued to the client machine.

Figure 8: Sample Bank Application Scenario

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.example.bank.ExternalInterface</td>
<td>[20/Apr/2013:00:00:12]</td>
</tr>
<tr>
<td>External Interface starting</td>
<td></td>
</tr>
<tr>
<td>com.example.bank.TransactionManager</td>
<td>[20/Apr/2013:00:00:13]</td>
</tr>
<tr>
<td>Transaction Manager starting</td>
<td></td>
</tr>
<tr>
<td>com.example.bank.AccountsManager</td>
<td>[20/Apr/2013:00:00:14]</td>
</tr>
<tr>
<td>Accounts Manager starting</td>
<td></td>
</tr>
</tbody>
</table>
External Interface successfully started

Transaction Manager successfully started

Accounts Manager successfully started

received ABM request from XYZ machine located in Beijing, China, for client id 123456, account number 456789, request to withdraw money, $100

Transaction Manager creating the Transaction Record

Account Manager checked account balance, successful (enough balance)

Transaction Manager approving transaction, completing transaction record

Account Manager updating account balance

Transaction Manager transaction completed

External Interface dispatching transaction

received POS request from QWE machine located in Toronto, Canada
for client id 741258, account number 963369, request to withdraw money, $15

com.example.bank.TransactionManager - [29/May/2013:00:04:14] Transaction Manager creating the transaction record

com.example.bank.AccountManager - [29/May/2013:00:04:14] Account Manager checked account balance, failure (not enough balance)

com.example.bank.TransactionManager - [29/May/2013:00:04:14] Transaction Manager Rejecting Transaction

com.example.bank.TransactionManager - [29/May/2013:00:04:14] Transaction Manager Transaction completed

com.example.bank.ExternalInterface - [29/May/2013:00:00:12] External Interface dispatching transaction failure information

com.example.bank.ExternalInterface - [30/May/2013:00:00:12] received POS request from TDS machine located in Winnipeg, Canada for client id 741456, account number 654852, request to withdraw money, $74

com.example.bank.TransactionManager - [30/May/2013:00:04:14] Transaction Manager creating the transaction record

com.example.bank.TransactionManager - [30/May/2013:00:04:14] Transaction Manager failed. System Exception. Transaction Manager shutting down

com.example.bank.TransactionManager - [30/May/2013:00:04:14] Transaction Manager stopped

com.example.bank.ExternalInterface - [30/May/2013:00:04:16] External Interface dispatching transaction failure information
Table 2: Traditional logging as human readable logs

The various components in the banking application start with the External Interface Manager which receives any transactions for deposit or withdraw of money from within the same or different countries. After verification of client identity, the request is processed by the transaction manager within the banking system and creates the necessary transaction record for transaction management that may need to be carried out at a later stage. After the component transaction manager, the accounts manager carries out necessary checks against the banking database or repository in order to check for the account balance. Once the Accounts manager component completes the necessary checks on the banking database or repository, it releases a response to the transaction manager. The transaction manager updates the relevant transaction record and sends a response back to the External Interface Manager component which sends a response back to the client machine from where the client initiated the request. Figure 8: Sample Bank Application Scenario depicts the architectural perspective of the use-case banking application.
ex _"http://www.example.org/ex2#"}

ontology _"http://www.example.org/ex1"

startAnnotations
   ex#EventID hasValue 123456
   ex#EventName hasValue "Starting External Interface"
   ex#TimeStamp hasValue _date(2013,04,20:00:00:12)
   ex#EventStatus hasValue "Success"
   ex#InboundComponents hasValue {External Interface}
   ex#OutboundComponents hasValue {External Interface}
endAnnotations

startAnnotations
   ex#EventID hasValue 123457
   ex#EventName hasValue "Starting Transaction Manager"
   ex#TimeStamp hasValue _date(2013,04,20:00:00:13)
   ex#EventStatus hasValue "Success"
   ex#InboundComponents hasValue {TransactionManager}
   ex#OutboundComponents hasValue {TransactionManager}
endAnnotations

startAnnotations
   ex#EventID hasValue 123458
   ex#EventName hasValue "Starting Accounts Manager"
### Table 3: Semantically formalized logs for the initiated components

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Event Name</th>
<th>Time Stamp</th>
<th>Event Status</th>
<th>Inbound Components</th>
<th>Outbound Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>123459</td>
<td>Started External Interface Successfully</td>
<td>2013-04-20 00:00:02:01</td>
<td>Success</td>
<td>ExternalInterface</td>
<td>ExternalInterface</td>
</tr>
<tr>
<td>123460</td>
<td>Started Transaction Manager Successfully</td>
<td>2013-04-20 00:00:00:14</td>
<td>Success</td>
<td>AccountsManager</td>
<td>AccountsManager</td>
</tr>
</tbody>
</table>
Table 4: Semantically formalized logs declaring components being started successfully
ex#TimeStamp hasValue _date(2013,04,20:00:00:12)
ex#EventStatus hasValue “Success”
ex#InboundComponents hasValue {ExternalInterface}
ex#OutboundComponents hasValue {TransactionManager}
ex#Context hasValue “Foreign Transaction”
ex#KeyValuePairs hasValue {TransactionID = 98765432}
ex#KeyValuePairs hasValue {TransactionCountry = “China”}
ex#KeyValuePairs hasValue {MachineID = XYZ}
ex#KeyValuePairs hasValue {ClientAccount= 123456}
ex#KeyValuePairs hasValue {ClientID= 456789}
ex#KeyValuePairs hasValue {AccountRequest = Withdraw}
ex#KeyValuePairs hasValue {Amount = 100}
ex#KeyValuePairs hasValue {Currency = CAD}

endAnnotations

startAnnotations

ex#EventID hasValue 123463
ex#EventName hasValue “creating the Transaction Record”
ex#TimeStamp hasValue _date(2013,04,23:00:04:14)
ex#EventStatus hasValue “Success”
ex#InboundComponents hasValue {TransactionManager}
ex#OutboundComponents hasValue {AccountsManager}
ex#Context hasValue “Foreign Transaction”
ex#KeyValuePairs hasValue {TransactionID = 98765432}
ex#KeyValuePairs hasValue {TransactionCountry = “China”}
ex#KeyValuePairs hasValue {MachineID = XYZ}
ex#KeyValuePairs hasValue {ClientAccount= 123456}
ex#KeyValuePairs hasValue {ClientID= 456789}
ex#KeyValuePairs hasValue {AccountRequest = Withdraw}
ex#KeyValuePairs hasValue {Amount = 100}
ex#KeyValuePairs hasValue {Currency = CAD}

dendAnnotations

dstartAnnotations

ex#EventID hasValue 123464
ex#EventName hasValue “Account Manager checked account balance , successful (enough balance)”
ex#TimeStamp hasValue _date(2013,04,23:00:04:14)
ex#EventStatus hasValue “Success”
ex#InboundComponents hasValue {AccountsManager}
ex#OutboundComponents hasValue {TransactionManager}
ex#Context hasValue “Foreign Transaction”
ex#KeyValuePairs hasValue {TransactionID = 98765432}
ex#KeyValuePairs hasValue {TransactionCountry = “China”}
ex#KeyValuePairs hasValue {MachineID = XYZ}
ex#KeyValuePairs hasValue {ClientAccount= 123456}
ex#KeyValuePairs hasValue {ClientID= 456789}
ex#KeyValuePairs hasValue {AccountRequest = Withdraw}
ex#KeyValuePairs hasValue {Amount = 100}
ex#KeyValuePairs hasValue {Currency = CAD}
Table 5: Semantically formalized logs execution of user request

In Table 2: Traditional logging as human readable logs, we show a sample of logs that are obtained using a banking application which uses a common logging mechanism to obtain logs for the execution performed. The logging data is completely anonymized and the banking application details are not mentioned due to non-disclosure. We note that the logs are highly unstructured and scattered all over the log file. There is no standardization technique which has been followed during the execution. The logs are ambiguous and it is hard to process the logs using any standardized processing technique to perform any analysis on the execution.

We then applied our proposed solution of semantically formalized logging process on the same bank application scenario and got the following logs. These logs are well structured and formalized using our proposed solution. Each of the log events are annotated using our proposed model for Semantic Logs. For each log event, we have included information like Event ID, Event Name, Timestamp, Status of the Event, Inbound and Outbound Components, Context, and a set of key-value pairs that contain application specific data. Table 3: Semantically formalized logs for the initiated components provides the semantically formalized log snippets that contain three events for starting up the components, i.e., External Interface, Transaction Manager and Accounts Manager.
startAnnotations
  ex#EventID hasValue 123465
  ex#EventName hasValue “Transaction Manager approving transaction, completing transaction record”
  ex#TimeStamp hasValue _date(2013,04,23:00:04:14)
  ex#EventStatus hasValue “Success”
  ex#InboundComponents hasValue {TransactionManager}
  ex#OutboundComponents hasValue {AccountsManager}
  ex#Context hasValue “Foreign Transaction”
  ex#KeyValuePairs hasValue {TransactionID = 98765432}
  ex#KeyValuePairs hasValue {TransactionCountry = “China”}
  ex#KeyValuePairs hasValue {MachineID = XYZ}
  ex#KeyValuePairs hasValue {ClientAccount= 123456}
  ex#KeyValuePairs hasValue {ClientID= 456789}
  ex#KeyValuePairs hasValue {AccountRequest = Withdraw}
  ex#KeyValuePairs hasValue {Amount = 100}
  ex#KeyValuePairs hasValue {Currency = CAD}
endAnnotations

startAnnotations
  ex#EventID hasValue 123466
  ex#EventName hasValue “Account Manager updating account balance”
ex#TimeStamp hasValue _date(2013,04,23:00:04:14)
ex#EventStatus hasValue “Success”
ex#InboundComponents hasValue {AccountsManager}
ex#OutboundComponents hasValue {TransactionManager}
ex#Context hasValue “Foreign Transaction”
ex#KeyValuePairs hasValue {TransactionID = 98765432}
ex#KeyValuePairs hasValue {TransactionCountry = “China”}
ex#KeyValuePairs hasValue {MachineID = XYZ}
ex#KeyValuePairs hasValue {ClientAccount= 123456}
ex#KeyValuePairs hasValue {ClientID= 456789}
ex#KeyValuePairs hasValue {AccountRequest = Withdraw}
ex#KeyValuePairs hasValue {Amount = 100}
ex#KeyValuePairs hasValue {Currency = CAD}
endAnnotations

startAnnotations

ex#EventID hasValue 123467
ex#EventName hasValue “Transaction Manager transaction completed”
ex#TimeStamp hasValue _date(2013,04,23:00:04:14)
ex#EventStatus hasValue “Success”
ex#InboundComponents hasValue {TransactionManager}
ex#OutboundComponents hasValue {AccountsManager}
ex#Context hasValue “Foreign Transaction”
ex#KeyValuePairs hasValue {TransactionID = 98765432}
ex#KeyValuePairs hasValue {TransactionCountry = "China"}
ex#KeyValuePairs hasValue {MachineID = XYZ}
ex#KeyValuePairs hasValue {ClientAccount= 123456}
ex#KeyValuePairs hasValue {ClientID= 456789}
ex#KeyValuePairs hasValue {AccountRequest = Withdraw}
ex#KeyValuePairs hasValue {Amount = 100}
ex#KeyValuePairs hasValue {Currency = CAD}
endAnnotations

startAnnotations
    ex#EventID hasValue 123468
    ex#EventName hasValue “External Interface dispatching transaction”
    ex#TimeStamp hasValue _date(2013,04,23:00:04:15)
    ex#EventStatus hasValue “Success”
    ex#InboundComponents hasValue {ExternalInterface}
    ex#OutboundComponents hasValue {empty}
    ex#Context hasValue “Foreign Transaction”
ex#KeyValuePairs hasValue {TransactionID = 98765432}
ex#KeyValuePairs hasValue {TransactionCountry = “China”}
ex#KeyValuePairs hasValue {MachineID = XYZ}
ex#KeyValuePairs hasValue {ClientAccount= 123456}
ex#KeyValuePairs hasValue {ClientID= 456789}
ex#KeyValuePairs hasValue {AccountRequest = Withdraw}
ex#KeyValuePairs hasValue {Amount = 100}
Table 4: Semantically formalized logs declaring components being started successfully shows the semantically formalized logs for declaring the three components as started successfully. After getting the components started successfully, Table 5: Semantically formalized logs execution of user request shows the semantically formalized logs reflecting the External Interface processing of the user request received from a different country, passing the response to the Transaction Manager to create the necessary transaction record followed by a response to the Accounts Manager to perform the required checks in the bank database about the user account.

Once the Accounts Manager component approves the user request after performing necessary checks on the user account in the bank database, it sends a response back to the Transaction Manager, which then completes the transaction record, i.e., allows the Accounts Manager to close down the request and dispatches a response to the External Interface which dispatches a positive response to the client’s machine.

Table 6: Semantically formalized logs execution of user request shows semantically formalized logs for a transaction that was received locally, processed by the External Interface component, transferred to the Transaction Manager to create the necessary transaction record, followed by the Accounts Manager component to perform the
necessary checks and find out that there was not enough balance in the client’s account. Therefore, the Accounts Manager sends back a signal with this information to the Transaction Manager. The Transaction Manager updates the transaction record with failure information and sends back to the External Interface Manager the information to be dispatched back to the client’s machine to reject the transaction.

```
startAnnotations

  ex#EventID hasValue 123462

  ex#EventName hasValue “Processing POS request received locally”

  ex#TimeStamp hasValue _date(2013,05,29:00:00:12)

  ex#EventStatus hasValue “Success”

  ex#InboundComponents hasValue {ExternalInterface}

  ex#OutboundComponents hasValue {TransactionManager}

  ex#Context hasValue “Local Transaction”

  ex#KeyValuePairs hasValue {TransactionID = 98765433}

  ex#KeyValuePairs hasValue {TransactionCountry = “Canada”}

  ex#KeyValuePairs hasValue {MachineID = QWE}

  ex#KeyValuePairs hasValue {ClientAccount= 963369}

  ex#KeyValuePairs hasValue {ClientID= 741258}

  ex#KeyValuePairs hasValue {AccountRequest = Withdraw}

  ex#KeyValuePairs hasValue {Amount = 15}

  ex#KeyValuePairs hasValue {Currency = CAD}

endAnnotations
```
startAnnotations

ex#EventID hasValue 123463
ex#EventName hasValue “Creating the Transaction Record”
ex#TimeStamp hasValue _date(2013,05,29:00:00:14)
ex#EventStatus hasValue “Success”
ex#InboundComponents hasValue {TransactionManager}
ex#OutboundComponents hasValue {AccountsManager}
ex#Context hasValue “Local Transaction”
ex#KeyValuePairs hasValue {TransactionID = 98765433}
ex#KeyValuePairs hasValue {TransactionCountry = “Canada”}
ex#KeyValuePairs hasValue {MachineID = QWE}
ex#KeyValuePairs hasValue {ClientAccount= 963369}
ex#KeyValuePairs hasValue {ClientID= 741258}
ex#KeyValuePairs hasValue {AccountRequest = Withdraw}
ex#KeyValuePairs hasValue {Amount = 15}
ex#KeyValuePairs hasValue {Currency = CAD}
endAnnotations

startAnnotations

ex#EventID hasValue 123464
ex#EventName hasValue “Account Manager checked account balance, failure (not enough balance)”
ex#TimeStamp hasValue _date(2013,05,29:00:00:14)
ex#EventStatus hasValue “Success”
ex#InboundComponents hasValue {AccountsManager}

ex#OutboundComponents hasValue {TransactionManager}

ex#Context hasValue “Local Transaction”

ex#KeyValuePairs hasValue {TransactionID = 98765433}

ex#KeyValuePairs hasValue {TransactionCountry = “Canada”}

ex#KeyValuePairs hasValue {MachineID = QWE}

ex#KeyValuePairs hasValue {ClientAccount= 963369}

ex#KeyValuePairs hasValue {ClientID= 741258}

ex#KeyValuePairs hasValue {AccountRequest = Withdraw}

ex#KeyValuePairs hasValue {Amount = 15}

ex#KeyValuePairs hasValue {Currency = CAD}

endAnnotations

startAnnotations

    ex#EventID hasValue 123465

    ex#EventName hasValue “Transaction Manager rejecting transaction, completed transaction record”

    ex#TimeStamp hasValue _date(2013,05,29:00:00:14)

    ex#EventStatus hasValue “Success”

    ex#InboundComponents hasValue {TransactionManager}

    ex#OutboundComponents hasValue {AccountsManager}

    ex#Context hasValue “Local Transaction”

    ex#KeyValuePairs hasValue {TransactionID = 98765433}

    ex#KeyValuePairs hasValue {TransactionCountry = “Canada”}

    ex#KeyValuePairs hasValue {MachineID = QWE}
ex#KeyValuePairs hasValue {ClientAccount= 963369}
ex#KeyValuePairs hasValue {ClientID= 741258}
ex#KeyValuePairs hasValue {AccountRequest = Withdraw}
ex#KeyValuePairs hasValue {Amount = 15}
ex#KeyValuePairs hasValue {Currency = CAD}
endAnnotations

startAnnotations

    ex#EventID hasValue 123468
    ex#EventName hasValue “External Interface dispatching transaction failure information”
    ex#TimeStamp hasValue _date(2013,05,29:00:00:14)
    ex#EventStatus hasValue “Success”
    ex#InboundComponents hasValue {ExternalInterface}
    ex#OutboundComponents hasValue {empty}
    ex#Context hasValue “Local Transaction”
    ex#KeyValuePairs hasValue {TransactionID = 98765433}
    ex#KeyValuePairs hasValue {TransactionCountry = “Canada”}
    ex#KeyValuePairs hasValue {MachineID = QWE}
    ex#KeyValuePairs hasValue {ClientAccount= 963369}
    ex#KeyValuePairs hasValue {ClientID= 741258}
    ex#KeyValuePairs hasValue {AccountRequest = Withdraw}
    ex#KeyValuePairs hasValue {Amount = 15}
    ex#KeyValuePairs hasValue {Currency = CAD}

endAnnotations
Table 7: Semantically formalized logs execution of user request

<table>
<thead>
<tr>
<th>EventID</th>
<th>EventName</th>
<th>TimeStamp</th>
<th>EventStatus</th>
<th>InboundComponents</th>
<th>OutboundComponents</th>
<th>Context</th>
<th>KeyValPairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>123469</td>
<td>&quot;Processing POS request received locally&quot;</td>
<td>_date(2013,05,30:00:00:14)</td>
<td>&quot;Success&quot;</td>
<td>{ExternalInterface}</td>
<td>{TransactionManager}</td>
<td>&quot;Local Transaction&quot;</td>
<td>{TransactionID = 98765434}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>{TransactionCountry = &quot;Canada&quot;}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>{MachineID = TDS}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>{ClientAccount = 654852}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>{ClientID= 741456}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>{AccountRequest = Withdraw}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>{Amount = 74}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>{Currency = CAD}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EventID</th>
<th>EventName</th>
<th>TimeStamp</th>
<th>EventStatus</th>
<th>InboundComponents</th>
<th>OutboundComponents</th>
<th>Context</th>
<th>KeyValPairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>123470</td>
<td></td>
<td>_date(2013,05,30:00:00:14)</td>
<td>&quot;Success&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

startAnnotations

startAnnotations
ex#EventName hasValue “Creating the Transaction Record”
ex#TimeStamp hasValue _date(2013,05,30:00:04:14)
ex#EventStatus hasValue “Success”
ex#InboundComponents hasValue {TransactionManager}
ex#OutboundComponents hasValue {AccountsManager}
ex#Context hasValue “Local Transaction”
ex#KeyValuePairs hasValue {TransactionID = 98765434}
ex#KeyValuePairs hasValue {TransactionCountry = “Canada”}
ex#KeyValuePairs hasValue {MachineID = TDS}
ex#KeyValuePairs hasValue {ClientAccount= 654852}
ex#KeyValuePairs hasValue {ClientID= 741456}
ex#KeyValuePairs hasValue {AccountRequest = Withdraw}
ex#KeyValuePairs hasValue {Amount = 74}
ex#KeyValuePairs hasValue {Currency = CAD}

startAnnotations

ex#EventID hasValue 123471
ex#EventName hasValue “Transaction Manager failed. System Exception. Transaction Manager shutting down”
ex#TimeStamp hasValue _date(2013,05,30:00:04:14)
ex#EventStatus hasValue “Failure”
ex#InboundComponents hasValue {TransactionManager}
ex#OutboundComponents hasValue {ExternalInterface}
ex#Context hasValue “Local Transaction”

endAnnotations
ex#KeyValuePairs hasValue {TransactionID = 98765434}
ex#KeyValuePairs hasValue {TransactionCountry = “Canada”}
ex#KeyValuePairs hasValue {MachineID = TDS}
ex#KeyValuePairs hasValue {ClientAccount= 654852}
ex#KeyValuePairs hasValue {ClientID= 741456}
ex#KeyValuePairs hasValue {AccountRequest = Withdraw}
ex#KeyValuePairs hasValue {Amount = 74}
ex#KeyValuePairs hasValue {Currency = CAD}

startAnnotations

ex#EventID hasValue 123472
ex#EventName hasValue “Transaction Manager stopped”
ex#TimeStamp hasValue _date(2013,05,30:00:04:14)
ex#EventStatus hasValue “Failure”
ex#InboundComponents hasValue {TransactionManager}
ex#OutboundComponents hasValue {ExternalInterface}
ex#Context hasValue “Local Transaction”
ex#KeyValuePairs hasValue {TransactionID = 98765434}
ex#KeyValuePairs hasValue {TransactionCountry = “Canada”}
ex#KeyValuePairs hasValue {MachineID = TDS}
ex#KeyValuePairs hasValue {ClientAccount= 654852}
ex#KeyValuePairs hasValue {ClientID= 741456}
ex#KeyValuePairs hasValue {AccountRequest = Withdraw}
ex#KeyValuePairs hasValue {Amount = 74}
Table 8: Semantically formalized logs execution of user request

```latex
ex#KeyValuePairs hasValue {Currency = CAD}
endAnnotations

startAnnotations
  ex#EventID hasValue 123473
  ex#EventName hasValue "External Interface dispatching transaction failure information"
  ex#TimeStamp hasValue _date(2013,05,30:00:04:16)
  ex#EventStatus hasValue "Success"
  ex#InboundComponents hasValue {ExternalInterface}
  ex#OutboundComponents hasValue {empty}
  ex#Context hasValue "Local Transaction"
  ex#KeyValuePairs hasValue {TransactionID = 98765434}
  ex#KeyValuePairs hasValue {TransactionCountry = "Canada"}
  ex#KeyValuePairs hasValue {MachineID = TDS}
  ex#KeyValuePairs hasValue {ClientAccount= 654852}
  ex#KeyValuePairs hasValue {ClientID= 741456}
  ex#KeyValuePairs hasValue {AccountRequest = Withdraw}
  ex#KeyValuePairs hasValue {Amount = 74}
  ex#KeyValuePairs hasValue {Currency = CAD}
endAnnotations
```
Table 8: Semantically formalized logs execution of user request shows semantically formalized logs for another transaction that was received locally, processed by the External Interface component, and then transferred to the Transaction Manager to create the necessary transaction record. In this case, while the Transaction Manager is creating the transaction record, it fails with an exception. Due to the system exception, the Transaction Manager shuts down and the relevant information is stored in the semantically formalized logs accordingly. The Transaction Manager then sends a failure of transaction information to the External Interface to be dispatched to the client machine that initiated the transaction, and closes the transaction.

We have shown two different scenarios for the same set of transactions execution in the banking use-case application. In the first scenario, we have shown the logs generated using a commonly used logging approach. In the second scenario, we have shown semantically formalized logs generated for the same situation as that of the first scenario. In the first scenario, we find it hard to interpret the unstructured logs and use the logs in analytical solutions to perform any analysis in the logs. On the other side, we find it convenient to use semantically formalized logs in performing monitoring and management of the application execution. We are using different analytical solutions like, association rule mining, social network analysis and classification to find out interesting and non-obvious patterns in events in the log. This will help in performing advanced as well as automated monitoring and management which is highly desired in applications, especially for large-scale applications.
Automated Ranking is a crucial step in the process of automated Web Services execution after discovery. Often adaptation and ranking (used interchangeably) of the discovered Web services is carried out using functional and non-functional information of Web Services. Such approaches are dependent on heavy and rich semantic descriptions as well as unstructured and scattered information about any past interactions between clients and Web Services. Existing approaches are either found to be only focusing on semantic modeling and representation only, or using data mining and machine learning based approaches on unstructured and raw data to perform discovery and ranking. We propose an approach to allow semantically formalized representation of logs during Web Service execution and then use such logs to perform ranking and adaptation of the discovered Web Services. We have found that combining both approaches together into a hybrid approach would enable formal representation of Web Services data which would boost data mining as well as machine learning based solutions to process such data. We have built Semantic FP-Tree based technique to perform association rule learning based on functional and non-functional characteristics of Web Services. The process of automated execution of Web Services is improved in two steps, i.e., (1) we provide semantically formalized logs that maintain well-structured and formalized information about past interactions of Services Consumers and Web Services, (2) we perform an extended

---

4 Contents of this chapter were published in the following paper: Omair Shafiq, Reda Alhajj, Jon G. Rokne, “Frequent Pattern Mining using Semantic FP-Growth for Effective
association rule mining on semantically formalized logs to find out any possible correlation in functional and non-functional characteristics of Web Services during past execution which is then used in automated ranking and adaptation of Web Services. We have conducted comprehensive evaluation to demonstrate the efficiency, effectiveness and usability of our proposed approach.

4.1 Introduction
Web Services [62] have changed the Web from static to dynamic nature where applications may act as Service Consumers in order to invoke and utilize Web Services over the Web. Applications as Service Consumers can dynamically invoke a Web Service by providing input and can get a response back as output processed based on the functionality provided by the Web Service. Because of the open nature of the Web, it is not possible for Service Consumers to have a pre-knowledge of all the available Web Services over the Web [1]. Dynamic invocation of Web Services requires dynamic discovery and ranking of Web Services that are found over the Web. In order to bring dynamism in the process of Web Service invocation and execution, it is crucial to make the process of Web Service discovery and ranking automated [20]. Several approaches have been proposed to make the process of discovery and ranking of Web Services automated. However, we have seen major lacking in such approaches. Traditional discovery and ranking approaches for Web Services have been found too limited and are based only on syntactic and pre-known information of services which causes limitations for user-applications to use newly available services. Such approaches are found to be
limited to use information from the Web Service Description Language (WSDL) [15] or the Universal Description Discovery and Integration (UDDI) [16] of Web Services. One proof of the limitation of such approaches is that in the last few years usage of UDDI based Web Service discovery approaches has rather become unpopular [63]. Due to the limited extent of human readable descriptions in the UDDI based business service registries, the process discovery of Web Services has become a rather limited and imprecise task that can be made useful with human intervention only. This drastically limits one of the key properties of Service Oriented Architectures and Web Services as to allow dynamic machine-to-machine interaction. Current technologies and Web-based search engines are also not well suited for Web Service discovery because search engines operate on HTML based characteristics of Web pages, and cannot take into account the features and properties of Web Services which are important for precise discovery.

Instead of using syntactic approaches, new approaches have been built which are based on using information from semantically enriched descriptions of Web Services. These approaches require precise, expressive and machine interpretable description of services with an aim to make it easier for users to search for the services required. These approaches have shown a good potential towards enabling automation in Web Services and because of that Semantic Web Services research have gained momentum. The latest trends on Semantic Web Service discovery [64] and Web Service Modeling Ontology (WSMO) [3] based Semantic Web Service discovery have presented discovery framework [26] that helps in proceeding towards dynamic Web Service discovery. The discovery framework uses reasoning approaches to try dynamically matching semantic
descriptions of requirements of service consumers as Goals with semantic descriptions of service providers. However, most of the existing semantic based service discovery approaches are still naïve, i.e., only support the discovery of a few services and take quite some significant amount of time while performing the discovery. Such approaches are still found to be not in their full potential to be used in practical scenarios for automated discovery and ranking of Web Services as it would be impractical to assume that every user and service provider will incorporate full-fledge semantics in requests as well as Web Service descriptions, respectively. On the other hand, using only the basic information about Web Services (i.e., WSDL based Web Service descriptions) does not provide enough information to be able to discover the required Web Services out of the available ones. This puts the dynamic discovery and ranking of Web Services in a dilemma of using semantics to bring enough information about Web Services, and at the same time keeping the Web Service discovery process simpler and reasonably efficient, usable and practical.

We try to solve this dilemma by proposing a hybrid approach of partially using semantics (such as functional and non-functional properties of Web Services), and use this information to perform discovery and ranking of Web Services. For this purpose, we have proposed a way to specify formalized and well-structured logs as Semantic Logs about past interactions of client applications with Web Services, and then use these Semantic Logs incorporating light-weight semantic specifications of Web Services to perform frequent pattern mining. This way, the process of automated ranking and adaptation of Web Services is enhanced in two steps. First, it makes use of semantic information of Web Services as well as past interactions between users and Web Services
which is available in a formalized and well-structured way as Semantic Logs. Second, it incorporates an adapted data mining approach called Semantic FP-Growth, which is based on the existing FP-Growth [65] data mining technique. It use Semantic Logs to perform association rule mining. The ranking and adaptation of Web Services is done using the rules learned from the developed association rule mining process.

The rest of the chapter is organized as follows. Section 2 presents related work in the area of automated ranking and adaptation of Web Services and outlines pros and cons of such approaches. Section 3 presents the proposed solution of Semantic Logs for incorporating past interactions between users and Web Services. Section 4 presents Semantic Logs for Web Services. Section 5 uses our proposed Semantic FP-Growth algorithm to process such logs to use such information in ranking and adaptation of Web Services. Section 6 presents experiments and evaluates the results as well as compares them with that of existing solutions. Section 7 presents conclusions.

4.2 Related Work

There has been a lot of related work in the area of automated ranking and adaptation of Web Services. Such related work spans from using highly formalized and semantically enriched descriptions of Web Services and user queries, to the usage of data mining and machine learning approaches on raw data of Web Services. Several approaches have been found that have used association rule mining for adaptation and ranking of Web Services and other similar systems. Given below are related and existing approaches followed by comparative analysis of such approaches.
A personalized Web Services Ranking has been proposed using user groups with association rule mining [66]. Based on the collaborative filtering idea, users with similar interests are identified. They are then used by association rule mining to deduce association rules by analyzing Web Service composition transactions related to that particular set of similar users rather than all the users. The authors found out that combining user group and association rule mining with relevant users only helped in building personalized Web Service ranking. This approach uses association rule mining on a subset of users, but is limited because the Web Service composition transactions include very limited information.

Web Service Relevancy Function (WsRF) [68] is another effort for measuring relevancy and ranking of a particular Web Service based on the preferences of users and the corresponding Non-Functional Properties like Quality of Service (QoS). Such QoS parameters are Response Time, Throughput, Availability, Accessibility, Interoperability Analysis, as well as Cost to invoke the Service. The QoS parameters can be specified by clients manually by a GUI, and by taking into account computing the relevance of known Web Services or discovering Web Services over the Web [67]. The limitation in this approach is that it only focuses on non-functional aspects of Web Services which are to be calculated by the client application and hence impose an overhead.

The usage of ontological representations of non-functional properties has been another way to explore ranking for Semantic Web Services [32]. Non-functional properties of Web Services are considered as a multi-criteria mechanism that takes the multiple nonfunctional properties as different possible dimensions of ranking. The proposed algorithm as described in [32] takes into account the associated importance for
non-functional properties from the perspective of users. The limitation in this approach is that it only focuses on non-functional aspects of Web Services.

In [69], a context based method has been proposed where Web Services are analyzed using Web Service Description Language (WSDL) from semantics perspective to try extracting more accurate and correct answer that could match user’s queries. After the discovery is performed, the degree of nearness as proximity of similar Web Services with context is determined in order to generate a list of finally ranked Web Services. In this approach, the authors take into account the context from specific sites; they do not take into account user perspective on the context and this makes it limited. Web Services that are determined as similar with contextual information are then used to perform another level of filtering to determine a final result of ranking. This approach is rather limited to the information provided in WSDLs of Web Services only.

An Association Rule Mining based approach described in the literature is used for discovering related items like queries [70]. We believe that this approach can also be extended to other items like documents or Web Services. In this approach, the log of query execution is viewed as a set of transactions, with each transaction representing a session in which a single user submits a sequence of related queries in a time interval. The method shows good results, but two problems arise. First, it is difficult to determine sessions of queries belonging to the same search process. Second, most interesting related queries submitted by different users cannot be discovered, since the support of a rule increases only if its queries appear in the same query sessions, i.e., the queries are submitted together by the same users. In this approach, authors do not attempt to structure or formalize the logs, but only apply an algorithm similar to Apriori for Association Rule
Mining. The log is viewed as a set of transactions and related queries are discovered to help in computing an answer faster by using previous answers already computed for a similar set of queries. In conclusion, this approach is also limited to query answering only.

Ontology semantics have also been explored and used for matching Web Services [71]. The XML documents of Web Services as Web Service Description Language (WSDL) are scanned and the inputs as well as outputs of similar Web Services are compared and used for ranking such Web Services. In addition to the information obtained from WSDLs of Web Services, an ontology is also built based on Quality of Service (QoS) aspects of Web Services that includes descriptions for various metrics for the Web Services. An algorithm has been developed that uses QoS based matrices for comparing and matching similar Web Services to find the best ones as per user requirements.

We have observed from the analysis and review of existing and related approaches overviewed above that almost all the approaches are either focused towards applying data mining and heuristic techniques on syntactic data of Web Services as Web Service Description Language (WSDL). The problem with such approaches is that the information in WSDL description of Web Services is syntactic and limited. We believe that such information is limited and is not enough to find the ranking of Web Services. We also explored other approaches which are based on semantically enriched descriptions of Web Services, like Non-Functional Properties (NFPs), which try to perform automated discovery, selection or ranking of Web Services. Such approaches are a good step towards performing automated discovery and ranking of Web Services, but
are still limited in two aspects. First, such approaches do not take into account any past history of interactions of users and Web Services, and second, such approaches do not take into account any extensive data mining or machine learning based approaches to make use of such semantically formalized and well-structured data. Therefore, such approaches are still not in their full potential to perform automated ranking and adaptation of Web Services. Such approaches are not only limited from the perspective of accuracy and completeness, but are also limited from the perspective of scalability and hence take significant amount of time to perform the task of automated discovery and ranking. This takes us to the dilemma of either user highly enriched and formal semantics of Web Services which would provide a lot of information about Web Services. However, this would be impractical to enforce all users and providers of Web Services to provide all such information as well as would cause inefficiency in processing such information. On the other hand, keeping the process of discovery and ranking of Web Services rather simpler, i.e., by using limited information with data mining as well as using heuristic techniques, does not bring us the level of accuracy and correctness that users require. Our proposed solution uses a hybrid approach of partially using light-weight semantics for Web Services and then uses an enhanced association rule mining technique to process such information for the discovery and ranking of Web Services. In the previous work, we used only non-functional aspects of Web Services to see the viability of our proposed approach [8] [13]. The work has been extended in this chapter to the next level by using functional as well as non-functional aspects of Web Services while correlating with execution logs modeled as Semantic Logs.
4.3 Proposed Solution

In this section, we present our proposed solution for applying frequent pattern mining using our proposed Semantic FP-Growth algorithm on Semantic Logs in order to perform effective and efficient ranking and adaptation of Web Services. Our proposed solution is unique because of several reasons. First, it proposes to take into account past interactions of users and providers of Web Services during the process of ranking and proposes to semantically formalize logs for past interactions between users and providers of Web Services. Second, it uses light-weight semantics for formalization of logs that include functional and non-functional aspects of Web Services as well as their past interactions. Third, it provides an enhanced association rule mining algorithm as Semantic FP-Growth to perform association rule mining based analysis on Semantic Logs which is then used to perform ranking and adaptation for Web Services. Given below are a few definitions which are important to present the proposed solution.

Definition 1 (Service Consumers - SC): SC stands for Service Consumers that act as Web Service clients.

Definition 2 (Service Providers - SP): SP stands for Service Providers that provide Web Services denoted as WS.

Definition 3 (Functional Properties - FPs): FPs stand for the Functional Properties that are required by Service Consumers and offered by Service Providers. We assume that there can be k functional properties denoted FP₁, FP₂, FP₃ … FPₖ. Functional Properties may include Inputs, Outputs, Preconditions and Effects, denoted I, O, P and E, respectively.
**Definition 4 (Non-Functional Properties - NFPs):** NFPs stand for the Non-Functional Properties that are required by Service Consumers, and offered by Service Providers. We assume that there can be \( l \) non-functional properties denoted NFP\(_1\), NFP\(_2\), NFP\(_3\) … NFP\(_l\).

### 4.3.1 The Architecture

Figure 9: Overall architecture for Ranking and Adaptation of Web Services using Association Rule Mining depicts the overall picture of ranking and adaptation of Web Services using Association Rule Mining based on Semantic FP-Growth. User applications as Service Consumers search for Web Services using a middleware application that performs discovery, ranking and adaptation and finally invokes the required Web Services. For each interaction, users as Service Consumers encapsulate their requests in our prescribed form for Semantic Logs and Service Providers model Web Services using prescribed specifications as per Semantic Web Services [3].

Each of the requests from user applications for discovering and invoking Web Services are modeled and stored as Semantic Logs in a repository. Such Semantic Logs are later on retrieved and represented in the form of Semantic FP-Tree and are processed by our proposed semantic extension to the FP-Growth algorithm. Semantic FP-Tree is an extended form of FP-Tree that contains items as semantic axioms. Semantic FP-Tree is translated into a normal FP-Tree after performing inference on axioms and semantic annotations that are stored at each node. Association Rules among different events in the logs are then discovered using the normal FP-Tree that is derived from the Semantic FP-Tree. Advantage of the Semantic FP-Tree is that it has higher expressivity than that of normal FP-Tree. It can represent complex conditions, for example an event occurred that
a particular instant of time with particular data. The discovered association rules are then used during the process of ranking and adaptation of Web Services selection out of the discovered set of Web Services to select the best one. Our solution uniquely takes the process of ranking and adaptation to the next level by making the information about Web Services and past interactions formalized and well-structured; it then uses association rule mining technique to process the information. The formalized and well-structured approach makes it easier for the association rule mining based approach to utilize the available information of Web Services and events from past interactions to the maximum.

Figure 9: Overall architecture for Ranking and Adaptation of Web Services using Association Rule Mining
4.4 Semantic Logs for Web Services

Logs are produced during the process of discovery, ranking, adaptation and invocation of Web Services by user applications. Logs represent the foot-print (informative summary) of the whole process of execution. The description of logs is highly dependent upon Web Service descriptions. It contains a set of events called Log Events.

Figure 10: Model of Semantic Logs for Web Services

Log Events include contextual information in which the event took place. Events have a unique identifier to distinguish them. Events also have names, date/time of events as
well as event status as compulsory fields to be filled-in. Status of an event is also derived from our formally defined vocabulary. Events also have n key-value pairs to enclose any Web Services specific information both from users (as Service Consumers) and Service Providers perspective like Inputs, Outputs, Pre-Conditions, Effects, Non-Functional Properties, and Functional Properties as Capabilities. Both the semantically formalized descriptions of Web Services as well as events from logs are correlated with each other in order to have a global view of events of their execution and to use this information for ranking and adaptation of Web Services. Figure 10: Model of Semantic Logs for Web Services depicts the model of Semantic Logs containing information in Log Events for Web Services execution.

4.5 Ranking and Adaptation using Semantic FP-Growth

We chose FP-Growth over Apriori because of several reasons. FP-Growth is in general better in terms of memory utilization whereas Apriori requires larger space as a larger number of candidates have to be generated. Apriori has to scan data multiple times (roughly number of database scans equals the size of the largest itemset which must be checked as potential frequent itemset) whereas FP-Growth scans only twice to build the FP-Tree and make it ready for discovering frequent itemsets and then the target association rules. Based on these reasons,

Let $LE = \{ le_1, le_2, le_3, \ldots le_n \}$ be a set of Log Events

Let $\Delta T$ be a Log Interval which is a set of Log Events LE that occur in a given time interval.
Let $I = \{ i_1, i_2, i_3, \ldots, i_n \}$ be a set of items in a Log Event LE, or a Service Consumer SC or a Web Service WS.

Let WS be a Web Service with a set of Binding, Type, Inputs, Outputs, Events, and Environment Variables.

Let $i_x$ and $i_y$ be two items, where $i_x$ is antecedent and $i_y$ is consequence. Support is the frequency of occurrence of a given nonempty itemset. The rule $i_x \rightarrow i_y$ has support $s$ if $s\%$ of Log Intervals in the set of all log intervals contains $i_x \cup i_y$. Confidence is the measure of strength of the rule. A rule $i_x \rightarrow i_y$ has confidence $c$ if $c\%$ of Log Intervals in $i_x$ contains $i_x \cap i_y$.

An FP-Tree is constructed for Semantic Logs as set of items I in a Log Interval $\Delta T$. We call this FP-Tree as Semantic FP-Tree as it contains semantically formalized information based on axioms; other than this it is the same as the normal FP-Tree. The root of the Semantic FP-Tree is labeled as “null” with a set of item-prefix sub-trees as children, and a frequent-item-header table. Each node in the item-prefix sub-tree consists of three fields, i.e., (1) item-identifier where item is represented by the node, (2) count as the number of transactions represented by the portion of the path reaching the node, (3) Node-link: links to the next node in the FP-tree carrying the same item-name, or null if there is none. Each entry in the frequent-item-header table consists of two fields, i.e., (1) item-name as the same to the node, and (2) head of node-link which is a pointer to the first node in the FP-tree carrying the item-name.

We have extended the FP-Growth algorithm to Semantic FP-Growth algorithm to construct and process a Semantic FP-Tree. Just like a normal FP-Growth, it allows
frequent itemset discovery without candidate itemset generation. It is carried out in multiple steps. First, a data structure as Semantic FP-Tree is built in two scans over Semantic Logs. Then, the Semantic FP-Tree is translated into a normal FP-Tree after performing inference on axioms and semantic annotations that are stored at each node with instance data. Then frequent itemsets are extracted from the FP-Tree that was translated from Semantic FP-Tree to get the Semantic Logs. Given below are the formal definitions of algorithms to generate the Semantic FP-Tree for events in Semantic Logs and extracting frequent patterns from it.

| Input: Semantic Logs for Log Interval ΔT |
| Output: Semantic FP-Tree               |
| Method: Semantic-FP-Tree (Semantic Logs, attributes, minimum support) |

//FP Tree construction using 2 passes over dataset
//Pass 1
Scan Semantic Log and find support for each Log Event lei by matching given attributes as SC, SP, NFP, FP.
Discard infrequent events with support less than the minimum support given.
Sort frequent events in decreasing order based on their support.

//Pass 2
Read each log set at a time and map it to a path after translating with each of the events in parent nodes.
Use fixed order so that paths can overlap when semantic log
sets share attribute values.
Maintain pointers between nodes containing the same attribute values.

**Table 9: Semantic FP-Tree Generation Algorithm**

<table>
<thead>
<tr>
<th>Input: Semantic Logs represented by Semantic FP-tree constructed and translated according to previous algorithm, and a minimum support threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: Set of frequent patterns of events occurring together</td>
</tr>
<tr>
<td>Method: Semantic-FP-Growth(Semantic-FP-tree, null)</td>
</tr>
<tr>
<td>1. if the Tree contains a single prefix path then (Mining single prefix-path FP-tree)</td>
</tr>
<tr>
<td>1a. let P be the single prefix-path part of Tree</td>
</tr>
<tr>
<td>1b. let Q be the multipath part with the top branching node replaced by a null root</td>
</tr>
<tr>
<td>1c. for each combination (denoted as ß) of the nodes in the path P do</td>
</tr>
<tr>
<td>1c1. generate pattern ß ∪ a with support = minimum support of nodes in ß</td>
</tr>
<tr>
<td>1c2. let freq pattern set(ß) be the set of patterns so far generated</td>
</tr>
<tr>
<td>2. else let Q be Tree</td>
</tr>
<tr>
<td>2a. for each item ai in Q do (Mining multipath FP-tree)</td>
</tr>
</tbody>
</table>
2a1. generate pattern $\beta = ai \cup a$ with support $= ai$.support
2a2. construct $\beta$’s conditional pattern-base and then $\beta$’s conditional FP-tree Tree $\beta$
2a3. if Tree $\beta \neq \emptyset$ then
2a3a. call Semantic-FP-growth(Tree $\beta$, $\beta$)
2a4. let freq pattern set(\(Q\)) be the set of patterns generated
3. Return (freq pattern set(P) \(\cup\) freq pattern set(Q) \(\cup\) (freq pattern set(P) \(\times\) freq pattern set(Q))

Table 10: Semantic FP-Growth Algorithm

The developed process involves generating Semantic FP-Tree, performing inference on axioms as semantic annotations to nodes of the Semantic FP-Tree and generating frequent itemsets of different events based on attributes given to the algorithm. We use these frequent itemsets to perform ranking of Web Services. The association rule mining on Semantic Logs is performed after several intervals of time to keep association rules for frequent itemsets up-to-date. Whenever a user application submits a request to discover Web Services, we use the discovered association rules to perform ranking. The discovered association rules contain correlations among different Log Events based on attributes like SC, SP, NFP or FP. The usage of such association rules in ranking Web Services brings different benefits. First, association rules are based on probability and statistical techniques as described above, this leads to taking into account overall
preferences of user applications and SCs from past invocations rather than just looking for a smaller subset only. Second, Web Services are provided by Service Providers SPs and used by user applications as Service Consumers SCs where SPs and SCs are isolated from each other over the Web. It calls for finding out any hidden associations as well as dependencies between different Web Services based on different attributes so that all possible Web Services could be considered that could help in fulfilling user requirements.

Figure 11: Semantic FP-Tree of items in Semantic Logs shows a Semantic FP-Tree that is constructed with items as Log Events, SC, SP or WS, using the definitions and algorithms mentioned in this section.

Figure 11: Semantic FP-Tree of items in Semantic Logs
After building association rules from Semantic Logs for different items and attributes, these association rules are used to rank the list of discovered Web Services to facilitate the final ranking process of the Web Services. The final ranking algorithm is outlined as follows.

<table>
<thead>
<tr>
<th><strong>Input:</strong></th>
<th>Set of discovered WS, set of Association Rules AS and Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output:</strong></td>
<td>Ranked list of Web Services</td>
</tr>
<tr>
<td><strong>Method:</strong></td>
<td>RankingWS (Set of WS, set of AS, Goal)</td>
</tr>
<tr>
<td></td>
<td>For each Web Service WS, find confidence for its attributes from attributes required in Goal from the set of related Association Rules AS</td>
</tr>
<tr>
<td></td>
<td>Calculate average confidence for each of the Web Services WS</td>
</tr>
<tr>
<td></td>
<td>Sort the list of Web Services based on the average confidence</td>
</tr>
</tbody>
</table>

**Table 11: Final Ranking Algorithm based on Association Rules**

The above mentioned algorithm for final ranking produces a sorted list of discovered Web Services from which a top N number of Web Services are returned to the user application to select one of the Web Services to be invoked. In the next section, we present experiments for evaluating the proposed solution and analyzing the results.
4.6 Evaluation and Results

Association Rules are discovered and generated after processing and mining Semantic Logs using our proposed approach which is a semantic extension to FP-Growth. Once the association rules are produced, the discovered set of Web Services are then matched and ranked accordingly.

4.6.1 Data Set and Experimental Setup

There is no prescribed dataset or available set of datasets using which we could validate Web Services discovery and especially perform validation of the ranking for such Web Services. However, some of the related works like [67] and [72] have collected Web Services data by proactively crawling Web Services over the Web. Similarly, Seekda.com also provides a Web Services based search engine which provides a crawled set of Web Services over the Web. However, such datasets are not publicly available. We used and adapted a dataset from [73] and www.webservicelist.com which provides different parameters including functional and non-functional properties of Web Services. We had up to 500 Web Services in our dataset which is enough to perform experiments and validate our approach as this number is comparable to the total number of Web Services that we may have over the Web up to date [67]. We have carried out some pre-processing on this dataset in order to be able to perform and validate our proposed solution. The experiments were carried out on Intel Core 2 CPU 2.40 GHz, with 4 GB of RAM, and on Microsoft Windows 7, 32-bit operating system. We used Weka (www.cs.waikato.ac.nz/ml/weka/) in order to perform Association Rule Mining on the data derived from the Semantic Logs.
4.6.2 Snapshots of Semantic Logs and Association Rules

In this section, we present some of the snapshots of the case-study dataset and the application that we used for our evaluation in the context of currency exchange Web Services. This case-study application has multiple components that execute concurrently in order to process user requests for discovering, ranking and finally invoking Web Services. Users initiate their requests through a component called Communication Manager. Matchmaking of user requests with the available Web Services is carried out by a component called Discovery Manager. Requests for invocation of Web Services by users are carried out by a component called Invocation Manager.

```
wsmlVariant
    _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

namespace { _"http://www.example.org/ex1#",
    wsml _"http://www.wsmo.org/wsml/wsml-syntax#",
    ex _"http://www.example.org/ex2#"}

ontology _http://www.example.org/ex1

startAnnotations
    ex#EventID hasValue 656218
    ex#EventName hasValue "Search for Foreign Currency Exchange WS"
    ex#TimeStamp hasValue _date(2013,04,20:00:02:01)
    ex#EventStatus hasValue "Success"
    ex#InboundComponents hasValue {CommunicationInterface}
```
ex#OutboundComponents hasValue {DiscoveryManager}
ex#Context hasValue "Discovery Request"
ex#KeyValuePairs hasValue {Input = ex:Currency:USD}
ex#KeyValuePairs hasValue {Output = ex:Currency:CAD}
ex#KeyValuePairs hasValue {PreCondition = value>0}
ex#KeyValuePairs hasValue {Effect = WebService}
ex#KeyValuePairs hasValue {NFPPrice = High}
ex#KeyValuePairs hasValue {NFPQoS = High}
ex#KeyValuePairs hasValue {Capability = ex:CurrencyConversion}
endAnnotations

startAnnotations
ex#EventID hasValue 656219
ex#EventName hasValue "Response for Foreign Currency Exchange WS"
ex#TimeStamp hasValue _date(2013,04,20:00:02:11)
ex#EventStatus hasValue "Success"
ex#InboundComponents hasValue {DiscoveryManager}
ex#OutboundComponents hasValue {CommunicationManager}
ex#Context hasValue "Discovery Response"
ex#KeyValuePairs hasValue {Input = ex:Currency:USD}
ex#KeyValuePairs hasValue {Output = ex:Currency:CAD}
ex#KeyValuePairs hasValue {PreCondition = value>0}
ex#KeyValuePairs hasValue {Effect = WebService}
ex#KeyValuePairs hasValue {NFPPrice = High}
ex#KeyValuePairs hasValue {NFPQoS = High}
ex#KeyValuePairs hasValue {Capability = ex:CurrencyConversion}
ex#KeyValuePairs hasValue {URL =
Table 12: Sample Semantically Formalized Log Events for a Discovery Request

<table>
<thead>
<tr>
<th>Event</th>
<th>Event Status</th>
<th>NFP Price</th>
<th>NFP QoS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Failure</td>
</tr>
<tr>
<td>Low</td>
<td>Failure</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Low</td>
<td>Success</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Low</td>
<td>Failure</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

Table 13: Sample Association Rules Found

{ NFPPrice=Low, NFPQoS=Low } -> { EventStatus=Failure }

{ EventStatus=Failure, NFPQoS=Low } -> { NFPPrice=Low }

{ Input = ex:Currency:USD, Input = ex:Currency:CAD } -> { Capability = ex:CurrencyConversion }

{ Capability = ex:CurrencyConversion, NFPQoS=Low } -> { NFPPrice = High }

{ NFPPrice=High, NFPQoS=High } -> { EventStatus=Success }

{ TimeStamp greaterThan _date(2013,05,18:00:00:00), TimeStamp lessThan _date(2013,05,18:01:00:00), URL= URL:http://www.example.com/ccWebService } -> { EventStatus=Failure }
Table 12: Sample Semantically Formalized Log Events for a Discovery Request shows some of the Semantic Logs that are stored while carrying out the process of discovering Web Services that can convert a given currency in US dollars into Canadian dollars. The two events shown record discovery request as well as corresponding response using Semantic Logs.

Semantic Logs are processed using our proposed solution described in the previous section to extract and discovery association rules which are then used during the process of ranking of Web Services. The logs being semantically formalized help during the processing and mining of the logs to discover association rules. If the logs are not well-structured and semantically formalized, like in traditional applications where logs are rather unstructured and not formalized, it makes it hard to process and mine the logs in order to get better. We found out that it was hard to interpret unstructured logs and to use such logs in analytical solutions to perform any analysis in the logs. On the other hand, once we used semantically formalized logs generated for the same situation, it was easier and meaningful to use such logs to process and mine them as well as to discover association rules which are then used in ranking the discovered Web services. Table 13: Sample Association Rules Found shows some of the association rules that were discovered and used for the ranking purpose.

### 4.6.3 Analysis and Discussion

We conducted a number of tests on the dataset used using our proposed solution. Our evaluation results include algorithmic complexity analysis, precision as well as accuracy, stability, robustness and efficiency. Regarding the complexity analysis of our approach
for ranking Web Services, the algorithm takes \( O(n) \) in order to scan the Semantic Logs and generate the Semantic FP-Tree. Once the Semantic FP-Tree is built, it takes \( O(n) \) to translate the Semantic FP-Tree into a normal FP-Tree. All the data is represented in the form of FP-Tree. This requires each path in the tree to be at least partially traversed the number of items existing in that tree path. Therefore, this leads to the complexity of the depth of the tree path as well as the number of items in the header [74]; association rules are discovered using our proposed extension to FP-Growth. Finally, ranking of the Web Services is also carried out by having one pass over the list of the discovered Web Services based on the association rules.

While evaluating our approach for ranking, we started with comparison against a naïve discovery engine for Web Services that does not use any optimization or ranking technique. We compared the behavior of both approaches and found out that the naïve discovery engine has to go through the descriptions of all the Web Services, whereas, our proposed approach short lists and ranks Web Services to find out the best one and hence it requires to process a smaller set of Web Service descriptions. The naïve discovery engine has to process the whole search space which makes its processing time proportional to the number of Web Service descriptions available irrespective of the number of Web Services that may be able to fulfill user requirements. We used and adapted a significantly extensive test design in order to make statistically firm statements on the behavior of traditional naïve discovery approach as well as our own proposed approach. We performed several repetitive test runs for search spaces for up to 500 available Web Services descriptions out of which only a few of the Web Services could match user requirements.
Figure 12: Comparison of variance for number of Web Services

Figure 12: Comparison of variance for number of Web Services presents the variance of the comparison between our proposed approach, the naïve discovery engine as well as another ranking approach [75]. It is evident that our proposed solution could limit the search space by performing the ranking, and even better than the other ranking approach. On the other hand, the traditional discovery engine had to carry out search into almost all the given search space.

The next metric used for the evaluation of our proposed approach is ‘precision’. Precision means the ratio of correct Web Services out of all the Web Services retrieved. Precision is defined as follows:

\[
\text{Precision} = \frac{\text{CorrectWS} \cap \text{RetrievedWS}}{\text{RetrievedWS}}
\]
CorrectWS refers to set of Web Services that actually matches user requirements. Whereas, RetrievedWS refers to set of Web Services that are actually discovered and ranked. After calculating the precision for each test run, using the above mentioned formula, we calculate Mean Average Precision (MAP) as the mean of the average precision scores for each Web Service discovery and ranking task.

Table 14: Comparison based on Precision presents the precision calculated for different test runs in three cases, i.e., the naïve approach (without using our approach), the other ranking approach and our proposed approach, as case 1, case 2 and case 3, respectively. We noticed that for Web Services search query involving lesser number of Web Services as retrieved had higher precision rate both, with and without using our proposed solution. Queries involving higher number of Web Services to be retrieved showed significant difference in precision. Table 15: Comparison based on MAP provides an overview of the Mean Average Precision calculated for different test runs, i.e., the naïve approach without using any ranking techniques, the other ranking technique and our proposed approach for ranking, as case 1, case 2 and case 3, respectively. We noticed that we had lower Mean Average Precision for validation of ranked results because the naïve approach has to go through the whole search space. Whereas, ranking approaches case 2 and case 3 got to pre-filter Web Services. Our proposed approach pre-filtered Web Services using association rules and then perform discovery and ranking on a smaller search space. We further noticed a higher Mean Average Precision for results using our proposed solution and the need to perform discovery and ranking on a small targeted as well as relevant search space.
In most of the cases during our experiments, precision was found to be reasonably good. We also found out that the overall accuracy depends upon how accurately Service Providers modeled Web Services as well as how accurately Service Consumers annotated their requests using Functional and Non-Functional Properties. Although our approach performed better compared to existing approaches, it still could not achieve 100% precision, which is of course impossible as achieving highest level of precision would only be under ideal circumstances which cannot exist in real-life scenarios. Our approach still performed better in terms of precision. It also helps in reducing the search space which eventually reduces the time required to perform discovery and ranking. This also reduces the overall variance factor for different test runs. Our proposed approach is eventually based on our earlier work [8] [13] on trying to achieve a suitable trade-off between the accuracy required vs time-based efficiency of the matchmaking and ranking mechanism by partially utilizing semantics that keep data well-expressed and well-structured and makes it easier for data mining based approaches to use it rather than only focusing on modeling Web Service descriptions with overly complex semantics or trying to employ data mining solution on unstructured as well as dispersed data. We believe that our proposed solution is practical for real-life scenarios as Service Consumers and Service Providers find it easier to model requests as well as Web Service descriptions using Non-Functional and Functional properties [13].

<table>
<thead>
<tr>
<th>Test Runs</th>
<th>Precision for case 1</th>
<th>Precision for case 2</th>
<th>Precision for case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Run 1</td>
<td>0.41</td>
<td>0.60</td>
<td>0.92</td>
</tr>
<tr>
<td>Test Run 2</td>
<td>0.42</td>
<td>0.53</td>
<td>0.84</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>Test Run 3</td>
<td>0.80</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Test Run 4</td>
<td>0.19</td>
<td>0.24</td>
<td>0.27</td>
</tr>
<tr>
<td>Test Run 5</td>
<td>0.52</td>
<td>0.65</td>
<td>0.76</td>
</tr>
<tr>
<td>Test Run 6</td>
<td>0.80</td>
<td>0.81</td>
<td>0.89</td>
</tr>
<tr>
<td>Test Run 7</td>
<td>0.53</td>
<td>0.64</td>
<td>0.90</td>
</tr>
<tr>
<td>Test Run 8</td>
<td>0.71</td>
<td>0.71</td>
<td>0.83</td>
</tr>
</tbody>
</table>

Table 14: Comparison based on Precision

<table>
<thead>
<tr>
<th>Mean Average Precision (MAP)</th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.55</td>
<td>0.64</td>
<td>0.78</td>
</tr>
</tbody>
</table>

Table 15: Comparison based on MAP

The usage of association rule mining with Semantic Logs helped us in two fold manner, i.e., (1) semantic logs helped in providing well-structured and formalized data from which it was easier for our technique to deduce and collect information, and (2) the association rule mining approach helped in finding out potential benefits and drawbacks of using some Web Services in certain scenarios; this helped us in pre-filtering Web
Services to have a smaller and more targeted search space and hence lead to more efficient and effective ranking to find the required Web Services.

We have found out that semantic annotations to Web Services are of high novelty if used reasonably with properly tuned and adapted reasoning and mining process. As a next step, we will investigate and build further hybrid techniques involving semantic annotations and data mining to address more issues for enhanced monitoring and management of Web Services as well as related applications execution.

4.7 Conclusions

In this chapter, we proposed a unique approach for ranking and adapting Web Services using Association Rule Mining based on our proposed Semantic Logs as well as Semantic extension of FP-Growth. We analyzed the related and existing approaches and found out that such approaches are limited as such approaches either focus only on semantically formalizing description of Web Services with limited mechanisms to utilize such descriptions or use heuristic based techniques on limited and syntactic data of Web Services for ranking and adaptation of Web Services. Such approaches also merely take into account past interaction of Service Consumers and Service Providers.

Our proposed approach allows semantically formalized representation of logs during Web Service execution which are then used to perform ranking and adaptation of the discovered Web Services. This hybrid approach of partially using semantic annotations to Web Services combined with semantically adapted FP-Growth for Association Rule Mining allows the preprocessing of requests for searching Web Services. This helps in
improving Web Service selection experience from performance as well as precision perspective.

We also presented our experimental results and showed how the trade-off of partially using semantics with semantically adapted Association Rule Mining techniques helps in improving Web Services selection. Our next steps are to design and develop more data mining techniques that could be adapted to semantically formalized data to further enhance the management of Web Services and related applications execution.
CHAPTER 5: REDUCING PROBLEM SPACE USING BAYESIAN CLASSIFICATION ON SEMANTIC LOGS FOR ENHANCED APPLICATION MONITORING AND MANAGEMENT

Monitoring and management of large scale applications has always been a crucial and complex task. Enormous research efforts have been carried out towards making the process of monitoring and managing applications efficient, effective and automated. However, the process still stays complex, lacks efficiency and effectiveness because execution workflow representation and logging (outcome from real-time execution) is rendered out in a syntactic and unstructured manner. This makes the information quite limited and requires manual interpretation and hence makes the monitoring and management process slow, cumbersome and hard. We propose our solution by semantically (highly structured, formalized and expressive) modeling of execution workflow and logs, and then using adapted Bayesian Classification based inference technique to process formalized logs to help in enhancing the process of monitoring and management by reducing the problem space. Our hybrid approach of partially using semantics to formalize log and workflow data, and adapted classification technique combines the best of both. Semantics help in providing high-level of precision, structure and expressivity to execution workflow and logs. Such kind of formalized data can be used in an effective manner to effectively interpret and process highly structured information from the generated logs during the execution by classification technique to

reduce problem space during the process of monitoring and management of applications. This chapter presents review of the related approaches, methodology towards the hybrid solution, design of our proposed solution and implementation, followed by evaluation of our proposed solution on real-life application scenario.

5.1 Introduction

With the increase in complexity of requirements as well as Web-scale open, dynamic and heterogeneous environment, software applications are becoming increasingly large as well as complex in order to be able to fulfill such requirements under such challenging environment. This not only makes the process of building software applications hard, but also monitoring and managing such applications also has become a challenging task. Several platforms have been built that take into account openness, dynamism and heterogeneity of environments for software applications to be built as well as run operations, however, not much focus has been made on the monitoring and management of such applications. Monitoring the execution of software applications is carried out using logging mechanism which is a basic and fundamental part of an application design and development process to allow applications to produce execution logs which is then used by software developers and administrators to monitor the execution and to debug as well as track any events during the application execution. Unfortunately, the process of logging is mostly taken lightly and is not given the expected significant attention that values its important nature and role in monitoring and managing applications. A lot of the effort is spent on design and implementation of software applications but spending some extra effort on the process of logging software applications can significantly improve the
process of monitoring and management of such applications. However, most of the logging mechanisms available today are quite limited. Some important limitations are that the logs are syntactic, not well-structured and have very basic event correlation capability.

Most of the logging mechanisms available so far are based on manual process to use such logs which makes the monitoring and management process hard, cumbersome as well as inefficient. This becomes even more crucial for large and Web-scale applications, where the process of monitoring and management of applications is even more difficult, complex and require maximum level of automation, i.e., Service Oriented System (SOS) which has received considerable attention in the industry [1] as well as academia [2] which aims for software applications to be able to flexibly adapt and deal with dynamic changes that may occur in distributed and large-scale environments like the Web. However, this is not possible with the use of traditional, syntactic and limited logging mechanisms and because of that the ability of monitoring and management mechanisms to sustain in dynamically changing and open environment remains limited [3] [4]. Therefore, currently available middleware based solutions for Service-Oriented Systems known as Enterprise Service Bus (ESB) solutions are limited to a closed environment and to a limited set of components with limited manual monitoring and management.

Our proposed framework allows having a systematic way of logging in software applications and then using such logging for effective and enhanced monitoring and management in such software applications. It is based on highly structured, formalized (semantic) descriptions [5] [6] to the components, events in the logs. Semantic descriptions for the components helps in precisely defining the descriptions of
components; and the semantics are modeled based on widely-accepted standards [3].

As a first step, we have built a model for semantically describing the components and logs. Secondly, we have built advanced log processing mechanism that processes semantically formalized logs to monitor the execution of such software applications by adapted Bayesian Classification technique [76]. Applications based on such Web-scale platforms are often based on multiple components which may communicate with each other to execute transactions. In such cases, it is crucial to find out the right event and track it in all the application across multiple components which brings the necessity that logging information should be modeled precisely and with higher level of expressivity. Therefore, Semantic Logging as semantic annotations to components, execution workflow and logs have been proposed. Semantics can be utilized for finding, monitoring and managing the components required in execution workflow. Semantic Logging allows highly structured, expressive and machine interpretable logs to be produced during the execution that are used for monitoring and managing such applications. Highly structured and expressive nature of the log information enables the monitoring and management process to be automated and such logs are utilized by Data Mining based techniques, i.e., Bayesian Classification to monitor execution, track events and deduce knowledge that helps in application monitoring and management.

The rest of the chapter is organized as follows. Section 2 presents related work in the area of automated monitoring and management of applications. Section 3 presents proposed solution of Bayesian Classification on Semantic Logs for reducing problem space in monitoring and management of software applications. Section 4 presents
experiments and discusses evaluation of results as well as compares it with that of existing solutions. Section 5 presents conclusions.

5.2 Related Work

We have found a number of related works done in the area of enhanced the monitoring and management of applications. These works span from monitoring of stand-alone applications to monitoring of large-scale applications, middleware solutions and service based systems [32]. Below we discuss some of the related approaches.

In [77], Web usage mining has been proposed that plays an important role in the personalization of Web services. Users’ access to pages of the Website is separated into user sessions in this approach. The required user sessions are then extracted from the log of the hosting Web Server. The authors consider a ‘process-centric view’ that defines Web mining as a sequence of tasks. Second is a ‘data-centric view’, which defines Web mining in proportion to the types of Web data that was used in the mining process. In this work, authors’ proposal of Web mining is merely a parsing issue of logs and does not focus on formalizing or even structuring logs. The proposed solution is limited to use syntactic information from unstructured logs which can provide basic level of classification to discover different types of usage patterns from users.

In [78], the authors proposed to use classification for identifying interesting visitors of a website by performing classification on Web logs. Web log classification in this case is also merely parsing and classifying of logs from a Web server. Attributes taken into account for classification are temporal attributes, page attributes and communication attributes. Authors found out during the classification process that the lower the recall
and precision are, the more important the attribute is, i.e., if such attribute is removed, the accuracy drops. However, the issue with this approach is that it merely parses and discretizes logs from the Web server for different users visiting the website. It does not attempt to provide any standardized formalism or structuring of logs.

Resource Description Framework (RDF) has also been used to enable semantic logging. In [36], RDF has been proposed to be used for formalizing logs which can be searched and analyzed to gain a further understanding of the system of interest. However, this approach does not attempts to build any such mechanism that could use logs represented using RDF.

Splunk [86] is a comprehensive framework for semantically logging and mining information from application execution to perform enhanced monitoring and management of applications. Authors argue that logs (especially unorganized logs) can be a hassle to deal with as there is no real structure, nor any standardized format. Such logs may become useful once stored with proper structure. Analyzing such logs may help in finding problems, get more insight information about IT infrastructure for an enterprise, behavior of users, and identify potential problems. However, this approach merely uses some structuring techniques and does not focus on formalism and standardization of logs which could be used with advanced data mining techniques to perform rigorous analysis on such logs.

In addition to these solutions, several semantics based solutions have been proposed for automated Web Service execution. These approaches do not focus on semantic logging in particular. However, they are still useful to review as the nature of the problem is very similar as these approaches semantically formalize Web Services to enable
automated discovery, selection, composition and execution. Similarly we are seeking to semantically formalize logging to enable extensive analysis of logs to allow enhanced and automated monitoring of applications.

Ontology Web Language for Services (OWL-S) [21] [38], part of the DAML Program [40], specifies a set of ontologies based on OWL to describe different aspects of a Semantic Web Service [26]. It includes a set of ontologies which only allow describing Web Services formally but do not leave any recommendation for formally representing execution events and logs. Another promising approach known as Web Service Modeling Framework (WSMF) [26] was proposed as a fully-fledged framework to model Semantic Web Services [4]. It gives two complementary principles (maximal de-coupling and scalable mediation [41]) and four elements (Ontology, Goal, Web Service and Mediator) to model any aspects related with the services’ definition and usage. To finally enable the framework, a set of corresponding technologies have been developed, such as the modeling ontology WSMO [3], the description language WSML [5], and the execution environment WSMX [28]. It includes a basic micro-kernel [44] and grounding support [49] with existing Web Service standards. This approach does take care of formally modeling Web Service descriptions and user requests, but do not leave any recommendations for modeling event logs. Semantic Web Services Framework (SWSF) is a specification produced by the SWSL Committee of the Semantic Web Service Initiative (SWSI). SWSF has its own conceptual model Semantic Web Service Ontology (SWSO) and relevant language Semantic Web Service Language (SWSL). SWSO has been influenced by OWL-S and adopts its three ontologies, namely service profile, model and grounding. The key contribution of SWSO is its rich behavioral process model. With
such extensions, SWSO supports more powerful descriptions and reasoning on Web Services [79]. This approach still focuses only on formalizing description of Web Services as well as user requests but lacks on formal description of events in logs. Same is the case with Web Service Description Language - Semantics (WSDL-S) which proposes a mechanism to augment WSDL with semantics, in particular focusing on the services’ functional descriptions. WSDL-S has the advantage of attaining semantics building on existing Web services; in the meantime, it does not prescribe any language for semantic descriptions [23].

Approaches like Adiscon LogAnalyzer [55] and WebLog Expert [56], GitHub Log-analyzer [58], Retrospective Log Viewer Software [59] and XpoLog Log Analysis Platform [61] provide practical tools to analyze log data. However, these approaches do not make any attempt in structuring the logs. Also the data mining and analysis techniques employed to mine the log data are also naïve and only provide basic performance reports about software execution. SysLog Monitor [57] provides rule based method to access and read the logs, but still does not make any attempt to structure or formalize logs. Also, it applies basic rule based monitoring techniques to generate reports like host system performance analysis, identifying faults in execution and identifying different types of events in application execution.

All these approaches have made significant efforts towards automated execution and monitoring, but are limited in different aspects. For example, all the Semantic Web Service based solutions that have been discussed are too focused on formalism on Web Service descriptions and user goal descriptions and do not specify issues related to execution monitoring. Other approaches also have been focused on specific log parsing
or mere structuring issues and hence are limited. Other approaches like Splunk.com and semantic logging using RDF have been too basic and limited in terms of formal semantics used to semantically model logs. It limits the expressivity of log events, relationships among log events and constraints in the logs. Our proposed solution takes into account higher formal semantics used in Semantic Web Services and uses it in a generic way to enable semantically formalized log that helps in enhanced monitoring and management of large-scale and complex applications.

5.3 The Proposed Solution

Our proposed solution includes building semantic models to formally describe components as well as events in the logs during application execution. This allows having more explicit information available with higher level of expressivity. Advanced Data Mining technique, e.g., classification is used to process highly structured information about components and logs. Our solution is unique as it followed a hybrid approach to (1) make the information highly structured and formalized, and (2) use classification technique to process the information, hence combines the best of both. It solves the problem in a two-fold manner. First, it provides semantic descriptions to the components and logs, so that the information about components and logs will be available more explicitly and with higher level of expressivity. Second, it uses classification to process the highly structured information about components and logs.
Our proposed models for semantically describing components and logs contain necessary information about Components and Log Events that are usually required by mining and analysis techniques in the process of application monitoring. Such information helps in tracking inflow and outflow of input data and output data from individual components within an application. It also takes into account information that a Log Event should contain, including contextual information and application specific information as key-value pairs. The highly structured, formally described nature of the information enables the algorithms and methodologies to be able to monitor and manage...
the components within applications. It involves reasoning solution based on Bayesian classification to process semantic descriptions of the components and correlate it with execution-workflow and execution logs. Figure 13: Overall scenario for Enhanced Monitoring and Management of Large Scale Applications depicts the overall scenario.

![Hierarchical representation in Semantic Logs](image)

**Figure 14: Hierarchical representation in Semantic Logs, adapted from [80]**

A layered view of hierarchical representation of concepts, objects, attributes and their relationships is depicted in Figure 14: Hierarchical representation in Semantic Logs, adapted from [80], which is inspired from the Concept Algebra [80]. Concepts at knowledge level are seen as processes based on business logic, which are further individualized at object level as Log Events, followed by attributes of such Log Events at
attribute level. Different Log Events may belong to different Concepts and similarly
different attributes may belong to different Log Events. Relationships between Concepts,
Log Events and Attributes are also depicted in the figure which can be formalized using
semantic expression as \( R(I, J) \), where \( I \) and \( J \) could be Concept, Log Event or Attribute.

Given below are definitions to formally define and represent aspects of our proposed
Semantic Logging.

**Definition 1 (Log):** Log is a footprint of a software application recorded during its
execution in a given time period.

**Definition 2 (Component - C):** Consider \( C \) as a component in an application that may
be involved in the execution of an event. It prescribes meta-model for any component to
contain necessary information. It can be represented as a tuple:

\[
C = (Binding, Type, Inputs(h), Outputs(j), Events(l), EnvironmentVariables(p))
\]

*Binding* contains information about protocol binding and protocol information for
invocation of the component. *Type* contains information about the different possible kinds
of components an application may have which could be defined and implemented by
application developers. *Inputs(h)* represents \( h \) key-value pairs that a component may
accept as input. *Outputs(j)* represents \( j \) key-value pairs that a component may accept as
output. *Events(l)* contain \( l \) events that a component might be involved in executing,
including state of component and any action that may need to be taken. *EnvironmentVariables(p)* contains \( p \) possible variables that may contain information
about the computing and storage environment that a component may encounter during the execution.

**Definition 3 (Log Event - LE):** Let LE be Log Event that prescribes meta-model for any event in the log to contain necessary information. It can be represented as a tuple:

\[
LE = (EventID, EventName, TimeStamp, EventStatus, InboundComponents(k), \text{OutboundComponents}(m), \text{Context}, \text{KeyValPairs}(n))
\]

where EventID is a unique identifier for any event defined for a software execution; EventName is a human readable name of an Event with a unique identifier. TimeStamp contains exact date and time of any update that may take place for an event. InboundComponents(k) represents k inbound components that may affect an event during the execution. OutboundComponents(m) represents m outbound components that may get effected by an event during the execution. Context represents the application execution context out of many possible contexts an application execution may have and defined by application developer. KeyValPairs(n) represent n Key-Value pairs that may contain application specific data and variables to be logged.

**Definition 4 (∆T):** ∆T denotes a Log Interval which is a set of Log Events LE that occur in a given time interval.

\[
LEs = \{ LE_1, LE_2, LE_3, \ldots LE_n \}
\]

**Definition 5 (I):** Consider I as a set of items in a given Log Event LE.
\[ I = \{ i_1, i_2, i_3, \ldots i_n \} \]

Let \( i_x \) and \( i_y \) be items in a Log Event LE with particular characteristics or a Component with particular Functional Properties (FPs) or Non Functional Properties (NFPs). In the following subsections, we formally define semantic models for components as well as logs.

**Definition 6 (FP – Functional Property):** FP denotes Functional Property that could be an item in a Log Event or Component.

\[ \text{FPs} = (\text{Inputs}(h), \text{Outputs}(j), \text{Capability}, \text{Interface}) \]

where Inputs(h) represents \( h \) key-value pairs that a component may accept as input. Outputs(j) represents \( j \) key-value pairs that a component may accept as output. Capability may include any pre-conditions and post-conditions which represent information space before and after execution as well as assumptions and effects which represent state of the world before and after execution. Interface includes Choreography and Orchestration which describe behavior and interaction patterns.

**Definition 7 (NFP – Non Functional Property):** NFP denotes Non Functional Property that could be an item in a Log Event or Component.

\[ \text{NFPs} = (\text{NFPLocation}, \text{NFPPrice}, \text{NFPTrust}, \text{NFPQoS}) \]

We take into account the following aspects in Non Functional Properties: (1) Location details of a Component or Log Event, (2) Quantitative description of Pricing that is involved in a Component or Log Event, (3) information required to describe trust information in description of a Component or a Log Event, and (4) Quality of Service (QoS) which is the level of rating for a Component or Log Event.
5.3.1 Semantic Model for Components and Logs

This section presents our models for semantically describing component descriptions and log events. These models prescribe overall template of how the Components and Log Events are modeled in a standardized manner. A component is a part of an application that encapsulates a functionality based on implementation and an interface that is used to provide input to the component to get the functionality. An implementation neutral description to this component is provided in the application which is used by the execution engine to find out the component and to communicate with it. On the other hand, logs are produced by applications that contain footprint of the application execution. We propose semantic annotations to the component description, as well as the logs that are produced by the applications.

The proposed model for semantic description of components is based on Definition 2 in this section. It includes obvious information about inputs and outputs. It also precisely contains information about the functionality this component provides in the context of a particular domain. The model for semantic descriptions of components also allows to precisely specify the conditions under which the component should be used (i.e., if some particular event occurs) and allows having precise information about a component and the action this component should perform if a particular event occurs.

The proposed model for semantic description of Log Events is based on Definition 3 in this section. The description of Log Event is also connected to the description of Components. It contains information about Components that originate a Log Event or the Component where Log Events end. It includes the context in which the event has taken
place. It is based on formally defined context vocabulary by domain expert based on the type of application to be monitored. Events have unique identifier to be distinguished among different events, names, date/time (i.e., timestamps) of events as well as status as compulsory fields. Status of an event is also derived formally from a defined vocabulary by a domain expert based on the type of application to be monitored. Each Log Event has a number of key-value pairs to enclose any application specific information. Both semantic models for describing Components and Log Events are correlated with each other in order to have a global view of events of their execution across different components. Events are produced and recorded in a structured way; they are modeled with semantic descriptions. This highly structured and formalized way of recording logs facilitates the execution and monitoring task to enable automated and enhanced monitoring of an application during its execution. Table 16: Formal description of Components and Log Events specifies formal description, inspired from WSMO [3] and using Meta-Object Facility (MOF) [81], for modeling Components, Log Events and related data in Semantic Logs.

```
Class LogEvent

    hasInBoundComponent type Component

    hasOutBoundComponent type Component

    hasContext type Context

    hasEventID type Number

    hasEventName type String

    hasEventTimeStamp type Date:Time
```
hasEventStatus type String

hasKeyValuePair type KeyValuePair

multiplicity = multi-valued

Class Component

hasName type String

hasType type String

hasInput type Input

multiplicity = multi-valued

hasOutput type String

multiplicity = multi-valued

hasState type String

hasAction type String

hasFunctionalProperty type FunctionalProperty

multiplicity = multi-valued

hasNonFunctionalProperty type NonFunctionalProperty

multiplicity = multi-valued

Class FunctionalProperty

hasName type String

hasDescription type String

hasDefinition type Axiom

Class NonFunctionalProperty

hasName type String

hasDescription type String

hasDefinition type Axiom
5.3.2 Bayesian Classification for Semantic Logs

This section presents our solution for using Bayesian classification model for reducing the Problem Space during application execution based on different aspects of Components and Log Events in Semantic Logs. As a first step, we have built naïve Bayesian classifier model which is a simple probabilistic classifier based on applying Bayes' theorem with strong independence assumptions among different classification factors. Such independence is important in the process of classifying problem types based on values obtained from features, because in the application execution, different factors whose values are independent of values of all other factors are taken into account for classification. Most of the applications are multi-component in which each component has its own independent execution. Similarly, most of the latest application design
requires components to be hosted as remote and stand-alone services. In such case, it is important to treat values of factors involved in the classification process as independent. We have used naïve Bayesian classification as in recent surveys it has been proved to outperform more current approaches [8] [13]. It mines semantically modeled log and takes the requirements from Log Events and uses Bayesian classifier model (which is trained with sample data) to classify different possible problems that may occur during application execution.

Here we provide a formal description of the Bayesian classifier for the classification of Problems from application execution based on the information extracted from Log Events. We take properties of Log Events as set of features for Bayesian classifier. The dependent class variable is the set of available possible Problems that may occur during application execution. We denote PS as class variable for possible Problems in application execution that may require monitoring and management activities, and LE as Log Event properties. Log Event properties are features of Bayesian classifier based on which the class is determined. These features are determined from semantic log as outcome of the execution of an application.

The Bayesian classifier takes into account the presence or absence of a particular feature and determines the class (PS) as a possible problem that is determined based on the features as Log Events (LE). For a general solution, there are $n$ numbers of possible properties that may be found in Log Events LE and they will be denoted as $LE_1, LE_2, LE_3 \ldots LE_n$. Based on this, our classifier model will be based on conditional probability of PS class variable over the set of available properties of Log Events. This is expressed in formal notation as follows:
Using the theoretical foundations of Bayesian classifier, the joint probabilistic model will be interpreted as below:

\[ P \left( PS \ | \ LE_1, LE_2, LE_3 \ldots LE_n \right) \ldots (1) \]

\[ P \left( PS \right) \ast \prod_{i=1}^{n} P \left( LE_i \ | \ PS \right) \ldots (2) \]

Given the above interpreted independence assumptions, the conditional distribution over the class variable PS are expressed as follows:

\[ P \left( PS \ | \ LE_1, LE_2, LE_3 \ldots LE_n \right) = \frac{1}{Z} \ast P(PS) \ast \prod_{i=1}^{n} P \left( LE_i \ | \ PS \right) \ldots (3) \]

Z is the scaling factor dependent only on LE1, LE2, LE3 … LEN, from values of feature variables that are known through the execution log generated during application execution as per our prescribed semantic model for Log.

Parameter estimation can be performed by calculating relative frequencies from the training dataset. These are taken as maximum likelihood estimates of the probabilities while values of the properties of Log Events are well discretized due to the fact that our proposed solution enables properties of Log Events to be well-defined and well-structured as per the proposed semantic model for Log Events. As per our derivation of Bayes probabilistic model, the naïve Bayes classifier combines the model (as mentioned above) with a decision rule which is as simple as, selecting the most probable. Therefore, the Bayesian classifier can be expressed as follows:

\[ \text{Classify} \ (LE_1, LE_2, LE_3 \ldots LE_n) = \text{argmax} \]

\[ P \left( PS = ps \right) \ast \prod_{i=1}^{n} P \left( LE_i = le_i \ | \ PS = ps \right) \ldots (4) \]
PS is the overall class variable, whereas ps is any particular value in the class. The same applies to \( LE_i \) and \( le_i \). Each distribution can be independently estimated as a one dimensional distribution. This helps in handling the datasets which may continuously increase and scale with more number of features. To represent properties of Log Events, we use discrete parameters as naïve a Bayesian classifier is dependent on the usage of discretized values of features.

We have also used Bayesian Network Classifier \([82]\) to take into account dependencies between Log Events.

\[
P(LE_i, F, C_i) = \prod P(LE_i) \times P(F|LE_1, LE_2, \ldots, LE_n) \times \prod P(C_i|F) \quad \ldots (5)
\]

Bayesian Network Classifier in Equation (5) is based on the Bayesian Network principle which is given below:

\[
P(X_1, \ldots, X_n) = \prod P( X_i | parents(X_i) ) \quad \ldots (6)
\]

Bayesian classifier uses these values from the features and determines the class or outcome as possible problem type which helps in reducing the Problem Space (PS). There are different types of inferences that can be carried out using our proposed solution of employing Bayesian Classification on Semantic Logs. The first type is diagnostic inference which helps in finding out any possible Log Events (LE) which caused any possible failures (denoted as F) in the Problem Space (PS) with any possible conditions (denoted as Cond). It helps in deriving effect (i.e., a set of occurring Log Events) from cause (i.e., a particular type of failure with any possible conditions). A generic diagnostic inference equation is represented in Equation (7).
Second type is predictive inference which helps in predicting any types of possible failure (denoted as $F$) in Problem Space (PS) with any possible conditions (denoted as $Cond$) that could be caused due to any possible Log Events (LE). It helps in deriving from cause (i.e., a particular type of failure) to effect (i.e., a set of occurring Log Events). A generic diagnostic inference equation is represented in Equation (8).

$$P \left( LE_i \ldots LE_k \mid F \cap \bigcup (Cond_j \ldots Cond_m) \right) \ldots (8)$$

Before carrying out the classification of problem types from the features extracted from Log Events of Semantic Logs, the Bayesian classifier is trained with a sample or training dataset iteratively once the accuracy drops across a certain threshold. Such training dataset is prepared using historical Semantic Log obtained from execution of the application. After training the Bayesian classifier, it is able to classify the problem type using semantic logs generated from the on-going execution of the application being monitored. This kind of automated application monitoring using classification of problem type helps in significantly decreasing the Problem Space to quickly dig down into specific problem and fix it. Experimental results and the analysis of the classification of problem types from processing semantic logs are described in the next section.

5.4 Evaluation and Results

We have performed experiments and evaluated the results based on our use-case application for a financial institution. It uses our proposed way of Semantic Logging and
employs Bayesian classification for processing such logs in order to help reducing the Problem Space to find out or predict any possible upcoming failures during an application execution. The experiments were carried out on Intel Core 2 CPU 2.40 GHz, with 4 GB of RAM, and on Microsoft Windows 7, 32-bit operating system.

The following application specific information has been taken into account: (1) Event-Status which contains information about current status of an event during application execution, (2) Context that contains information about background information about the execution event in the Log Event, and finally (3) one of the key value pairs that contain application specific information, i.e., Transaction Country, value, currency, etc. There are different possible problem types that are identified during the use-case application execution based on adaptive measures that can be taken by the application. Such types of problems can be classified by mining semantically formalized log and respective measures could be taken by the applications fault-handling mechanism by deducing information, thus allowing the application to automatically identify problem type and take actions accordingly to handle or at least mitigate the problem.

We have run tests based on the dataset; we provided the Bayesian classifier initial dataset to perform supervised learning. Once the supervised learning was completed, we further processed the incoming requests based on requests from users containing the values of required properties of the Log Events recorded during the application execution. We discretized the parameter values from the dataset in order to make it available for the Bayesian classifier to process it. Table 17: Outlook of the dataset used provides a discretized overview of the data.
<table>
<thead>
<tr>
<th>Event Status</th>
<th>Inbound Component</th>
<th>Context</th>
<th>Key Value (App data)</th>
<th>Select Problem Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Started</td>
<td>Transaction Manager</td>
<td>Foreign Transaction</td>
<td>China</td>
<td>“Security”</td>
</tr>
<tr>
<td>To be Started</td>
<td>Accounts Manager</td>
<td>National Transaction</td>
<td>USA</td>
<td>“Accounts Database”</td>
</tr>
<tr>
<td>Success</td>
<td>Transaction Manager</td>
<td>Local Transaction</td>
<td>USA</td>
<td>“Accounts Database”</td>
</tr>
<tr>
<td>Failure</td>
<td>Communication Manager</td>
<td>Local Transaction</td>
<td>USA</td>
<td>“Accounts Database”</td>
</tr>
<tr>
<td>Shutting Down</td>
<td>Communication Manager</td>
<td>Foreign Transaction</td>
<td>China</td>
<td>“External Communication”</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Table 17: Outlook of the dataset used**

Given the set of values for Log Event properties, the dataset contains record of the semantically formalized logs containing the Log Events. We used some of the dataset to perform the supervised learning for the Bayesian classifier, and the rest of the data was used to classify and select one out of four possible Problems in the application execution upon any failure occurred, based on the information from properties of the Log Events. We used cross validation to check the level of accuracy of the results obtained from the classifier. In order to use the Bayesian classification mechanism, we used the Weka tool
which is available at the URL: http://www.cs.waikato.ac.nz/ml/weka. It has a collection of machine learning algorithms implemented for data mining tasks.

<table>
<thead>
<tr>
<th>#</th>
<th>Classified Problem Types</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>External Communication</td>
<td>0.88</td>
</tr>
<tr>
<td>2</td>
<td>Internal Communication</td>
<td>0.94</td>
</tr>
<tr>
<td>3</td>
<td>Database Manager</td>
<td>0.71</td>
</tr>
<tr>
<td>4</td>
<td>Customer address validation from foreign station</td>
<td>0.98</td>
</tr>
<tr>
<td>5</td>
<td>Customer id validation from foreign station</td>
<td>0.84</td>
</tr>
<tr>
<td>6</td>
<td>Login failure</td>
<td>0.51</td>
</tr>
<tr>
<td>7</td>
<td>Transaction Timeout</td>
<td>0.79</td>
</tr>
<tr>
<td>8</td>
<td>Gateway down</td>
<td>0.84</td>
</tr>
<tr>
<td>9</td>
<td>External currency conversion</td>
<td>0.90</td>
</tr>
</tbody>
</table>

Table 18: Accuracy in Classification Results

Table 18: Accuracy in Classification Results presents the results of our classification analysis based on the dataset we have. We used one-third of the data for supervised learning, in order to train the dataset. The rest two-third of the dataset was used for testing the classification. The overall Mean Average Precision (MAP) is observed to be 82.11% which is of course based on the accuracy as well as diversity of training dataset. We learned that although, we do not have very high accuracy rate, the time taken in
performing the classification and identifying the problem was highly automated, and helped the fault handling process to identify the possible problems and handle them accordingly. Hence, it is therefore, a trade-off between the accuracy required vs. time-based efficiency to achieve the automated fault handling process during application execution.

Figure 15: Analysis of Problem Type classification

In addition to precision, Figure 15: Analysis of Problem Type classification shows different Problem Types that were classified as faults from mining the Semantic Logs produced during application execution. Each fault is numbered and should be interpreted as per Table 18: Accuracy in Classification Results. The X-axis shows the number of
functionalities that were affected in the application due to a particular fault. The Y-axis shows the number of failures that occurred due to the fault. Whereas, the size of the bubble shows the number of times a fault occurred (i.e., the more a fault occurred, bigger is the size of the corresponding bubble).

Figure 16: Comparison of number of steps in fault detection shows a comparison of the number of steps required to be followed to detect the fault in an application in three different cases: (1) without using any monitoring solution, (2) using another similar solution, and (3) using our proposed solution. The X-axis shows different faults, and the Y-axis shows the number of steps required for detecting the fault. We noticed a reasonable reduction in the number of steps required in detecting the fault as semantically formalized logs with the help of Bayesian classification helped in automatically reducing the problem space which decreased the number of steps required for detecting the issue and the fault.

In comparison to the related work, most of the approaches found to be either focusing only on formalizing or structuring logs, or focusing on employing data mining based approaches for processing unstructured log data for monitoring and managing the applications. We did not find any of the approaches to be comprehensive enough to address the issue of application monitoring and management from both aspects, i.e., to structure and formalize logging as well as employing Data Mining based techniques to process such logs. Because of this lacking, approaches that focus on making the logs structured and formalized are still limited because such approaches do not make use of semantics based formalism but merely try to structure the logs and try to provide a basic level of formalism; they do not address the issue of using such formalized and structured
logs to deduce new information. Some of the related works try to use a certain level of formalism to logs but do not try to make use of it in log processing. Our proposed solution is unique because it is hybrid. It attempts to combine the best of both, i.e., formalizing the logs to make them well structured and highly expressive, and then using Bayesian Classification based technique for making use of such formalism and enabling the monitoring and management of applications using such formalized logs.

![Figure 16: Comparison of number of steps in fault detection](image)

5.5 Conclusions

In this chapter, we proposed a hybrid approach for enhanced and automated monitoring and management of applications by using Semantics with Data Mining. Semantics are used to formalize and structure logs from application execution which are then utilized by Data Mining based approach (i.e., Bayesian Classification) to classify different types of
possible issues. This helps in reducing problem space for application administrators to focus on the problematic part of the application rather than the whole application. We also analyzed and compared existing approaches and found out that such approaches are limited because they either focus only on semantically formalizing the description of logs with limited mechanisms to utilize such descriptions or just focus on using heuristic based techniques on limited, syntactic and unstructured log and other execution related data of applications which makes the process of application monitoring and management limited. Our proposed hybrid approach partially uses semantically formalized and well-structured logs with adapted Bayesian classification to allow for automatically pre-selecting and reducing the problem space and thus helps in improving application monitoring and management experience from the perspective of efficiency and precision. It helps in reducing the number of steps that are required to detect a problem and reach it in order to recover an application from a fault. It further helps in predicting any possible fault or failure that could occur during application execution so that it could be mitigated and avoided. We also carried out experimental evaluation and analyzed results that show how it is better to enable and use semantically formalized logs with Bayesian classification for enhancing and automating application monitoring and management. Our next steps will be use and adapt more data mining techniques to use semantically formalized data to further enhance application monitoring and management.
Monitoring and management of large scale applications is already a complex task because of syntactic and unstructured nature of the execution data. Traditional application monitoring and management solutions focused on employing analysis techniques on unstructured and syntactic log information become limited as unstructured information cannot be well utilized to find out related events information or correlate such information with other related information from applications. Our proposed solution of semantically formalized logging fills this gap by bringing formal semantics and combining it in a meaningful way to enable automated monitoring and management of applications. Such formalized and well-structured log information helps analytical solution to maximally automate the process of monitoring and management of applications. However, while formalizing and structuring the log information, we came across several missing and incomplete data which causes hindrance in this process. In this chapter, we tackle this problem and propose a social network analysis based solution to handle incomplete and missing data from application execution. Possibly compute and use it by our proposed solution of semantically formalizing and structured logs with adapted data mining techniques to enable automated and effective application monitoring.

---

6 Contents of this chapter were published as: Omair Shafiq, Reda Alhajj, Jon G. Rokne, “Handling incomplete data using Semantic Logging based Social Network Analysis Hexagon for Effective Application Monitoring and Management”, in the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (IEEE/ACM ASONAM 2014), 17-20 August 2014, Beijing, China.
and management. We demonstrate from an industrial use-case application how historical data from application execution is stored using semantic logging and utilized with standard social-network analysis techniques to find out missing values in incomplete data and perform application monitoring and management.

6.1 Introduction

With the increase in complexity of user requirements, and computing power, software applications are also becoming increasingly complex and large. This makes the process of application monitoring and management a challenging task, especially when the requirement is to have automated monitoring and management of the application. Logging is a basic and fundamental part of an application design and development which allows an application to produce an execution log which is then used by software developers and administrators to monitor the execution and to debug as well as track any events during the application execution. The process of logging is often taken lightly and is not given the right attention as it deserves. A well-developed logging mechanism always helps in better monitoring and management of application execution. However, most of the logging mechanisms available today are quite limited. Some of the important limitations are that the logs are syntactic, not well-structured and have very basic event correlation capability.

There are a number of solutions available today that attempt to automate the process of monitoring and management of applications. Such solutions are still limited to unstructured data, limited and incomplete information from application execution and hence still require significant manual effort which makes this process hard, cumbersome
and inefficient. The larger the applications are to be monitored and manage, the more significant this problem becomes. Therefore, with very large and web-scale applications, the process of monitoring and management of applications becomes more difficult, complex and demand of maximum level of automation becomes more desirable. Latest and key developments in the area of Web-scale applications are known as Service Oriented System (SOS) [1]. Such systems have received considerable attention from industry [1] as well as academia [2]. With the increase in complexity of user requirements, it is becoming increasingly important that such Service Oriented Systems (SOS) should be able to flexibly adapt and deal with dynamic changes that may occur in distributed and large-scale environments like the Web. However, this is not possible with the use of traditional, syntactic and limited logging mechanisms and because of that the ability of monitoring and management mechanisms to sustain in dynamically changing and open environment remains limited [34, 35]. Therefore, currently available middleware based solutions for Service Oriented Systems solutions are limited to a closed environment and to a limited set of components with limited manual monitoring and management.

We have proposed Semantic Logging [83] which allows applications, especially complex applications like middleware based solutions for services (often called Service Bus), to adapt to the dynamically changing environments and automate the process of execution and monitoring using highly structured, formalized (semantic) descriptions [84] to the components, events in the execution logs. Semantic descriptions for the components and events help in precisely defining the descriptions in a formalized and well-structured manner based on widely-accepted standards [2]. We have built models for
semantically describing the components and events in the logs. Secondly, we have also built processing mechanisms to process semantically formalized logs and monitor the execution by adapting advanced data mining and analytical approaches like classification and association rule mining [84] [85] for application monitoring and web services ranking.

In order to find out interesting information, we model the information from semantic logs in our proposed social network analysis model and perform standard social network analysis techniques to compute any missing data, based on historical execution data. It is based on our previous work on social aspects of personalized ranking for Web Services [13]. Our proposed Semantic Logging approach attempts to formalize the information from application execution but faces hindrance when some of the data is found missing from the application. More details on Semantic Logging can be found in our work described in [85] [84] [87]. In this chapter, we present our solution of handling incomplete and missing data. It is based on standard Social Network Analysis techniques. Social Network Analysis (SNA) based on Graph theory techniques [13] help in analyzing the social network in terms of network and graph consisting of nodes and edges. Nodes are individual actors in social networks. In our proposed solution, we model different items from semantically formalized logs, like Log Events (LE), Components (C), Functional Properties (FP), Non Functional Properties (NFP), Users (U) and Problem Space (PS) as social network. SNA techniques are then utilized after modeling the items in social networks; the target is to compute any missing and incomplete data.

In order to find out incomplete and missing information, we perform social network analysis based computation on the data obtained by mining logs from application
execution. Our proposed solution, using execution logs based on past application execution, foresees the problem of computing incomplete and missing data from different perspectives like correlations between Log Events (LE), Components (C), Users (U), Problem Space (PS), Functional Properties (FP) and Non Functional Properties (NFP).

Under ideal circumstances, information should be available in execution logs from all the given aspects, and therefore, it would be even easier to compute the data and perform monitoring. However, we have found out in real-life scenarios that this is not the case. Not all the information is always available. Application execution comes across missing values and incomplete data which could be because of errors or fault during execution or invalid data submitted by application users. Therefore, our proposed solution will show how it is possible to use partial information from application execution data as logs and use it to compute missing values and compute possible correlations of different log elements with faults, failures and exceptions.

The rest of the chapter is organized as follows. Section 2 presents related work in the area of automated monitoring and management of software applications and outlines pros and cons of such approaches. Section 3 presents proposed solution of using Social Network Analysis using Semantic Logs for finding out missing data based on past execution of application. Section 4 presents application of our proposed solution on industrial case-study. Section 5 presents experiments and discusses evaluation of the results as well as compares them with those of existing solutions. Section 6 presents conclusions.
6.2 Related Work

This section discusses related work in the area of automated application monitoring and management as well as any efforts made towards handling missing values and incomplete data. Some of the approaches use semantic languages for the purpose of formalizing and structuring logs that are recorded during application execution but lack on utilizing such well-structured and formalized logs. Some of the approaches only focus on using data mining based approaches without any attempt to structure or formalize logs and hence are limited to utilize such unstructured and scattered logs.

Approaches like [33] [34] propose to use logs generated from the execution of queries to deduce semantic relationships among different queries to find related queries. Analysis is carried out on a large log of past query execution and relationships among queries is extracted and stored using cover graphs that are defined by authors. Such cover graphs also record the answers that are click by users. The main benefit achieved is faster and efficient computation of answers by using information from past execution of similar and related queries. In this approach authors attempt to use data mining based approaches like the Apriori algorithm for carrying out association rule mining, but do not attempt to structure and formalize the logs.

In [35], the authors attempt to build a framework for semantic logging that enables structuring of logs from the perspective of agent-based distributed systems for chemical incident response. Semantics are utilized in this approach to help, using relationships that are defined between different but related events of the application, in the reconstruction of sequence of events that occurred during response to particular chemical incident. This structured logging also helps in having a detailed view of the system execution trace, as
well as of agents' decisions taken at various decision points during the incident management workflow.

Resource Description Framework (RDF) as one of the key building blocks towards Semantic Web is also used in an attempt [36] to enable semantically formalized logs. However, RDF is found to be too simple and hence has been accepted as a preliminary specification for semantic modeling of log data. Authors of this approach proposed to use logs modeled with RDF as a source to evaluate and diagnose the performance and other related characteristics of distributed systems. This approach lacks prescription or usage of any data mining or other related approach to utilize logs modeled using RDF.

smartFIX [37] is an approach that has been proposed for building product portfolio for knowledge-based extraction of data from any document format. This approach attempts to automatically determine the document type and extracts all relevant data for a given business process. This approach is based on using semantic technologies that enable semantic logging. The semantically formalized logs contain all process relevant information to enable explanation facility and to generate customized and understandable explanations which could be easily interpreted by users. This approach also lacks prescription or usage of any data mining or other related approach to utilize semantically modeled logs.

Splunk [86] is a comprehensive framework for semantically modeling logs and using analysis techniques for mining information from application execution to perform possibly monitoring of software applications. The authors argue that unorganized logs could be a hassle to deal with as there is no real structure, nor any standardized format. Such logs could be made more useful once stored with proper structure. Analyzing such
well-structured logs may help in finding problems, get more insight information about application execution, infrastructure for an enterprise, behavior of users, and identify potential problems. However, this approach is limited to mere structuring of logs and using basic analysis techniques for generating reports on log execution.

Another relevant approach is our own previous work on social aspects of personalized ranking for Web Services [13]. It is based on using light-weight semantics for modeling interactions of Service Consumers (SC), Service Providers (SP) and Non Functional Properties (NFP). We then used standard social network analysis techniques to compute any missing data, most importantly match between Service Consumer (SC) and Service Provider (SP) based on their part correlations with Non Functional Properties (NFP). Such correlation helped in computing possible match between Service Consumers (SC) and Service Providers (SP) which significantly help in effective ranking of Web Services.

In addition to all the related work presented, several semantics based solutions have been proposed for automated Web Service execution, including discovery, selection, composition and invocation. These approaches do not focus on semantic logging, but are highly relevant and useful for us to review given similarity in the nature of the problem. These approaches aim at semantically formalizing description of Web Services to enable the automated discovery, selection, composition and execution. Whereas, we are seeking to semantically formalize logging to enable effective analysis of logs to allow for enhanced and automated monitoring of applications.

Web Ontology Language for Services (OWL-S) [39] specifies a set of ontologies based on OWL language to describe different aspects of a semantic Web service using three core ontologies, i.e., service profile, service model and grounding. These core
ontologies model what a service does, how it works and how to access it. Web Service Modeling Framework (WSMF) [26] was introduced as a fully-fledged framework to model semantic Web services [4]. It is unique based on two complementary principles (maximal de-coupling and scalable mediation [41]) and four key elements (ontology, goal, Web service and mediator) to model different aspects of services. A set of corresponding technologies have been developed, i.e., the modeling ontology WSMO [3], the description language WSML [5], and the execution environment WSMX [28]. Web Service Description Language - Semantics (WSDL-S) [23] proposes to enrich functional description of WSDL with semantics. Based on the WSDL, WSDL-S has advantage of attaining semantics building on existing Web Service standards.

All the above mentioned approaches have made reasonable efforts towards achieving effective and automated monitoring and management of applications, however, some of the approaches lack structuring and formalizing log data to be mined and process, some approaches lack usage of advanced data mining approaches to utilize well-structured and formalized logs, and some approaches lack even both aspects. The Semantic Web Services based solutions that we discussed made promising contributions to formalize web services descriptions but do not attempt to formalize and utilize log and event descriptions. None of the approaches, except our previous related work [13], attempted to handle missing values and incomplete data which are very crucial for monitoring and managing applications. Having missing values and incomplete data in execution is a common problem due to invalid inputs from users, faults and errors that may occur during application execution. Such missing values and incomplete data could be identified in logs once logs are well-structured and formalized, before it could be
attempted to computed and predicted. Our proposed solution, by using formal semantics, enables semantically formalized logs which could be utilized by advanced data mining approaches to performing effective monitoring and management of large-scale and complex applications. This also opens space for identifying any missing values and incomplete log execution data and gives us an opportunity to attempt to resolve it by computing and predicting the missing information.

6.3 The Proposed Solution

This section presents our proposed solution to compute any missing values and incomplete data from application execution that is modeled in semantically formalized logs. Our proposed solution of semantic logs includes semantic models to formally describe components as well as events in logs during application execution. This enables having explicit information available with higher level of expressivity. Detailed description of Semantic Logs can be seen in our earlier work [83]. Our model of Semantic Logs contains key elements as Users, Problem Space, Functional Properties, Non Functional Properties, Log Events and Components. We define each of the elements as follows:

**Definition 1 (User - U):** U stands for User that acts as user applications or users. There can be g Users denoted as U₁, U₂, U₃ ... U₉.
**Definition 2 (Problem Space - PS):** PS stands for Problem Space that includes different possible types of problems as faults, failures, error or exceptions that may occur in an application. There can be $h$ Problems in Problem Space denoted as $PS_1, PS_2, PS_3, ..., PS_h$.

**Definition 3 (Functional Property - FP):** FP stands for the Functional Properties that are required by Service Consumers, and offered by Service Providers. There can be $i$ functional properties denoted as $FP_1, FP_2, FP_3, ..., FP_i$. Functional Properties may include Inputs, Outputs, Preconditions and Effects, each denoted as I, O, P and E, respectively.

**Definition 4 (Non Functional Property - NFP):** NFP stands for the Non Functional Properties that are required by Service Consumers, and offered by Service Providers. There can be $j$ non-functional properties denoted as $NFP_1, NFP_2, NFP_3, ..., NFP_j$.

**Definition 5 (Log Event – LE):** LE stands for Log Event which may occur during an application execution. There can be $k$ log events denoted as $LE_1, LE_2, LE_3, ..., LE_k$.

**Definition 6 (Component – C):** C stands for Component that is a software module to be used by Service Consumers, and offered by Service Providers. We assume that there can be $l$ non-functional properties denoted $C_1, C_2, C_3, ..., C_l$. 
Let $LEs = \{ le_1, le_2, le_3, \ldots le_n \}$ be a set of Log Events.

Let $\Delta T$ be a Log Interval which is a set of Log Events $LE$ that may occur in a given time interval.

Figure 17: Social Network Hexagon between Log Events, Components, Problem Space, Users, Functional Properties and Non Functional Properties depicts a Social Network Analysis Hexagon that shows connections between the key elements in Semantic Logs. If connections between the key elements are modeled graphically, it depicts a hexagon shape. The hexagon has several triangles between different elements of logs. We have extended and used our technique that we initially proposed in [13] to compute missing values, find out incomplete data and reveal hidden and non-obvious correlations between different elements of logs with possible problems in application execution. The Social Network Analysis Hexagon gives a generic model to represent our solution for using these connections between the elements and compute any missing values and incomplete data. We explore social network between each of these elements which can be carried out through analysis of execution log of an application with respect such elements. Each of the edges in the given hexagon represents a social network between any two elements. For example, social network between Components (C) and Functional Properties (FP) can be denoted as $SN(C, FP)$. Social Network between Components C and Non Functional Properties NFP can be denoted as $SN(C, NFP)$. Social network between Functional Properties FP and Non Functional Properties NFP can be denoted as $SN(FP, NFP)$. In this chapter, we will take two triangles of the hexagon and compute any missing values using the other information available, which is an extension of our earlier proposed solution in [13].
This computation also reveals hidden and non-obvious correlations among different elements of logs. These triangles can be noted in the figure as triangles (1) \{LE,C,U\}, (2) \{LE,C,PS\}, (3) \{LE,C,FP\}, (4) \{LE,C,NFP\}, (5) \{LE,U,PS\}, (6) \{LE,U,FP\}, (7) \{LE,U,NFP\}, (8) \{LE,PS,FP\}, (9) \{LE,PS,NFP\}, (10) \{LE,FP,NFP\}, (11) \{C,U,PS\}, (12) \{C,U,FP\}, (13) \{C,U,NFP\}, (14) \{C,PS,FP\}, (15) \{C,PS,NFP\}, (16) \{C,FP,NFP\}, (17) \{U,PS,FP\}, (18) \{U,PS,NFP\}, (19) \{U,FP,NFP\}, 20 \{PS,FP,NFP\}. We can take any of these triangles, and by using two social networks (edges) between any two out of three elements in each of these triangles, we can calculate the third social network.

![Figure 17: Social Network Hexagon between Log Events, Components, Problem Space, Users, Functional Properties and Non Functional Properties](image)

The first triangle is between elements C, FP and NFP. As per the definitions given in this section and for the purpose of generality, we may have \(l\) Components, \(i\) Functional
Properties, and \( j \) Non Functional Properties. A two dimensional social network triangle between Components, Functional and Non Functional Properties are given in Figure 18: Two dimensional Social Networks between Components, Functional and Non Functional Properties. If we have any two of the social networks (edges) data available from semantic logs, we can compute the third social network (third edge). Suppose the two social networks (SN (C, FP) and SN (C, NFP)) are available. We can use these two social networks to compute the third social network (SN (FP, NFP) as follows. Social network between \( l \) Components and \( i \) Functional Properties can be denoted as:

\[
A_{lxi} = SN(C, FP) \tag{1}
\]

Social network between \( l \) Components and \( j \) Non Functional Properties can be denoted as:

\[
B_{lxj} = SN(C, NFP) \tag{2}
\]

Using the social networks in equation (1) and (2), we can deduce social network between Functional and Non Functional Properties using the matrix multiplication steps given below:

\[
C_{ixj} = A_{lxi} \ast B_{lxj} \tag{3}
\]

\[
C_{ixj} = SN(C, FP)^T \ast SN(C, NFP) \text{ using (1) & (2)}
\]

\[
C_{ixj} = SN(FP, C) \ast SN(C, NFP) \tag{4}
\]

\[
C_{ixj} = SN(FP, NFP) \tag{5}
\]

Using this solution, we are able to compute the social network between FP and NFP, using the social networks between C, FP, and C, NFP. Similarly, if we have social networks between C, FP and FP, NFP available from semantic logs, we can compute the social network between C and NFP as follows:
\[ A_{i,k} = SN(C, FP) \]  \hspace{1cm} (6)

Social network between Functional and Non Functional Properties can be denoted as:

\[ B_{i,j} = SN(FP, NFP) \]  \hspace{1cm} (7)

Using the social networks in equation (6) and (7), we can deduce social network between Components and Non Functional Properties using the matrix multiplication steps given below:

\[ C_{j,l} = B_{i,j}^T \cdot A_{i,l}^T \]  \hspace{1cm} (8)

\[ C_{j,l} = SN(FP, NFP)^T \cdot SN(C, FP)^T \text{ using (6) \& (7)} \]

\[ C_{j,l} = SN(NFP, FP) \cdot SN(FP, C) \]  \hspace{1cm} (9)

\[ C_{j,l} = SN(NFP, C) \]  \hspace{1cm} (10)

It can be seen that this time we are able to compute the social network between C and FP, using the social networks between C, FP, and FP, NFP. We can also compute the social network between C and FP if we are given the social networks between FP, NFP and C, NFP in similar way.

This solution was for the triangle between C, FP and NFP. In a similar way, we can compute missing values in the other triangles like the triangle between LE, C and PS, the triangle between LE, FP and NFP, the triangle between LE, U and PS, etc. We show one more situation using the triangle between LE, C and PS.

Our proposed solution is very practical because from semantic logs of application execution, we may have information about correlation between Components and Functional Properties as well as Components and Non Functional Properties, but we may not have correlation between Functional and Non Functional Properties. It may be important in monitoring and managing application execution and now can be computed
using our proposed solution. Similarly, it may be easy to identify the correlation between Log Events and Users as well as Log Events and Components from semantic logs, but the correlation between Log Events and Problem Types may not be visible explicitly. This correlation between Log Events and Problem Types in the Problem Space may be very helpful discovery for monitoring and management of application and can be computed using our proposed solution.

Figure 18: Two dimensional Social Networks between Components, Functional and Non Functional Properties

6.4 Application of the proposed solution on an industrial case-study

In this section, we apply our proposed solution on an industrial use-case application for banking. This application has a number of components like ‘Transaction manager’,

```

Our proposed solution for semantic logs can benefit from our proposed methods of exploring possible social networks between Components, Functional and Non Functional Properties. Different Components may share different Functional or Non Functional Properties, like ‘Transaction manager’ and ‘Accounts manager’ both need connectivity to database and offer high quality of service. Same applies to Log Event associated with a Component as well as its Functional and Non Functional Properties. Analysis of correlation of Components with Functional and Non Functional Properties using our proposed solution may reveal many hidden and non-obvious correlations between Functional and Non Functional Properties.

Figure 19: Overall architecture for the user-case application
Semantic logging, identification of hidden and non-obvious correlations, computation of missing values and incomplete data using our proposed approach based on the social network hexagon between Components, Log Events, Service Consumers, Service Providers and Functional, Non Functional Properties, are then used by our proposed adapted classification mechanism [83] to classify problem types and to reveal any possible faults, error or exceptions that may occur during execution of the application. Semantically formalized logs make it easier for our social network hexagon based solution to find out missing values, incomplete data as well as reveal hidden or non-obvious correlations between different elements of semantic logs.

```
startAnnotations

  ex#EventID hasValue 264667

  ex#EventName hasValue “Account balance being checked by Database Manager, failure (balance not enough)”

  ex#TimeStamp hasValue _date(2014,04,29:01:08:16)

  ex#EventStatus hasValue “Success”

  ex#InboundComponents hasValue {DBManager}

  ex#OutboundComponents hasValue {TransactionManager}

  ex#Context hasValue “Local Transaction”

  ex#KeyValuePairs hasValue {TransactionID = 85645714}

  ex#KeyValuePairs hasValue {TransactionCountry = “Canada”}

  ex#KeyValuePairs hasValue {MachineID = ABM1}
```
Table 19: A glimpse of Semantic Logs in use-case application

Table 19: A glimpse of Semantic Logs in use-case application shows a glimpse of semantically formalized logs for a transaction in the use-case application. It shows a log event for a transaction that was received locally, processed by the Communication manager, transferred to the Transaction Manager in order to create necessary transaction record, followed by the Database Manager to perform necessary checks and find out whether there is enough balance in client’s account. The Database manager sends back a response with this information to the Transaction Manager. The Transaction Manager then updates the transaction record accordingly with the failure information and sends it back to the Communication Manager where this information is dispatched to the user to notify rejection of the transaction along with the reason. It is to be noted that all the activities during the use-case application execution are recoded using semantic logs analogous to the semantic log example shown in Table 19: A glimpse of Semantic Logs in use-case application.
6.5 Evaluation and Results

In this section, we present experiments and evaluation results on our proposed solution with its application on the use-case application. We carried out these experiments on Intel Core 2 CPU 2.40 GHz, 4 GB of RAM, Operating system as Microsoft Windows 7, 32-bit operating system. The use-case application was executed to record execution foot-print as semantic logs. The logs were recorded analogous to the sample shown in the previous subsection. There are different possible problem types that are identified during the monitoring of the semantic logs as recorded during the use-case application execution using our proposed solution. Table 17: Outlook of the dataset used provides a discretized overview of the execution logs of the use-case application. Semantically formalized and well-structured logs make it easier for identifying different elements. Some of the dataset was used to perform supervised learning for the Bayesian classifier, and the rest of the data was used to classify and select one out of the possible Problems in the application execution upon any failure occurred, based on the information from properties of Log Events.

We further used our social network analysis hexagon based solution to deduce hidden but interesting and useful correlations between different elements. We deduced and used two correlations in our experimental results, (1) between Functional and Non Functional Properties from social networks of Functional Properties and Components as well as Non Functional Properties and Components, (2) between Service Consumers and Service Providers using social networks of Service Consumers and Log Events as well as Service Providers and Log Events. In order to use the Bayesian classification mechanism, we used the Weka tool which is available at the URL: http://www.cs.waikato.ac.nz/ml/weka.
For Social Network Analysis purposes, the ORA tool was used which is available at [http://www.casos.cs.cmu.edu/projects/ora/](http://www.casos.cs.cmu.edu/projects/ora/).

<table>
<thead>
<tr>
<th>Event Status</th>
<th>Inbound Component</th>
<th>Context</th>
<th>Key Value (App. data)</th>
<th>Problem Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>To be started</td>
<td>Transaction Manager</td>
<td>Foreign Transaction</td>
<td>China</td>
<td>“Security”</td>
</tr>
<tr>
<td>Started</td>
<td>Database Manager</td>
<td>National Transaction</td>
<td>France</td>
<td>“Database issue”</td>
</tr>
<tr>
<td>Failure</td>
<td>Transaction Manager</td>
<td>Local Transaction</td>
<td>Canada</td>
<td>“Database issue”</td>
</tr>
<tr>
<td>Success</td>
<td>Communication Manager</td>
<td>Local Transaction</td>
<td>Canada</td>
<td>“Accounts Database”</td>
</tr>
<tr>
<td>Starting</td>
<td>Communication Manager</td>
<td>Foreign Transaction</td>
<td>South Korea</td>
<td>“External Communication”</td>
</tr>
</tbody>
</table>

Table 20: Outlook of the dataset used

Table 21: Accuracy in Classifying Problem Types lists possible problem types that were detected and classified by our Bayesian classification mechanism for Semantic
Logs. Some of the data from logs was used for supervised learning, while the rest of the data was used in performing our proposed social network analysis based calculations and classification.

<table>
<thead>
<tr>
<th>#</th>
<th>Possible Problem Types</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Customer late response / timeout</td>
<td>0.96</td>
</tr>
<tr>
<td>2</td>
<td>External communication issue</td>
<td>0.86</td>
</tr>
<tr>
<td>3</td>
<td>Internal communication issue</td>
<td>0.89</td>
</tr>
<tr>
<td>4</td>
<td>Database connectivity issue</td>
<td>0.90</td>
</tr>
<tr>
<td>5</td>
<td>Customer address validation issue</td>
<td>0.91</td>
</tr>
<tr>
<td>6</td>
<td>External B2B connectivity issue</td>
<td>0.81</td>
</tr>
<tr>
<td>7</td>
<td>External Gateway down/inaccessible</td>
<td>0.88</td>
</tr>
<tr>
<td>8</td>
<td>Web server out of memory error</td>
<td>0.84</td>
</tr>
</tbody>
</table>

Table 21: Accuracy in Classifying Problem Types

We observed an overall Mean Average Precision (MAP) of almost 88% which is dependent on the diversity of data as well as elements in the training dataset. We observed that although we do not have 100% precision rate, the achieved precision can be used to classify and shorten the problem space for administrator monitoring and managing the application, and hence may get the task of fault detection and handling fairly enhanced and automated. However, in order to achieve this enhancement and
automation, the applications have to use our proposed solution of Semantic Logs with Social Network Analysis based techniques with Classification.

<table>
<thead>
<tr>
<th>#</th>
<th>Predicted Problem Types</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>External communication issue</td>
<td>0.73</td>
</tr>
<tr>
<td>2</td>
<td>Internal communication issue</td>
<td>0.64</td>
</tr>
<tr>
<td>3</td>
<td>External Gateway down/inaccessible</td>
<td>0.82</td>
</tr>
<tr>
<td>4</td>
<td>Database connectivity issue</td>
<td>0.68</td>
</tr>
</tbody>
</table>

Table 22: Predicted Problem Types with accuracy

Table 22: Predicted Problem Types with accuracy presents some of the problem types that were predicted in advance by using our solution of social network analysis based computation associations (from historical execution data) between Log Events, Components, Users, and other key elements, with a reasonable Mean Average Precision (MAP) of 71%. The prediction was carried out for 4 problem types. Associations between Problem Space, as well as Log Events, Components, Users, Functional and Non-Functional Properties were used from historical data to compute prediction values of possible problems that may occur during application execution.
Figure 20: Comparison of number of steps in Problem detection presents a graph that outlines a number of steps for detecting a Problem which could be an exception, fault, failure or any other related problem during application execution. The graph also compares the number of steps required to be followed to detect problems in different cases, i.e., without using any monitoring solution, using another similar solutions [33] [34], and using our proposed solution. The X-axis shows different problems that occurred during application execution, and the Y-axis shows number of steps that were required for detecting the problem. We noticed a reasonable reduction in the number of steps required in the detection and classification of problems because semantically formalized logs with the help of data mining based techniques helped in not only reducing the problem space but also predicting any possible problems during application execution.

![Figure 20: Comparison of number of steps in Problem detection](image-url)
While comparing our proposed approach with other related approach as well as traditional methods of manually detecting and handling problems in application execution, we found out that most of the related works were found to be either limited to use formalizing or structuring logs, or limited to use data mining based approaches for processing unstructured log data for monitoring and management of applications. None of the related approaches were found to be comprehensive enough that could address the issue of application monitoring and management from both aspects, i.e., to structure and formalize logs as well as use Data Mining based techniques for processing unstructured log data for monitoring and management of applications. None of the related approaches were found to be comprehensive enough that could address the issue of application monitoring and management from both aspects, i.e., to structure and formalize logs as well as use Data Mining based techniques for processing such formalized and well-structured logs. Formalism and well-structuring of logs enables us to identify correlations between different key elements of logs and compute other hidden and non-obvious correlations which help in not only classifying but also predicting any possible problems during application execution. The combination of the best of both, i.e., formalized and well-structured logs with advanced data mining based techniques makes best use of such formalism and enables enhanced and effective monitoring and management of applications.

6.6 Conclusions

In this chapter, we proposed a social network based solution to handle missing values and incomplete data during the execution of applications. Our proposed solution is based on our proposed solution of semantically formalized logging for recording execution footprint of applications, and use it to deduce possibly new or hidden information which may otherwise be not available. We modeled the correlation of key elements in logs into a social network analysis hexagon and further showed how we can use the correlation
between different key elements of semantic logs and use them to deduced new and non-obvious correlations between other elements of semantic logs and use this information in monitoring and management of applications. Our proposed adapted data mining based approaches can intelligently use the newly deduced information to predict any upcoming possible faults or errors in applications and avoid such risks in advance.

We presented and evaluated methods to compute such hidden and non-obvious correlations and complete any missing values or incomplete data in execution footprint of applications. Semantic logs played a key role in our proposed solution by providing formalized and well-structured logs with adapted data mining based approaches to process such logs. We presented an industrial use-case application and applied our proposed solution to that followed by extensive experiments and evaluation. This evaluation showed how the semantically formalized logs, with adapted data mining based approaches, could make use of the new and deduced information to perform effective monitoring and management, especially for large-scale distributed applications like the use-case application.
CHAPTER 7: A CLUSTERING BASED INTEGRATED APPROACH FOR SEMANTIC LOGS AND ANALYTICAL SOLUTIONS

We proposed semantically formalized logs with advanced analytical solutions to enable enhanced monitoring and management of software application. In this chapter, we discuss a clustering based solution for overall integration of all the approaches. During application execution, events are executed and produced in a continuous stream which is recorded as logs. Our proposed solution is of hybrid nature for semantically formalized modeled execution workflow and logs used by advanced analytical solutions to process semantic information to help in enhancing the process of monitoring and management of software applications. We also have discussed and analyzed recall of computation by analytical techniques to computing. Recall of computation for logs by analytical solutions imposes cost in terms of computation and space and also brings value to the process of monitoring and management that such analytical solutions refresh knowledge (e.g., association rules, clusters, classifier and missing data) by processing logs. The information obtained from this knowledge is then used for monitoring and management of software applications. However, this value may be smaller for some types of applications under a scenario and higher for other types of applications under another scenario. This chapter presents how this subjective measure of recall should be used with our proposed solution in order to achieve best value for the cost spent. This chapter discusses a customized stream clustering solution for the integration of stream clustering solution in overall semantic logging framework, followed by analysis of recall with respect to the value gained for different types of applications.
7.1 Introduction and Related Work

We have proposed our hybrid solution of semantically formalized logging with advanced analytical solutions for enhanced monitoring and management of software applications. As the complexity in user requirements is increasing, software applications are also getting more and more complex, huge in terms of size, computation as well as storage resources required. Our proposed solution of semantic logs and advanced analytics for enhanced monitoring and management of software applications is based on building semantic models to formally describe components as well as events descriptions in execution logs of software applications and then build adapted analytical solutions to effectively process such logs. This allows having more explicit information available with higher level of expressiveness. Highly expressive, formalized and well-structured information makes it easier for the monitoring solutions to process such logs in order to have an enhanced and effective way to view the activities in the application execution.

We proposed an Association Rule Mining based approach to use our proposed Semantic Logs leading to Semantic extension of FP-Growth for automated ranking and adaptation of Web Services. Our hybrid approach of partially using semantic annotations to Web Services combined with semantically adapted FP-Growth for Association Rule Mining allows the preprocessing of requests for searching Web Services which help in improving the Web Service selection experience from performance as well as precision perspective. We further used Bayesian Classification and proposed a hybrid approach for enhanced and automated monitoring and management of applications by using Semantics with Bayesian Classification. Semantics are used to formalize and structure logs from
application execution which are then utilized by Bayesian Classification to classify different types of possible issues. This helps in reducing problem space for application administrators to focus on the problematic part of application rather than the whole application. We also used a social network based solution with Semantic Logs to handle missing values and incomplete data during execution of applications. Our proposed solution is based on our work on semantically formalized logging for recording execution footprint of applications and later on using it to deduce possibly new or hidden information which may not be available otherwise. We modeled correlations of key elements in logs into a social network analysis hexagon and further showed how we can use such correlations between different key elements of semantic logs and use them to deduce new and non-obvious correlations between other elements of semantic logs and then utilize this information in monitoring and management of applications. Our proposed adapted data mining based approaches can intelligently use the newly deduced information to predict any upcoming possible faults or errors in applications and avoid such risks in advance.

In this chapter, we propose a stream clustering based overall integration approach for each of the components of our proposed solution. There could be several other ways to perform integration of all of the components together; however we keep our proposed solution generic and open to different possibilities and scenarios to handle the monitoring and management of different types of applications. We use stream clustering based approach because logs are produced in a stream like manner as an application executes.

In [88], the authors introduce a way to cluster log events based on different features. They have employed different clustering algorithms [89] [90] to cluster log events into
different categories. They view different lines in log files as objects and use clustering algorithms to cluster lines into different categories. After the clusters (event types) have been identified, the authors employ different analysis techniques for detecting temporal associations between event types. They believe that clustering may identify many line patterns that reflect normal system activity and that can be immediately included in the system profile, since the user does not wish to analyze them further. Clusters of outliers may contain infrequent lines that could represent unexpected behavior of the system including faults, exceptions or errors. Authors have built a clustering tool called SLCT (Simple Logfile Clustering Tool). However, the limitation of this approach is that authors do not make any attempt to formalize or structure log information. They build their solution to rely on unstructured and less expressive data and cluster events based on fault or no fault basis. This limits their approach in terms of detecting different possible events from different perspectives.

In [91], the authors proposed to cluster logs from network management software to have a better view to system and network administrators. Clustering can let network administrators to view faulty parts of log data rather than being overwhelmed with a large amount of log data. In fact, large amount of log data with a lot of irrelevant information may make the monitoring process slow and may also cause a lot of unnecessary delay. The authors based their work on the Simple Log file Clustering Tool (SLCT) [88] and developed a visualization tool that can be used to view log files based on the clusters produced by SLCT. They claim that their results based on different application log files help in easing the summarization of a vast amount of data contained in the log files. It may help in speeding up the analysis of event data in order to detect any possible errors,
faults or exceptions in the application. However, drawbacks of this approach are the same as those of [88], i.e., the approach is dependent on using unstructured and less expressive data and cluster events based on fault or no fault basis. This limits their approach in terms of detecting different possible events from different perspectives.

In [92], the authors apply clustering on search engine query log in order to mine a collection of user transactions with an internet search engine to discover clusters of similar queries and similar URLs. Using clustering for different queries from query log, the authors claim to enhance web search. Clustering of queries into different clusters helps in computing results faster for new queries that are similar to older queries. While this approach does help in enhancing the process of search to some extent, this approach is also limited to unstructured and raw log data (also known as click-through data). This limits their approach in terms of detecting and correlating different possible events from different perspectives.

7.2 Overall Integration

This section presents the overall stream clustering based integration of our proposed solution of Semantic Logs with adapted analytical solutions based on Association Rule Mining, Classification and Social Network Analysis. We chose stream clustering because events are executed in application in a stream like manner where logs are produced as event execution progresses in an application. There could be multiple ways to perform integration of all of the components together. Our approach is to keep our proposed solution generic and open to different possibilities and scenarios to handle the monitoring and management of different types of applications. Therefore, we have used stream
clustering based approach because logs are produced in a stream like manner as application executes. Our proposed solution has been designed in a way which is generic and open to system analysts to use one, multiple or all of the analytical solutions together as required. Figure 21: Overall Integration of Classification, Association Rule Mining and Social Network Analysis over Clustering depicts the overall integration scenario.

Figure 21: Overall Integration of Classification, Association Rule Mining and Social Network Analysis over Clustering

7.3 Stream Clustering of Log Events

Logs are produced as events in an application are executed. We have used STREAM [93] based approach to cluster events into different clusters. We can cluster events based on
different features of events in the logs. These features could be category, status, components, functional, non-functional or any other application specific feature. Clustering of logs based on data stream of events from logs is carried out by STREAM approached as outlined in Table 23: Stream Clustering Algorithm for Log Events.

**Input:** a sequence of n Log Events from Semantic Logs and an integer k for number of clusters to be determined.

**Algorithm:**

1. Input the first m points; using the randomized algorithm presented in [93] reduce these to O(k).
2. Repeat the above till we have $m^2/(2k)$ of the original data points to have m intermediate medians.
3. Using a local search algorithm, cluster these m first-level medians into 2k second-level medians and proceed.
4. In general, maintain at most m level-i medians, and, on seeing m, generate 2k level-i+1 medians, with the weight of a new median as the sum of the weights of the intermediate medians assigned to it.
5. When we have seen all the original data points, we cluster all the intermediate medians into k final medians, using the primal dual algorithm.[94]

**Output:** n centers in the set of the m Log Events so as to minimize the sum of distances from data points to their closest cluster centers.
7.4 Computing missing values using Social Network Analysis for each cluster

In chapter 6, we proposed a social network based solution to handle missing values and incomplete data during the execution of applications where key elements of logs are modeled into a social network analysis hexagon. As an input, it takes log events from the stream clustering component and performs computation on data from each cluster. It uses our technique as described in chapter 6 to compute missing values, find out incomplete data and reveal hidden and non-obvious correlations between different elements of logs with possible problems in application execution. We take each triangle in the SNA hexagon and compute any missing values using the other information available where each edge of the triangle represents a two-mode social network. For example, if we take the triangle between elements C, FP and NFP, we can use any two of the social networks (as two edges) to compute the third social network (as the third edge) of the triangle. In a similar way, we can perform computation on other triangles of elements in the SNA hexagon. It makes the computation of missing values and incomplete data more efficient as the processing is based on data from each cluster in which log events that are similar in characteristics are categorized in the same cluster.

7.5 Discovering Association Rules from each cluster

In chapter 4, we proposed a Semantic FP-Tree based technique to perform association rule learning on different characteristics of logs. Applications encapsulate the execution
outcome in the form of Semantic Logs. Each of the execution and event processing is stored as Semantic Logs in a repository. Such Semantic Logs are later on retrieved and represented in the form of Semantic FP-Tree and are processed by our proposed semantic extension to the FP-Growth algorithm. The constructed Semantic FP-Tree is then discretized after translating semantic axioms and grounded into a normal FP-Tree from which Association Rules among different events in the logs are discovered. The discovered association rules are then used during the process of monitoring and management of applications. Semantic Logs are processed using our proposed solution described in the previous section to extract and discover association rules which are then used during the process of ranking of Web Services. The logs being semantically formalized help during the process of processing and mining the logs to discover association rules. It makes the process of discovering association rules more efficient as it is based on the data from each cluster in which log events with similar characteristics are categorized in the same cluster and have any missing values are computed.

7.6 Integrating results from each cluster and performing classification

In chapter 5, we proposed a Bayesian classification based approach to perform classification on semantic logs in order to reduce the problem space. Our hybrid approach of partially using semantics to formalize log and workflow data, and adapted classification technique combines the best of both. Semantics help in providing high-level of precision, structure and expressivity to execution workflow and logs. Such kind of formalized data can be used in an effective manner to effectively interpret and process highly structured information from the generated logs during the execution by
classification technique to reduce problem space during the process of monitoring and management of applications. This helps in reducing problem space for application administrators to focus on the problematic part of application rather than the whole application. The process of classification becomes more effective when it takes as input log events with similar characteristics that are clustered in same cluster using our clustering technique, have any incomplete data and missing values computed using our proposed social network analysis hexagon based computation technique and have different possible association rules that are discovered using our proposed Semantic FP-Growth technique.

7.7 Experiments and evaluation

Once clusters of logs are produced from stream of log events generated from application execution, analytical solutions including association rule mining, classification and social network analysis are applied on individual clusters. This helps in narrowing down to different types of similar events, e.g., a cluster containing events related to a particular failure, error or exception. After such clusters are identified, further analytical solutions can be applied. Clustering could carry out different patterns that may either reflect normal application execution with routine events, or non-routine events related to a system failure and hence may require a closer inspection by narrowing down the problem space even further.

We have performed experiments and evaluated results based on our use-case application for a financial institution with outlook of data shown in Table 24: Outlook of the dataset used. The experiments were carried out on Intel Core 2 CPU 2.40 GHz, with 4
GB of RAM, and on Microsoft Windows 7, 32-bit operating system. We have run tests based on the dataset and provided the Bayesian classifier initial dataset to perform supervised learning. Once the supervised learning was completed, we further processed the incoming requests based on the incoming requests from users containing the values about the required properties of the Log Events recorded during the application execution. We then performed clustering of Log Events into two different clusters, i.e., events with status “failure” and events with status other than “failure” and carried out the same experiment of classification cluster with events having status “failure”. The same training dataset was used to train the classifier in both cases.

<table>
<thead>
<tr>
<th>Event Status</th>
<th>Inbound Component</th>
<th>Context</th>
<th>Key Value (App data)</th>
<th>Select Problem Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Started</td>
<td>Transaction Manager</td>
<td>Foreign Transaction</td>
<td>China</td>
<td>“Security”</td>
</tr>
<tr>
<td>To be Started</td>
<td>Accounts Manager</td>
<td>National Transaction</td>
<td>USA</td>
<td>“Accounts Database”</td>
</tr>
<tr>
<td>Success</td>
<td>Transaction Manager</td>
<td>Local Transaction</td>
<td>USA</td>
<td>“Accounts Database”</td>
</tr>
<tr>
<td>Failure</td>
<td>Communication Manager</td>
<td>Local Transaction</td>
<td>USA</td>
<td>“Accounts Database”</td>
</tr>
<tr>
<td>Shutting Down</td>
<td>Communication Manager</td>
<td>Foreign Transaction</td>
<td>China</td>
<td>“External Communication”</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
Table 24: Outlook of the dataset used

<table>
<thead>
<tr>
<th>#</th>
<th>Classified Problem Types</th>
<th>Precision without clustering</th>
<th>Precision with clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>External Communication</td>
<td>0.88</td>
<td>0.89</td>
</tr>
<tr>
<td>2</td>
<td>Internal Communication</td>
<td>0.94</td>
<td>0.91</td>
</tr>
<tr>
<td>3</td>
<td>Database Manager</td>
<td>0.71</td>
<td>0.89</td>
</tr>
<tr>
<td>4</td>
<td>Customer address validation foreign station</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>5</td>
<td>Customer id validation from foreign station</td>
<td>0.84</td>
<td>0.78</td>
</tr>
<tr>
<td>6</td>
<td>Login failure</td>
<td>0.51</td>
<td>0.64</td>
</tr>
<tr>
<td>7</td>
<td>Transaction Timeout</td>
<td>0.79</td>
<td>0.82</td>
</tr>
<tr>
<td>8</td>
<td>Gateway down</td>
<td>0.84</td>
<td>0.88</td>
</tr>
<tr>
<td>9</td>
<td>External currency conversion</td>
<td>0.90</td>
<td>0.87</td>
</tr>
</tbody>
</table>

Table 25: Comparison of Accuracy in Classification Results with or without Clustering

Table 25: Comparison of Accuracy in Classification Results presents a comparison of the results on our classification analysis on all log data versus classification analysis on
clustered data with event status “failure”. We used one-third of the data for supervised learning, in order to train the model. The rest two-third of the dataset was used in testing. The overall Mean Average Precision (MAP) was observed to be approximately 82% from carrying out classification on log data without clustering, and approximately 85% on data that was clustered as mentioned previously. We can notice a slight increase in precision after carrying out classification on clustered data with event status as “failure”. The reason is that the classifier had to narrow down the problem space after carrying out classification on clusters of data with event status as “failure”. However, we also notice that there is a slight increase and slight decrease in precision for classifying individual problem types. This difference is due to the fact that some of the problem types had events with status “failure” as well as “successful” and after narrowing down classification on a cluster of events with status “failure” only, reduced the probability for the classification mechanism to detect the problem type as accurately as in the previous case. Problem types where an increase in precision type has been noticed had events with status “failure” only. Narrowing down the classification on a cluster of events with status “failure” only increased the probability for the classification mechanism to detect the problem type as accurately as in the previous case. Therefore, it depends on the variety of events that different problem types may have and depending on that system administrators can choose to apply or not to apply clustering before classification or any other analytical technique. This keeps the overall framework of our proposed solution of Semantic Logging with Advanced Analytics generic and open to adapt to different types of application as required.
7.8 Analysis of recall with respect to value gained for different applications

In this section, we discuss and analyze recall of computation by analytical techniques to computing. Recall of computation is required, for the analytical solutions being used in our proposed solution, in order to let such analytical solutions re-compute set of rules with latest logs that are produced during application execution. Recall for the classification mechanism would be to re-compute probabilities of different features for its classification mechanism to determine the value of class variable from latest logs produced from application execution. Recall for the association rule mining mechanism would be to re-compute a set of association rules using FP-Growth from latest logs produced from application execution. Recall for the social network analysis mechanism would be to re-compute set of missing values and incomplete information in the SNA-Hexagon from the latest logs produced from application execution. Recall for clustering based mechanism would be to re-compute a set of clusters for log events from the latest logs produced from application execution.

Recall of computation for logs by analytical solutions imposes cost in terms of computation and space. It also brings a value to the process of monitoring and management that such analytical solutions refresh knowledge (e.g., association rules, classifier, missing data and clusters) by processing newly generated logs from application execution. The information obtained from this knowledge is then used for monitoring and management software applications. However, this value may be smaller for some types of applications under a scenario and higher for other types of applications under another scenario. More frequently recall is made, the more the latest the information will become available for analytical solutions and vice versa. Moreover, more frequently recall is
made, more cost will be imposed from re-computation of data for analytical solutions and vice versa. Therefore, setting frequency of recall is a subject measure which depends on several factors like type of the application in terms of level of criticalness, resources available in terms of time and space, etc. Figure 22: Analysis of frequency of recall versus value gained for different types of applications depicts the correlation of frequency of recall versus the value it may bring to different types of applications.

Light-weight applications and utilities may include different types of games, non-critical software utilities, notes or reader applications. For such types of applications and utilities, it is less critical to perform monitoring and management on execution. Therefore, it is better to save cost in terms of time and space and keeping frequency of recall to a lower rate providing lower value from the generation of a set knowledge for analytical solutions. The lower rate is a fuzzy term and is dependent on the computing and storage resources available for the application monitoring and management platform.

Critical applications may include hospital systems, defense systems, weather watch systems, airport communication systems and different applications used by military and other law enforcement agencies. For such types of applications and utilities, it is highly critical to perform monitoring and management on execution. Therefore, it is better to invest more cost in terms of time and space and to keep frequency of recall to a higher rate providing higher value from the generation of a set of knowledge for analytical solutions. The higher rate is also a fuzzy term and is dependent on the computing and storage resources available for the application monitoring and management platform.
Day to day and targeted applications with medium level of criticalness may include applications like excel spread sheets, inventory management systems, office tools and data entry systems For such types of applications and utilities, it is critical to a medium extent to perform monitoring and management on execution. Therefore, it is better to save cost in terms of time and space and keeping frequency of recall to a medium rate providing medium value from the generation of a set knowledge for analytical solutions. The medium rate is also a fuzzy term and is dependent on the computing and storage resources available for the application monitoring and management platform.
Figure 22: Analysis of frequency of recall versus value gained for different types of applications

7.9 Conclusions

In this chapter, we presented an overall clustering based integration framework for our proposed solution of Semantic Logging using different adapted analytical solutions to enable enhanced monitoring and management of software applications. We presented the related work and presented a stream clustering based integration solution. Stream clustering was used because events in the execution logs are produced like a stream. Such log events are clustered using stream clustering based solution. Each of the analytical
solutions, like association rule mining, classification and social network analysis are carried out on different clusters of log events. We carried out experimentation and analysis of our proposed integrated solution. We also presented recommendations to set frequency of recall for different analytical solutions (e.g., association rules, clusters, classifier and computing missing data).

We found out that setting frequency of recall is very subjective and is dependent on different types of applications. We discussed that more frequently a recall is made, more latest the information will become available for analytical solutions and vice versa. However, higher frequency of recall imposes higher cost from computation and storage perspective, for analytical solutions and vice versa. It depends on the nature of application, i.e., how critical is the monitoring and management for such application. We recommended that the more critical it is for an application to have monitoring and management, the more value it will bring by investing on higher frequency of recalls and vice versa.
CHAPTER 8: CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this thesis, we have enlightened the issue of manual labour and shortcomings in the process of monitoring and management of application and proposed a Semantic Logging with Advanced Analytics based solution to tackle the problem. We first carried out a detailed comparative analysis and discussed shortfalls, limitations to point out need for flexibility and enhancement in monitoring and management of software applications. We reviewed several related approaches, currently available tools and technologies. From our study, we could classify currently available approaches into four different categories, i.e., (1) approaches focusing on semantic formalism of logs, (2) approaches focusing on data mining based processing and analysis of logs, (3) approaches performing mere structuring of logs, and (4) approaches focusing on the combination of semantic formalism as well as data mining based processing and analysis of logs. The key lacking we found in the existing approaches was that most of the approaches were either about semantic formalism of logs, or mere structuring of logs or only tried to process the logs using data mining related approaches. We found out that such approaches faced challenges, e.g., while trying to formalize the logs, the approaches included basic information related to application execution and did not consider information about components as well as event logs together. Because of this limitation, such approaches do not have the ability to correlate event execution across multiple components of a software application and hence make the process of monitoring and management of large-scale as well as multi-component applications complicated and limited. Many other approaches found were only focusing on either mere structuring of logs or only applying data mining
and other related approaches to process the logs. We found out that such approaches did not focus on combining the efforts to structure and perform mining as well as analysis on logs to achieve better results, as data mining and analytic approaches are dependent upon well-structuring and formalization of logs.

After completing the literature survey and analysis, we designed our proposed solution of semantically formalized logging for enhanced monitoring and management of software applications. Our approach takes into account the lacking found in the existing approaches and tries to cover that and attempts to collect comprehensive information about event logs, components as well as background information about the application and the software execution in the logs, which is later used by our log mining techniques for enhanced monitoring and management of software applications. Our proposed solution also correlates semantic formalism and structuring of logs along with mining the logs, which helps in maximizing the utilization of formalized logs to deduce the maximum possible useful information about log execution which eventually helps in enhanced monitoring and management of software applications. We further presented our methodology and design of our proposed solution. We also presented a use-case application scenario in which our proposed solution was utilized to perform enhanced monitoring and management of the use-case application by having higher-level automation as well as flexibility.

We then proposed a unique approach for ranking and adaptation of Web Services using Association Rule Mining based on our proposed Semantic Logs as well as Semantic extension of FP-Growth. We analyzed related and existing approaches and found out that such approaches are limited since such approaches either focus only on
semantically formalizing description of Web Services with limited mechanisms to utilize such descriptions or use heuristic based techniques on limited and syntactic data of Web Services for ranking and adaptation of Web Services. Such approaches also merely take into account past interaction of Service Consumers and Service Providers. Our proposed approach allows semantically formalized representation of logs during Web Service execution which are then used to perform ranking and adaptation of the discovered Web Services. This hybrid approach of partially using semantic annotations to Web Services combined with semantically adapted FP-Growth for Association Rule Mining allows the preprocessing of requests for searching Web Services which help in improving Web Service selection experience from performance as well as precision perspective. We also presented our experimental results and showed that how this trade-off of partially using semantics with semantically adapted Association Rule Mining techniques helps in improving Web Services selection.

We further used Bayesian Classification and proposed a hybrid approach for enhanced and automated monitoring and management of applications by using Semantics with Bayesian Classification. Semantics are used to formalize and structure logs from application execution which are then utilized by Bayesian Classification to classify different types of possible issues. This helps in reducing the problem space for application administrators to focus on the problematic part of application rather than the whole application. We also analyzed and compared existing approaches and found out that such approaches are limited because they either focus only for semantically formalizing description of logs with limited mechanisms to utilize such descriptions or just focus on using heuristic based techniques on limited, syntactic and unstructured log
and other execution related data of applications which makes the process of application monitoring and management limited. Our proposed hybrid approach partially used semantically formalized and well-structured logs with adapted Bayesian classification to allow automatically pre-selecting and reducing problem space and thus help in improving application monitoring and management experience from the perspective of efficiency and precision. It helps in reducing the number of steps that are required to detect a problem in order to recover an application from a fault. It further helps in predicting any possible fault or failure that could occur during application execution so that it could be mitigated and avoided. We also carried out experimental evaluation and analyzed results that show how it is better to enable and use semantically formalized logs with Bayesian classification for enhancing and automating application monitoring and management.

We then used a social network based solution with Semantic Logs to handle missing values and incomplete data during the execution of applications. Our proposed solution is based on our work on semantically formalized logging for recording execution footprint of applications and then later on using it to deduce possibly new or hidden information which may not be available otherwise. We modeled correlations of key elements in logs into a social network analysis hexagon and further showed how we can use such correlations between different key elements of semantic logs and use them to deduce new and non-obvious correlations between other elements of semantic logs and then utilize this information in the monitoring and management of applications. Our proposed adapted data mining based approaches can intelligently use the newly deduced information to predict any upcoming possible faults or errors in applications and avoid
such risks in advance. We presented and evaluated methods to compute such hidden
and non-obvious correlations and complete any missing values or incomplete data in
execution footprint of the applications. Semantic logs played a key role in our proposed
solution by providing formalized and well-structured logs with adapted data mining
based approaches to process such logs. We presented an industrial use-case application
and applied our proposed solution to that followed by extensive experiments and
evaluation. This evaluation showed how the semantically formalized logs, with adapted
data mining based approaches, could make use of the new and deduced information to
perform effective monitoring and management, especially for large-scale distributed
applications like the use-case application.

Last but not least, we presented overall integration framework for our proposed
solution of Semantic Logs with Advanced Analytical solutions based on Association
Rule Mining, Bayesian Classification and Social Network Analysis based on Clustering
of log events. We also discussed the process of recall in each of the analytical approaches
and discussed the cost associated versus the value it may bring which may depend on
different types of applications.

### 8.1 Future research directions

Our future work is to further extend our research methodology to use different types of
formal and semantic languages at different level of expressivity for Semantic Logs and
different types of analytical solutions, including Big Data Analytics. This will help in
capturing more data with larger number of constraints. However, while extending our
methodology further, we will try to stay with our design objectives, i.e., to keep our
methodology generic enough and not making it restricted to a particular software application for monitoring and management. Our aim is to keep our methodology generic enough, so that it could be used for monitoring and management of any software application. We also plan to extensively evaluate our methodology based on other real-life data sets that could be obtained from real-life applications. Major hurdles in getting access to logs of real-life applications are maintaining privacy of application users as well as the organization operating it and keeping the data secure.

This work also lays foundation towards Big Data Analytics. This thesis proposes semantic logs as well as advanced and adapted analytical solutions to formally represent and process machine generated data. Machine generated data is one form of Big Data that is produced in much faster speed than that of data that is produced by humans directly. Big Data and the information that is obtained from it is often maintained at different heterogeneous data sources. Big Data is already evident in several related domains like Oil and Gas information integration, Banking Systems, Business Intelligence, Energy and Environmental monitoring systems, Health and Clinical systems, and any other kinds of systems that produce and deal with large amounts of data. This work on semantic logs can be extended towards building standardized and effective ways to model Big Data which can be analyzed, integrated and managed efficiently and effectively.
REFERENCES


[38] B. Rohallah, M. Ramdane, S. Zaidi, "Agents and owl-s based semantic web
service discovery with user preference support", International Journal of Web &
Semantic Technology (IJWesT), April 2013, Volume 4, Number 2, pp 57-75.

University. Available at: www.sei.cmu.edu/isis/guide/technologies/owl-
s.htm

[40] M. Paolucci, N. Srinivasan, K. Sycara, T. Nishimura, "Towards a Semantic
Choreography of Web Services: from WSDL to DAML-S", in the proceedings of
International Conference on Web Services (ICWS 2003), Las Vegas, Nevada,
USA, June 23-26 2003.

[41] E. Cimpian, A. Mocan, F. Scharffe, J. Scicluna, M. Stollberg: D29v0.1 WSMO
Mediators, WSMO Final Draft, December 2005, Available at:
http://www.wsmo.org/TR/d29/v0.1/

[42] J. Kopecký, M. Moran, D. Roman, A. Mocan, WSMO Grounding, a WSMO
working draft D24.2v0.1, September 2005. Available at
http://www.wsmo.org/TR/d24/d24.2/v0.1

[43] A. Mocan, E. Cimpian: Mapping Creation Using a View Based Approach, 1st
International Workshop on Mediation in Semantic Web Services (Mediate 2005),
December 2005, Amsterdam, Netherlands

[44] E. Cimpian, A. Mocan: WSMX Process Mediation Based on Choreographies, 1st
International Workshop on Web Service Choreography and Orchestration for
Business Process Management (BPM 2005), September 2005, Nancy, France

Available at http://tools.deri.org/wsml/OverviewReasoner.html


[62] Web Services at W3C: W3C Recommendations on WSDL and SOAP. Available at http://www.w3.org/2002/ws


[65] J. Han, H. Pei, and Y. Yin, “Mining Frequent Patterns without Candidate Generation”, in proceedings of International Conference on the Management of Data (SIGMOD 2000), held in Dallas TX, USA, published in ACM Press, New York, NY, USA, 2000.


[88] Risto Vaarandi, "A Data Clustering Algorithm for Mining Patterns From Event Logs", In proceedings of the 2003 IEEE Workshop on IP Operations and Management (IPOM 2003), 1-3 October 2003, Kansas City, Missouri, USA.


