2015-07-30

A Methodology for a Data-Driven Model and Approach for Content-Specific Assessment of Service Workflows

Damian A. Clarke
University of Miami, dclarke6@umiami.edu

UNIVERSITY OF MIAMI

A METHODOLOGY FOR A DATA-DRIVEN MODEL AND APPROACH FOR CONTENT-SPECIFIC ASSESSMENT OF SERVICE WORKFLOWS

By

Damian Clarke

A DISSERTATION

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Coral Gables, Florida

August 2015

UNIVERSITY OF MIAMI

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

A METHODOLOGY FOR A DATA-DRIVEN MODEL AND APPROACH FOR CONTENT-SPECIFIC ASSESSMENT OF SERVICE WORKFLOWS

Damian Clarke

Approved:

_____ M. Brian Blake, Ph.D. Professor of Computer Science	_____ Geoff Sutcliffe, Ph.D. Professor of Computer Science
_____ Stefan Wuchty, Ph.D. Associate Professor of Computer Science	_____ Bonnie Kirkpatrick, Ph.D. Assistant Professor of Computer Science
_____ Aubrey Rembert, Ph.D. Assistant Professor of Business Florida Institute of Technology	_____ Dean of the Graduate School

CLARKE, DAMIAN (Ph.D., Computer Science)
A Methodology for a Data-Driven Model and Approach for Content-Specific Assessment of Service Workflows (August 2015)

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Professor M. Brian Blake.
No. of pages in text. (152)

The increase in the demand for online services has resulted in approximately 1 trillion connected objects and devices on the Internet, which generate billions of gigabytes of new data each day. As more devices become web-enabled, the number and type of services available on the Internet are likely to increase. The growing number of services adds to the complexity of connectedness and interoperability of computing systems. Moreover, interactions between organizations and individuals are also increasing. As a result, data produced by internal and external business processes are in turn growing at an exponential rate creating a data deluge.

Companies need to do more than just connect people and integrate processes to become smarter service providers. Business applications must now be able to react to inherently dynamic and uncertain business situations. Companies need to improve and modify, act in an agile manner, optimize and adapt business processes to their customers, and thereby improve the responsiveness of the whole company. Only companies that can quickly and efficiently adapt to the changing business needs can stay competitive in the global market.

Initial approaches that sustain competitiveness focus on increasing collaboration and interoperability within an organization and with its suppliers and consumers. These early

approaches required a robust, and interoperable computing architecture. A popular and deeply collaborative approach to incorporating the realizations of connectivity and interoperability has been the Web service framework. In this context, services are realized as autonomous, platform-independent, computational elements that are described, published, discovered, orchestrated and programmed using standard protocols. Loosely coupled services have traditionally been orchestrated by a central processing service executed in the Business Process Execution Language (BPEL). BPEL is the defacto standard for service-oriented composition and orchestration. It is used to build and execute workflows of collaborating applications distributed within and across organizational boundaries. These workflows scale to Web-scale *Web service workflows*, which realize the business logic of organizations, its partners, and customers across geographic boundaries.

However, business logic realized by BPEL is usually formulated in the early stages of the software design lifecycle, which renders workflows *non-responsive* to just in time changes to time-dependent and often uncertain business trends. Manual changes to service activities can be a delicate, time-consuming activity that might require a designer to have extensive knowledge about the underlying business logic and representations. Also, the deluge of data generated from workflows create daunting challenges in bringing data together in a form that can be efficiently analyzed to make data-driven decisions promptly. The result is an *information gap* where more data does not necessarily increase the ability to process symbiotic pieces of data to arrive at needed insights and decisions in uncertain environments. Currently, there is a strong requirement for methods that allow Web service workflows to adapt transparently to changes in a dynamic and uncertain environment to meet the demands of a business conditions. Also, a by-product of workflows is data silos.

These silos have been known to contain valuable and detailed records of operations, manufacturing, supply-chain management, customer behavior, marketing campaign performance and workflow procedures. Converting data silos into a knowledge base that can provide data-driven inference is also necessary to create added value, deliver value more efficiently and prevent commoditization of the goods and services.

The major contribution of this dissertation is a methodology whereas the BPEL process is represented as an agile model that mimics an *expert system.* This agile model enables greater responsiveness to the evolution of data and efficiently narrows the information gap. The agile model is implemented as a probabilistic network that, in effect, realizes the knowledge base and an inference engine that underlies any traditional expert system. The knowledge base formulates knowledge about the workflow problem domain in a structured way, and the inference engine supports reasoning about events and decisions in the workflow's domain. Two steps are used to accomplish structuring and reasoning. Firstly, sets of (conditional) dependence and independence statements among workflow data transitions, and casual relations are encoded as a directed acyclic graph. Secondly, the strengths of dependency relationships using probability and graphical theory are specified for the inference engine. A core aspect of this approach is that *historical domain-specific functional data (extracted from an operational Web service workflow) is leveraged to capture an initial snapshot of probabilistic beliefs of the underlying phenomenon of the workflow.* Then ongoing run-time data update those beliefs to capture the run-time trends of the workflow and provide just-in-time decision support.

The abstract representation of BPEL process as an expert system addresses both nonresponsiveness and the information gap by providing a *data-derived inferential visibility*

of workflow semantics. The particular knowledge base used is a Bayesian network, and the inference engine is the set of graphical and probabilistic methods that applies to the probabilistic network. The framework uses data as evidence and provides a framework to represent and update beliefs about the workflow behavior. Moreover, data silos can now be used to provide integrated, actionable information and insights. Consider the Web service workflow of an online retailer with a global supply chain. Queries to non-explicitly defined trends about the behavior of workflow users such as "*Which product or supplier is best?"* or "*Should a product or supplier be changed?"* can now be responsive to time-dependent trends. This dissertation contributes a methodology for the construction of this unique model and the framework for the transformation of BPEL into a more agile representation. Ultimately, together, these contributions represent a paradigm that is more responsive to dynamic business trends and uncertainty by the holistic inferential view of business processes.

ACKNOWLEDGEMENTS

This journey would not have been possible without the support and love of my wife Allison and children Isabella and Charles. To my family, thank you for encouraging me in all of my pursuits and inspiring me to follow my dreams. My family to whom this dissertation is dedicated to has been a constant source of love, concern, support, and strength all these years. I would like to express my heartfelt gratitude.

I would like to give special thanks to my dissertation committee. I owe a debt of gratitude to my advisor Dr. Brian Blake for his time and careful attention to detail. I thank him for his untiring support and guidance throughout my journey. To Professor Geoff Sutcliffe and the rest of my committee, whose encouragement and mentoring has inspired this dissertation by challenging my thinking by helping me question assumptions and view issues from multiple perspectives.

DAMIAN CLARKE

University of Miami

August 2015

Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

Chapter 1
INTRODUCTION

In the last 30 years, more new information has been produced than the previous 5000 [1]. The publishing world has seen a deluge of magazines, journals, and books. Major libraries are doubling in size every 14 years; 1000 new books are published every day, and the number of scientific journals has increased to the extent that the vast majority go unread. Communications networks have enabled us to communicate across the globe via voice, fax and e-mail. The World Wide Web (WWW), henceforth referred to the Web, allows us to also send text, pictures, and sound. With over 600 million websites worldwide, 500 million tweets are sent every day, and an estimated 2.4 billion people go online everyday. Individuals can now access information and share things in a way that was not possible in previous generations. The rise of the online use of the Web and social media have changed the way we work, the way we live and the way we communicate and make and maintain friendships. Companies use the Web as their defacto communication platform to interact with their core computer systems, vendors and perform electronic commerce transactions and services.

Also, interactions between people and businesses have become increasingly complex and dynamic, and there has been a fundamental change in the axis of IT innovation. In prior decades, new systems were introduced at the very high end of the economic spectrum, typically within large public agencies and Fortune 500 companies. Over time these systems trickled down to smaller businesses, and then to home office applications, and finally to consumers, students, and even children. In this past decade, however, that flow

has been reversed with customers leading the way, with businesses that are agile enough following.

For instance, a visit to a medical center evolved around the "all knowing" doctor telling patients what was best for them. In the new evolving medical diagnosing model, many patients research the latest medical research online, consult support groups and fully participate along with doctors to make a more collaborative health care decision. In the political and governmental landscape, the public can take part in political campaigns and interact with candidates and Government officials in easier and more efficient ways. Also, blogging enable citizens with varying opinions too easily and quickly post online political views, criticism, rumors, and the latest political conspiracy theory swaying opinions of potential voters and ultimately the political race.

More interestingly, enterprises use the Web not only as an efficient and cost-effective way to interact with their core systems but also as a platform for providing services to other businesses and individual customers. As a way to facilitate this, Web services technology [2] is a frequently used method to conduct web-based commercial transactions such as online banking, investments, loan approval and bill payment. These services are likely to increase at an exponential rate as access to the Web becomes ubiquitous and pervasive on Web-enable devices [3]. Web-enabled devices are in innumerable forms and sizes. Embedded in clothing, wristwatches, eyeglasses, electrical appliances, the electrical grid, other public utilities, motor vehicles, televisions and other physical devices they are becoming ubiquitous. The portability of Web-enabled devices, together with their ability to connect conveniently to networks in different places add to the many possible connectivities of individuals to services.

The rise in the need for connectivity has heralded the symbiotic development of interoperable disparate systems and networks. The Web proved its effectiveness to meet the demand by using simple protocols over the internet to provide services and applications. In particular, the HTTP request-reply protocol [4] allows general-purpose clients such as browsers to display web pages and other resources regarding their web address (URL). Despite these features, the use of general-purpose clients such as browsers even with the enhancements of embedded application-specific programs in HTML restricted the potential scope of applications. In the original client-server model, both the client and server were functionally specialized [5]. Ironically, this model has seen a return in the form of Web Service Technologies, in which application-specific clients, instead of general-purpose clients such as browsers, interact with services with a functionally specialized interface over the Internet.

The Web service model [6] has proven to provide an infrastructure for maintaining a richer more structured form of interoperability between disparate clients and servers. It provides a basis whereby client programs in one organization may interact with servers in another agency without human supervision. In particular, Web services allow for complex business functionality to be developed by integrating services that are possibly offered by different companies.

Advances in connectivity and interoperability have resulted in few business transactions that involve just one vendor. Supply chains, delivery chains, customer support services and partner ecosystems across firms are integrated making companies much more collaborative than ever before. The increase in the demand for online services has resulted in approximately one trillion connected objects and devices on the Internet, which generate

billions of gigabytes of new data each day [7]. The exponential growth in data has resulted in a data deluge [8]. Various data silos include operations, manufacturing, supply-chain management, customer behavior, marketing campaign performance and workflow procedures. The data deluge in turn motivated investments in business infrastructure such as Hadoop [9], HBase [10] and MongoDB [11], which have improved the ability to collect data throughout the enterprise. Records of data are now virtually in every aspect of the business that is often inherently instrumented for data collection. With the efficiency of data collection improving there is an increasing interest in determining whether useful information and knowledge can be extracted from the data is on the front burner.

Despite advances in connectivity, interoperability and data collection, enterprises of all sizes and shapes are still being forced to sharpen their competitive advantage or suffer commoditization of the goods and services [12]. Moreover, companies expand their reach both organically and through acquisitions which mean they have to focus more resources on their core businesses, core competencies, and core differentiation. Data-driven decision-making (DDD) [13] is an approach to address these challenges. Studies in [13] show firms that are more data-driven are more productive. DDD has proved to be correlated with a higher return on assets, return on equity, asset utilization, and market value, and the relationship seems to be causal. This dissertation addresses the challenge of infusing DDD in Web service technologies, particularly in the BPEL process.

1.1. PROBLEM DESCRIPTION

The triage of concerns; connectivity, interoperability, and the data deluge are an impetus for firms to work towards becoming smarter. Web service technologies have adequately solved the ability to connect people and services, and integrate disparate

services. However, data-driven decision-making to create added value, deliver value more efficiently and avoid suffering commoditization of the goods and services still proves to be imperative and elusive. Companies have become increasingly capable of processing massive amounts of data [9][10][11]. However, questions such as: *"What can I now do that I could not do before?"* or *"What can I do better than I could do before?"* are still not easily answered. This dissertation introduces an approach deriving responses to these questions, which are hidden within workflow specifications such as the Business Process Execution Language (BPEL) process and in user trends embedded in data silos.

Web-service technologies have been a popular choice for many organizations to implement business processes workflows to match the need for connectivity and interoperability. These techniques provide for the efficient and effective selecting and integrating of inter-organizational and heterogeneous services on the Web. The Service Oriented Computing (SOC) [14] design paradigm is used to implement the Web service model by positioning business processes as software components exposed through network-accessible, platform, and language independent interfaces. Loosely coupled services are orchestrated by a central processing service executed in BPEL [15]. BPEL realizes the composition and execution workflows of collaborating applications distributed within and across organizational boundaries. Also, the BPEL language describes business processes by specifying message exchanges behavior between different parties while specifying the full implementation logic of the company workflow. As the orchestrating service, it describes the execution order of services and message exchanges from a centralized perspective. These workflows scale to complex and distributed Web-scale Web Service Workflows [16] that realize the business logic of organizations, its partners, and customers.

The creation of business logic in the BPEL process is usually statically formulated in the early stages of the software design lifecycle such as in the requirement and design analysis documentation [17]. These documents are time-consuming to make and leaves little room for iteration. In turn, this renders the output of a software design lifecycle such as workflows non-responsive to just in time changes to time-dependent and often uncertain business trends.

Shortcomings in BPEL language features provide opportunities for improvements. As a brief overview, the Web Services Business Process Execution Language (WS-BPEL), referred to as BPEL, is an XML-based language for describing the behavior of business processes. The language gives its users the freedom to describe business processes in two ways: executable or abstract. The abstract process is the business protocol. It specifies the message exchange behavior between different service without revealing the internal behavior for any one of them. The executable process determines the full implementation logic of the business process and is meant to be executed by an execution engine. An Orchestration describes how services can interact with each other at the message level, including the business logic and execution order of the interactions from the perspective and under the control of a single endpoint.

BPEL processes use the XML [18] tag <*partnerLinks*> to model conversational relationships with its partners. The relationship is established by specifying the roles of each party and the interfaces that each provides. Process participants (partner‘s web services) must be defined and bound to the process flow at design time. Design time only partners discovery and bounding, limits BPEL ability to be responsive and adaptable to run-time trends. Manual changes to service activities can be a delicate, time-consuming activity that

might require a designer to have extensive knowledge about the underlying business logic and representations.

BPEL defines data variables in terms of the Web Service Definition Language (WSDL) [19] message types, XML Schema types (simple or complex), or XML Schema elements [18]. Variables allow processes to maintain state between message exchanges. In addition, using <*assign*> and <*copy*> message data can be copied and manipulated between variables. However, dynamic message mediation is not possible. Mediation processes in-flight messages by either modifying or transforming a message or routing messages (or cloning messages) to other or additional destinations that were not statically defined. Dynamically allowing or disallowing a message to be delivered based on run-time conditional logic requirements can greatly enhance the ability of BPEL to adapt to runtime demands.

Even though, BPEL engines can support some ad-hoc deviations from pre-specified BPEL schema, approaches has fallen short particularly once the process has begun execution [20]. BPEL applicability to only well structured, rigid service flows limits its ability to be responsive. Barriers towards adaptivity in BPEL stems from its complexity and the lack of formal semantics in its language. Several approaches exist in this context, e.g., [21][22][23]. However, current formalizations and verification methods are either incomplete (i.e., focussing on selected BPEL language elements only) or are not standardized. In particular, one will not be able to reason about the uncertainty of a dynamic business environment if there exist no formal basis and mechanism for inference.

Finally, data has become the economic lifeblood of organizations, whether as intellectual property, critical methods, business records, operational data, and especially the

higher-order knowledge that drives an organization's strategic activities. Data -- especially the actionable knowledge derived from it -- has become the dominant supply-chain component of whatever products and services provided to the marketplace. Unfortunately more data does not necessarily increase the ability to process symbiotic pieces of data to arrive at needed insights and decisions in uncertain environments thereby resulting in an information gap.

In response to today's increasingly data-centric business environment, Web-service workflows lack the ability to analyze the deluge of data generated by its infrastructure. This data can contain a definitive record of all the activity and behavior of customers, users, transactions, applications, servers, networks and mobile devices. Data can be captured in service logs, configurations, data from APIs, message queues, change events, the output of diagnostic commands, call detail records, sensor data from industrial systems and more. Influential time dependent trends may be implicitly and explicitly present in workflow data such as customer buying patterns and expectations. Unlocking the intelligence in workflow data and understanding how the data interrelates, can resolve problems faster, reduce downtime and improve user satisfaction resulting in valuable business and operational intelligence and insight.

1.2. MOTIVATION

Within both business-to-business (B2B) and business-to-consumer (B2C) transactions, organizations are transforming themselves to become more data-centric by embracing data-driven decision-making based on transactional data and analytics [24]. Analytics is the discovery and communication of meaningful patterns in data. A framework that can apply analytical methods to workflow semantics and data can be used to gain a far greater

understanding of business processes. Identifying underlying trends and, which outcomes are correlated, and, most importantly, which actions are most likely to produce concrete results that can provide a competitive advantage. It is important to note that while some of the relationships present in workflow data may arise from the business logic embedded in the enterprise process, other relationships may have no business explanation. For example, consumer buying trends are not explicitly defined in the business logic, yet holds potential for competitive advantages. Many agencies currently use data to create reports on what is currently happening but opportunities to use data to make predictions about what will happen and to understand the decision levers available to influence what will happen remain underexploited.

We are living in the big data era where new distributed data storage and processing technologies such as Hadoop [9] and NoSQL [25] are becoming the standard techniques. Also, self-service business intelligence has become the preferred approach to analyzing data. Data has turned from a mere reporting source or administrative tasks to a critical asset for many lines of businesses. With the right data, organizations can optimize their business processes, improve their customer relationships, and differentiate themselves from the competition. With that, the dependency of agencies on data has intensified. The change in data has in turn changed the organizations.

Data is a necessary ingredient to reason about uncertainty when performing analytics. Data analytics is a process for obtaining raw data and converting it into information useful for decision-making by users. Data is collected and analyzed to answer questions, test hypotheses or disprove theories [26]. The continued proliferation of data and the relative ease of its capture and storage, organizations now have more information than

they ever thought possible. The resulting data deluge has contributed to an increasing interest in determining whether useful information and knowledge can be extracted to derive Data-driven decision-making (DDD) [13].

In [27], the extent to which big data technologies add value to firms were examined. After controlling confounding factors big data technologies associated with significant additional productivity growth. In [13], a measure of DDD was developed that rates firms as to how strongly they use data to make decisions throughout the company. The result shows that statistically, the more data-driven a firm is, the more productive it is. A data-driven firm is correlated with a higher return on assets, return on equity, asset utilization, and market value, and the relationship seems to be causal.

Data analytics can influence decisions from a company's marketing and customer care to its mergers and acquisitions. Analytics has improved organizations' ability to innovate in areas such as marketing, operations, and finance. The idea that data and analytics can be used to build competitive advantage and advance innovation is essential to the business models of many online companies, such as Match.com, PayPal, eBay, Amazon, and Google.

For example, Match.com analyzes billions of data points from the last 17 years to continuously improve a series of more than 15 matching algorithms. In contrast, their competition uses psychological-based methodologies while working closely with psychologists. However, there are many Psychological theories, and they are difficult to represent concretely as opposed to a mathematical equation derived from data. Match.com is among a small but growing cadre of companies — both online and off — that are mastering the use of data and analytics to drive innovation and build competitive advantage.

Data-driven decision insights can enable agencies to monitor their environment more proactively.

1.3. RESEARCH QUESTIONS AND APPROACH

The general objective of this dissertation is to advance the current efforts in utilizing Web service workflows orchestrated by a BPEL process, and the data it generates to provide inferential visibility on workflow domains. This approach provides a graphical and probabilistic abstraction of the BPEL process, which allows for inferential visibility of the workflow.

The objectives are twofold:

i) Provide a methodology to make Web service workflows responsive to time-dependent trends and,

ii) Narrow the information gap created by increasing data by providing opportunities for content assessment of service workflows.

We introduce a methodology for transforming BPEL processes into a normative expert system consisting of a knowledge base and an inference engine. We use a probabilistic network, specifically a Bayesian Network, as the knowledge base to represent our workflow activities in a structured format. To support reasoning about events and decisions in the workflow's domain we use graphical and probabilistic theories for the inference engine.

To meet these objectives, the following research questions are addressed:

1. What is an effective representation or model to assess service workflows to determine workflow or user optimality?
    a. What is the process for BPEL transformation?
    b. How is model robustness achieved?

2. Can such a model be used for decision support?
    a. Can we assess if a workflow can reach the content/domain-specific objectives of the user?
    b. Can we automatically optimize (choose viable paths) within the workflow?

The first question addresses the responsiveness of Web service workflows. Responsiveness is the cornerstone of usability and utility, but more than that, responsiveness means that problems may be detected quickly and dealt with effectively. Responsive Workflows focus on providing rapid and consistent response times, establishing reliable upper bounds, so they deliver a consistent quality of service. This consistent behavior in turn simplifies error handling, builds end user confidence, and encourages further by utilizing We address responsiveness by utilizing probabilistic networks as an effective representation or model to assess service workflows.

Probabilistic networks are graphical models that provide an intuitive language for capturing the set of (conditional) dependence and independence properties. Embedded business logic rules among workflow data transitions, informational precedence, and preference relations are modeled as the structure of the probabilistic network as nodes and edges. The nodes are the variables and the edges represent a casual relationship between variables or parameters. The resulting graph, known as the qualitative component or knowledge base, encodes the set of (conditional) dependence and independence properties associated with BPEL variables to form the structuer of the Bayesian Network.

Dependencies within the probabilistic network are realized by joint prior and posterior probability distributions [28]. Marginal distribution realizes the strength of individual nodes. These distributions represent the strength of dependence and

independence relations and evidence before and after data is applied to the network. The resulting structure of the graph realizes the behavior of a workflow from the perspective of relationships between BPEL variables and where probability values represent the strengths of relationships. On demand querying of relationships and its strength realize responsiveness in our model of a Web Service workflow.

The second question addresses narrowing the information gap in silos of data generated by Web service workflows. Bayesian Networks [28] are the specific class of probabilistic network used to represent workflows. The quantitative component or inference engine of the probabilistic network, or more precisely, the Bayesian Network specifies the strengths of dependence relations between variables and is represented as a joint probability distribution. Probability and graphical theories are used to apply probabilistic values to relationships between variables. Prior marginals are determined from expert or domain knowledge. Posterior marginals are computed from historical data from silos. The result is the parameterization of the quantitative component, which captures a probabilistic view of workflow trends embedded in data silos. Also, the quantitative component serves as an inference engine. An inference engine derives conclusions from facts and rules contained in the knowledge base using various query types. Ongoing workflow data are treated as evidence to update beliefs withing the framework. Ongoing run-time data are used to update beliefs thereby providing run-time trends and hence just-in-time decision support.

In general, creating and determing the probabilistic values of a Bayesian network model of a workflow allows us to exploit both the qualitative and quantitative components to reason efficiently about workflow events and make decisions in the workflow domain while considering its inherent uncertainty. Our framework provides an intuitive interface

for leading efforts that is focused specifically on empowering knowledge workers to negotiate the complexity of web-scale Web Service workflows in real time.

1.4. CONTRIBUTIONS

The original, scientific contributions of this dissertation are two fold:

1. A representation, principled methodology, model for capturing historical and ongoing data and process details for long-standing workflow including *data-centric details.*
2. An analysis approach for providing decision support when optimizing long-standing workflow operations, providing service workflow prediction, capturing user specific causal factors and enabling on-demand service workflow intervention.

1.5. ORGANIZATION

The structure of the dissertation is as follows:-

Chapter 2. Literature Review.

This chapter reviews the literature most related this work. We focus on the dynamic service composition methods, verification techniques, process mining and list other similarly related works. We detail the state of the art in the various techniques.

Chapter 3. An Overview of Web Services.

An outline of Web services technology and its architecture is provided.

Chapter 4. Theoretical Explanations.

This chapter describes the theoretical explanations needed to transform a Web service workflow realized as a BPEL process into a normative expert system representation. Topics include graphs, conditional independence, probabilistic networks, probabilistic calculus and finally Bayesian networks.

Chapter 5. Methodology, Evaluation, and Discussion

This chapter describes the main contribution of this dissertating by demonstrating the method of transforming a BPEL process into a normative expert system. Two use cases are described to show modeling steps.

Chapter 6. Conclusion and Future Work

This chapter concludes the dissertation and proposes some directions for future work.

Chapter 2
LITERATURE REVIEW

The state-of-the-art in service-oriented computing is primarily focused on the design specifications of services and not the running operational systems [29][30][31]. In addition, considerable amount of work has been done on modelling (parts of) BPEL and developing verification techniques and tools for BPEL [32][33][34][35][36]. As such, these projects are not related to the approach in this dissertation as we look at the real data content of messages as a method to re-engineer operational web service workflow systems. However, we discuss those approaches, which serve to add adaptability in the workflow without the use of the real data content. Section 2.1 list static and dynamic workflow techniques. Section 2.2 list Artificial Intelligence (AI) approaches to runtime adaptability. We discuss various verification techniques in Section 2.3. Finally, Section 2.4 investigates the closest related body of work, process mining.

2.1. WORKFLOW TECHNIQUES

Workflow systems are designed to assist in carrying out work procedures and contain organizational knowledge of organizations. A workflow is defined as "systems that help organizations to specify, execute, monitor, and coordinate the flow of work items within a distributed environment" [37]. In the workflow-based composition methods, techniques can be divided into *static* and *dynamic*. In static workflow technologies, an abstract process model is created before the composition activities initiates. The conceptual process model includes a set of tasks and their data dependency. Each task contains a query clause used to search atomic Web services to fulfill the task. The automation is done in the selection and

binding of atomic Web service. The most commonly used static method is to specify the process model as a graph. Whereas, dynamic workflow techniques creates both the process model and selects atomic services automatically. This requires the requester to specify several constraints, including the dependency of atomic services and the user's preference.

An early contribution concerning dynamic change to procedures in the context of workflow systems came from [38], which model procedures with a particular kind of Petri nets referred to as sequel flow nets. The steps within procedures or activities are modeled as transitions and precedence (i.e. the ordering relation between events) as arcs in the flow net. A flow net is a Petri Net with two distinguished places; namely the input place and the output place. The activities of the procedure are modeled by transitions, each of which has a name, at least one input place and at least one output place. The approach is called synthetic cut-over change: Given a particular procedural change, its change region is defined as the part of the net containing all the activities directly affected by the change. The old region is the change region prior to the change, and the new region is the change region after the change. The jobs evolving outside the transition region are not affected by the change. The jobs inside the old region are "transferred" to the new region. This transfer can result in the creation of new jobs or the destruction of old jobs. The change is said to be correct if the resumption is intended to finish the in-progress work according to either the old or the new procedure. The question as to how the change regions are selected remains unanswered.

Application Development Based on Encapsulated Premodeled Process Templates (ADEPT) [39] defines a complete and minimal set of change operations that support users in modifying the structure of a running workflow, while maintaining its (structural) correctness and consistency. Correctness properties are defined to determine whether a

particular change can be applied to a given instance. If these constraints are violated, the change is either rejected, or the correctness must be explicitly restored with exception handling techniques. A directed acyclic graph represents control flow and data flow between tasks is defined by connecting their parameters with elements from data element where the input (output) parameters of the WF schema are logically treated as the output (input) parameters of its start (end) node.

Eflow [40] is a commercial process management system that supports adaptive and dynamic composite e-services. Composite services are modeled as business processes and are enacted by a service process engine. Eflow uses a static workflow technique. It models services as a graph, which defines the execution order among the nodes in the process. The graph may include service, decision, and event nodes. Service nodes represent the invocation of a basic or composite service; decision nodes specify the alternatives and rules controlling the execution flow while event nodes enable service processes to send and receive several types of events. Arcs in the graph may be labeled with transition predicates defined over process data, meaning that as a node is completed, nodes connected to outgoing arcs are executed only if the corresponding transition predicate evaluates to true. The eFlow model also includes the notion of transactional regions. A transactional region identifies a portion of the process graph that should be executed in an atomic fashion. If for any reason the part of the process defined by the transactional region cannot be completed, then all running services in the region are aborted and completed ones are compensated, by executing a service-specific compensating action. Compensating actions may be defined for each service or may be defined at the region level. Transactional regions may also include the specification of different isolation modes, that prevent data read or modified by nodes

in the regions to be accessed by services that are outside the transactional region. Eflow addresses the challenges of an adapting environment in the following ways dynamic service discovery, multiservice nodes, and generic nodes.

Dynamic service discovery is incorporated using service selection rules specified in service nodes and are executed by a service broker to return the appropriate service. A mapping function must be defined for a <service node, service description> pair before the service can be invoked in the context of the service node. The default or other plugged-in brokers can be used to dynamically discover local and external services respectively. This feature allows for:

- selecting appropriate services depending on the customer's requirements
- decouple service selection from the process definition
- dynamically discover the best currently available service that fits the need of a specific customer.

Multiservice nodes is a particular kind of node that allows for multiple, parallel activation of the same service node. The number of service nodes to be activated is determined at run time in one of the following ways:

1. It can be determined by the number of service providers able to provide a given service.
2. It can be equal to the number of list items of input parameter.

A key feature of the multiservice node is the ability to select either all or any of the parallel service to invoke and wait for completion.

Dynamic service node creation is the support of the notion generic service nodes. Unlike ordinary service nodes, generic nodes are not statically bound or limited to a specific

set of services. Instead, they include a configuration parameter that can be set with a list of actual service nodes either at process instantiation time or at runtime. The specified services will be executed in parallel or sequentially depending on an *executionMode* attribute of the generic service node.

2.2. ARTIFICIAL INTELLIGENCE PLANNING

Broadly, AI planning finds a sequence of actions that transform an initial state into a state in which the goal is satisfied. In general, a planning problem can be described as a five tuple $\{S, S_0, G, A, \Gamma\}$, where S is the set of all possible states of the world, $S_0 \subset S$ denotes the initial state of the world, $G \subset S$ denotes the goal state of the world. The planning system attempts to reach A, which is the set of actions the planner can perform in attempting to change one state to another state in the world, and the translation relation $\Gamma \subseteq S \times A \times S$ defines the precondition and effects for the execution of each action.

In the terms of Web services, S_0 and G are the initial states and the goal states specified in the requirement of the Web service requesters, A is a set of available services and Γ denotes the state change function of each service. The general assumption of such kind of methods is that each Web service can be specified by its preconditions and effects in the planning context. Firstly, a Web service is a software component that takes the input data and produces the output data. Thus, the preconditions and effects are the input and the output parameters of the service respectively. Secondly, the Web service also alters the states of the world after its execution. If the user can specify the preconditions and effects required by the composite service, a plan or process is generated automatically by logical

theorem prover or AI planners without knowledge of predefined workflow. However, during the planning, the business logic can provide constraints in the planning setting.

The body of work under the heading Semantic Web services [41], support the construction of tools and methodologies, and promote the use of semantically well-founded reasoning about services. The Web Ontology Language for Service (OWL-S) [42] standard derived from the semantic Web research is the only Web service language that announces a direct connection with AI planning by using Description Logics [43] as its logical foundation allowing for logical expressions. In the following we introduces a list of Web service composition methods based on AI planning. We do not claim that we have an exhaustive list of the methods. We classify the methods into two categories, namely, the situation calculus and rule-based planning.

2.2.1. SITUATION CALCULUS

The situation calculus is a logical language for representing changes [44] .The basic concepts in the situation calculus are situations, actions and fluents. Briefly, situations are the finite sequence of actions that has been performed since the initial situation S_0, actions are what make the dynamic world change from one situation to another when performed by agents. Finally, fluents are situation-dependent functions used to describe the effects of actions. Situation calculus based on first-order logic (FOL), a situation variable models new states of the world, action objects model activities, uses inference methods developed for FOL to do the reasoning

In the Web service domain, software agents reason about Web services to perform automatic Web service discovery, execution, composition and inter-operation. The user's request (generic procedure) and constraints can be presented by the first-order language of

the situation calculus [45]. Golog, a logic programming language built on top of the situation calculus and has been adopted as a natural formalism for representing and reasoning in the Web service composition problem [46]. The software agent knowledge base provides a logical encoding of the preconditions and effects of the Web service actions in the language of the situation calculus. The agents use procedural programming language constructs composed with concepts defined for the services and constraints using deductive machinery. A composite service is a set of atomic services which is connected by procedural programming language constructs (if-then-else, while and so forth).

Situation calculus suffers from some disadvantages, one of which is related to the fact that the Situation Calculus is based on predicate logic, which means that the structure of situations has to be axiomatized. Axioms are used to ensure that in every model, the situations form an infinite tree of which S_0 is the root, and where each sequence of actions corresponds to a path to a particular node in that tree. Apart from the somewhat cumbersome notation, the problem with this representation of situations is that model theoretic proofs tend to be quite tedious and involved. Other issues include large search space, large number of axioms to be defined for one action and the proof may not lead to the best (shortest) plan.

2.2.2. RULE BASED

Rule based learning consists of a knowledge base of rules and an inference engine to apply those rules. Rules are of the form *IF some condition THEN some action*.

Medjahed [47] present a technique to generate composite services from high-level declarative description. The method uses composability rules to determine whether two services are composable. The composition approach consists of four phases. First, the specification phase enables high-level description of the desired compositions using a

language called Composite Service Specification Language (CSSL). Second, the matchmaking phase uses composability rules to generate composition plans that conform to service requester's specifications. The third phase is selection phase. If more than one plan is generated, in the selection phase, the service requester selects a plan based on quality of composition (QoC) parameters (e.g. rank, cost, etc.). The final phase is the generation phase. A detailed description of the composite service is automatically generated and presented to the service requester. The main contribution of this method is the composability rules, because they define the possible Web service's attributes that could be used in service composition. Those rules can be used as a guideline for other Web service methods based on planning.

The disadvantages of rule based planning are the number of rules to represent a BPEL process can be very large and the resulting rules can be very lengthy. Also, rules specification can be error prone [48].

2.3. VERIFICATION TECHNIQUES

Different verification techniques and tools have been developed for BPEL [49]. In this section, we present an overview of some those techniques and tools.

2.3.1. PETRI NETS

Petri nets was first introduced by Carl Adam Petri in 1962, it is able to describe complex processes which are characterized as being concurrent, asynchronous, parallel and nondeterministic with both formal notion and graphical representation. It is a directed, connected, and bipartite graph in which each node is either a place or a transition and is used to model concurrency and synchronization in distributed systems [50]. Tokens occupy

places. When there is at least one token in every place connected to a transition, the transition is enabled. Any enabled transition may fire removing one token from every input place, and depositing one token in each output place.

The elements and the rules of Petri nets are described as follows:

- *Places*: graphically represented as circles. A place may have zero or more tokens. Places are passive components and model the system states.
- *Transitions*: graphically represented as squares. Transitions are active components modelling activities, processes or events.
- *Arcs*: graphically represented as arrows. Arcs link a place to a transition or vice versa, never places or transitions.
- *Tokens*: graphically represented as black dots and occupy places. The distribution of tokens in the Petri nets is changed by the occurrence of transitions.
- *Marking*. A distribution of tokens over places that represents a configuration of the net is called a marking. A marking represents the current status of the Petri net.
- *Enabled* Transition: A transition is enabled if each of its input places contains at least one token.
- *Firing.* An enabled transition can fire (i.e., it occurs). When it fires it consumes a token from each input place and produces a token for each output place.

Since the execution of Petri nets is nondeterministic i.e. if a transition is enabled, it may or may not fire and multiple tokens may be present anywhere in the net (even in the same place), Petri nets are well suited for modelling the concurrent behavior of distributed systems. There are currently several Petri net extensions (e.g.: colored, timed, stochastic,

continuous, hybrid, hierarchical, functional), forming a very versatile framework for both qualitative and quantitative analysis. Petri nets have been extensively used to model and verify business processes [32]. A mapping from BPEL activities to petri Nets is given in [51].

Advantages include:

- Formal semantics in addition to the graphical nature Unlike other graphical modelling languages, the semantics of the classical Petri net and several enhancements (color, time, and hierarchy) has also been defined formally [52]. This means that a workflow procedure specified in terms of a Petri net is unambiguous, i.e., the meaning of each construction is clear and there is no room for multiple interpretations.
- Expressiveness: Petri nets support all the primitives needed to model a workflow process such as including sequential, parallel, conditional and iterative routings.
- Straightforward mapping: Mapping BPEL concepts onto Petri nets is rather straightforward: tasks are modeled by transitions, conditions are modeled by places, and cases are modeled by tokens.
- Abundance of analysis techniques: Petri nets benefit from the availability of many analysis techniques. These techniques can be used to prove properties (safety properties, invariance properties, deadlock, etc.) and to calculate performance measures (response times, waiting times, occupation rates, etc.).

Disadvantages include:

- Complexity: A major weakness of Petri nets is complexity. In the real world, Petri-net based models tend to become too large for design and analysis even for a modest-size system. Moreover, since Petri net is an abstract mathematical language, people that have no prior experience and technical training may have troubles to model business processes properly using Petri nets.
- Execution: Petri nets are regarded as a formalism rather than an execution language. Due to the lack of available Petri net execution engine, it is infeasible to directly deploy the Petri net-based workflow specifications and execute their instances

2.3.2. PROCESS ALGEBRA

Process Algebras are mathematically rigorous languages with well defined semantics that permit describing and verifying properties of concurrent communicating systems. They can be seen as models of processes, regarded as agents that act and interact continuously with other similar agents and with their common environment. The agents may be real-world objects (even people), or they may be artifacts, such as or software systems [53]. Numerous process algebras have been introduced including, for example, the Calculus of Communicating Systems (CCS) [54], LOTOS [55] and the π-calculus [56]. Process algebras are usually modelled by means of labelled transition systems. The transition relation of the labelled transition system is generally defined by a collection of axioms and rules. Many different equivalence relations on the set of states of the labelled transition system have been introduced. These behavioral equivalences capture which states behave the same. An overview of the applicability of process algebras in the context of web services

is presented in [34]. Existing process algebras and also new process algebras have been used to model BPEL.

In [57], Kramer and Magee present a process algebra named FSP (Finite State Process). Each FSP represents a finite labelled transition system. They also present a tool for FSP named Labelled Transition System Analyzer (LTSA). This tool takes as input an FSP and translates it into a labelled transition system. Subsequently, this labelled transition is analyzed. LTSA can check for safety and progress properties.

In [58], Foster et al. show how LTSA can be exploited to check if a Web service composition implemented in BPEL satisfies a web service composition specification captured by Message Sequence Charts (MSCs). Both the BPEL process and the MSC are translated into FSPs. In [59], Foster et al. use LTSA to check compatibility of web service compositions in BPEL. A case study by Foster et al. is presented in [60].

In [61][62], Ferrara, Salaun and Chirichiello present a two-way mapping between the process algebra LOTOS and BPEL. Most BPEL activities including fault handlers, compensation handlers and event handlers are considered. By going from BPEL to LOTOS, the toolbox CADP [63], standing for Construction and Analysis of Distributed Processes, can be exploited for the verification of BPEL processes. Counterexamples produced by CADP, given in LOTOS, are mapped back to BPEL.

Each different process algebra has its own advantages and disadvantages. The fact that an expression of a process algebra always reveals its syntactic structure, can be viewed as an advantage. For example, it is often possible to argue ‘by syntactic induction’; or build proof systems ‘by syntactic definition’ around such an algebra; and the availability of operators for the modular construction of systems is usually appreciated by practitioners.

On the other hand, one of the disadvantages of such a syntactic view is that some effort has to be invested into the definition of the behavior of expressions. While this can usually be done inductively, it is still necessary to go through all operators one by one and to define their semantics individually. Sometimes, this leads to ad-hoc (and very disparate) definitions [64].

2.3.3. ABSTRACT STATE MACHINE

Abstract state machines (ASMs) have been used to model a large variety of languages. A basic ASM consists of a finite set of transition rules. Each transition rule consists of two parts: a Boolean expression and a finite set of assignments. The transition rules captures which transitions the ASM can make. A transition takes the ASM from one state to another. The latter state is obtained from the former state by performing the assignments of those transition rules whose Boolean expressions evaluate to true. For an introduction to the ASM approach, we refer the reader to, for example, [65]. ASMs have also been used to model BPEL. Below, we provided a brief overview of this work.

Farahbod, Glasser and Vajihollahi [66] model all key aspects of BPEL. For example, the basic and structured activities, correlation, and compensation, event and fault handling are modelled. To model interaction, Farahbod et al. introduce so-called inbox and outbox managers that deal with the message exchanges. For dealing with some of real time aspects of BPEL, like time-outs, an abstract notion of global system time is introduced and additional constraints on the sequences of transition are imposed. The ASM model for BPEL proposed by Fahland and Reisig [67] extends and refines Farahbod et al. model. Reisig discusses the model by means of an example in [68]. In [67], the focus is on fault handlers

and event handlers. It is shown how these BPEL features can be modelled within the ASM framework.

The advantage of ASMs is that they are close to logic, which makes the overall design easily amenable to well-understood mathematical techniques. Essentially the mathematical foundation of ASMs supports the formal verification of dynamic systems designed by means of ASMs.

2.4. PROCESS MINING

The closest related body of work is in the area of process mining, first in the context of software processes [14] then in relation to business processes [15] [16]. Process mining is focused on the ability to derive correct process models from real-life data. Currently, a broad variety of approaches are available; the α-algorithm according to van der Aalst et al. [69], the $++\alpha$ algorithm according to Wen et al. [70], the heuristicmining algorithm according to Weijters et al. [71], the DWS-algorithm according to Greco et al. [72], the multiphase-algorithm according to van Dongen and van der Aalst et al. [73], the genetic-mining algorithm according to de Medeiros et al. [74], and the theory-of-regions-based algorithm according to van Dongen and Busi et al. [75], as provided in the ProM tool [76].

In the area of discovering business processes, Petri Nets have been used along with expert knowledge to encode the control flow [17]. We gave an overview of Petri Nets in section 2.3.1. with an accompanying list of detailed references. A limitation in this work is that the domain expert's control flow model can, at times, be erroneous. Also, process-mining approaches have problems when analyzing noisy, incomplete data, multiple occurrences of activities, or richness of process types and variants [77].

2.5. OTHERS

There are other service engineering projects that investigate the real data content. One such work is in the area of automated or semi-automated web service testing. These projects evaluate real, web service outputs to predict the specific data content that requires testing[13] [5]. Consequently, message data can be varied and Web service operations tested. Unlike these approaches that tend to work on just one Web service, models in our work leverages models across multiple services in a web service workflow.

2.6. DISSERTATION FOCUS

One strength of formal approaches such as Petri Nets, state and activity charts, temporal logic or process algebra lies in the offered mechanisms for specifying, analyzing, and verifying the properties of static workflow structures, e.g., regarding state transitions, deadlocks, or the reachability of states [39]. Adequate mechanisms for modifying these structures at run-time, however, are missing for the most part [38]. General purpose models mentioned above makes the analysis of more complex workflow models extremely costly [78], which may cause a significant overhead when complex structural changes become necessary at run-time.

This dissertation follows the concept of a static workflow technique by manually modelling workflow task and data dependency as an abstract process model. In contrast to workflow and service composition techniques the focus of service selection is based on finding the optimal services in an existing workflow that optimizes specific user needs. These optimal services define a workflow path for user specific needs. In such a setting, creating compositions from scratch is costly. Instead, we advocate automatic adaptation of

existing compositions to fit the requested service demands. Our approach starts from existing BPEL processes. The details and constraints on the environment are expressed by a Bayesian network. The network automatically decides which services are necessary to complete a desired service in a given BPEL process and which services can be removed from the process by using the Bayesian network inference engine.

Instead of design-time service rules, multiservice and generic node specification, the main underlying idea of this dissertation is based on the use of historical and ongoing workflow data to facilitate data-driven decision-making. This allows user specific trends to be captured from data and determine "selection rules" for user optimized workflow paths. Thus, even though there is an initial workflow, the automatic modification allows the workflow to adapt to dynamic changes in the environment. Without such an automatic adaptation, the workflow would have to be changed manually at compile time. This is clearly not acceptable in dynamic situations. Another advantage of adaptable workflows is that it supports process-oriented structure. In other words, sometimes—based on the current situation—a process does not have to execute all of its tasks. If the workflow can adapt itself, then it can decide not to execute these unnecessary tasks. Such an adaptation would improve the efficiency considerably, because the most suitable workflow would be generated at run time based on the request.

2.6.1. FEATURES

- One type of reasoning help is to reason about procedural change (both temporary and permanent) within structured workflows. To perform this type of reasoning, it is useful to have formal definitions and apply mathematical analyses.

- Presentation of multiple views of how an organization is perceived to work (or how it has done procedures in the past,) as well as other information presented by the model, can be very useful to workers without any automation.
- Changes/path selection is based on the business trends in the data. Other methods show change can be made but do not demonstrate when and where to make these changes
- Normally, several instances of a specific WF type are active at the same time. As changes of different kinds may be applied to these instances during their execution, several issues must be addressed. This work does not change any instance
- Historical and ongoing data is used to update probabilities.

Chapter 3
AN OVERVIEW OF WEB SERVICES

Over the past years, the Internet has evolved from just a communication media into a platform for B2B integration. Emerging technologies and industrial standards in the field of Web services enable a much faster and easier cooperation of distributed partners. This chapter gives an overview of the application of Web services realize distributed, cross-organizational business processes. A Web service [79] is a self-describing, self-contained modular application that can be published, located, and invoked over a network, e. g. the Internet. A service-oriented architecture is a way to design, implement, and assemble services to support or automate business functions. Standardized interface realizes accessibility to Web services.

3.1. SERVICE ORIENTED COMPUTING

The evolution of software development began as modular mechanisms that perform concise, domain-specific task. Advancements to this approach took place in the early 1960s, with the development of the object-oriented paradigm [80] and later in the same decade through component-based programming [81]. An evolutionary leap occurred in the late 1990s with the connection of components on the WWW. This lead to the emergence of the Service Oriented Computing (SOC) paradigm and the Service Oriented Architecture (SOA) [82].

SOC is a paradigm for designing software applications known as *services*. Services are self-contained, self-describing, modular applications that can be published, located, and invoked in a distributed computing environment. There are two types of services: atomic and composite. An *atomic* service is a well-defined, self-contained function that does not

depend on the context or state of other services. A *composite* service is an assembly of atomic or other composite services. A service within a composite service may depend on the context or state of another service that is also within the same composite service.

Figure 3-1 illustrates a basic SOA process. It shows a service consumer at the right sending a service request message to a service provider at the left. The service provider returns a response message to the service consumer. The definition of the request and subsequent response connections are understandable to both service consumer and the service provider through standard Web services communication protocols [83][84].

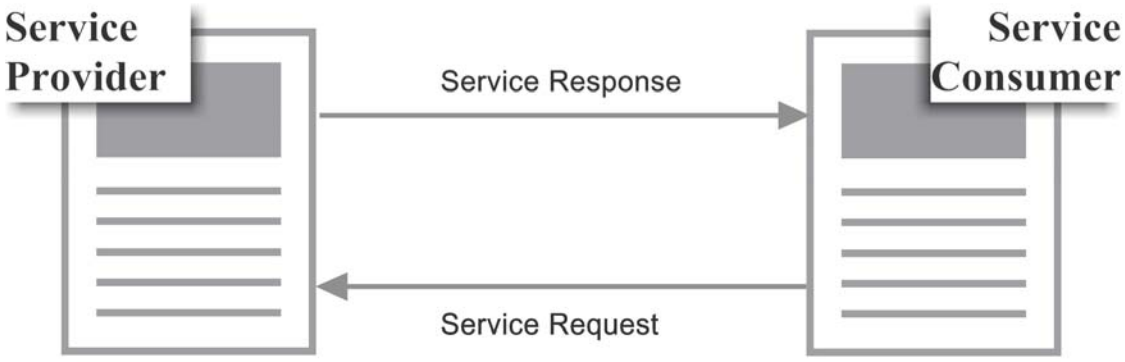

Figure 3-1. SOA Basics

SOA provide the autonomous and platform-independent environment for services to communicate. Services are orchestrated and programmed using standard protocols [15]. Networks of collaborating services distributed within and across organizational boundaries allow for heterogeneous integration in incompatible environments, such as Windows and UNIX. The use of standard protocols for communication requires little or no knowledge of or control over client applications to access the service. Various types of client formats can be supported, such as browser clients, rich desktop clients, spreadsheets, wireless devices, interactive voice response (IVR) systems, and other business applications.

SOC and SOA provide a design paradigm and architecture for implementing and assembling services to support and/or automate business functions. The main idea is to allow organizations to add services or modify the composition of services to provide added-value to evolving business functions. Business logic can be realized from the orchestration order of services and the exchange of data between services. Web services provide the technological foundation for achieving connectivity and interoperability between firms, its partners, and consumers.

3.2. WEB SERVICES

Web services provide the means of connecting services and support interoperable human-to-machine and machine-to-machine interaction on a variety of platforms over a network in a distributed architecture. Web service technologies define a technical framework to implement distributed business processes while a minimum of syntactic consistency is guaranteed.

A brief history of the distributed computing paradigm is provided here. The distributed paradigm initially started with the message-passing paradigm [85] such as Distributed Computing Environment (DCE) [86], Remote Procedure Call (RPC) [87]. Message-oriented middleware [88] such as MQ Series [89], MSMQ [90] was developed to facilitate interoperability within the message-passing paradigm. Message-passing paradigm mimics inter-human communications where network services interact with each other through the exchanges of messages.

However, the abstraction provided by this paradigm does not meet the needs of the complexity of sophisticated network applications. Message passing requires the participating processes to be *tightly-coupled*: throughout their interaction and the processes

must be in direct communication with each other. If communication is lost between processes (due to failures in the communication link, in the systems, or in one of the processes), then collaboration fails. The message-passing paradigm is data-oriented. Each message contains data marshaled in a mutually agreed upon format making the task of interpreting messages a challenge.

Distributed objects and Object Request Brokers (ORBs) [91] evolved from the message-passing paradigm. Some examples are the Common Object Request Broker (COBRA) [91], and the Distributed Component Object Model (DCOM) [92]. The distributed object paradigm provides abstractions beyond those of the message-passing model. In object-oriented programming [93], objects are used to represent an entity significant to an application. Each object encapsulates the state or data of the entity. In Java [94], data is contained in the instance variables of each object. The operations of the entity, can be accessed or updated throgh methods. Distributed objects methods can be invoked by a remote process; a process running on a computer connected via a network to the computer on which the object exists.

Some of the problems with distributed objects include the following: The behavior of objects were encapsulates. This resulted in an ambiguity in the interface definition for dependencies among object in the distributed configuration. Also, programmers were exposed to many relatively low-level details associated with the middleware architecture. A separation of concern was lacking between code related to operation in a middleware framework and code associated with the application. Finally, programmers had to explicitly deal with non-functional concerns related to issues such as security, coordination, and replication. Complexities should be hidden whenever possible.

Component models were developed to deal with the problems associated with distributed computing. [95]. A software component is a unit of composition with contractually specified interfaces and explicit content decencies only. Some exmaples of component models are Enterprise Java Beans (EJB) [96], Component Object Model COM+ [97], .NET Enterprise Services [98], and CORBA Component Model (CCM). Components are specified in terms of a contract. Components include both a set of provided interfaces and required interfaces. Provided interfaces are offered as services to other components. Required interfaces are the dependencies that one component has in terms of other components that must be present and connected to the components. Problems arises when componets are hindered with incompatable interfaces, different archetectural assumptions, and conflicts with other system components. RPC, ORBs and component models, share similar communication models based on synchronous operation invocation while only messaging systems are based on the asynchronous communication model.

Web services introduced several important changes compared to earlier architectures. It is the first technology to fulfill the promise of universal interoperability between applications running on disparate platforms. Support is provided for both asynchronous and synchronous interactions communication models. Connections are *loosely coupled,* a design concept where the internal workings of one service are not "known" to another service, only the external behavior of the service in known. This way, the underlying programming of a service can be modified without disrupting the system. As long as an external behavior has not changed, consumers of that service can continue to function as expected. This is an integral part of the inevitable evolution of interconnectedness that is driving the way we use the Internet.

Web services provide middleware integration between different “islands” inside an enterprise, intra-enterprise, and between different enterprises, inter-enterprise by utilizing standard Internet protocols. As opposed to other middleware technologies, which are mostly intra-enterprise solutions, Web services overcame the challenges of inter-organization operability with the use of standard specification messaging and data transfer protocols such as HTTP [4]. The Web Services Description Language (WSDL) [19]; Universal Description, Discovery, and Integration (UDDI) [99]; and Simple Object Access Protocol (SOAP) [83] formed the initial basis for the original Web service specifications [100]. Figure 3-2 illustrates the use of the WSDL and other specifications above. The steps involved in providing and consuming a service are as follows:

1. A service provider describes its service using WSDL. This definition is published to a registry of services. The registry uses UDDI.
2. A service consumer issues one or more queries to the registry to locate a service and determine how to communicate with the service.
3. Part of the WSDL provided by the service provider is passed to the service consumer. This tells the service consumer what the requests and responses are for the service provider.
4. The service consumer uses the WSDL to send a request to the service provider.

   Finally, the service provider provides the expected response to the service consumer.

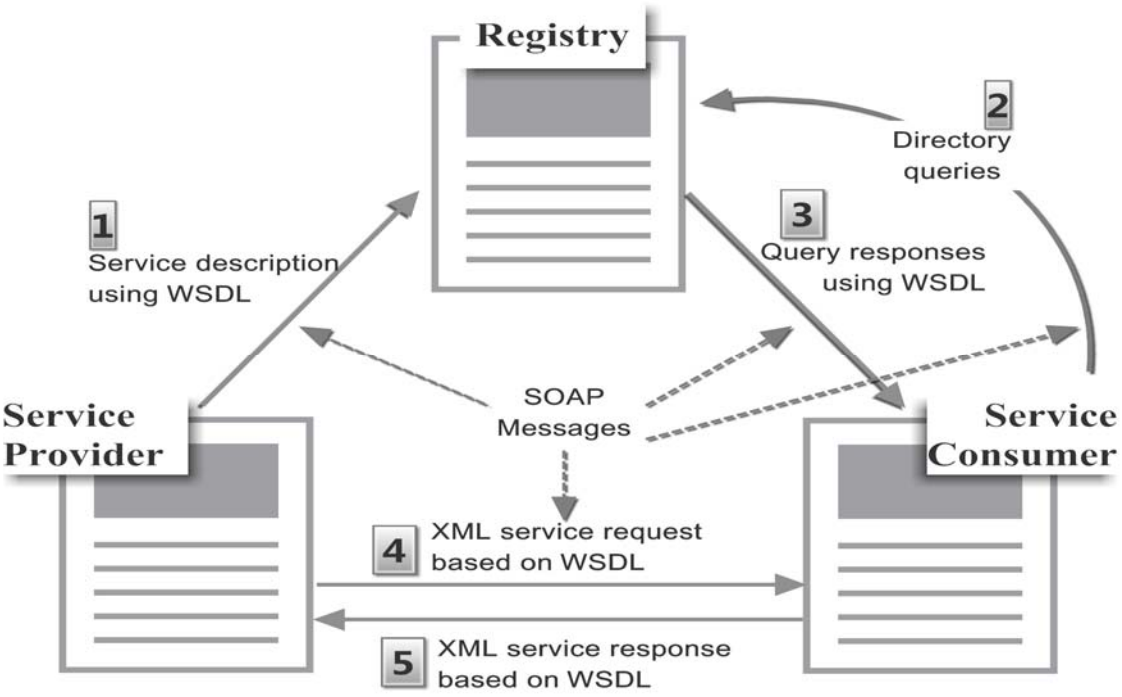

Figure 3-2. Web Service Basics

The UDDI registry was intended to serve as a means of "discovering" Web services described using WSDL. The idea was that the UDDI registry could be accessed and searched in various ways to obtain contact information and the services available from various organization. Coupled with varying performance results from different UDDI implementation [101] and other issues such as doubts about trust and acceptance (the UDDI registry was not maintained by an independent authority but rather private companies) caused UDDI to not be widely implemented. Other specifications to extend and replace UDDI were developed [102].

All messages shown in Figure 3-2 were initially only sent using SOAP, which provides the envelope for sending data. The first alternative to SOAP was the Representational State Transfer (REST) [103] protocol. REST is based on a set of principles that describes how networked resources are defined and addressed. REST appeals to

developers because it has a simpler style that makes it easier to use than SOAP, it is less verbose, and it used in a way that other resources are used on the Internet [84][104]. Both SOAP and REST response message can be implemented using the Extensible Markup Language (XML) [18]. XML was designed to describe data in XML tags and is a software- and hardware-independent tool for carrying information.

The XML-tagged format provides a level of resilience not available with fixed record formats commonly used before the advent of XML. For example, if a service provider adds element not expected by a service consumer, the XML-tagged format allows processing to continue without any problems occurring. Besides XML, JavaScript Object Notation (JSON) [105] is an emerging option to encode messages between service, Table 3-1 shows a comparison between XML and JSON. JSON uses name/value pairs instead of tags used by XML. For example, the name "city" is paired with the values "Orangeburg". This is illustrated on the right side of Table 3-1. The name/value pairs in JSON provide the same type of resilience as the XML-tagged format for data exchange. The name/value pairs do not have to be in any particular order to work. The choice of format for data exchange is dependent on the format service providers decide to use. Service consumers may normally have an "all of the above" implantation to interact with external services that they require.

Table 3-1. Comparison of XML and JSON

XML	**JSON**
…>	{
<name> Alice and Associates </name>	"name." :"Alice and Associates."
<phone> 123-456-7890 </phone>	"phone." :"123-456-7890."
<street> 1 Main Street </street>	"street." :"1 Main Street."
<city> New City </city>	"city." :"New City."
<state> New State </state>	"state." :"New State."
<zip> 98765</zip>	"zip." :"98765."
<country> USA </country>	"country." :"USA."
..>	}

Web services provide the basis for the development and execution of business processes that are distributed over a network and are available via standard interfaces and protocols. In particular, no single Web service can satisfy the functionality required by the user, but rather a combination of existing services work together to fulfill requests. A service composition is an aggregate of services collectively composed to automate a particular task or business process.

3.3. SERVICE COMPOSITION

The realization of domain specific business process logic is accomplished by specific aggregation of services known as service composition [31]. Services are composed in a particular order and follow a set of rules to provide support for business processes. Some recent methods that provide automation to the Web service composition include workflow based [40] [106] and AI planning [41][107][47][108][109] composition methods.

For the former, we argue that the interconnectedness of service is analogous to a *workflow* [110]. The definition of a composite service includes a set of atomic and composite services together with the control and data flow among the services. Similarly, a workflow has to specify the flow of work items. The current achievements on flexible workflow, automatic process adaption and cross-enterprise integration provide the means for automated Web services composition.

Composite services are modeled as business processes, enacted by a service process engine. To qualify as a composition, at least two participating services plus one composition initiator needs to be present. The composition initiator is itself a service, and it orchestrates other services to perform specific business processes. The *orchestration service* is designed in the Business Process Execution Language for Web Service (BPEL4WS) [111], referred

to as just BPEL. BPEL is the process engine, which enables the top down realization of SOA through composition, orchestration, and coordination of Web services. It focuses on representing service compositions where the flow of a process and bindings between services are known a priori.

3.4. SERVICE ORCHESTRATION: BPEL

General adoption of business process automation solutions requires a standard foundation and a specilized language for composing services into business processes. BPEL is such a language and has beocme a standard. BPEL is used to standardize enterprise application integration and extend the integration to previously isolated systems. BPEL is the main technology in environments where functionalities already are or will be exposed via Web services.

IBM, BEA, Microsoft developed the first version of BPEL in 2002. In 2003 SAP and Siebel provided modifications and improvements resulting in the adoption of the first standardized version. BPEL is a convergence of two early workflow languages, WSFL (Web services Flow Language) [112] and XLANG [113]. WSFL, designed by IBM, is based on the concept of directed graphs. XLANG, designed by Microsoft, is a block-structured language. BPEL combines both approaches to provide a directed acyclic graphical and block-structured description of business process.

To an extent, BPEL is similar to traditional programming languages. It offers constructs, such as loops, branches, variables, assignments and fault event handling that allow business process to be define in an algorithmic way.

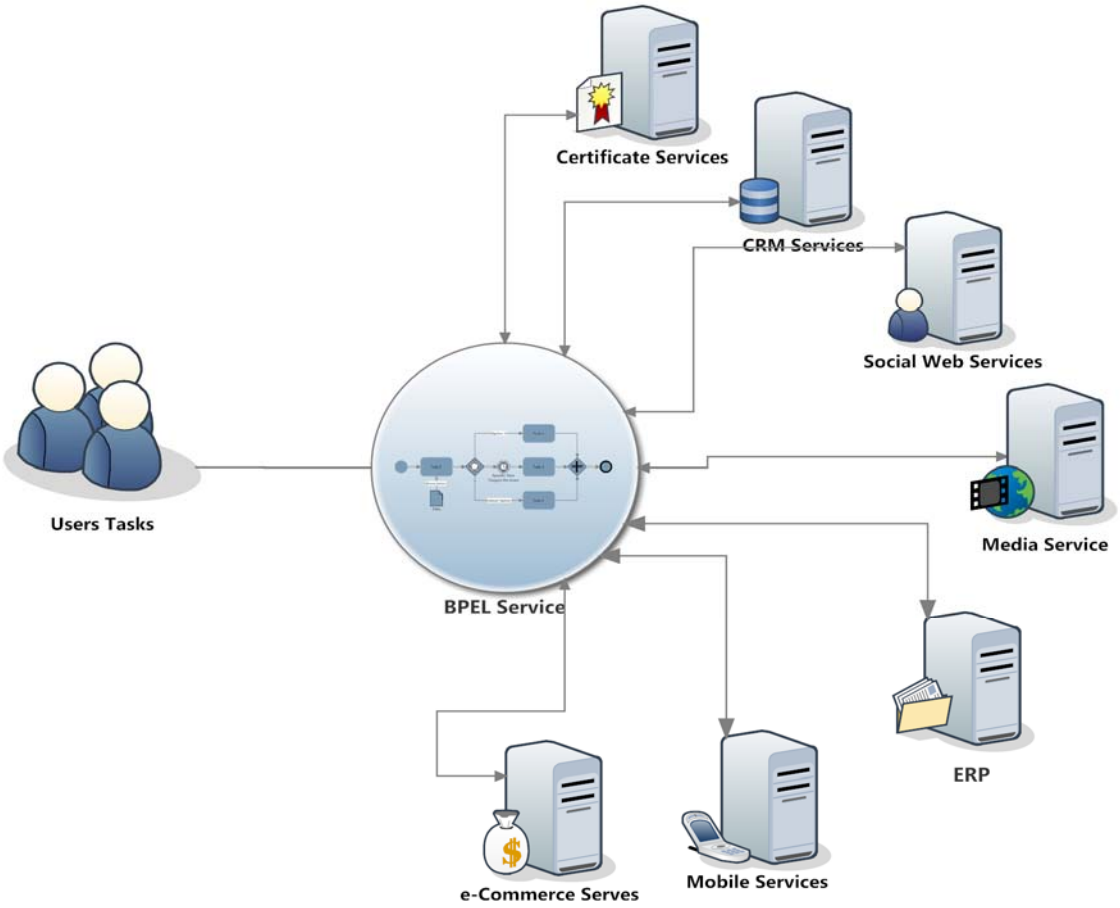

Figure 3-3. BPEL Service Orchestration

On the other hand, it focuses on business process logic making it less complex than traditional languages. The most important constructs are related to the invocation of web services and the data flow between services to realize user tasks. Services can be invoked either synchronously or asynchronously, in a sequence or in parallel. Data can be manipulated in the BPEL service or within any of the invoked services

Orchestration is the process flow of a workflow. S*ervice orchestration* is the implementation of workflows whose invoked applications are implemented as a Web service.

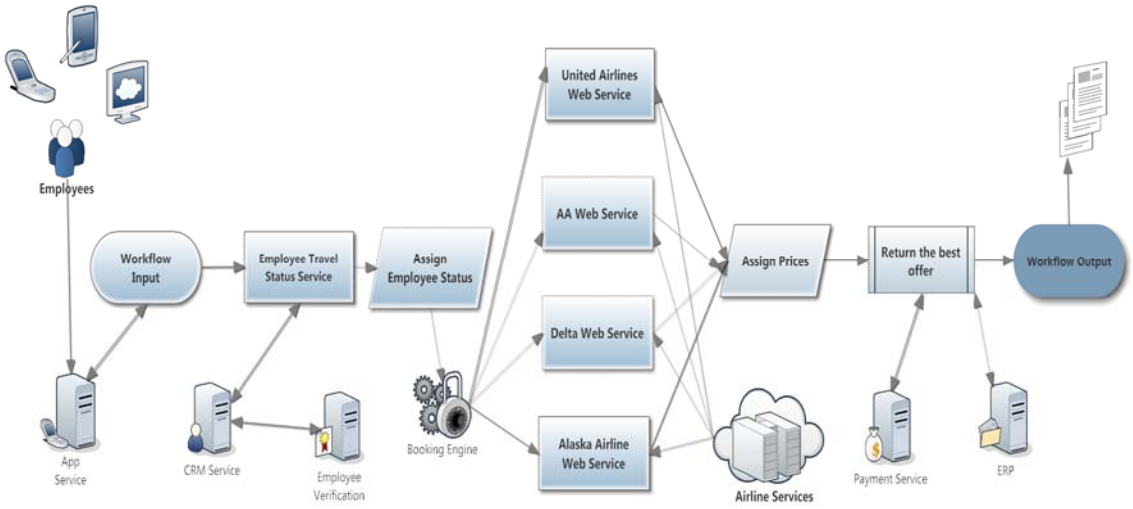

Figure 3-4. Travel Booking Workflow Schema

In this dissertation, we focus on BPEL for creating and orchestrating workflow activites where applications such as e-Commerce, Enterprise resource planning (ERP) and Customer relationship management (CRM) are invoked in different orderings to implement the business logic of user tasks, Figure 3-3.

Figure 3-4, shows an example of workflow schema for a travel booking. Rectangle boxes represent workflow steps realized by BPEL activities. Here a user may use any web-enabled device to access the travel website service. This service provides consolidated air travel services for a specific company employee. Each travel service is provided by an airline service. When a user selects a travel destination, the website service takes information from the user including employee status. The employee status is used to determine what type benefits the user is entitled too, such as first class or economy class seating. Next, requests are sent to various airline services on the specific destination and accompanying benefits. The airline services then return available flight information to the travel website service. The travel website service then sort flight information according to user constrains and returns the best offer.

The process flow of a BPEL process is based on *activities.* BPEL distinguishes between *basic activities* and *structured activities*. Basic activities are used to implement the behavior and data flow of workflow activities such as *receive*, *invoke*, *reply* for message exchange and *assign* for data manipulation. Variables can be used as data containers for messages or other XML-based content. Structured activities are used to create a block-based control flow structure for a BPEL process such as *sequence* for sequential, *repeat until* for iterative, and *flow* for concurrent execution. They can recursively contain other basic and structured activities. An excerpt of a BPEL process with a *flow* element as a root structured activity is shown in Listing 3-1

The structured activity *flow* provides the definition of an acyclic graph-based control flow, Figure 3-5. All basic and structured activities contained in a *flow* are regarded as concurrent. This concurrency can be synchronized by adding a control flow dependency as a *link* between activities. Each *link* has exactly one target and one source activity. An activity may have multiple ingoing and outgoing *link* elements. The creation of loops with *link* elements is not permissible. Each activity in a *flow* can define a Boolean condition for all ingoing *link* elements as the *joinCondition* and each outgoing *link* as the *transitionCondition*. A *transitionCondition* sets an outgoing *link*-state to either *true* or *false*.

A *joinCondition* evaluates whether an activity is executed (true) or not (false) based on all incoming *link* states. The behavior of a *false joinCondition* can be further controlled with the Boolean attribute *suppressJoinFailure*, which can be defined globally for the flow activity and locally for each contained activity. If it is set to *false*, a join failure is created.

Listing 3-1. Flow activity with links and transition conditions.

```
<flow ...>
    <links>
        <link name="request-to-approve" />
        <link name="request-to-decline" />
    </links>
    <receive name="ReceiveCreditRequest"
        createInstance="yes"
        partnerLink="creditRequestPLT"
        operation="creditRequest"
        variable="creditVariable">
        <sources>
            <source linkName="request-to-approve">
                <transitionCondition>
                    $creditVariable/value &lt; 5000
                </transitionCondition>
            </source>
            <source linkName="request-to-decline">
                <transitionCondition>
                    $creditVariable/value >= 5000
                </transitionCondition>
            </source>
        </sources>
    </receive>
    <invoke name="approveCredit" ...>
        <targets>
            <target linkName="request-to-approve" />
        </targets>
    </invoke>
    <invoke name="declineCredit" ...>
        <targets>
            <target linkName="request-to-decline" />
        </targets>
    </invoke>
</flow>
```

If it is set *true*, the activity is not executed, and all outgoing *link* elements are set to false. The *false* link state is further propogated until an activity throws a join failure, or a *joinCOndition* is reached that evaluates to *true.* This mechanism is called *Dead-Path-Elimination* (DPE). It ensures that dead paths with *false link* elements are eliminated. Figure 3-5 shows a more detailed view of the aforementioned travel example.

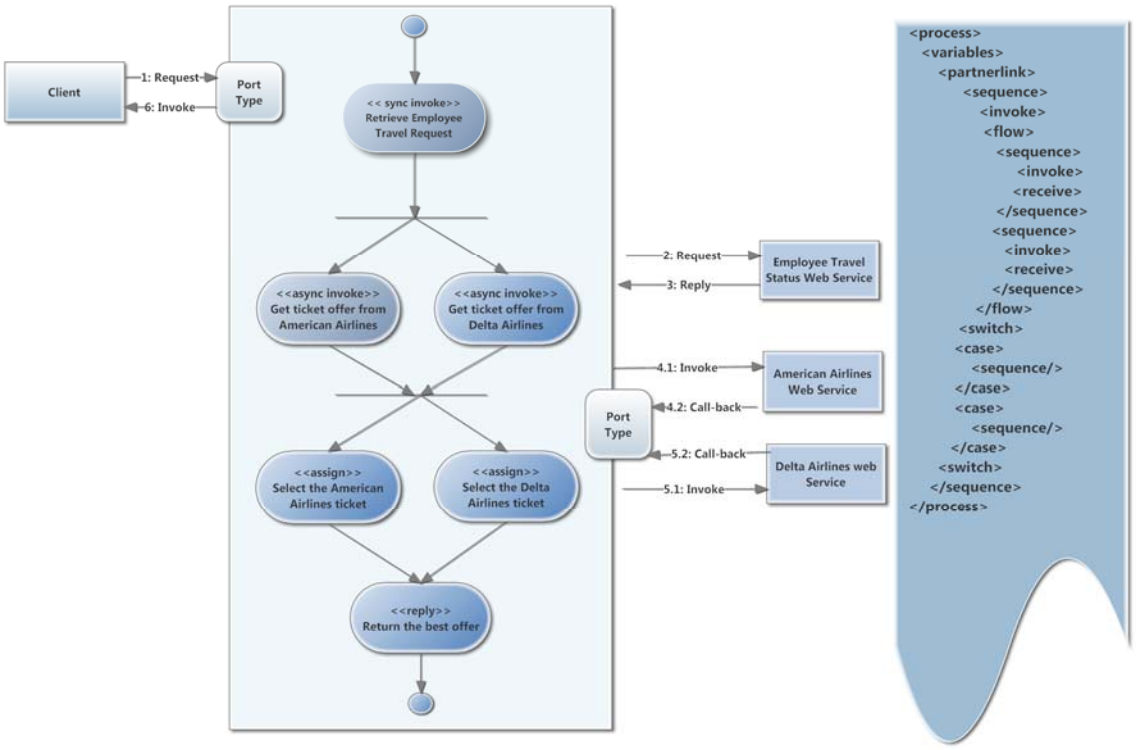

Figure 3-5. Travel Example

The message exchange between Web services can be described as a WSDL extension called *partner link type*. A partner link type defines roles for all participating Web services based on the WSDL port types from the respective WSDL interfaces. Each role is associated with one WSDL port type. Partner link types are instantiated by *partner links* within a BPEL process. A partner link references one or two roles from the corresponding partner link type via the attributes *partnerRole* and *myRole*. A *partnerROLE* indicates that the associated WSDL port type is used to send messages (e.g. via *invoke* and *reply*) and must be provided by the WSDL interface of an external service. A *myRole* indicates that the associated WSDL port type is used to receive messages (e.g. via *receive*) and must be provided by the WSDL interface of the BPEL process. The WS-Addressing [114] endpoint for an invoked Web service is also assigned to the partner link. It is possible to modify this

endpoint with the *assign* activity during process execution, which allows the dynamic invocation of Web Services.

A BPEL process may be executed in different process instances. To route an incoming message to the correct process instance, BPEL provides a *correlation* mechanism. It basically utilizes unique identifiers in the content of exchanged message, for example, a customer number. *Properties* can be defined with a corresponding WSDL extension to reference values in WSDL messages. One or more properties can be used to define a *correlation set* with a BPEL process. Each correlation set has a unique name so that several correlation sets can be defines for a BPEL process. All properties of one correlation set must represent a unique identifier. The values for these properties are retrieved from the message that is currently being received or sent by a corresponding activity. A correlation set must be utilized once by message exchange activity to associate the unique identifier with the process instance. Initialized correlation set are immutable and can be used for message routing by all activities that receive the message. Therefore, a correlation set is determined for an incoming message and compared with the initialized correlation set of each process instance. The message is routed to the process instance with the matching initialized correlation set.

A BPEL process is deployed and executed by a compliant process engine such as Apache ODE (Orchestration Director Engine) [115]. Each BPEL process engine provides a vendor-specific deployment descriptor that usually contains binding information for all partner links.

BPEL provides a language extension mechanism, e.g., to define custom activities or to extend existing activities by adding additional information. Extensions are identified by

a namespace and specify an additional Boolean attribute *mustUndersand* to indicate whether it is essential for workflow execution and, therefore, must be supported by a process engine or may be ignored. Extensions can further be classified as *design time only extension, design, and runtime extension,* and *runtime only extension* [116].

3.5. WEB SCALE WEB SERVICE WORKFLOWS

A Workflow is the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules [117]. Workflows have their origin in the early 90's and were initially used to support and coordinate mainly human-centric *business processes* with information technology (IT). Business processes are a set of one or more linked procedures or *activities*, which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships [117]. An activity is a description of a piece of work that forms one logical step within a process. An activity may be a manual activity, which does not support computer automation, or a workflow (automated) activity. A workflow activity may require both human and/or machine resources to support process execution. An activity, which is capable of computer automation, uses a workflow management system to manage the activity during execution of the business process.

Workflows provide a means to bridge the gap between the IT and the business domain. Today, workflows play a central role in IT infrastructures of enterprises, especially in the context of SOAs. Web service technology executed in SOAs are business processes and are often completely automated workflows. This dissertation focuses on SOAs as the execution infrastructure for workflows.

With SOC emerged the idea of service composition [31]; the creation of complex functionalities by integrating loosely-coupled services that form Web service workflows (WSW) [118]. Web service workflows are an ordered set of sequential and concurrent messages among services producing realizing domain-specific functions. As such, services can exist both inside and outside an enterprise creating Web-scale workflows [16], Figure 3-6. A variety of services in step 1 are combined into an inter-organizational process in step 2. Business intelligence is realized as modular infrastructure in step 3.

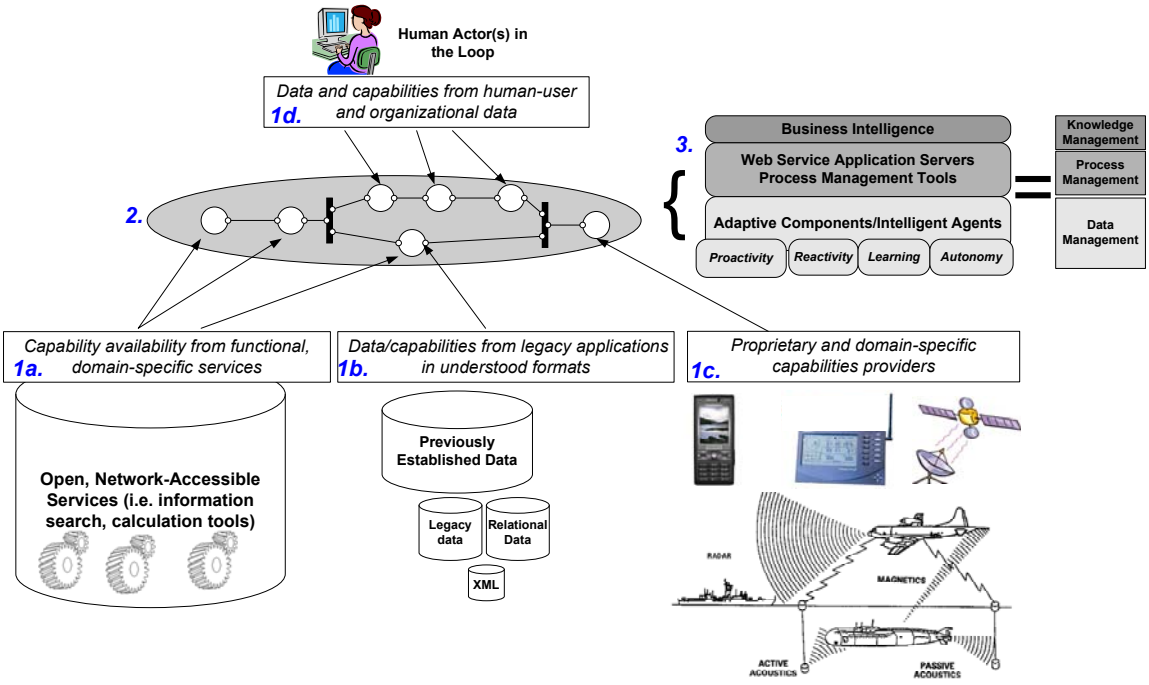

Figure 3-6: Web-scale workflow (adapted from [16])

Major challenges for WSW environments is the ability to design services capabilities into higher-level inter-organizational processes [16]. Also, as environments changes, mechanism in WSW are unable to adapt and detect designed induced error rendering them both nonreactive and non-proactive to learn to act autonomously. Moreover, WSW produces a deluge of data, which creates a daunting challenge to bring all of the available data together in a form that is manageable for making accurate decisions promptly.

The result is an information gap where more data does not necessarily increase our ability to process symbiotic pieces of data to arrive at the needed information. An interface is needed that allow for the efficient management of data to gain high-level understanding at runtime. Moreover, workflows need to be responsive to time-dependent and business specific trends at runtime.

Responsiveness can be stymied by the creation of the data flow, i.e. the message assignments between the activities, which can be complex and might require the user to have extensive knowledge about the underlying type representations. Currently, full workflow automation is limited to narrow and formally well-defined application domains [7]. An impetus for an agile WSWs is the fostering of collaboration and orchestrating processes among business partners.

Chapter 4
THEORETICAL EXPLANATIONS

To formally represent workflow semantics we use a Probabilistic Graphical Model (PGM) [119]. PGMs defines a family of probability distributions that represented its attributes in terms of a graph. Nodes in the graph correspond to random variables. Edges represent its structure, which translates into conditional dependencies (among variables) that drive the computation of joint, conditional, and marginal probabilities of interest [119]. The particular type of PGM used is a *Bayesian Network* [120]. Bayesian Networks rely on the basic insight that independence form a significant aspect of belief and that it can be exploited easily using the language of graphs.

This chapter introduces some of the background theory behind graphical modeling and Bayesian networks. First, we describe the properties of the data we are using in the model, and then the basic concepts of graphs, independence, and conditional independence. Then an explanation of how conditional independence structures can be represented graphically is given. Finally, Bayesian networks is discussed with a working example demonstrates the modeling approach.

4.1. DATA PROPERTIES

This dissertation is only concerned with the analysis of datasets on the form

$$\begin{array}{cccc} v^{(1)} & w^{(1)} & \dots & z^{(1)} \\ v^{(2)} & w^{(2)} & \dots & z^{(2)} \\ \vdots & \vdots & \vdots & \vdots \end{array}$$

$$v^{(n)} \qquad w^{(n)} \quad \dots \qquad z^{(n)}$$

that is, consisting of a set of workflow measurements, $V, W, \dots, Z$,taken on N data entities. The measurements are called *variables*, and the entities represent observations, for example, on different objects or at different times. A variable represents an exhaustive set of mutually exclusive events, referred to as the domain of the variable. These events are also often called states, levels, values, choices, options, etc. The domain of a variable can be discrete or continuous; discrete domains are always finite and continuous infinite.

Throughout the dissertation capital letters (possibly indexed) denote variables or sets of variables and lower case letters (possibly indexed) to denote particular values of variables. Thus, $X = x$ may either denote the fact that variable X attains the value x or the fact that the set of variables $X = X_1, \dots X_n$ attains the (vector) of values $X = X_1, \dots X_n$. By $dom(X) = X_1, \dots X_{||x||}$ we denote the domain of X, where $\big||X|\big| = |dom(x)|$is the number of possible distinct values of X. If $X = X_1, \dots X_n$, then $dom(X)$ is the Cartesian product (or product space) over the domains of the variables in X. Formally, $dom(X) = dom(X_1)\ x, \dots x\ (domX_n)$. For example, assume that dom(X) = {F,T} and dom(Y) = {red, green, blue}. Then dom(X, Y) = {(F, red), (F, green), (F, blue), (T, red), (T, green), (T, blue)}. For z = (F, blue) we get zX = F and zY = blue.

A fundamental assumption of this dissertation is that the data generated by Web Service Workflows are stochastic in nature; namely, workflow measurements $V, W, \dots, Z$ are random variable. A random variable is a variable whose value depends on the outcome of a probabilistic experiment. Its value is a priori unknown, but it becomes known once a workflow activity outcome is realized. We distinguish between two types of variables: continuous variables, whose values lie in the real line R, and discrete variables (also called

factors), which can take values from a finite set. For convenience, the label of the values in this finite set are $\{1, 2, \ldots, \#X\}$, where $\#X$ is the number of levels of X. When X is a random variable we write its density or mass function as $f_X(x)$. If X is discrete, we write this as $P(X = j)$, for level $j \in \{1, 2, \ldots, \#X\}$.

In addition to the assumption that workflow measurements $V, W, \ldots, Z$ are random variables, they are also random variables with a joint probability distribution function $f_\theta(v, w, \ldots, z)$, where θ is some unknown parameter. We conceive of the data as resulting from workflow invocations in which the N observations are sampled independently from f_θ. We base our inference on statements about the unknown parameter θ. Usually this inference rests more or less explicitly on historical and ongoing data produced by the workflow. We define another category of variables called *chance variables* or *decision variables* representing random events and variables representing choices under the control of some, typically human, agent.

Particular workflow cases are simulated by making certain variables fixed, so that their observed values become, as it were, a constant part of the workflow activity. We regard the remaining variables as random variables drawn from a conditional density given the fixed variables. Thus, in any hypothetical replications, the fixed values are held constant.

4.2. GRAPHS

A graph may be defined as a pair $G = (V, E)$, where V is a set of *vertices* or *nodes* and E is a set of *edges* [121]. Each edge is associated with a pair of nodes, its *endpoints*. Edges may in general be *undirected*, *bi-directed* or *directed*.

In the context of this dissertation, nodes correspond to data variables in the BPEL activities and the edges to data dependencies between activities, we focus only on directed edges.

A subset $A \subseteq V$ is *complete* if an edge connects all vertex pairs in A. For example, in Figure 4-1, the graph is complete. A graph G is complete if the vertex A is complete. A *clique* is a maximal complete subset, a complete subset that is not contained in a larger complete subset. The set of cliques of a graph G is denoted by $C(G)$.

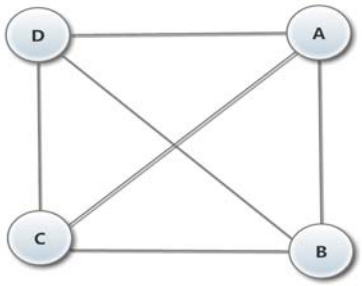

Figure 4-1. Complete graph

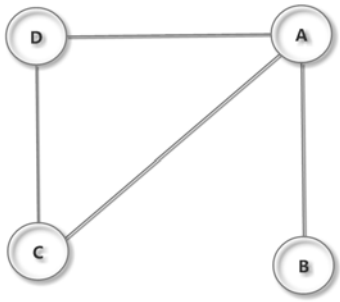

Figure 4-2. Incomplete graph

Two vertices α and β in G are said to be *adjacent*, written $\alpha \sim \beta$, if there is an edge between α and β in *G*. In Figure. 4-2, B, A, D, C is path of length 3 between B and C. If a

path $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_n = \beta$ has $\alpha = \beta$ then the path is a *cycle* of length n. A graph is said to be connected if there is a path between every pair of vertices. A path $X_1, X_2, \ldots X_n, X_1$ is called an *n-cycle*, or a cycle of length n. For example, in Figure 4-1, A, B, D, C, A is a 4-cycle. If the n vertices $X_1, X_2, \ldots X_n$ of an n-cycle $X_1, X_2, \ldots X_n, X_1$ are distinct, and if X_j $adjacent$ X_k only if |j-k| = 1 or n -1, then we call it a *chordless cycle*. A graph *triangulated* if it has no chordless cycles of length greater than or equal to four. The triangulated property turns out to be closely related to the existence of closed-form *maximum likelihood estimates*, discussed later.

A subset $D \subset V$ *separates* $A \subset V$ from $B \subset V$ if every path between a vertex in A and a vertex in B contains a vertex from D. The graph $G_0 = (V_0, E_0)$ is a *subgraph* of $G = (V, G)$ if $V_0 \subseteq V$ and $E_0 \subseteq E$. For $A \subseteq V$, let E_A denote the set of edges in E between vertices in A. Then $G_A = (A, E_A)$ is the *subgraph induced by* A. In Fig. 3-2, the subset {A, C, D} is complete. The *boundary* $bd(\alpha) = adj(\alpha)$, is the set of vertices adjacent to α. Two vertices are adjacent if there is an edge between them. The boundary is equal to the set of *neighbors* $ne(\alpha)$. The *closure* $cl(\alpha)$ is $bd(\alpha) \cup \{\alpha\}$.

In a directed graph E is a set of directed edges and the edges are represented as arrows. A directed graph is *acyclic* if it has no directed cycles. A DAG is a directed graph that is acyclic. A *path* (of length) from α to β in a directed graph is a sequence of vertices $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_n = \beta$ such that $\alpha_{i-1} \rightarrow \alpha_i$ is an edge in the graph. If there is a path from α to β we write $\alpha \longmapsto \beta$. The *parents* $pa(\beta)$ of a node β are those nodes α for which $\alpha \rightarrow \beta$. The descendants $des(\alpha)$ of a node α are those nodes β for which $\alpha \rightarrow \beta$. The *non-descendants* $non\_des(\alpha)$ node α are those nodes other than α, $pa(\alpha)$ and $des(\alpha)$ The *ancestral set* $an(A)$ of a set A is the union of A with its ancestors. The *ancestral graph* of a

set A is the subgraph induced by the ancestral set of A is the subgraph induced by the ancestral set of A. *Moralization* is the adding of edges between the parents of each node and then replacing all directed edges with undirected ones, thereby returning an undirected graph. The moralized DAG is transformed into a triangulated graph. A graph is *triangulated* if and only if every cycle of length four or greater contains an edge between two nonadjacent nodes in the cycle.

In the BPEL language, activities perform process logic. Activities can be the source or target of a process. A source activity is equivalent to a node with an outgoing edge in graph G. A target activity is equivalent to an incoming edge to of a node in graph G. Combinations of source and target activities represent a directed graphs. The BPEL standard ensures that a source activity cannot have a target activity as a logically preceding activity. This implies that directed graphs formed by source and target activities are always acyclic [15], hence we use DAG to represent data dependencies in BPEL process logic. For example, a price calculation for a product can only be started immediately after a request for the product is received, or a shipping price can only be added to an invoice after the shipper information has been obtained.

4.3. INDEPENDENCE AND CONDITIONAL INDEPENDENCE

As *statistical objects,* graphical nodes represent variables (and sometimes parameters) in such a way that the independence structure can be read directly off the graph. Two events, A and B are *independent* if the probability of A and B is the product of their respective probabilities, written as $P(A,B) = P(A)P(B)$. This is automatic but uninteresting if $P(A) = 0\ or\ 1$, likewise for B. If $0 < P(A)\ and\ P(B)$, then an equivalent statement for

independence is $P(A|B) = P(A)$, that is, the probability of A given B is the probability of A, alternatively $P(B|A) = P(B)$ [122]. Independence means that the occurrence or lack of A, alternatively B, does not influence the occurrence or lack of it of the other.

From the context of graphs, *conditional independence* is represented in the following way. Let $(X_v)_{v\in A}$ be a collection of random variables with a joint density. Let A, B and C be subsets of V and let $X_A = (X_v)_{v\in A}$ and similarly for X_B and X_c. Then the statement that X_A and X_B are conditionally independent given X_C, written $A \perp B|C$, means that for each possible value of x_C of X_C, X_A and X_B are independent in the conditional distribution given $X_C = x_c$. Otherwise, if C is not observed, then A and B are potentially dependent.

Let $f()$ be a generic density or probability mass function, then one characterization of $A \perp B|C$ is $f(x_A, x_B|x_C) = f(x_A|x_C)\, f(x_B|x_C)$. An equivalent characterization is the joint density of (X_A, X_B, X_C) factorizes as $f(x_A, x_B|x_C) = g(x_A, x_C)\, h(x_{B,}x_C)$ [123]. The two functions $g()$ and $h()$ does not depend on x_B and x_A respectively. This is known as the *factorization criterion*.

Conditional independence in undirected graphs is represented in the following way: let $G = (V, E)$ be an undirected graph with cliques $C_1, \dots, C_k$ and $f()$ be a joint density of the variables in V. If this admits a factorization of the form $f(x_V) = \prod_{i=1}^{k} g_i(x_{Ci})$ for some functions $g_1() \dots g_k()$ where $g_j()$ depends on x only through x_{Cj} then we say the $f()$ factorizes according to G. If all the densities under a model factorizes according to G, the G encodes the conditional independence structure of the model through the *global Markov property* [124].

Patterns of conditional independence occurs in DAGs if the joint density factorizes as follows $f(x_V = \prod_{v \in V} f(x_v | x_{pa(v)}$ for some variable sets $\{pa(v)\}_{v \in V}$ such that the variables in $pa(v)$ precede v in the ordering [125]. In DAGs, conditional independence is represented by a property called *d-separation* [120]. The notion of d-separation can be defined in various ways, but one characterization is as follow: A and B are d-separated by a set C if and only if they are separated in the graph formed by moralizing the anterior graph of $A \cup B \cup C$ [126]. This dissertation uses DAGs to model Web service workflows.

In BPEL, the <*flow*> activity provides concurrency and synchronization. A <*flow*> activity creates a set of concurrent activities directly nested within it. It enables synchronization dependencies between activities that are nested within it to any depth. The <*link*> construct is used only to express synchronization dependencies. In the following example, the two <*invoke*> activities are enabled to start concurrently when the <*flow*> starts. Assuming the <*invoke*> operations are request-response operations, the completion of the <*flow*> occurs after both the seller and the shipper respond. The "transferMoney" activity is executed after the <*flow*> completes.

```
<sequence>
   <flow>
      <invoke partnerLink="Seller" ... />
      <invoke partnerLink="Shipper" ... />
   </flow>
   <invoke partnerLink="Bank" name="transferMoney" ... />
</sequence>
```

To represent the BPEL snippet in a graphical and conditional perspective, "Seller", "Shipper" and "transferMoney" are variables and are represented as nodes in a graph. "Seller" and "Shipper" are parents of "transferMoney" and is conditional independent of "transferMoney", Figure 4-3. Additional, there is no edge connecting "Seller" and

"Shipper". Hence, we can say "Seller" and "Shipper" are conditional independent given "transferMoney" written as: Seller $\cap Shipper \mid tranferMoney$

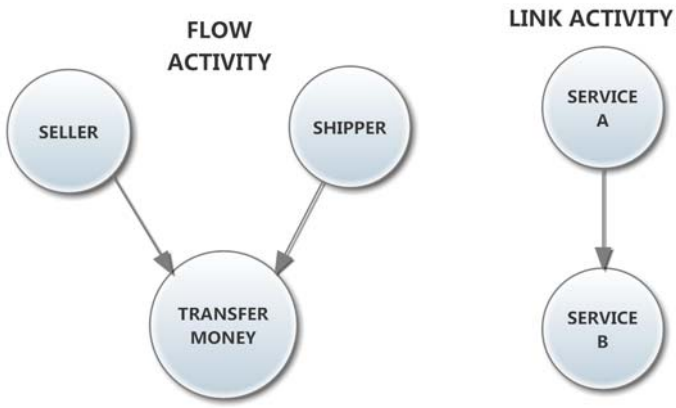

Figure 4-3. BPEL Concurrent and Sequential Activities.

4.4. PROBABILISTIC NETWORKS

Probabilistic networks are graphical models of (causal) interactions among a set of variables. Nodes represent variables or parameters and the interactions (direct dependencies) as directed links (also known as arcs and edges) between the nodes. Any pair of unconnected/nonadjacent nodes of such a graph indicates (conditional) independence between the variables represented by these nodes under particular circumstances. Hence, probabilistic networks capture a set of (conditional) dependence and independence properties associated with the variables represented in the network. A domain of random variables can, for instance, form the basis of a decision support system to help decision makers identify the most beneficial decision in a given situation [127]. Two branches of graphical representations of distributions are commonly used, namely, *Bayesian networks* and *Markov networks*. Both families encompass the properties of factorization and independences, but they differ on the set of independences they can encode and the

factorization of the distribution that they induce [119]. We use Bayesian Network branch of probabilistic networks in this dissertation to model BPEL processes.

The graphical or structural aspect of a probabilistic network is referred to as it a *qualitative* component, and the probabilistic, numerical part or calculus of uncertainty as its *quantitative* component. The qualitative component encodes a set of (conditional) dependence and independence statements among a set of random variables, informational precedence, and preference relations. They are visually encoded using a graphical language. The quantitative component specifies the strengths of dependence relations using probability theory and preference relations using probability theory.

Since a probabilistic network consists of two components, it is customary to consider its constructions as a two-phase process. As the first step, the qualitative structure of the model is constructed using a graphical language. This step consists of identifying variables and relations between variables. As the second step, the values of probabilistic values are computed. Probabilistic Graphical modeling has two main advantages:

1. It allows for a multivariate approach by including all relevant variables in the analysis to study the conditional as well as the marginal associations.
2. Markov properties of pairwise, local and global [125][128][129], allows for a simple translation from a graph-theoretic property and separation perspective to a statistical property of conditional independence.

The key inference task with a probabilistic network is computation of posterior probabilities of the form $P(X|e)$, where, in general e is evidence (i.e., information) received from external sources about the (possible) states/values of a subset of the variables of the network. The ability to perform inter-causal inference is unique for graphical models and is

one of the key differences between automatic reasoning systems based on probabilistic networks and systems based on, for example, production rules. In a rule-based system, we would need dedicated rules for taking care of inter-causal reasoning.

In our approach, we manually create the qualitative component of the probabilistic network from BPEL activities. Activities are realized as variables and are represented as nodes in a network. To create edges between the nodes, we distinguish between source and target activities and determine their sequential or concurrent dependencies. For example, we demonstrate the creation of the qualitative component from the following BPEL snippet:

```
<flow suppressJoinFailure="yes">

<links>
   <link name="buyToSettle" />
   <link name="sellToSettle" />
   <link name="toBuyConfirm" />
   <link name="toSellConfirm" />
</links>
<receive name="receiveBuyerInformation" ...>
   <sources>
      <source linkName="buyToSettle" />
   </sources>
</receive>
<receive name="receiveSellerInformation" ...>
   <sources>
      <source linkName="sellToSettle" />
   </sources>
</receive>
<invoke name="settleTrade" ...>
   <targets>
      <joinCondition>$buyToSettle and $sellToSettle</joinCondition>
      <target linkName="buyToSettle" />
      <target linkName="sellToSettle" />
   </targets>
   <sources>
      <source linkName="toBuyConfirm" />
      <source linkName="toSellConfirm" />
   </sources>
</invoke>
<reply name="confirmBuyer" ...>
   <targets>
      <target linkName="toBuyConfirm" />
   </targets>
</reply>
<reply name="confirmSeller" ...>
   <targets>
      <target linkName="toSellConfirm" />
```

```
      </targets>
   </reply>
</flow>
```

The <*flow*> activity is used to specify one or more activities to be performed concurrently. <*links*> can be used within a <flow> to define explicit control dependencies between nested child activities. The previous <*link*>'s are defined as:

- "buyToSettle" starts at "receiveBuyerInformation" (specified in the corresponding <*source*> element nested in "receiveBuyerInformation") and ends at "settleTrade" (specified in the corresponding <*target*> element nested in "settleTrade").
- "sellToSettle" starts at "receiveSellerInformation" and ends at "settleTrade".
- "toBuyConfirm" starts at "settleTrade" and ends at "confirmBuyer."
- "toSellConfirm" starts at "settleTrade" and ends at "confirmSeller."

The corresponding qualitative component defined in the <*flow*> activities cause "receiveBuyerInformation" and receiveSellerInformation to run concurrently. The settleTrade activity is performed only after both of these activities are completed. After settleTrade completes the two activities, confirmBuyer and confirmSeller are performed concurrently again.

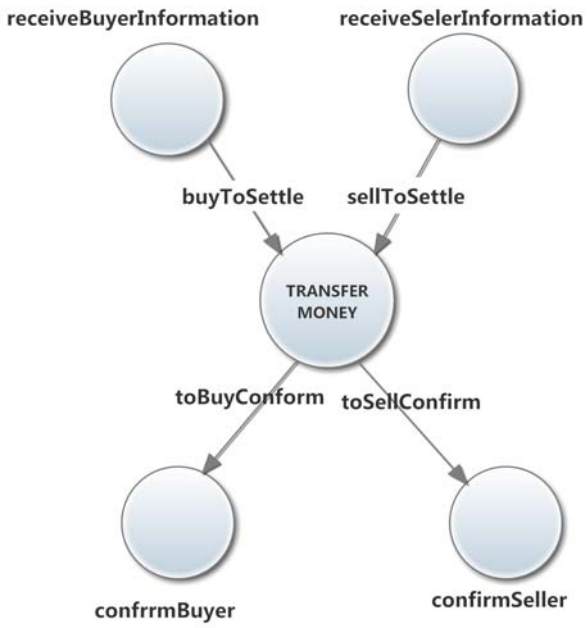

Figure 4-4. Quality Component for BPEL Snippet.

4.5. PROBABILISTIC CALCULUS

In this section, we provide the framework for representing and reasoning with uncertainty in a BPEL process. Each BPEL activity that represents a node is assigned a degree of belief which is interpreted as a probability that quantifies the belief in that event. The focus of this section is to describe the semantics of degrees of belief, its properties and the methods used for revising them in light of new evidence.

Degrees of Belief. We assign a *degree of belief* or *probability* in [0, 1] to each world ω and denote it by $P(\omega)$. A *world* is defined as an instantiation of a state in the statespace of each BPEL activity cause by a workflow invocation. The belief in, or probability of, a BPEL activity α is defined as:

$$P(\alpha) \triangleq \sum_{\omega \vDash \alpha} P(\omega), \qquad \textit{(EQ 4-1)}$$

which is the sum of probabilities assigned to worlds to which α is true. As an example, table 4.1, lists a set of worlds and their corresponding degrees of belief. Each world represents a Web service workflow invocation and its set of BPEL activities. This is known as the *state of belief* or the *joint probability distribution* of the probability network or Web Service workflow. As a demonstrated example we consider a situation where a solution to a problem is being derived by Human Computing Services (HCS) and Machine Computing Services (MCS). A value of true or false is returned in the event that either an HCS or MCS thinks they or it have determined a solution. Likewise, a value of true or false is returned if the problem has been solved. Based on Table 4.1, we have the following initial or *prior beliefs*:

$$\mathrm{P}(HCS) = P(\omega_1) + P(\omega_2) + P(\omega_3) + P(\omega_4) = .1$$

$$\Pr(MCS) = .2$$

$$\mathrm{P}(Solution) = .2442$$

$$\sum_{\omega} P \stackrel{\text{def}}{=} (\omega) = 1$$

Table 4-1. A State of Belief, also known as a Joint Distribution.

World	HCS Activities	MCS Activities	Solution	P(.)
ω_1	true	true	true	0.0190
ω_2	true	true	false	0.0010
ω_3	true	false	true	0.0560
ω_4	true	false	false	0.0240
ω_5	false	true	true	0.1620
ω_6	false	true	false	0.0180
ω_7	false	false	true	0.0072
ω_8	false	false	false	0.7128

Properties of Belief.

- $0 \leq P(\alpha) \leq 1$ for any event α
- $P(\alpha) = 0$ when α is inconsistent
- $P(\alpha) = 1$ when α is valid
- $P(\alpha) + P(\neg\alpha) = 1$ The complement of a belief could be derived from the belief
- $P(\alpha) \vee P(\beta) = P(\alpha) + P(\beta) - P(\alpha \wedge \beta)$ Belief can be computed in disjunction
- $P(\alpha) \vee P(\beta) = P(\alpha) + P(\beta)$ If α and β are mutually exclusive

Quantify uncertainty. Uncertainty about a variable X is formally quantified using the notion of *entropy* measure [130][131]:

$$ENT(X) \stackrel{\text{def}}{=} -\sum_{x} P(x) \log_2 x \qquad \textit{(EQ 4-2)}$$

Intuitively, beliefs in Table 4-1 may seem more certain about HCE events as opposed to solutions events. However, the entropies associated with these events show that we are more certain about the solutions events as opposed to HCE events.

Table 4-2. Entropy vs. Probability

	$P(HCS)$	$P(MCS)$	$P(Solution)$
true	0.1	.2	0.2442
false	0.9	0.8	0.7558
ENT(.)	**0.469**	**0.722**	**0.802**

Updating Beliefs. Beliefs are updated to accommodate new information, which we refer to as *evidence*. More generally, evidence will be represented by an arbitrary event, for example, β. Given evidence β, current belief $P(\omega)$ will be updated into a new state of belief denoted by $P(\omega|\beta)$ and hence every ω the satisfies $\neg\beta$ is assigned the belief 0 $P(\omega|\beta) = 1$. New beliefs with a positive probability in normalized by the old beliefs, with

the normalizing constant being our old belief in the evidence, $P(\beta)$. New states of beliefs are define as:

$$P(\omega|\beta) \stackrel{\text{def}}{=} \begin{Bmatrix} 0, & if\ \omega \vDash \neg\beta \\ \frac{P(\omega)}{P(\beta)} & if\ \omega \vDash \beta \end{Bmatrix} \qquad \textit{(EQ 4-3)}$$

The new state of belief $P(\omega|\beta)$ is referred to as the result of *conditioning* the old state on evidence β. Table 4.3 shows the updated beliefs given evidence "Solution = true". Some of the changes induced by this new evidence are; Prior belief in $P(MCS) = 0.2$ moved to $P(HCS|Soltion) = 0.741$, prior belief in $P(HCS) = 0.1$ moved to $P(HCS|Soltion) = 0.307$. To avoid having to explicitly computing belief for every ω, the closed form for updating belief, *Bayes Condition* [120], is used:

$$P(\alpha|\beta) = \frac{P(\alpha \wedge \beta)}{P(\beta)} \qquad \textit{(EQ 4-4)}$$

Bayes conditioning has the same commitments as belief updating:

1. Worlds that contradict the evidence β will have zero probability
2. Worlds that have zero probability will continue to have zero probability
3. Worlds that are consistent with evidence β and have positive probability will maintain their relative beliefs.

Table 4-3. A State of Belief and the Result of Conditioning it on Evidence Solution

World	HCS	MCS	Solution	P(.)	P(.|*Solution*)
ω_1	true	true	true	0.0190	0.0190/0.2442
ω_2	true	true	false	0.0010	0
ω_3	true	false	true	0.0560	0.0560/0.2442
ω_4	true	false	false	0.0240	0
ω_5	false	true	true	0.1620	0.1620/0.2442
ω_6	false	true	false	0.0180	0
ω_7	false	false	true	0.0072	0.0072/0.2442
ω_8	false	false	false	0.7128	0

Repeated applications of Bayes conditioning is performed using the *chain rule* [120]:

$$P(\alpha_1 \wedge \alpha_2 \wedge \ \ldots \ \alpha_n) = \ P(\alpha_1 | \alpha_2 \wedge \ \ldots \wedge \alpha_n)\, P(\alpha_2 | \alpha_3 \wedge \ \ldots \wedge \alpha_n) P(\alpha_n) \qquad \textit{(EQ 4-5)}$$

Bayes conditioning is used to perform *case analysis*:

$$P(\alpha) = \sum_{i=1}^{n} P(\alpha \wedge \beta_i) = \sum_{i=1}^{n} P(\alpha | \beta_i)\, P(\beta_i) \qquad \textit{(EQ 4-6)}$$

Case analysis is the computation in the belief of event α by adding up beliefs in a number of mutually exclusive cases, $\alpha \wedge \beta_1, \ldots \alpha \wedge \beta_n$, that cover the conditions under which α holds.

When an event α is perceived to be a cause of event β, *Bayes Rule* [120] is used to assess belief in the cause given the effect:

$$P(\alpha | \beta) = \frac{P(\beta | \alpha) P(\alpha)}{P(\beta)} \qquad \textit{(EQ 4-7)}$$

In addition to Bayes conditioning, we use *conditional entropy* of a variable X given another variable Y to qutatify the average uncertainty about the value of X after observing the value of Y:

$$ENT(X|y) \stackrel{\text{def}}{=} -\sum_{x} P(x|y) \log_2 P(x|y) \qquad \textit{(EQ 4-8)}$$

Table 4-4 shows some entropies for the variable MCE in our previous example.

Table 4-4. Conditional Entropies Results

	$P(MCS)$	$P(MCS\|Solution = true)$	$P(MCS\|Solution = false)$
true	0.2	0.741	0.025
false	0.8	0.259	0.975
ENT(.)	**0.722**	**0.825**	**0.169**

Soft Evidence. We differentiate between two types of evidence, hard and soft. Hard evidence reflects that some event has conclusively occurred while soft evidence is not conclusive. For example, we may get unreliable evidence β has occurred, which may increase belief in β but not to the point of certainty. Soft evidence is specified in terms of its strength as it relates to either its integration or independence with existing belief. *Jeffrey's rule* [132] is used to integrate soft evidence with initial belief:

$$P'(\omega) \triangleq \begin{Bmatrix} \frac{q}{P(\beta)}, & if\ \omega \vDash \neg\beta \\ \frac{1-q}{P(\neg\beta)} & if\ \omega \vDash \beta \end{Bmatrix} \qquad \textit{(EQ 4-9)}$$

Jeffrey's rule specifies soft evidence as a constraint $P'(\beta) = q$, where P' denotes the new state of belief after accommodating the evidence and β is the event to which the evidence pertains. Jeffery's rule follows the same principle as Bayes conditioning where updated belief from β to q must sum to one. In addition, $P'(\beta) = q$ imposes the following constraint $P'(\neg\beta) = 1 - q$ and now updated belief from $\neg\beta$ to $1 - q$ must also add up to one. Jeffrey's Rule can be generalized as follows:

$$P'(\alpha) = \sum_{i=1}^{n} q_i P(\alpha|\beta_i) \qquad \textit{(EQ 4-10)}$$

Consider a service TCS that tracks the sales of a toy which can have one of three colors green tcs_g, blue tcs_b, or yellow tcs_y. We want to know whether the next day the cloth will be sold (s). The first three columns of Table 4-5 demonstrates initial or prior beliefs. The original belief in the toy being sold is $P(0.56)$. Moreover, the original belief in toy colors tcs_g, blue tcs_b, and yellow tcs_y are 0.3, 0.3, and 0.4, respectively. Given new evidence we conclude update our belief in toy colors to 0.7, 0.25 and 0.05. By applying Jeffery's rule we get $P'(s) = 0.7\left(\frac{0.12}{.3}\right) + 0.25\left(\frac{0.12}{0.3}\right) + 0.05\left(\frac{0.32}{0.4}\right) = 0.42$. The new evidence reduces out belief in the toy being sold. The fourth column, in Table 4-5, represent updated or posterior belief using Jeffery' Rule.

Table 4-5. Updating Evidence Base of Jeffery's Rule

Worlds	S	TCS	P(.)	P'(.)
ω_1	s	tcs_g	0.12	0.28 = 0.12 x 0.7/0.3
ω_2	$\bar{s}$	tcs_g	0.18	0.42 = 0.18 x 0.7/0.3
ω_3	s	tcs_b	0.12	0.10 = 0.12 x 0.25/0.3
ω_4	$\bar{s}$	tcs_b	0.18	0.15 = 0.18 x 0.25/0.3
ω_5	s	tcs_y	0.32	0.04 = 0.32 x 0.5/0.4
ω_6	$\bar{s}$	tcs_y	0.08	0.01 = 0.08 x 0.5/0.4

The second method for specifying soft evidence is *Bayes factor*:

$$k = \frac{O'(\beta)}{O(\beta)} \qquad \textit{(EQ 4-11)}$$

where k is the ratio of the odds $O'(\beta)$, of β after accommodating the evidence, $P'(\beta)/P'(\neg\beta)$ and the odds $O(\beta)$, of β before accommodating the evidence, $P(\beta)/P(\neg\beta)$. That is, the odds of 1 indicates that we believe β and $\neg\beta$ equally, wile the odds of

10 indicates that we belive β ten time more the be believe $\neg\beta$. Bayes factor is based on declaring the strength if evidence indecently from of currently held beliefs. As an example, consider three Web services $XCS, YCS\ and\ ZCS$, each containing different computer vision algorithms $xcs_a, ycs_b\ and\ zcs_b$. Supposed the initial belief about the performance of each algorithm to solve a specific problem is describe in column three of Table 4-6. The accompanying odds of the evidence is listed in column 4. According to the initial evidence, we could specified, "Web service XCS has triple the odds of solving the problem." Given new evidence, Bayes factor = 2, we have $\frac{O\prime(solutin=XCS)}{O(soution=XCS)} = 2$. Using this new evidence, $P'(solution = XCS) = \frac{3x2/3}{\frac{3x2}{3}+1/3} = \frac{6}{7} \approx 86\%$. With new evidence independent from the previous evidence The following statement can now be made "The new (soft) evidence has strengthen belief by 86%" or more intuitively "Belief has doubled".

Table 4-6. Bayes Factor Results

world	Machine Learning Service	P(.) of solution	O(.)of solution
ω_1	XCS	2/3	3
ω_2	YCS	1/6	0.2
ω_3	ZCS	1/6	0.2

4.6. BAYESIAN NETWORKS

This dissertation considers the family of probabilistic graphical networks known as Bayesian networks (BN) [133] to represent BPEL data dependencies. Bayesian networks represent a (joint) probability distribution using a graphical model of a directed, acyclic graph (DAG). Every node in the graph corresponds to a random variable in the domain and is annotated by a Conditional Probability Distribution (CPD), defining the conditional distribution of the variable given its parent [120]. It provides a model-based domain

description, where the model is reflecting properties of the problem domain (rather than the domain expert), and probability calculus is used as the calculus of uncertainty.

To facilitate an efficient representation of a broad and complex domain with many random variables, the framework of Bayesian networks uses a graphical representation to encode dependence and independence relations among the random variables. The dependence and independence relations induce a compact representation of the joint probability distribution. A Bayesian network encodes a joint probability distribution over a set of random variables, X, of a problem domain. The set of conditional probability distributions, P, specifies a multiplicative factorization of the joint probability distribution over X as represented by the chain rule, equation 4-5 [134].

Bayesian network that represents some probability distribution and is used to answer some probabilistic query often in the form of computing $P(X|E = e)$, i.e. the probability distribution over some random variables X given some evidence $E = e$. For example, in a travel itinerary workflow we can query the probability of flights given specific destinations and/or price ranges.

The construction of a Bayesian network (or any probabilistic network for that matter) proceeds according to an iterative procedure where the set of nodes and their states, and the set of links are updated iteratively as the model becomes more and more refined. Also, Bayesian networks provide inference. To solve a Bayesian network $N = (\chi, G, P)$ is to compute all posterior marginals given a set of evidence E, i.e., $P(X|E)$ for all $X \epsilon \chi$. If the evidence set is empty, i.e., $E = \emptyset$, then the task is to compute all prior marginals, i.e., $P(X)$ for all $X \epsilon \chi$.

Implementation of BNs uses the fact that the joint probability distribution of all the variables in a network factorizes according to the structure of the graph. The distributions of interest can then be found by a series of local computations, involving only some of the variables at a time [127]. Bayesian networks are therefore suitable for problems where the variables exhibit a complicated dependency structure. For an overview of different applications of Bayesian Networks see [135]. In this dissertation, we discuss already developed methods for inference and constructing the knowledge base, and we demonstrate an exact approximation method to model Web Service Workflows.

Capturing Independence Graphically. The qualitative component is used to capture independence. Even though the joint probability distribution specified by a Bayesian network is defined in terms of conditional independence, a Bayesian network is most often constructed using the notion of cause-effect relations. In practice, cause-effect relations between entities of a problem domain can be represented in a Bayesian network using a graph of nodes representing random variables and links representing cause-effect relations between the entities.

A Bayesian network relies on the basic insight that independence forms a significant aspect of beliefs and that it can be elicited rather easily using the language of graphs. Figure 4-3 shows an example of a simple Bayesian network implied from a set of BPEL activities. The Bayesian network represents a subset of possible cause of flight delays taken from the U.S. Department of Transportation's (DOT) Bureau of Transportation Statistics (BTS) [136]. The network models a Web service workflow of services monitoring possible causes of a delayed arriving flight. Nodes represent the variables of a Web service workflow and the edges in the graph represent “direct casual influences” among these variables.

For example, the "*Departure delay*" is a direct cause of "*Arrival delay*". Given this causal structure, belief in "*Arrival delay*" is influenced by evidence in "*Alert*". If we get an alert that, a weather event has occurred, then the belief that there will be an arrival delay can increase. However, if hard evidence suggest that there was a delayed flight, then evidence can negate the *"Alert"* node's evidence suggesting independence arrival delay from alert given departure delay.

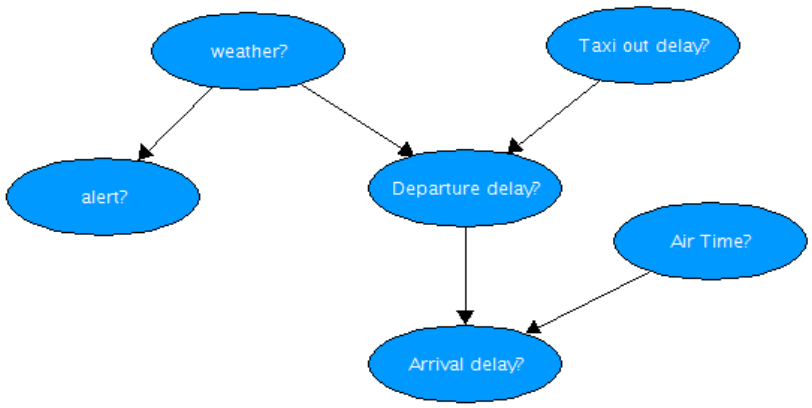

Figure 4-5. Flight Delay Web Service Bayesian Network.

Arrival Delays are represented by the random variable "*Arrival Delay"*, which is directly influenced by two variables, "*Departure Delay",* and "*Air time"*. These variables represent whether or not a plane was delayed at departure and the length of time the plane took during the flight. The "*Departure Delay"* variable is in turn directly influenced by variables "*Taxi out delay"* and "*weather"*. "*Taxi out delay"* represent whether or not there was a delay in the plane leaving the terminal gate. The "*weather"* variable represent whether or not there was a weather event that may cause a backlog. Finally, the "*alert"* variable is influence by the "*weather"* variable. The "*alert"* represents whether or not there was an alert indicating a weather event.

The *Markovian Assumption* is used to interpret formally the independence of the DAG of a Bayesian network:

$$I(\text{V, pa(V), non\_des(V)}) \qquad \textit{(EQ 4-12)}$$

where V is the set of nodes in graph $G = (V, E)$, $pa(V)$ are the parents of V and $non\_des(V)$ are the descendants of V. The Markovian assumption is denoted as $Markov(G)$. In general, $pa(V)$ denotes the *direct causes* of V and $non\_des(V)$ denotes the *effects* of V. The Markovian assumption dictates that given the direct causes of a variable, beliefs in that variable will be longer be influenced by any other variable except possibly by its effects. The following are all the independence statements of Figure 4-3:

$$I(AD, \{DD, AT\}, \{TOD, W, A\})$$

$$I(A, W\ \{AD, AT, DD, TOD\})$$

$$I(DD, \{W, TOD\}, \{A, AT\})$$

$$I(TOD, \emptyset, \{W, A, AT\})$$

$$I(W, \emptyset, \{TOD, AT\})$$

$$I(AT\ \emptyset, \{W, A, DD, TOD\})$$

Parameterizing the Independence Structure. *Parameterizing* is the quantifying of the dependencies between nodes and their parents. This quantity is represented as a joint probability distribution satisfying the independence assumptions of $Markov(G)$. After the structure is created as aforementioned, the set of conditional probabilities are outline as follows: For every variable X in the DAG G and its parents $\boldsymbol{U}$, we provide the probability $P(x|\boldsymbol{u})$ for every value x of $X$and every instantiation $\boldsymbol{u}$ of parent $\boldsymbol{U}$. For example in Figure 4-3, we provide the following conditional probabilities:

$$P(ad|dd,at), P(dd|w,tod), P(a|w), P(w), P(tod), P(at)$$

where $ad, dd, at, tod, w\ and\ a$ are values of variables $AD, DD, AT, TOD, W\ and\ A$. An example of the conditional probability quantities are:

Table 4-7. CPT for Variable A.

W	A	P(A|W)
true	true	0.80
true	false	0.20
false	true	0.001
false	false	0.999

This table is referred to as the *conditional probability table* (CPT) for variable AD. Each variable will have its own CPT.

We now formally define a Bayesian Network:

A *Bayesian Network* for variables $\boldsymbol{Z}$ is a pair (G, Θ), where:

- G is a directed acyclic graph over variables $\boldsymbol{Z}$, called the network *structure*.
- Θ is a set of CPTs, one for each variable in $\boldsymbol{Z}$, called the network *parameterization.*
- The probability distribution is given by the chain rule $P(z) \triangleq \prod_{\theta_{x|\boldsymbol{u}\sim z}} \theta_{x|\boldsymbol{u}}$

We use the following notations throughout the dissertation to represent the particulars of a Bayesian network:

- $\Theta_{X|\boldsymbol{U}}$ denote the CPT for variable X and its parents $\boldsymbol{U}$
- X**U** denote a *network* family
- $\theta_{x|\boldsymbol{u}}$ denote a specific value of a CPT and call it a network *parameter*
- $P(x|\boldsymbol{u})$ is the conditional probability of network parameter $\theta_{x|\boldsymbol{u}}$
- $\theta_{x|\boldsymbol{u}} \sim \boldsymbol{z}$ denote network parameter $\theta_{x|\boldsymbol{u}}$ is compatible with a network instantiation $\boldsymbol{z}$ when the instantiations $x\boldsymbol{u}$ and b are compatible, i.e. they agree on the values they assign to common variables.

Graphical Test of Independence. The Markov assumption cannot represent all of the independence assumptions of the JPD of a Bayesian network. We use a property called *d-separation* [120] which can derive independences not captured by $Markov(G)$ and are derived from graphoid axioms [123]. Given three sets of variables$\boldsymbol{X}, \boldsymbol{Y}, and\ \boldsymbol{Z}$ we test whether $\boldsymbol{X}$ and $\boldsymbol{Y}$ d-separated by $\boldsymbol{Z}$ in DAG G, written as $dsep_G(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ by considering whether every path between a node in $\boldsymbol{X}$ and a node in $\boldsymbol{Y}$ and ensuring that the path is blocked by $\boldsymbol{Z}$ if at least one node on the path is closed given $\boldsymbol{Z}$. The notation of a closed node is a node with hard evidence. We consider three types of paths and state the conditions for d-separation.

- A sequential path ($\rightarrow W \rightarrow$) is closed iff variable W appears in $\boldsymbol{Z}$.

  For example, the sequential path $W \rightarrow DD \rightarrow AD$ in Figure 4-3, is closed iff we know the value of variable DD, otherwise a weather event may change our belief in the plane arriving late.

- A divergent path ($\leftarrow W \rightarrow$) is closed iff variable W appears in $\boldsymbol{W}$.

  For example, the divergent path $A \leftarrow W \rightarrow DD$ in Figure 4-3, is closed iff we know the value W, otherwise an alert on a weather event may change our belief a departure delay.

- A convergent path ($\rightarrow W \leftarrow$) is closed iff neither variable W nor any of its descendants appears in $\boldsymbol{Z}$.

  For example, the convergent path $W \rightarrow DD \leftarrow TOD$in Figure 4-3 is closed iff neither the value of variable DD nor AD are known, otherwise, a taxi out delay may change the belief in weather.

Reasoning with Bayesian Networks. We reason with the Bayesian network in the following ways:

- *Probability of evidence*: the probability of some variable instantiation $\boldsymbol{e}$, $P(\boldsymbol{e})$
- *Prior and posterior marginal*: Given a joint probability distribution $P(x_1, \ldots x_n)$the *marginal distribution* or *prior* marginal $P(x_1, \ldots x_{m)}, m \leq n$ is:

$$P(x_1, \ldots x_{m)} = \sum_{x_{m+1}, \ldots x_n} P(x_1, \ldots x_n)$$

  The *posterior distribution* is the marginal distribution given some evidence $\boldsymbol{e}$

$$P(x_1, \ldots x_{m)}|\boldsymbol{e} = \sum_{x_{m+1}, \ldots x_n} P(x_1, \ldots x_n|\boldsymbol{e})$$

- *Soft evidence*: We use either Jeffery's rule or the Bayes factor as aforementioned
- *Most Probable Explanation (MPE)* [137]: Given $X_1, \ldots X_n$ represents all of the network variables and $\boldsymbol{e}$ is the given explanation, the goal is to identify an instantiation $x_1, \ldots x_n$ for which the probability $P(x_1, \ldots x_n|\boldsymbol{e})$ is maximal. When only a subset of network variables are used we consider the *Maximum a posterior hypothesis* (MAP) [138] measure. Given $\boldsymbol{M}$ is a subset of all network variables $\boldsymbol{X}$ and given some evidence $\boldsymbol{e}$, the goal is find an instantiation $\boldsymbol{m}$ of variables $\boldsymbol{M}$ for which the probability $P(\boldsymbol{m}|\boldsymbol{e})$ is maximal. Variables $\boldsymbol{M}$ are known as the *Map variables.*

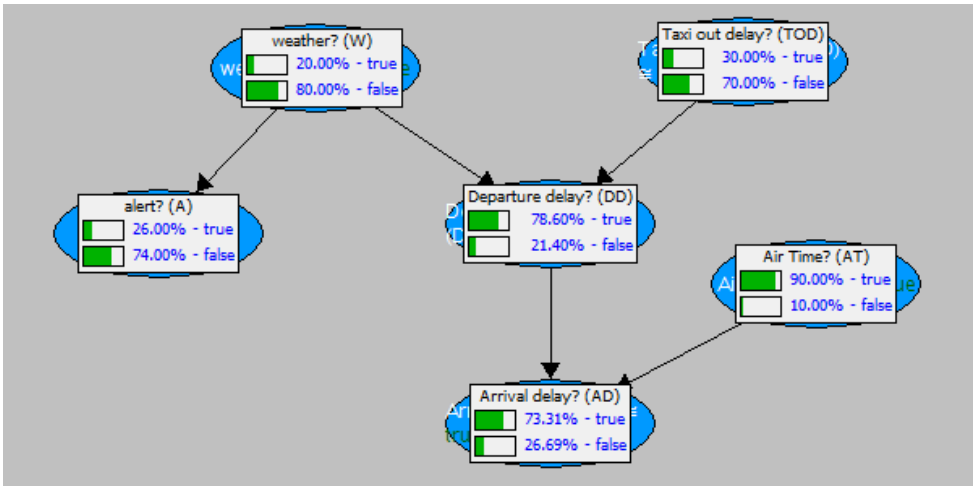

Figure 4-6. Prior Marginals for the Delay Service BN.

An example of the probability of evidence, assume we are interested in knowing the likelihood of an arrival delay, $P(AD = \text{"true"})$. The resulting probability is 76.3%.

To demonstrate the prior and posterior marginal respectively, assume we observe that there was an arrival delay, and we ask, "What is the probability that there was a weather event?" Stated more formally, we are looking for $P(W = Y|AD = F)$. Note that the position of the *Weather* node in the graph shows that this variable does not directly influence the *Arrival Delay* variable; however, these variables are not independent. If there is no arrival delay then there is a higher likelihood that the there is no weather event. Figure 4-4, shows the prior marginal values of the Delay Web service BN. Figure 4-5, shows the updated beliefs or posterior marginals after entering evidence $AD = F$. The initial belief that a weather could cause a delay was 20%, given hard evidence of no arrival delay the BN updated the belief that weather caused a delay to 25%. Intuitively, one would expect the belief in a weather event to decrease instead on increase. This is as a result of the strengths of the initial beliefs in both the

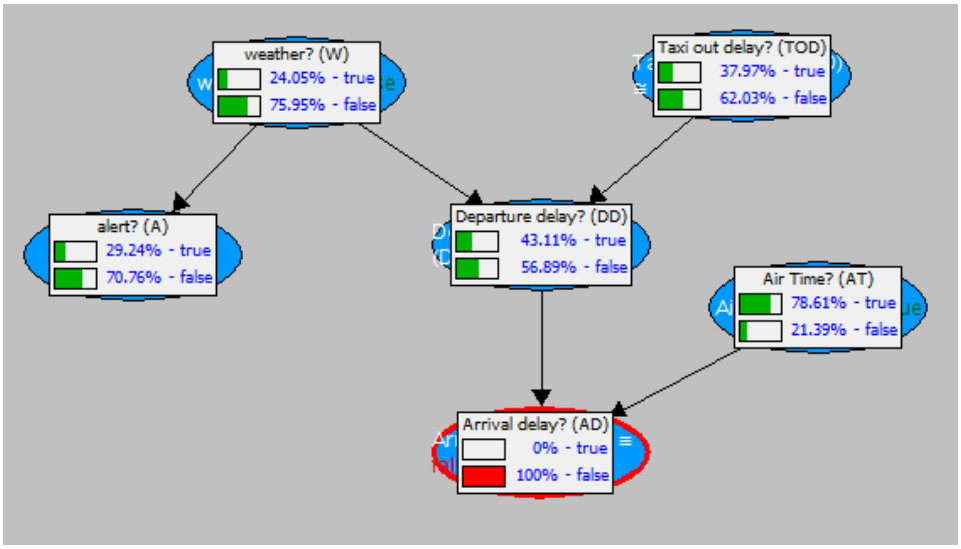

Figure 4-7. Posterior Marginals with Arrival Delay Evidence.

"*weather*" and "*Taxi delay out*" nodes and its effects on the "*Arrival delay*" node. Later we show a more informed way priming the network to reflect real world events.

As another example, consider the following query $P(AD|A = \text{"true"}, TOD = \text{"}false\text{"}$ Intuitively, since we have hard evidence on a weather alert we would expect that a weather event did in fact take place. Figure 4-6, shows that belief was updated to approximately 69% up from 20%. The arrival delay now shows approximately 72%, down from approximately 73%.

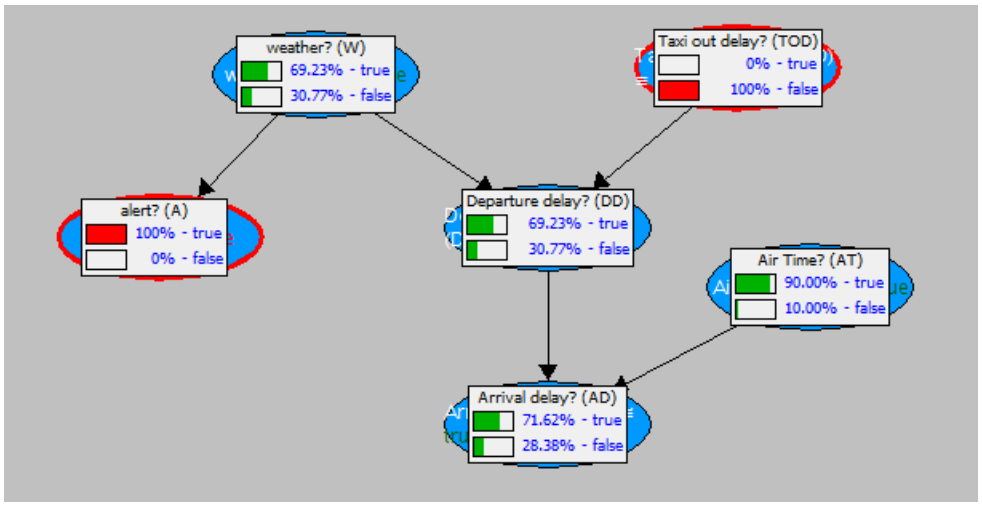

Figure 4-8.Posterior Marginals with Alert and Taxi out Delay Evidence

To demonstrate soft evidence, we use the Bayes factor. Suppose for example that we received the soft evidence that doubled the odds of a weather event or a taxi out delay. To represent the OR condition of two nodes and the Bayes factor condition, we add additional auxiliary nodes, "*W-or-TOD*" and "*Bayes Factor*".

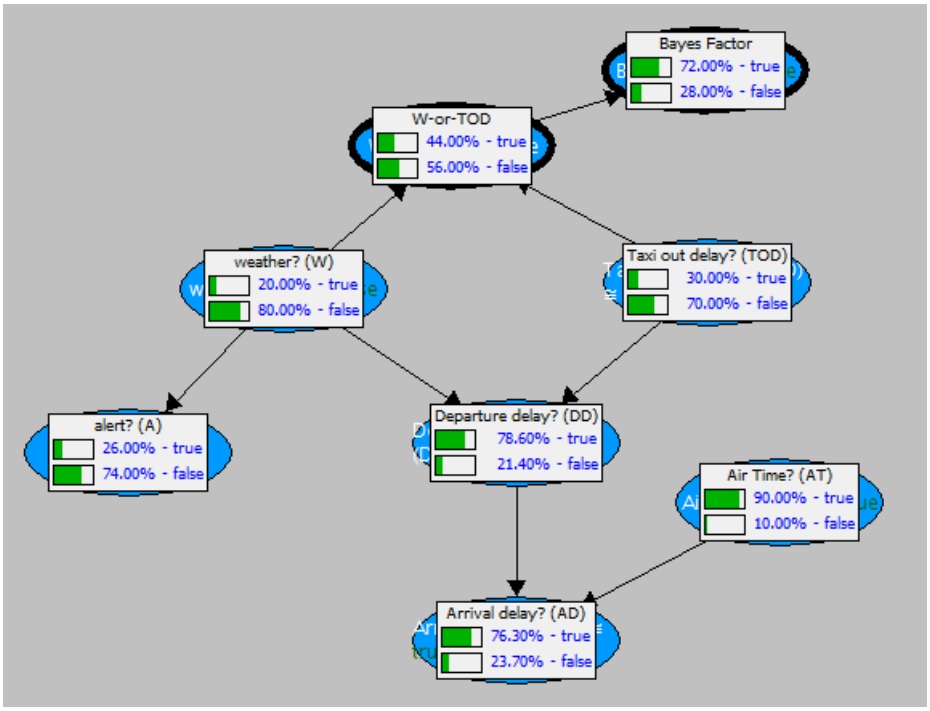

Figure 4-9. Auxiliary nodes for Soft Evidence.

Figure 4-9, show the prior marginals and Figure 4-10, shows the posterior marginal with the Bayes factor as soft evidence.

Finally, we use the MPE to compute the most viable path within the workflow given a particular evidence for which the probability $P(x_1, \ldots x_n|\boldsymbol{e})$ is maximal to determine. We use the variable elimination algorithm [139] to compute the MPE. The variable elimination algorithm uses *factors* to represent each instantiation of variables in a distribution. A factor is a function over a set of variables, mapping each instantiation of theses variables to a non-negative number. There are two key operations that apply to factors. The first is summing out a variable from a factor and the second is multiplying two factors. The variable

elimination algorithm first multiplies factors with common a variable and then sum out that variable from the resulting common factor. In contrast, MPE by variable elimination maximize out that variable instead of summing it out. The *MPE probability* given $\boldsymbol{e}$ and Bayesian network variable $\boldsymbol{Q}$ is defined as:

$$MPE_p(e) \stackrel{\text{def}}{=} maxP(q|\boldsymbol{e}) \qquad \textit{(EQ 4-13)}$$

Since there may be a number of instantiations $\boldsymbol{q}$ that attain this maximal probability then each instantiation is then an *MPE* instantiation, where the set of all such instantiations is defines as:

$$MPE_p(e) \stackrel{\text{def}}{=} argmaxP(q|\boldsymbol{e}) \qquad \textit{(EQ 4-14)}$$

Consider for example the MPE for arrival delay being true, $MPE(x_{BN}|AD = "true")$, where x_{BN} represent all the other variable in the BN. The results returns departure delay and weather event as the probable causes.

It is important to realize that the reasoning in these examples comes directly from the joint probability distribution over the domain variables. In other words, there is no need to create special reasoning rules since all the information that we need is encoded in the joint probability distribution. Thus, if we can represent the probability distribution and answer the probabilistic queries about it, we get this type of sophisticated reasoning "for free" using the probabilistic approach.

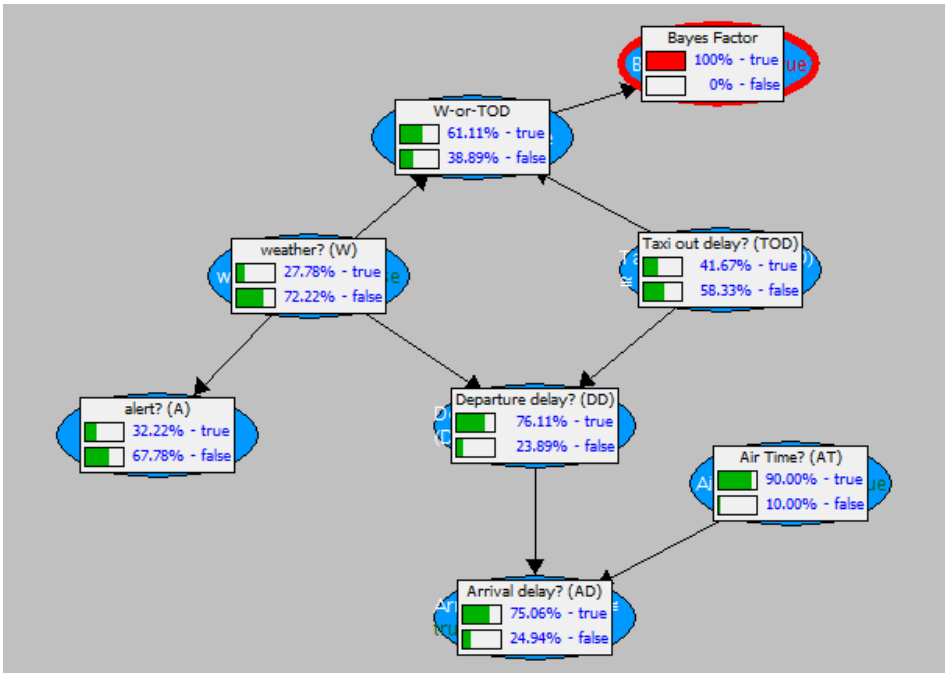

Figure 4-10. Bayes factor as Soft Evidence.

Learning from Data. Previous we parameterize the Bayesian network from prior knowledge of the domain of the workflow. For our first example we asked, given there was no arrival delay what is the likelihood that a weather event occurred $P(W = Y|AD = F)$. Intuitively, we would most likely guess that since there was no delay then most likely an weather event did not occur. However, the Bayesian network returned a higher posterior marginal than the prior marginal that a weather event has occurred. This is because we had to rely only on expert opinion to populate the BN which may skew the BN to non-real world scenarios. To rely less heavily on prior marginal, which may not reflect real world events, we use historical data to parameterize the Bayesian network. Historical data would capture historical trends of the workflow behavior and would hence better reflect real world events and its probabilities.

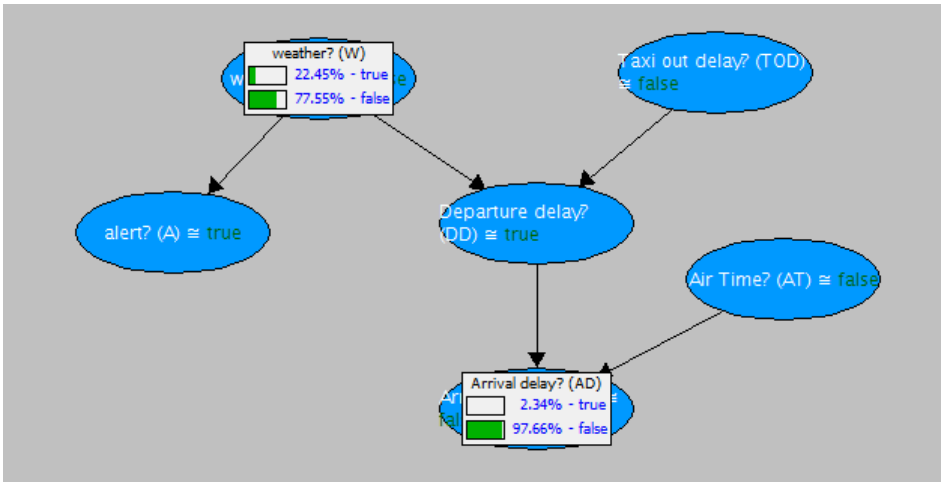

Figure 4-11. Parameterizing with Data

To parameterize the BN from data we use the Expectation Maximization (EM) learning algorithm [140], The EM algorithm determines the values of conditional dependencies between network variables quantitatively based on observed data. In particular, this method generates a belief model that maximizes the expectation of the data. We assume the data could be complete i.e. without missing values of incomplete.

Given a complete data set, Table 4-8, generated independently and according to their probabilities an *empirical distribution* can be used to summarize the data set as follows:

$$P_D(w, tod, ad) = \frac{D\#(w,tod,ad)}{N} \qquad \textit{(EQ 4-15)}$$

where $D\#(w, tod, ad)$ is the number of cases in the data set D that satisfy instantiation w, tod, ad and N is the data set size. Given EQ 4-13, we estimate the parameter $\theta_{x|\boldsymbol{u}}^{ml}$ by the empirical probability:

$$\theta_{x|\boldsymbol{u}}^{ml} \overset{\text{def}}{=} P_D(x|\boldsymbol{u}) = \frac{D\#(x|\boldsymbol{u})}{D\#(\boldsymbol{u})} \qquad \textit{(EQ 4-16)}$$

Let θ be the set of all parameter estimates for a given network structure G and let P_θ be the probability distribution induced by the structure G and estimates θ then the likelihood of these estimates is defined as:

$$L(\theta|D) \stackrel{\text{def}}{=} \prod_{i=1}^{N} P_\theta(d_i) \qquad \textit{(EQ 4-17)}$$

The probability of the data is maximized with parameters θ is the probability of the set D conditioned on the parameters. Equation 4-15, is maximized when $\theta_{x|\boldsymbol{u}} = P(x|\boldsymbol{u})$ the empirical probability and is called the *maximum likelihood (ML)*.

$$\theta^{ml} = argmaxL(\theta|\boldsymbol{D}) \qquad \textit{(EQ 4-18)}$$

For a complete data set, estimates are based on the empirical distribution and then the likelihood function is maximized. For incomplete data it common to start with the goal of maximizing the likelihood function and then derive the estimates.

Consider for example our initial query $P(W = Y|AD = F)$. Figure 4-11, shows the resulting probability values after learning from data after starting from a uniform distribution prior. More so, it shows a much lower probability that a weather event has occurred in the past. In addition, the event of an arrival delay is also very low. Figure 4-12,

Table 4-8, Complete Data

Case	W	TOD	AD
1	T	F	T
2	T	F	T
3	F	T	F
4	F	F	T
5	T	F	F
6	T	F	T
7	F	F	F
8	T	F	T

Table 4-9. Empirical distribution

W	TOD	AD	$P_D(.)$
T	T	T	0/8
T	T	F	0/8
T	F	T	4/8
T	F	F	1/8
F	T	T	0/8
F	T	F	1/8
F	F	T	1/8
F	F	F	1/8

shows the results after asserting the evidence from the query. The results indicate that the weather has a negligible increase and hence effect when the arrival delay is true. This result is more intuitive to expectations.

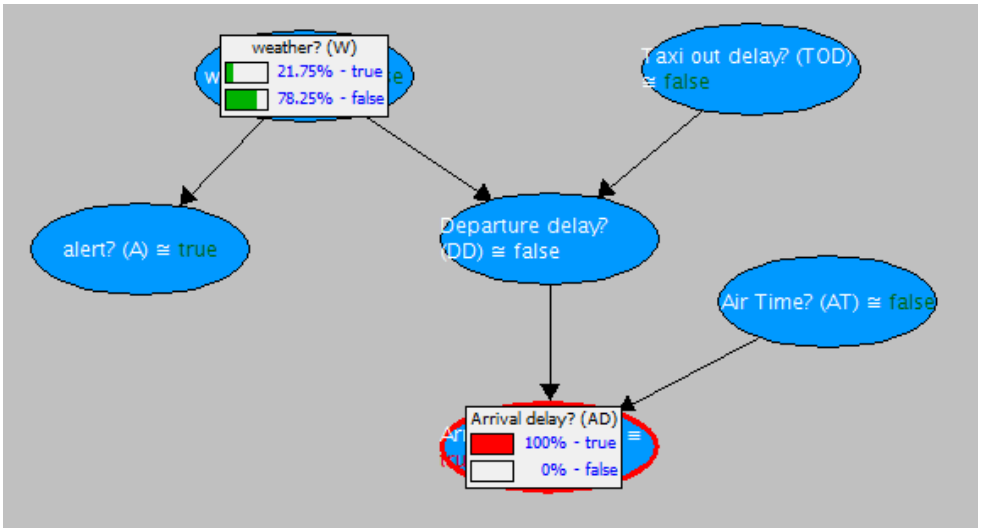

Figure 4-12.Asserting Evidence after Learning from Data.

Network Robustness. To measure the robustness of the network we use a property of ML, which minimizes the KL divergence [141] or the relative entropy between learned Bayesian networks. Since the ML estimates are unique for a given structure G and the complete data set D, the likelihood of these parameters is then a function of the structure G and the data set D. The *likelihood of structure* G given data D is given as:

$$L(G|D) \stackrel{\text{def}}{=} L(\theta^{ml}|D) \qquad \textit{(EQ 4-19)}$$

The log-likelihood is similarly defined as:

$$LL(G|D) \stackrel{\text{def}}{=} logL(\theta^{ml}|D) \qquad \textit{(EQ 4-20)}$$

The *minimum description length* (MDL) or *Bayesian information criterion* (BIC) is a measure used to avoid overfitting a complex Bayesian network to the data i.e. creating a model that has too many parameters compared to the available data [142]. We measure model complexity by the number of independent parameters within the model. Let $\psi(N).||G||$ be a penalty term that favors simpler models where $\psi(N)$ is the penalty weight and $||G||$ is equal to the number of independent parameters in $G's$ CPT or $G's$ dimension then MDL is defined as:

$$MDL(G|D) \stackrel{\text{def}}{=} LL(G|D) - \psi(N).||G|| \qquad \textit{(EQ 4-21)}$$

$\psi(N) = \frac{log_2 N}{2}$, which allow the penalty term to grow logarithmically in N, while the log-likelihood term grows linearly in N. The goal is to minimize the score instead of maximizing it [142].

In addition, we use the Receiver Operating Characteristic curve (or ROC curve) [143]. The ROC curve is a plot of the true positive rate against the false positive rate for the different possible cut points of a test. ROC graphs are commonly used in medical decision making, and in recent years have been used increasingly in machine learning and data mining research. The ROC curve inspects the performance of a given variable as a classifier for the data set. An ROC curve demonstrates several things: It shows the tradeoff between

sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity). The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the node. Finally, the area under the curve is a measure of text accuracy. An area of 1 represents a perfect test; an area of .5 represents a poor test.

To shows how well a given variable performs as a predictor of the actual class, we show confusion matrix [143]. The error rate is reported, which is the measure of the number of false classifications over a number of true classifications. The matrix is constructed as follows:

Table 4-10. Confusion Matrix.

True Negative actual class != target class P(target class) < threshold	False Negative actual class == target class P(target class) < threshold
False Positive actual class != target class P(target class) >= threshold	True Positive actual class == target class P(target class) > threshold

Sensitivity analysis. Sensitivity analysis considers the impact of parameter changes on query values, and the amount of parameter change needed to enforce some constraints on these values. Sensitivity analysis, able the finding of the set of network parameters that guarantee a required accuracy. We use the *CD Distance* [144], measure the impact of parameter changes. The CD distance measure is formally defined as:

$$D(P^0, P) \stackrel{\text{def}}{=} \ln max_x \frac{P(x)}{P^0(x)} - \ln min_x \frac{P(x)}{P^0(x)} \qquad \textit{(EQ 4-22)}$$

where P^0 and P are the probability result of a query in the form of $\alpha|\beta$ with respect to two networks N^0 and N after changing a parameter in N^0 to obtain a new network N. The CD distance is a network-independent measure, which allows for the efficient computation for

distributions that correspond to very similar Bayesian networks by bounding query results for N^0 distribution with respect to N distribution [144]. Assuming all parameters are the same between N^0 and N, the CD distance between the global distribution is exactly the same as the CD distance between the local distribution. As an example, consider the fact there was a weather event, Figure 4-11.

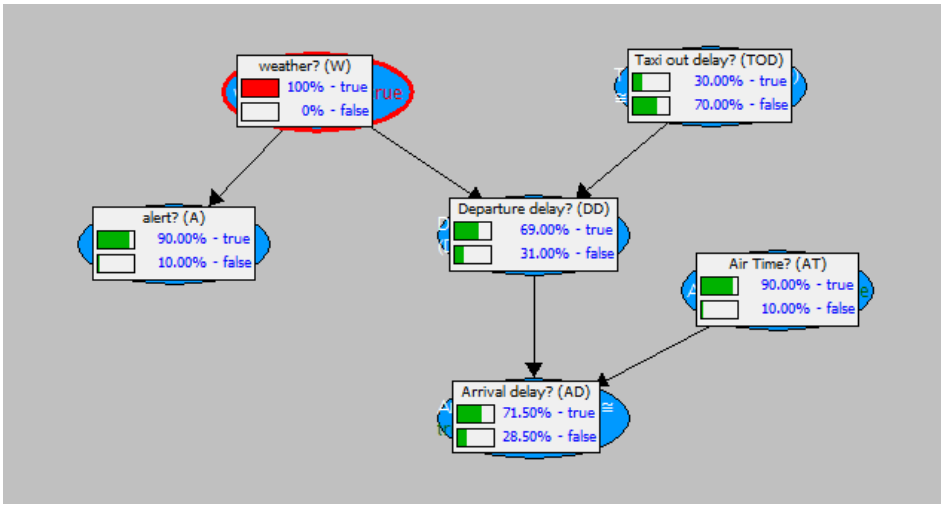

Figure 4-13. Network Probabilities with Weather Event as Hard Evidence.

We would like to know which parameters we would have to change in the network to reduce an arrival delay to 60%. Table 4-8, show the results from the CD measure. The results are sorted in increasing order of the log-odds change because we want to consider the smallest change as a top priority. There are three suggestions, fist given departure delay and air time true, then reduce arrival delay from 90% to 71%. Second, weather event occurred and taxi out delay is false then this will reduce departure time from 60% to 27%. Third, reduce air time from 90% to 52%. Finally, given a weather event occurred and there was a taxi delay

Table 4-11. Sensitivity Analysis Results.

Parameter	Current Value	Suggested Value	Absolute Values	Log-odds Change
$P(AD = T \mid DD = T, AT = T)$	0.9	<= 0.71	>= 0.19	>= 1.28
$P(DD = T \mid\mid W = T, TOD = F)$	0.6	<= 0.27	>= 0.33	>= 1.40
$P(AT = T)$	0.9	<= 0.52	>= 0.38	>= 2.13
$P(DD = T \mid\mid W = T, TOD = T)$	0.9	<= 0.13	>= 0.77	>= 4.06

then reduce departure delay from 90% to 13%. If our delay Web service Bayesian network represented all of the constraints of an actual workflow, then choice two would be most viable. The other may not implementable in a real world situation. Figure 4-12, show the updated BN after in cooperating the sensitivity analysis suggestions.

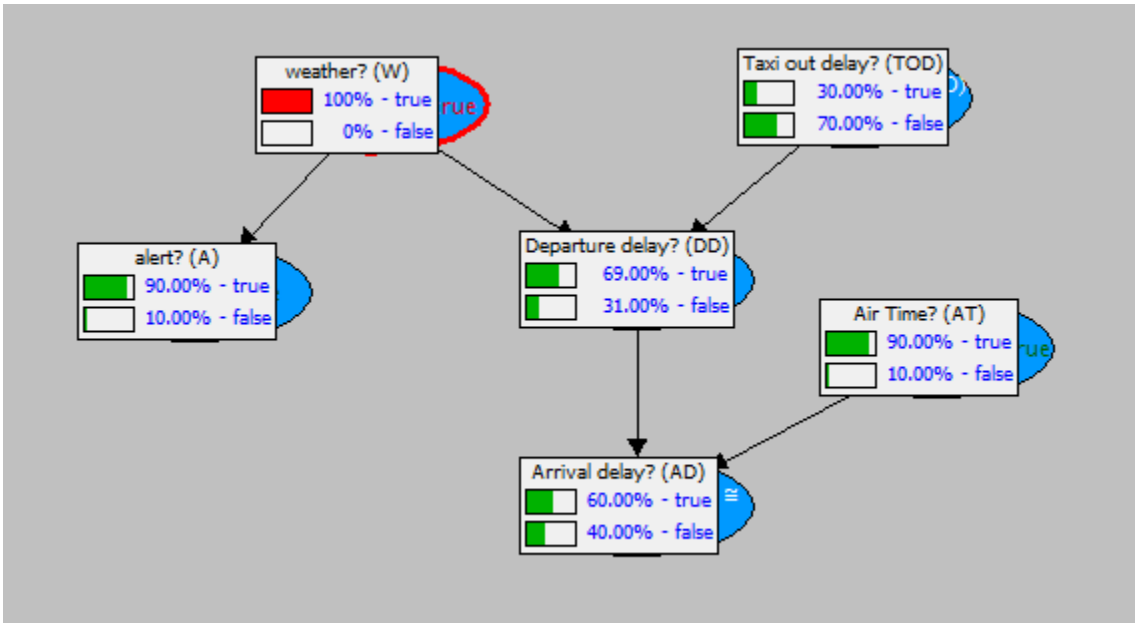

Figure 4-14. BN after Sensitivity Analysis.

Chapter 5
METHODOLOGY, EVALUATION, AND DISCUSSION

This chapter details our approach used for representing Web Service Workflows realized by a BPEL process as a normative expert system to provide a data-derived inferential visibility of workflow semantics. We formalize the expert system as a Bayesian network. The pattern of edges in the graph represents the knowledge base or qualitative dependencies between BPEL activities; the absence of an edge between two nodes means that any statistical dependency between two BPEL activities is mediated via some other activity or set of activities. The quantitative dependencies between activities that are connected via edges are specified via parameterized conditional distributions, or more generally non-negative "potential functions". The pattern of edges and the potential functions together specify a joint probability distribution over all the activities in the graph. We refer to the pattern of the edges as the *structure* of the graph while the parameters of the potential functions simply as the *parameters* of the graph.

In the case of Bayesian networks, there is a clear distinction between the knowledge base and the inference engine. The knowledge base is the structure of the Bayesian network, whereas the inference engine is a set of generic methods that applies the knowledge formulated in the knowledge base. The knowledge base is the structure of the graph and the probabilistic values of parameters are used to update our belief about the state of the world or to identify (optimal) decisions in the light of new knowledge. Data in existing workflow

data silos are used to provide initial probability values for parameters and ongoing data for belief updating.

Queries can be made to the knowledge base to answer non-explicitly defined trends about the behavior of workflow users. Examples of queries are "*Which product or supplier is best?* Also, *Should a product or supplier be changed?* Responses to these queries and others will now be responsive to time-dependent trends. A benefit of this framework is the transformation of BPEL into a more agile representation that is more responsive to uncertain and dynamic business trends by providing a holistic inferential view of business processes.

We contribute to the body of knowledge in this area by creating a principled methodology (shown in Figure 5-1) for capturing historical and ongoing data and process details for long-standing workflows. Moreover, we contribute an analysis approach for capturing user specific causal factors that can facilitate on-demand service workflow interventions, providing decision support when optimizing long-standing workflow operations, creating a mechanism to provide service workflow predictions.

We perform the computation of our Bayesian model in the SamIam [145] and the Hugin [146] modeling tools. Both tools support a graphical user interface and a reasoning engine. The graphical interface lets users develop Bayesian network models. The reasoning engine supports many tasks including classical inference; parameter estimation; time-space tradeoffs; sensitivity analysis; and explanation generation.

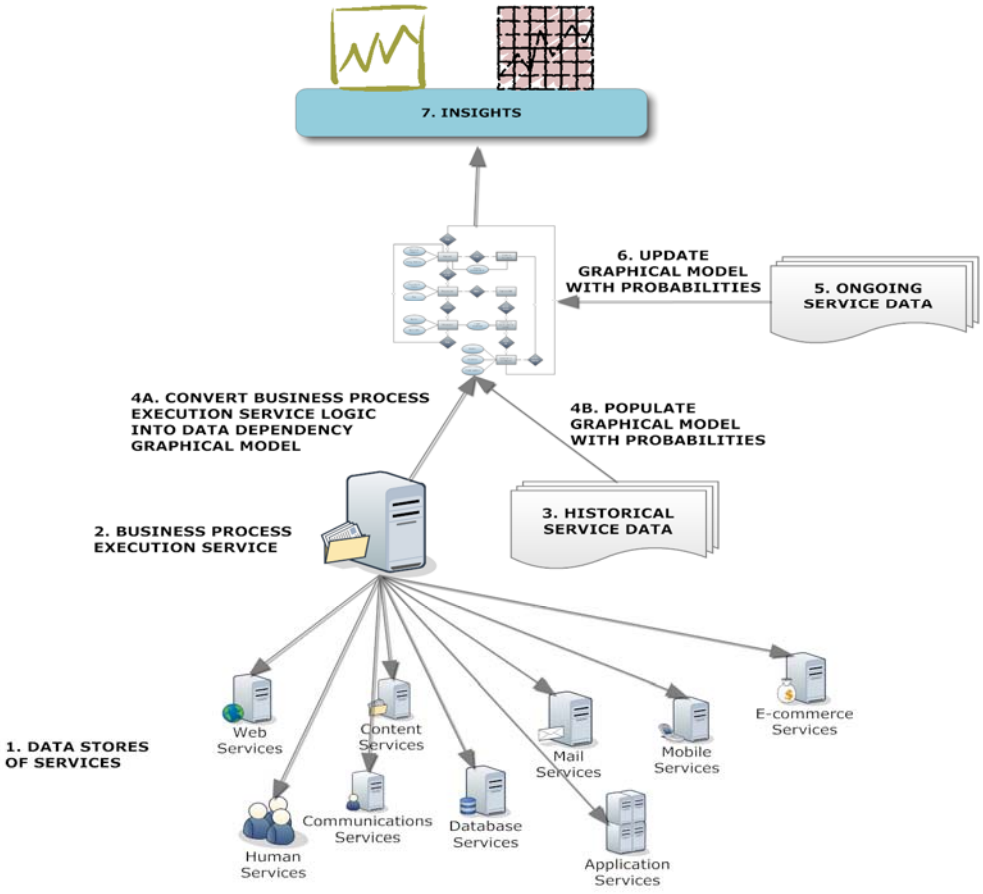

Figure 5-1. Methodology for Representing BPEL as a Normative Expert System.

The following modeling steps are performed to transform a BPEL process into a normative expert system.

1. *Workflow semantics.* We define a workflow semantics to represent formally the data dependencies within the messages of a BPEL process.
2. *Structure Learning.* The structure of the Bayesian network DAG is learned manually from BPEL activities. BPEL activities realize workflow scenarios that often include concurrency, cyclic/acyclic operations, and/or decisions/branches/merges. These causal relationships guide the implementation of network structure. A DAG is constructed ensuring dependence/independencies among variables is consistent with the underlying

domain of the workflow. The nodes of the network represent the variables while the edge patterns the data dependencies. A uniform distribution is used for the prior marginal. The resulting DAG variables are categorized either as a: query, evidence, and intermediary variable. A *query variable* is one that questions are asked about. An *evidence variable* is one about which we need to assert evidence. Finally, an *intermediary variable* is neither query nor evidence and is meant to aid the modeling process by detailing the relationships between evidence and query variables.

3. *Learn from Data:* We parameterize the Bayesian network from the historical data of the workflow. Define the network variables values the Joint Probability Distribution (JPD). For every variable X in the DAG G and its parents $\boldsymbol{U}$, we provide the probability $P(x|\boldsymbol{u})$ for every value x of $X$and every instantiation $\boldsymbol{u}$ of parent $\boldsymbol{U}$
4. *Evaluate Network robustness.* The MDL or BIC is the measure used to avoid overfitting the Bayesian network to the data i.e. creating a model that has too many parameters compared to the available data.
5. *Sensitivity analysis.* Sensitivity analysis considers the impact of parameter changes on query values, and the amount of parameter change needed to enforce some constraints on these values. We use the *CD Distance* to find the set of network parameters that guarantee a required accuracy.
6. *Querying the model*. Querying the model is the same as reasoning in a Bayesian network. To enable the query process, we use the query types aforementioned.

As a motivating scenario, consider the Airline Ticket Pricing Workflow shown in Figure 5-2. Such a workflow consists of a Select Airlines service that takes origin and destination cities from the user and devises a list of air carriers that fly between the two cities. Subsequently, a Collect Prices service contacts the list of airlines and develops the best ticket price. We define special cases of the state variable as defaulted variables; variables whose values are created at design time and values are time dependent. The Collect Prices service may have many defaulted variables that are not used in every instance of the workflow such as the number of connections, time of day of departure, and time of day of arrival. In this service workflow and others like it, the user may be able to leverage unused message types to find the most optimal price. For this scenario, the date and time of purchase and the time-of-day for the flight also may affect the optimal price.

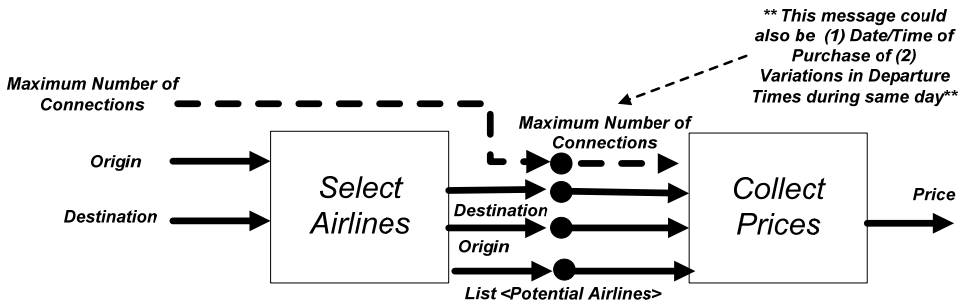

Figure 5-2. Airline Ticket Pricing Workflow

5.1. WORKFLOW SEMANTICS

Our workflow semantics define four types of data transitions to represent the set of sequential and concurrent BPEL activities [118].

Content Oriented Service Workflow Semantics (COCWS). Interfaces between services in workflow scenarios often include concurrency, cyclic/acyclic operations, and/or decisions/branches/merges. Our solution examines the data flow within the messages considering these interfaces. Services are modeled as black boxes where time dependent

trends in the exchange of their messages can be derived to understand more fully the optimality of the overall operations of the workflow.

We define a workflow, W, as illustrated in Figure 5-2,

- As the set of data exchanges between services.
- These data exchanges are defined as transitions, T where $W = < T_1, \dots, T_c >$, and c is the total number transitions within the workflow.
- Each transition, T can be defined as a set of parts, P where $T =< P_1, \dots, P_g >$, and g is the total number of parts.
- Parts can be further defined as the 2-tuple $P = \{P_{p,}P_r\}$. $P_{p,}$ represents the pairing of the particular providing service, S, and the specific outputs from that service, o, representing the tuple, $P_p = \{S, o\}$. P_r represents the pairing of the receiving service, S, and the specific inputs to that service, i, representing the tuple, $P_r = \{S, i\}$.

As with the workflow in Figure 5-3, the transitions are the shaded rectangles containing its parts and the arrows represent the message flow containing the workflow data and logic.

Service Workflows Element Types.

We divided the elements of a workflow into two parts:

1. The atomic components of a service workflow are the set of services, $\{S_1, S_2, \dots S_b\}$, where b is the total number of services comprised in the workflow.
2. The sub-atomic elements of a service, S, consists of a set of inputs, $\{i_1, i_2, \dots, i_m\}$, and a set of outputs $\{o, o_2, \dots, o_n\}$, where m is the total number of inputs and n is the total number of outputs.

Inputs and outputs for web services can be realized by input messages and output messages, each message separated into a set of transition parts.

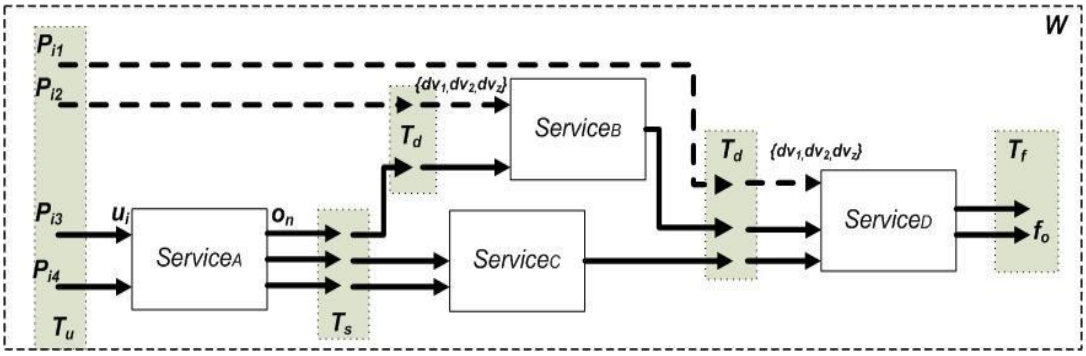

Figure 5-3. Service Workflow with Formal Annotations.

Data Transitions Types.

We defined four transitions types that assimilate specific conditions that will be considered in our workflow representation and optimization methodologies.

1. *The initial user-specified transition*, T_u represents the first transition in the workflow that contains user-designated parts. As such, this transition is defined as $T_u = \{P_p, P_i\}$, where P_p is not represented by a providing service but as:

   $P_p = \{S, o\}$, where $S = null$ and $o = null$

   $P_{r_i} = \{S, u_i\}$, where S is the initial service and u_i is the user input.

2. *The output transition*, T_f represents the final transition in the workflow that contains the result of the workflow operations. The transition is defined as $T_f = \{P_f, P_r\}$, where P_r is not represented by a receiving service but as:

   $P_{p_f} = \{S, f_o\}$, where S is the final service and o the workflow outputs

   $P_r = \{S, 0\}$, where $S = null$ and $o = null$

3. *The dynamic incremental transition*, T_d, represents a transition within the workflow where at least one part is composed of defaulted values, d_i, where $d_i =< d_1, d_2, \dots, d_z$

and z is the number of possible defaulted values. The transition is defined as $T_d = \{P_p, P_r\}$, where either P_p or P_r contains defaulted value:

$P_{p_d} = \{S, d_i\}$, where S is an intermediary service and d_i its defaulted outputs

$P_{p_r} = \{S, d_i\}$, where S is an intermediary service and di its defaulted inputs

4. Finally, we define the *stationary transition*, T_s, as $T_s = \{P_p, P_r\}$, where either P_p or P_r are stationary values that does not change within the workflow:

   $P_{p_s} = \{S, o\}$, where S is either an initial or intermediary service and o is the stationary output

   $P_{r_s} = \{S, i\}$, where S is either an initial or intermediary service and i is the stationary input

In the flight example, P_i and P_{p_s} could represent a user's personal information, departure and arrival time, flight origin and destination and S represent the Select Airlines web service. P_{p_f} represents the output of the workflow with are the specific airlines prices.

"Hidden" within the service workflow are built-in defaulted and business logic values, P_{p_d}. These hidden values can have a high impact factor on the output of the workflow such as the price of a flight and intermediary stages within the workflow. With respect to our example, defaulted values may capture the number of legs in a trip, the demand of a travel route (e.g. the availability of seats or the rate at which seats are being filled), as qualified by external events such as weather conditions or seasonal time of year.

To demonstrate, evaluate, and verify our approach, the remainder of the chapter will detail two evaluative examples. The first study models a Web service workflow that captures and monitors events that may cause a flight to be within the United States (US). The list of causes are defined by the DOT BTS and data was collected by their efforts. The

workflow monitors both sequential and concurrent causes of flight delays in the order they occur. A challenge is to determine the most influential cause/s for flight delays. The second study and example simulates a hybrid Web service workflow of Human computing services from crowdsourcing and machine services from algorithms. The workflow simulates a set computing services (CS) comprising of human computing service (HCE) and machine computing service (MCS), which tries to solve a common. A challenge is to predict which CS performs best.

5.2. CASE STUDY & EVALUATION 1:

FLIGHT DELAY WORKFLOW

Description.

We continue the modeling process with the flight delay workflow aforementioned in Chapter 4. We use an expanded list of flight delay causes as listed by the DOT, shown in Table 5-1. The DOT BTS tracks the on-time performance of domestic flights operated by large air carriers by collecting flight details to determine the causes of flight delays from airports across the United States [136]. Summary information on the number of on-time, delayed, canceled and diverted flights appears in its monthly Air Travel Consumer Report, published about 30 days after the month's end. Summary statistics and raw data are made available to the public at the time the Air Travel Consumer Report is released. The expanded delay Web service workflow has three conceptual sections, Figure 5-4. The delay profile management section, which records the occurrence of a delay. The delay management section, which captures the sequential or concurrent dependencies delays. Finally, the

learning management section is our proposed abstraction, which learns delay cause for given flights.

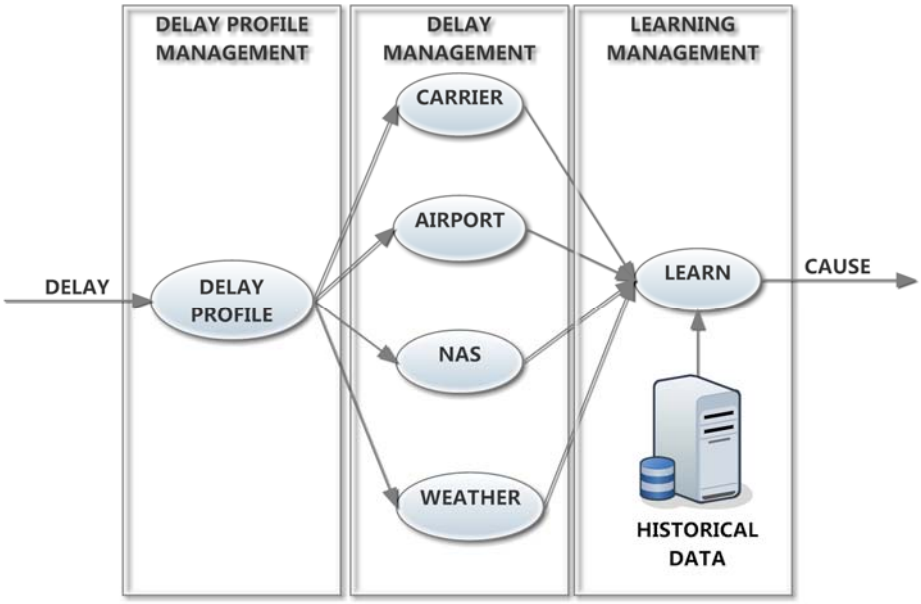

Figure 5-4. Flight Delay Web Service Workflow.

Dataset.

We made the underlying assumption that the data from the cause of each flight delay was collected by a Web service orchestrated by a BPEL process forming a Web service workflow infrastructure. In our demonstration, we only use the 2008 flight data set that contains data for January and February. The data consists of flight arrival and departure details for all commercial flights within the USA, in 2008 and contains approximately one million records in total. Table 5-1 list the description of each variable in the dataset.

Table 5-1. Dataset Variable Descriptions.

	Name	Description
1	Year	2008
2	Month	1-12
3	Day of Month	1-31

4	Day of Week	1 (Monday) - 7 (Sunday)
5	Departure Time	actual departure time (local, hhmm)
6	CRS Dep. Time	scheduled departure time (local, hhmm)
7	ArrTime	actual arrival time (local, hhmm)
8	CRSArrTime	scheduled arrival time (local, hhmm)
9	Unique Carrier	unique carrier code
10	FlightNum	flight number
11	TailNum	plane tail number
12	Actual Elapsed Time	in minutes
13	CRSElapsedTime	in minutes
14	AirTime	in minutes
15	ArrDelay	arrival delay, in minutes
16	DepDelay	departure delay, in minutes
17	Origin	origin IATA airport code
18	Dest	destination IATA airport code
19	Distance	in miles
20	Taxi In	taxi in time, in minutes
21	Taxi Out	taxi out time in minutes
22	Cancelled	was the flight canceled?
23	Cancellation Code	reason for cancelation (A = carrier, B = weather, C = NAS, D = security)
24	Diverted	1 = yes, 0 = no
25	Carrier Delay	in minutes
26	Weather Delay	in minutes
27	National Aviation System (NAS) Delay	in minutes
28	Security Delay	in minutes
29	Late Aircraft Delay	in minutes

BPEL mapping to Bayesian network.

For this use case, we are specifically concerned with the variables that represent the causal factors for a delayed flight. The data was first curated to remove redundant nodes by combining columns when necessary. For example, the "*Flight Delay (FD)*" node was obtain by combining columns ActualElapsedTime and CRSElapsedTime in the original data set using the following logic *If ((CRSElapsedTime +15) > (ActualElapsedTime) then FD = TRUE else FALSE*. In should be noted that the DOT BTS classifies a delay as any flight departing or arriving after fifteen minutes of the Customer Reservation Schedule (CRS) time. As a result, columns with continuous numeric values such as those that represented delay in minutes were converted to "TRUE" of "FALSE" if the number of minutes was greater than fifteen. Table 5-2 list the data curation methods used to obtain each node in the extended delay BN.

To accurately represent dependencies with the data, we made the following assumptions:

- Sequential Invocation/Storage. Service invocations or data storage was made in sequential order to capture real time occurring sequential events. For example, a

Table 5-2. Data Curation Methods.

Node	Original Column/s	Calculation
Departure Delay(DD)	DepTim and CRSDepTime and DepDelay	If (DepDelay>15) then DD = TRUE Else FALSE
Arrival Delay (AD)	ArrTime, CRSArrTime, ArrDelay	If (ArrDelay>15) then AD = TRUE Else FALSE
Weather Delay (WD)	WeatherDelay	If (WeatherDelay) > 15 then WD = TRUE else FALSE
Taxi Out Delay (TOD)	TaxiOut	If (TaxiOut) > 15 then TOD = TRUE else = FALSE

Late Air Craft Delay (LAD)	LateAircraftDelay	If (LateAircraftDelay) > 15 then LAD = TRUE else FALSE
Carrier Delay (CD)	CarrierDelay	If (CarrierDelay) > 15 then CD = TRUE else = FALSE
NAS Delay (NASD)	NASDelay	If (NASDela) > 15 then NASD = TRUE else FALSE
Taxi In Delay (TID)	TaxiIn	If (TaxiIn) > 15 then TID = TRUE else FALSE
Flight Delay (FD)	ActualElapsedTime, CRSElapsedTime	If ((CRSElapsedTime +15) > (ActualElapsedTime) then FD = TRUE else FALSE
Month (M)	Month	If (M) > 15 then M = TRUE else FALSE

Week of Month (WOM)	DayofMonth	If (DayofMonth) > 0 and < 8 then WOM = First
		If (DayofMonth) > 7 and < 15 then WOM = Second
		If (DayofMonth) > 14 and < 21 then WOM = Third
		If (DayofMonth) > 7 and < 15 then WOM = Fourth

departure delay would have to occur before an arrival delay and, as such, must be reflected in this way in each row of the data.

- Parallel invocation/storage. Service invocation or data storage were made in parallel order to capture real time occurring parallel events. For example, a carrier delay could occur at the same time a National Air System (NAS) delay is occurring and as such must be reflected in this way in each row of the data.

Figure 5-5 shows the extended delay BN with the assumptions above. The shape or structure of the BN is its qualitative characterized by the DAG. The nodes can represent random variables, decision variables, or utility functions, and the links represent direct dependencies, informational constraints, or they indicate the domains of utility functions. The labels of the nodes refer either to (i) the names of the nodes, (ii) the names of the variables represented by the nodes, or (iii) descriptive labels associated with the variables represented by the nodes.

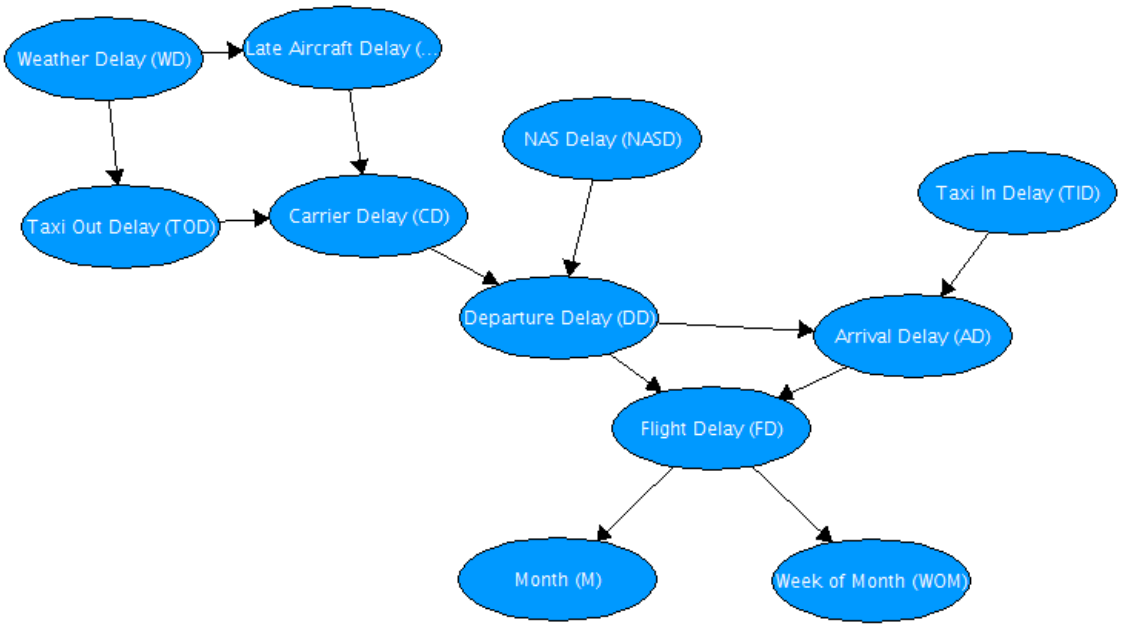

Figure 5-5. Extended Delay Bayesian Network.

Learning from Data.

One of the goals of this dissertation is to capture the uncertain user/system trends of a Web service workflow. We hypothesized that these trends were inherently embedded within the data generated from the workflow. To realize these trends in an explicit way, we parameterized our Bayesian networks from workflow data. Initially, we parameterize the network with a non-informative uniform distribution. Figure 5-6, shows the prior marginals. In this initial step, we are postulating that, other than the causal relationships between network variables, which derived from the BPEL process we have no initial knowledge of the casual strength or belief in the relationships.

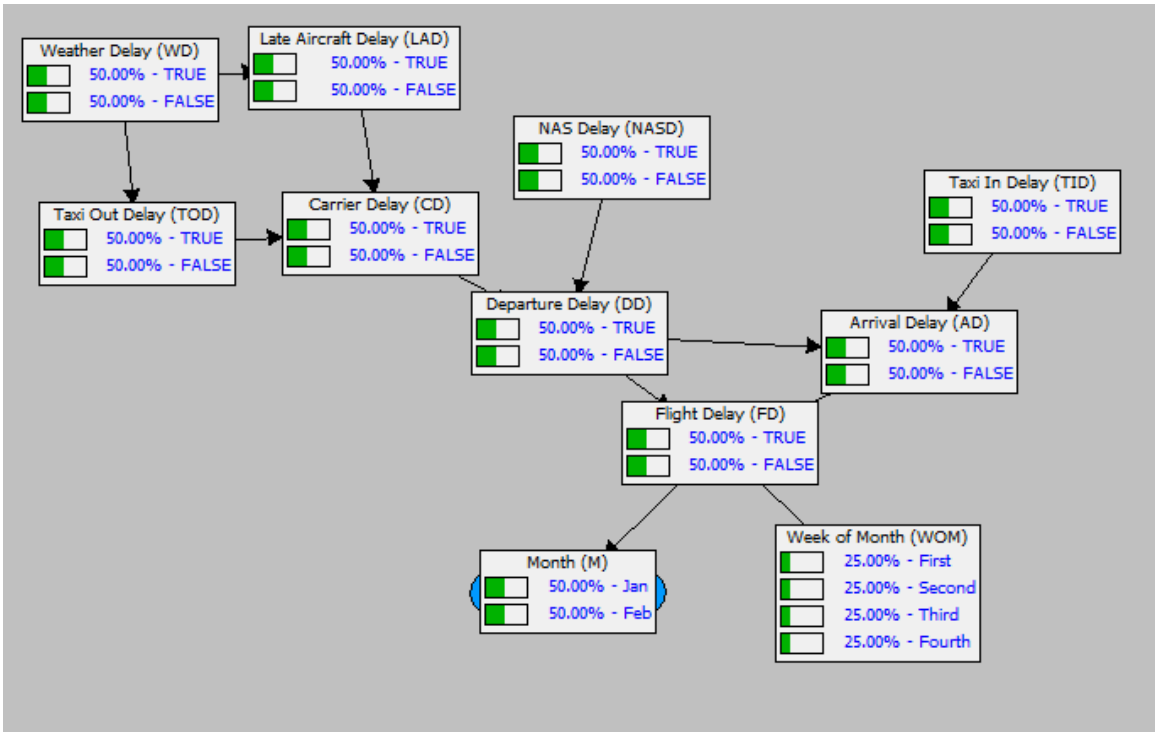

Figure 5-6. Initial Uniform Distribution of Extended Delay BN.

To obtain the posterior marginal, we parameterize the BN from data using the EM learning algorithm. The EM algorithm determines the strengths of casual relationships and hence our belief in the relationships. These are probabilistic values of conditional dependencies between variables quantitatively based on the observed data. In particular, this method generates a belief model that maximizes the expectation of the data. Figure 5-7 shows the posterior marginal for the Extended Delay BN. The probabilities values represented the quantitative attribute of the BN.

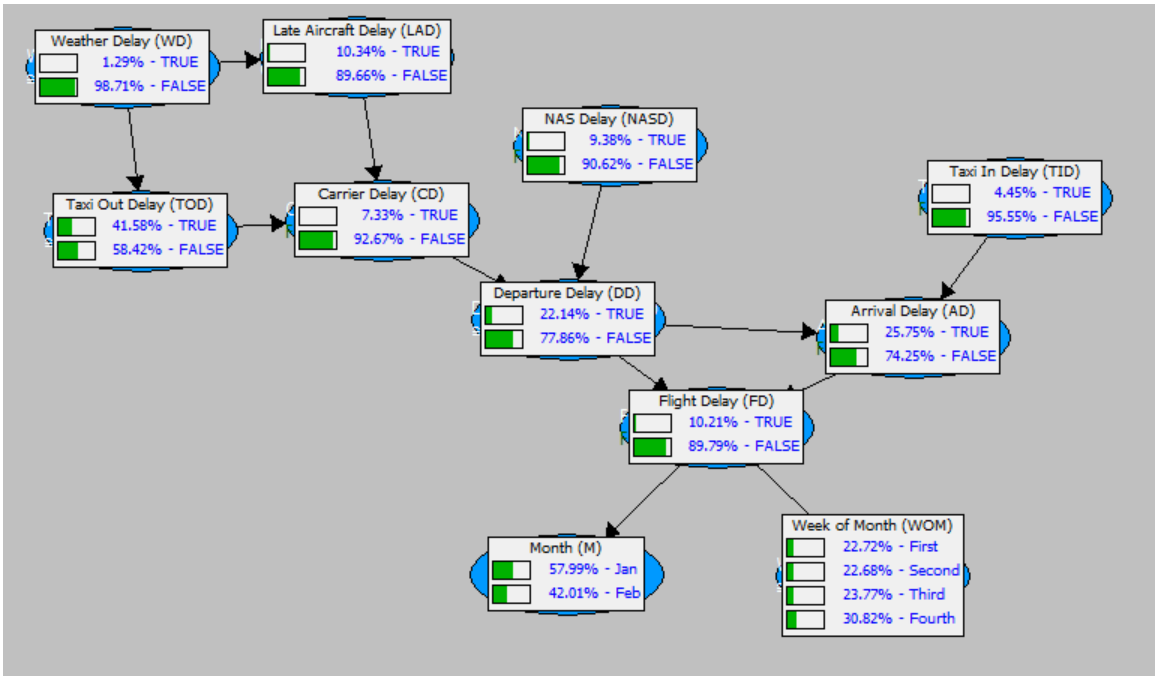

Figure 5-7. Posterior Marginals for Extended Delay BN.

Based on the BN's quantitative attribute, we can now articulate our belief about the behavior of the workflow in a sound mathematical manner. In other words, we can now make mathematically valid data-driven decisions. For example, we say with approximately 90% certainty that, within the first two months of 2008, there were no flight delays issued to the airlines as represented in the data. Also, the event with the highest probability to cause a delay is *taxiing out* and second the highest probability by *late aircraft delay*.

Evaluating Model Robustness.

Evaluating a Bayesian network is centered on how well the network structure "fits" the data or rather finding the maximum likelihood structure [147][142]. The brute-force approach to Bayesian network structure learning is to enumerate all possible DAGs, score each one and select the best one. In our approach, we learn the structure of the network from the BPEL process. In other words, we use expert knowledge to derive the network structure and hence

only generate one network structure. We use the follow evaluations methods each previously described;

- BIC: To evaluate the fitness of the overall network with the data
- ROC, Confusion Matrix, and its error rates: To assess the accuracy of a given node on the network.

First, we show the results for network fitness and the *Arrival Delay* node.

1. Network fitness: *BIC = -2419.06*
2. ROC Node Evaluation: *Arrival Delay*

Measure	Value
Cut off	0.0752
Area under the curve	0.92603

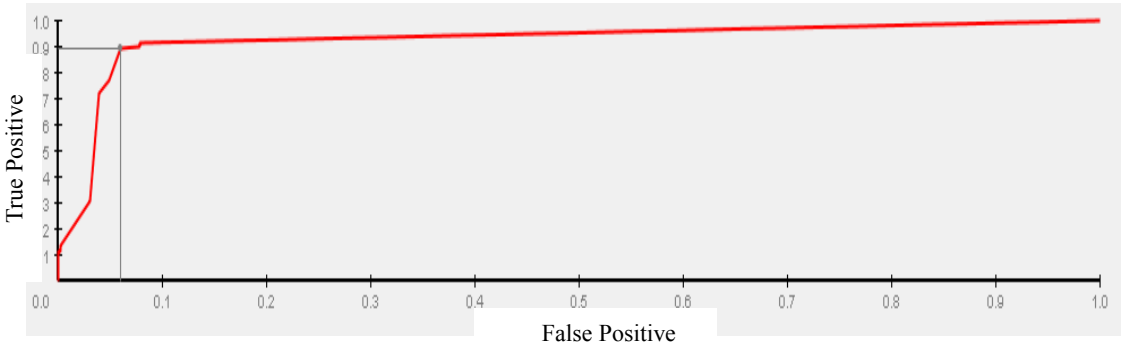

Figure 5-8. ROC for Arrival Delay

3. Confusion Matrix:

!True	True	
352	10	!True
30	107	True

4. Error rate = 8.02%

Since the goal is to minimize the BIC, for the extended delay BN the BIC score shows the network fits the data well. The ROC results for the *Arrival Delay* node shows that

this node has a high degree of accuracy since the area under the curve is close to one. The results from the confusion matrix with an error rate of 8.02% also supports this claim.

Next, we show the evaluation results for the *Carrier Delay* node

1. ROC Node Evaluation: *Carrier Delay*

Measure	Value
Cut off	0.3828
Area under the curve	0.87114

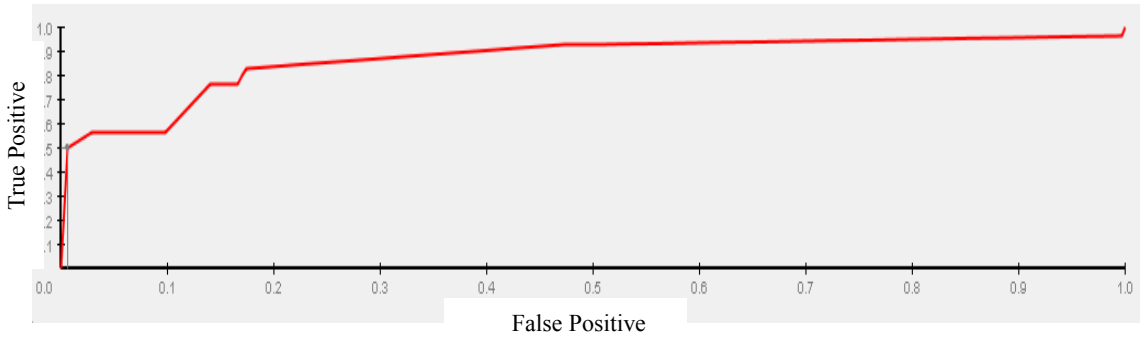

Figure 5-9. ROC for Carrier Delay.

2. Confusion Matrix:

!True	True	
469	30	!True
0	0	True

3. Error rate = 6.02%

The results for the *Carrier Delay* is a few percentage points less accurate than the *Arrival delay* but with an 87% accuracy it shows that this node can make reasonable predictions.

Finally, we display the evaluation results for the *Taxi Out Delay* node. You may recall that, after calculating the posterior marginal, this node had almost a 50/50 percent chance of causing a delay, as shown in Figure 5-8.

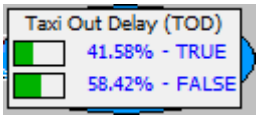

Figure 5-10. Taxi Out Delay Posterior Marginal.

1. ROC Node Evaluation: *Taxi Out Delay*

Measure	Value
Cut off	0.5461
Area under the curve	0.5252

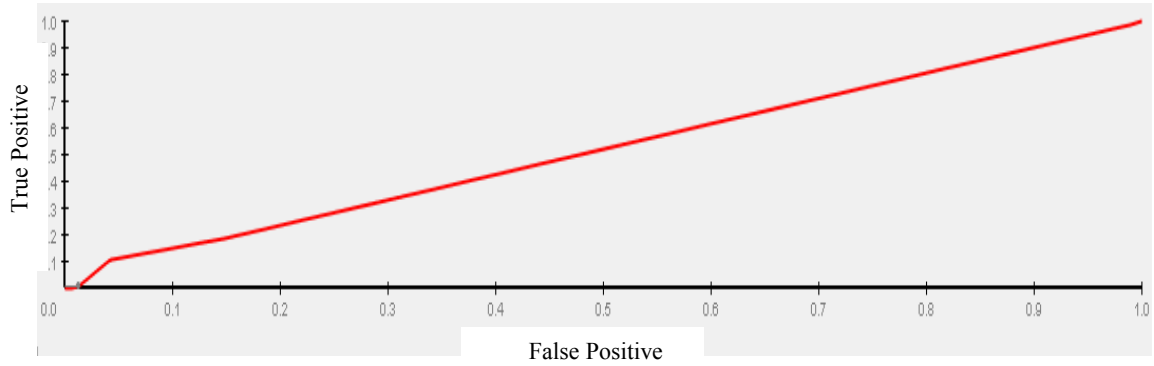

Figure 5-11. ROC for Taxi out Delay.

4. Confusion Matrix:

!True	True	
483	182	!True
13	21	True

5. Error rate = 39.08%

As anticipated, the *Taxi Out Delay* is not a good indicator for predicting delay. Its area under the curve is just above 50%, and ROC curve has an almost 45-degree angle.

Decision support and User specific assessments.

Some of the goals of this dissertation include providing decision support when optimizing long-standing workflow operations and providing Web service workflow prediction. To accomplish this, we perform probabilistic reasoning by using the four query above types: 1. The probability of evidence, 2. Prior and posterior marginal 3. Soft evidence and 4. Most Probable Explanation (MPE).

We performed the following queries:

1. When is the best time of time to fly to minimize delays, Jan or Feb and which week?
2. What is the primary cause of a carrier delay?
3. Are there critical links in the system?

Results for Query 1:

We use the *probability of evidence* $P(e)$ and *the posterior marginal* $P(FD|e)$ to answer this query: We enter hard evidence Month = "Jan" and Week of Month = "First", "Second", "Third" and then "Fourth". We then change the evidence for Month to February and record the results for each week of the month. In each instance, we also record the $P(e)$. Figure 5-11, shows the instance of hard evidence M = "Jan" and WOM = "First" and Table 5-3 show the results for $P(e)$ and $P(FD = false)$ for the hard evidence. Table 5-3, shows the answer to the query being the second week in January.

Table 5-3. Best Month and Week of Month to Fly.

Evidence Variables		$\boldsymbol{P(e)}$	$\boldsymbol{P(FD = false)}$
M = "Jan"	WOM = "First"	0.132	89.72%
	WOM = "Second"	0.132	91.5%
	WOM = "Third"	0.138	90.5%
	WOM = "Fourth"	0.179	88.9%
Evidence Variables		$\boldsymbol{P(e)}$	$\boldsymbol{P(FD = false)}$
M = "Feb"	WOM = "First"	0.1	89.13%
	WOM = "Second"	0.1	90.96%
	WOM = "Third"	0.1	89.94%
	WOM = "Fourth"	0.13	88.23%

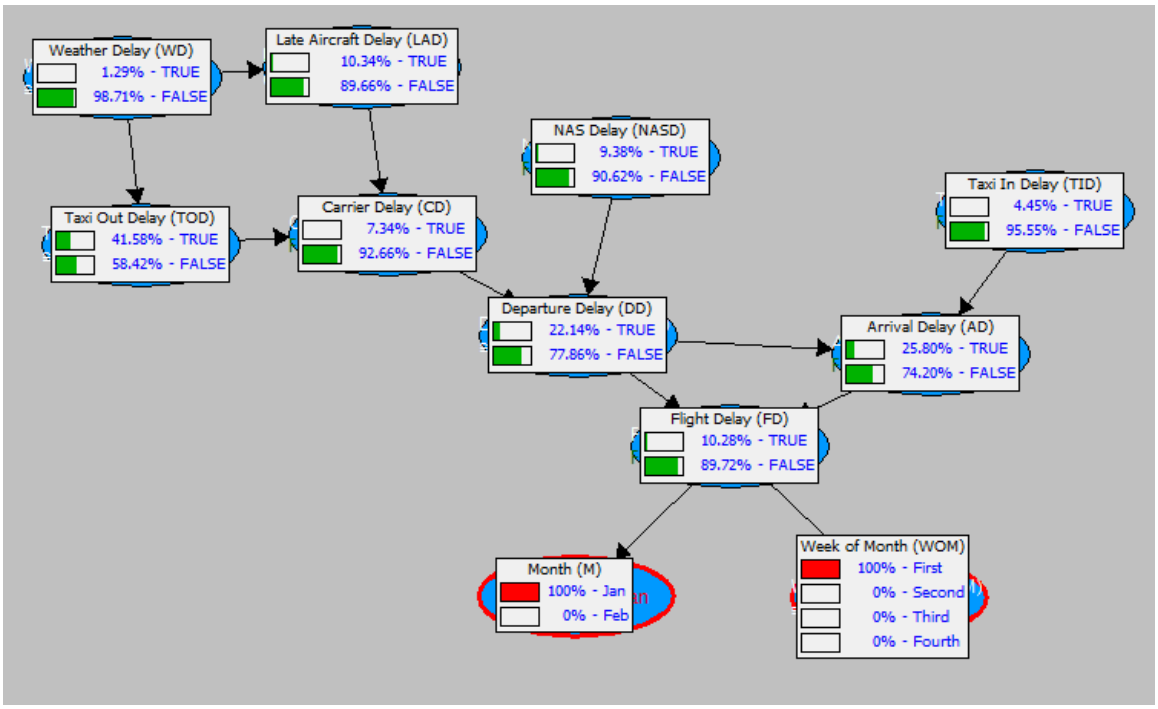

Figure 5-12. Hard Evidence M=Jan and WOM=First.

Prior and posterior marginal. Prior marginal for initially entered into the BN as the non-informative uniform distribution, see Figure 5-5. We did this to not make sure initial assumptions has little influence in the behavior of the workflow and to allow the data to influence mostly posterior marginal. Examples of posterior marginal can be seen in in the $P(FD = false$ column of Table 5-3 and in Figure 5-11. For example the prior $P(FD)$ and posterior marginal for *Flight Delay*, $P(FD|e)$, are:

FD	$P(FD)$	$P(FD\|e)$
TRUE	10.21%	10.28%
FALSE	89.79%	89.72%

Since the probability of the evidence $P(e)$ was very low $P(FD|e)$, see table 5-3, the effect of the evidence on the results was minimal.

Results for Query 2:

We use *MAP* to answer this query and later *soft evidence* to answer an updated query. To answer this query, we are finding the most probable instantiation of a subset of network variables. We use the MAP property with evidence (e) CD = True and MAP variables {TOD, WD, LAD} i.e. we want to know the most likely instantiation of these variables given the evidence, $P(MAP|e)$. The following Table 5-4 show the most probable explanation for $P(MAP|e)$ for variables LAD, TOD and WD, with taxi out delays being the most influential reason for a carrier delay and $P(MAP|e) \approx 48\%$.

Table 5-4. MAP for CD = true

Variable	Value
LAD	False
TOD	True
WD	False

Suppose we for example we received external evidence from the workflow that doubles the odds of an aircraft delay. We represent this evidence as an auxiliary node *Bayes Factor*, as shown in Figure 5-12. The prior marginal over node LAD, as shown in Figure 5-6 is 10.34%, with increase odds of late aircraft delivery the posterior marginal for node LAD is increased to 18.74%, as shown in Figure 5-12. If we evaluate the MAP property with the odds of an aircraft delay doubled $P(MAP|e) \approx 42\%$. However, the results in Table 5-4 still remain the same. This shows that a delay in taxiing out is still a strong factor in casing an aircraft to be late.

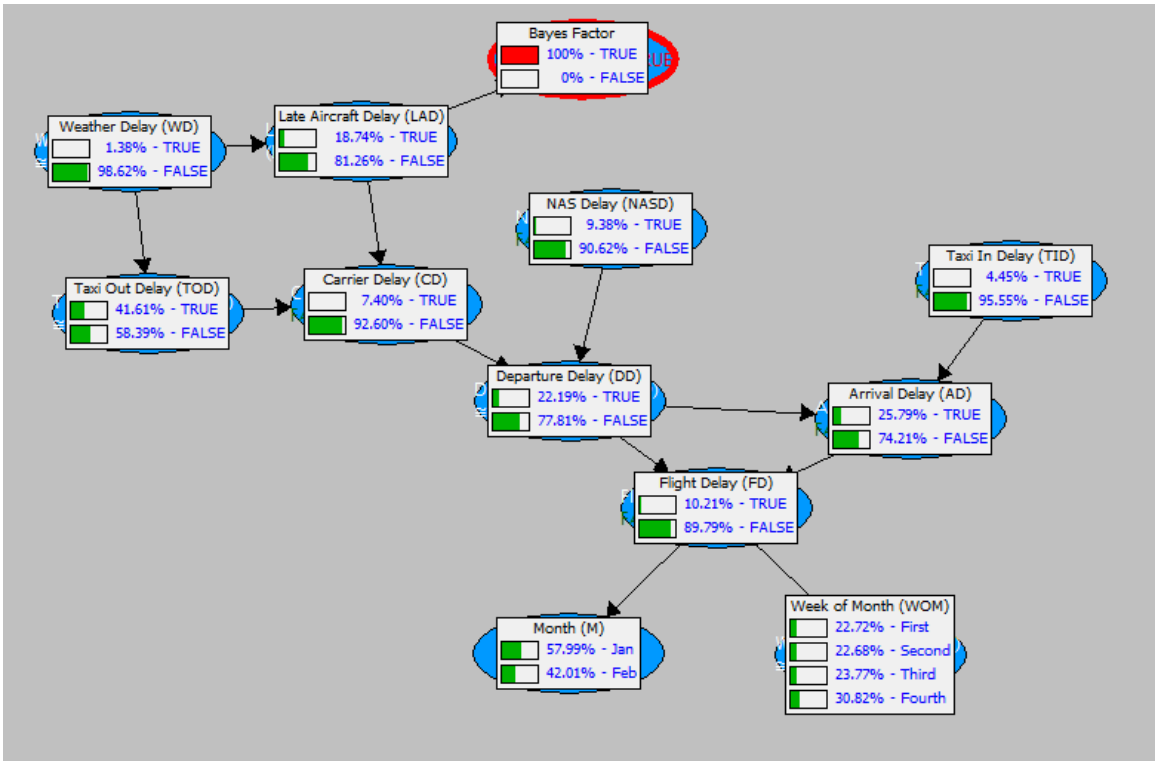

Figure 5-13. Updated BN showing Auxiliary Bayes Factor node.

Results for Query 3:

We use the *MPE* property to answer query 3. MPE is a special case of MAP when the MAP variables include all network variables. To determine the critical links in the workflow, we simulate an a*ll system fail* condition and then analysis possible causes. An all system fail in the extended delay BN would mean there was a carrier delay, departure delay, arrival delay and flight delay as shown in Figure 5-13. We then perform MPE to determine the most likely causes. $P(mpe|e = 0.04)$ and Table 5-5, shows the results for values of the network variables. Once again, taxi out delay is a likely cause. However, since MPE property analyzes all network variable, in this case we are given a time frame of when taxi out delays caused a system failure i.e. January in the fourth week. The MPE and MAP results allow us to analysis the specific areas of the workflow that may need improving.

Table 5-5. MPE foe All Systems Fail

Variable	Value
LAD	False
TOD	True
WD	False
NASD	False
WD	False
M	Jan
WOM	Fourth

Workflow Optimizations.

Finally, we demonstrate how our approach facilitates workflow optimization. From our previous queries, it is clear that taxi out delay is a major cause for delays. The question now arises, from the BN workflow's perspective, what needs to be done to reduce the effect of a taxi out delay on flight delays. To answer this question, we perform *sensitivity analysis.* Sensitivity analysis evaluates the relationships between network variables and the conclusions drawn based on the network. Specifically, we will determine which variables to change, and by how much, to ensure that taxi out delay could be reduced by 10%. Table 5-6 shows the results from the CD measure. The results are sorted in increasing order of the log-odds change because we want to consider the smallest change as a top priority. There are two suggestions, first given WD = false, then reduce $P(TOD = true|WD = false)$ from 41% to 30%. Second, increase $P(CD = true|TOD = ture, LAD = false)$ from 9% to 47%.

Table 5-5-6. Sensitivity Analysis Results.

Parameter	Current Value	Suggested Value	Absolute Values	Log-odds Change
$P(TOD = true\|WD = false)$	0.41	<= 0.303	>= 0.108	>= 0.47
$P(CD = true\|TOD = ture, LAD = false)$	0.09	>= 0.47	>= 0.377	>= 2.136

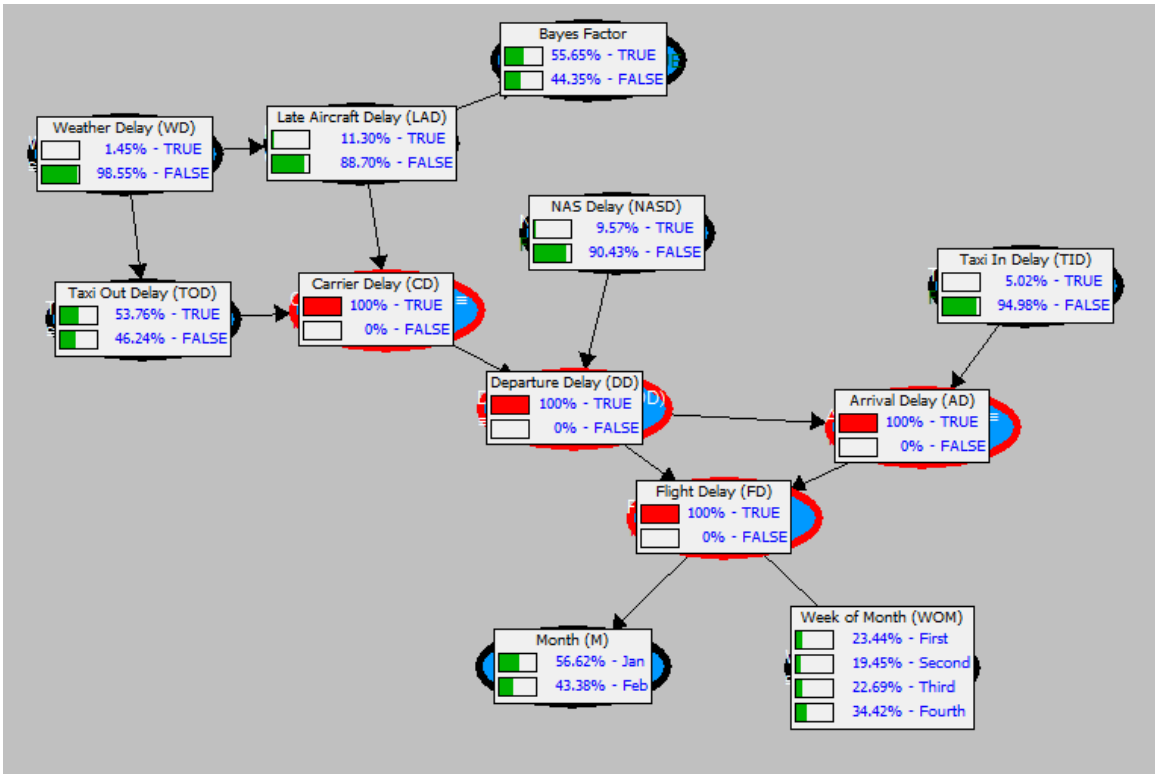

Figure 5-14. All System Fail.

Clearly, the second suggestion would not be possible in a real-world situation. The first suggestion demonstrates a reasonable method. Given a more detailed BN detailing the causes of a taxi out delay other than just weather, we can determine the exact causes of taxi out delay and begin to optimize the workflow from its practical real world event.

5.3. CASE STUDY & EVALUATION 2:

ELASTIC CROWDSOURCING WORKFLOW

In the previous example, we focused on the core Bayesian network features of the modeling approach, whereas, in this case, study we motivate the extensibility of our approach in another domain.

Description.

As a second example, we use the domain of crowdsourcing. Crowdsourcing is on online problem-solving paradigm that outsource tasks to large groups [148]. The crowdsourcing paradigm can be combined with traditional computing resources, forming hybrid service workflows that derive solutions that neither humans nor machines can solve alone. Such hybrid service workflows could involve large numbers of people with varying expertise, skills, interests, and incentives and varied computing resources. Also, computing services can have varied performances, which can be effected by attributes within the problem domain. For example, facial recognition performance is measured by comparing multiple algorithms on a single dataset. This type of analysis compares approaches for a particular task as exemplified by the dataset [149]. This leads to the choice of an algorithm being specific to the requirements of a given task. In crowdsourcing, humans create a profile of their expertise and are assigned task according to their ability. However, their performance in solving various task may not reflect their profile. A major challenge in the hybrid system is choosing the most efficient computing service for a given task. In this case study, we simulate a hybrid service workflow and describe our data-driven Bayesian approach to identify high performing computing services.

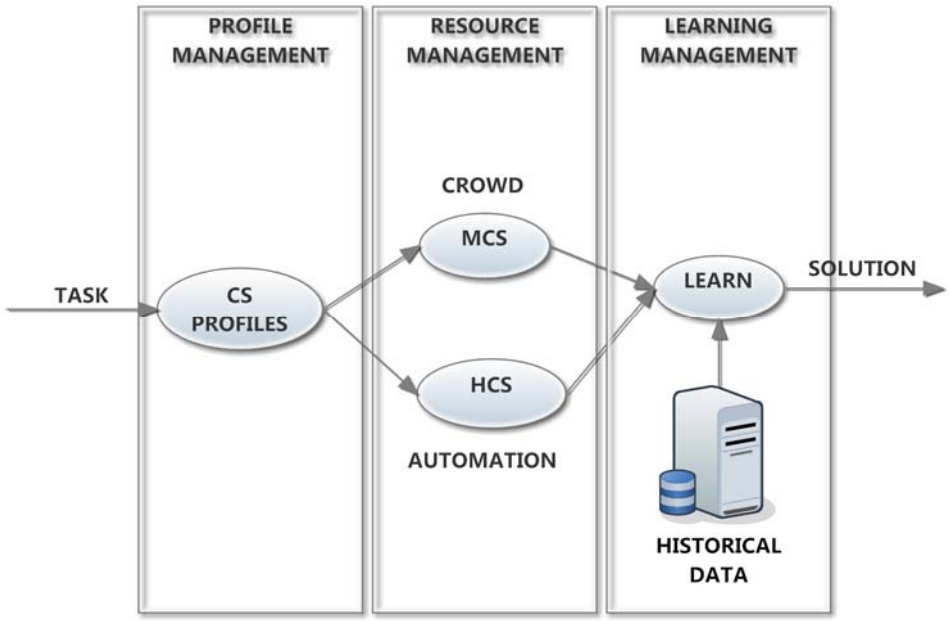

Figure 5-15. Hybrid Web Service Workflow adopted from [150]

Although many face recognition techniques have shown significant promise, robust face recognition is still difficult. There are at least three major challenges: illumination, pose, and recognition in outdoor imagery. Alternatively, while human perception has impressive facial recognition capabilities, the number and types of faces are not comparable to machines. In light of the advantages and disadvantages of the machine and human computing services in facial recognition, we demonstrate our methodology in an image recognition hybrid Web service workflow, Figure 5-15, as our running example. The hybrid Web service workflow has three conceptual levels. The profile management, which records the profiles of each CS. The resource management section, which assigns the task to each CS and finally the learning management section our proposed abstraction, which learns the best CS for given tasks.

Dataset.

Performing elicitation from the field is plagued with well-known biases as explained in the field of cognitive and computational psychology [151]. We capture some of these biases in

our hybrid workflow by using two generalized types of human services, each with different assumptions:

1. E1: An expert skewed towards getting the correct answer.
2. E2: An expert is randomly getting any answer.
3. NE1: A non-expert skewed towards getting the incorrect answer.
4. NE2: A non-expert skewed towards getting the correct answer.

We use a skew-normal distribution [152] to generate simulated data for the machine and human services to represent workflow results. The skew normal distribution is a continuous probability distribution that generalizes the normal distribution to allow for non-zero skewness. We adjust the sample size, location and scale parameters to affect the size of the sample, the mean, median and mode of the sample and finally the spread of the sample size distribution respectively. Table 5-7, shows the skewed results for the hybrid workflow.

We used 100 simulated data points for four MCS and four HCS. For HCS, we assumed two experts and two non-experts. We simulated good and bad results for expert 1 (E1) and expert (E2) respectively and did the same for the non-experts. We set high odds for our experts and low for non-experts, Table 5-8. We did this to demonstrate that our framework can update initial prior beliefs to predict accurately which CE will ultimately outperform the other by updating evidence.

Method.

To represent causal relationships between human and machine service outcomes we use a Dynamic Object Oriented Bayesian Network (DOOBN) [153]. A dynamic Bayesian network represents the evolution of some state space model through time, as a stochastic process, where parameters change their state dynamically.

Table 5-7. Simulated Data of Computing Elements Performance.

Computing Services	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	#false
Bayes	2	7	5	2	1	4	2	1	3	1	28
EBGM	0	18	6	18	7	11	11	22	4	25	122
LDA	27	16	11	30	0	45	5	13	42	38	227
PCA	12	15	2	10	4	10	17	2	12	6	90
E1	9	1	10	7	4	4	3	9	0	3	50
E2	79	86	76	58	99	50	73	63	63	94	741
NE1	98	93	86	93	86	83	83	85	97	80	884
NE2	3	16	17	11	19	6	1	12	19	4	108

Whereas, an Object Oriented Bayesian Network (OOBN) is a BN with additional features that make it reusable as part of a larger BN [154]. OOBNs are usually used when the network contains so many nodes that it becomes conceptually difficult to understand and when there are many similar repeated fragments.

Our DOOBN has three conceptual objects, Figure 5-15, a Machine Computing Service (MCS) object, and a Human Computing Service (HCS) object representing individual Bayesian Networks (BN) for each group of MCS and HCS respectively. Finally, the Solution object, another BN, consumes the output of the previous objects. The output consists of the updated odds of each computing service (CS) for each timestamp and hence this object is the decision object, since it performs decision analysis to choose the best computing service.

Each CS object represents a Bayesian network that has four conceptual levels, Fig. 2. The first level represent the prior belief obtained from previous research in the case of MCE and individual classification in the case of HCE. The second level, represent the posterior belief, our updated belief given new evidence.

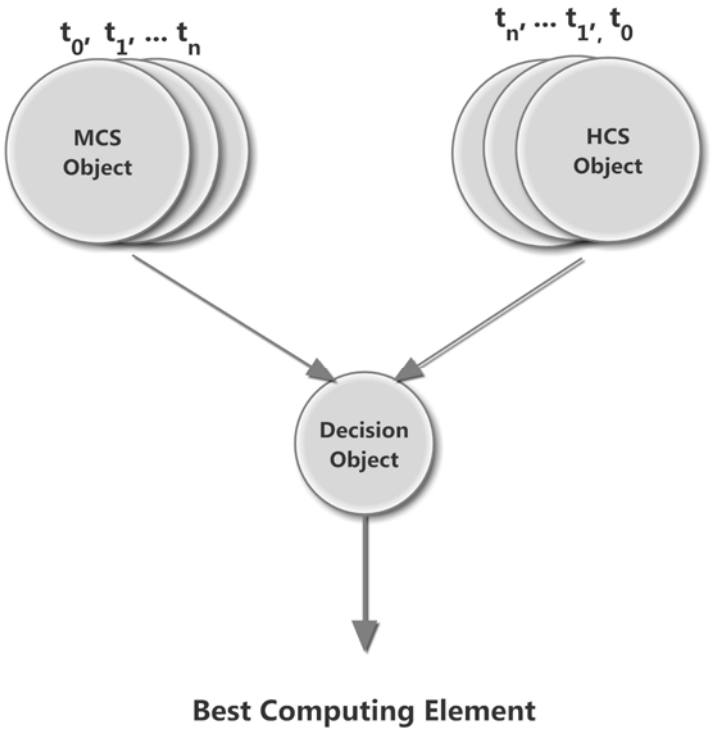

Figure 5-16. DOOBN Objects.

This level also serves as an output to new instances of the DOOBN. As timestamps progress, the odds from previous instances are updated to posterior odds and propagated as new prior odds to future instances. Updating odds in this way serve as a learning loop in the DOOBN that allows for more informed decisions based on ongoing data. The learning loop updates on each invocation of the DOOBN on each timestamp. In the third level, we compare the odds for each CS to determine which CS out-performs the other (new evidence is presented in each timestamp). This operation is performed on each MCS and HCS object separately to find the best CS in each OOBN object. The decision object compares the best MCS to the best HCS in the final level to determine the best CS.

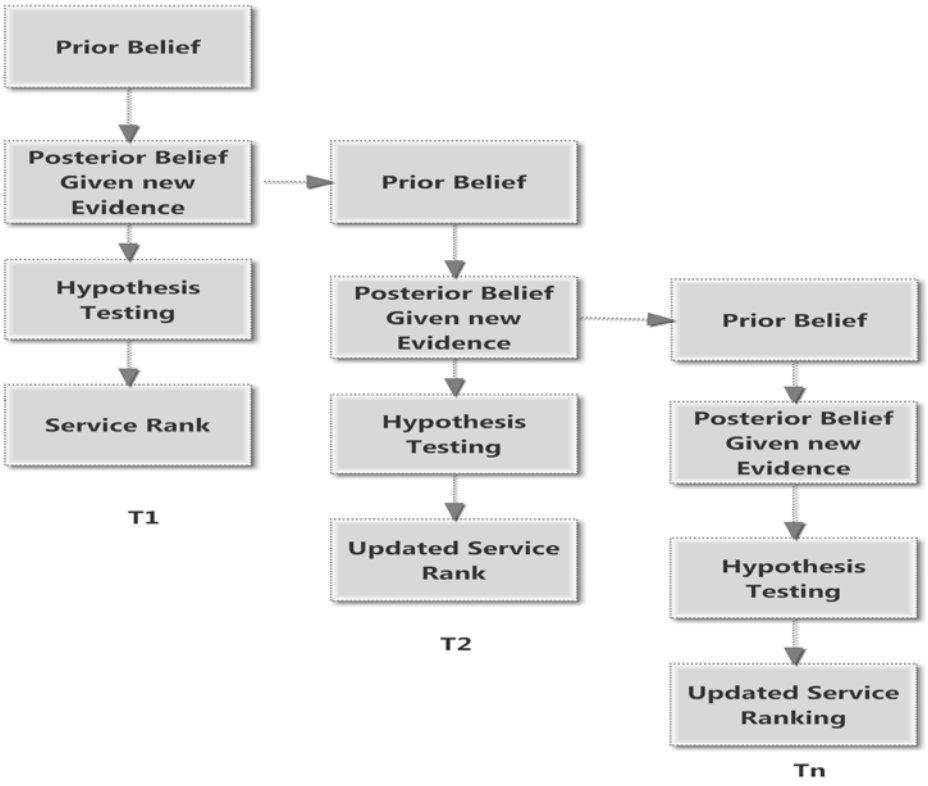

Figure 5-17. Conceptual Levels of CS Bayesian Networks.

To obtain prior expert knowledge for the MCS, we used baseline algorithms and evaluations from previous work from the Colorado State University's Evaluation of Face Recognition Algorithms [15]. It provides a standard set of established algorithms and experimental protocols. The goal of the evaluation is to provide a firm statistical basis for drawing conclusions about the relative performance of different algorithms. The evaluation also helps to explain better why evaluated facial recognition algorithms behave as they do. We use the following algorithms: Principal Component Analysis (PCA) [16], Linear Discriminant Analysis (LDA) [17], Elastic Bunch Graph Matching (EBGM) [18], and the Bayesian Intrapersonal/Extrapersonal Classifier (BIEC) [19]. Using the results from the evaluation above, we give higher ranked algorithms higher prior odds of success as opposed to lower ranks. Table 5-8 list out initial priors.

The Bernoulli likelihood function [8] and its conjugate Beta distribution [9] represent the likelihood and prior respectively. The Bernoulli distribution represents a success and failure hypothesis where a random variable takes value 1 (*true*) with success probability p and value 0 (*false*) with failure probability $q = 1 - p$. Whereas the beta distribution has parameters, $\alpha = mn$ and $\beta = (1 - m)n$, where m and n are the mean and sample size respectively. The Beta distribution,$B(\alpha, \beta)$ is ideally suited to represent odds, for example, $B(1,9)$ is equivalent to the odds of 1 in 10.

Table 5-8. Initial Priors for Computing Services..

Computing Element	Prior Odds	Description
Bayesian MAP	1:09	Excellent
EBGM Standard	2:08	Very Good
LDA Euclidian	3:07	Good
PCA Mah Cosine	4:06	Fair
Expert 1 (E1)	2:08	Very Good
Expert 2 (E2)	2:08	Very Good
Non-Expert 1 (NE2)	1:01	Poor
Non-Expert 1 (NE2)	1:01	Poor

Results and Evaluation.

To compute the hybrid service workflow Bayesian network, we used the AgenaRisk Professional modeling tool [155]. Figure 5-17 shows the model in AgenaRisk tool. Due to the size of the network, the level of details, that can be shown, is limited. The network has three distinct parts, left, right and bottom, each representing a Bayesian Network object. The left side shows the Bayesian network for the facial recognition algorithms or MCS, the right side the Bayesian network for the HCS and the bottom eight nodes is the Bayesian network for the decision object. The top four rows of both the MCS and HCS are the prior marginal nodes, and the next two rows are the posterior marginal nodes. Figure 5-18 shows the node mapping from different instances of the dynamic Bayesian network. For example, the

posterior nodes, from the Bayesian network instance in timestamp zero, are mapped to the prior nodes of the Bayesian network instance at timestamp 2. This mapping facilitates the propagation of odds, and hence beliefs as new evidence are updated for each timestamp.

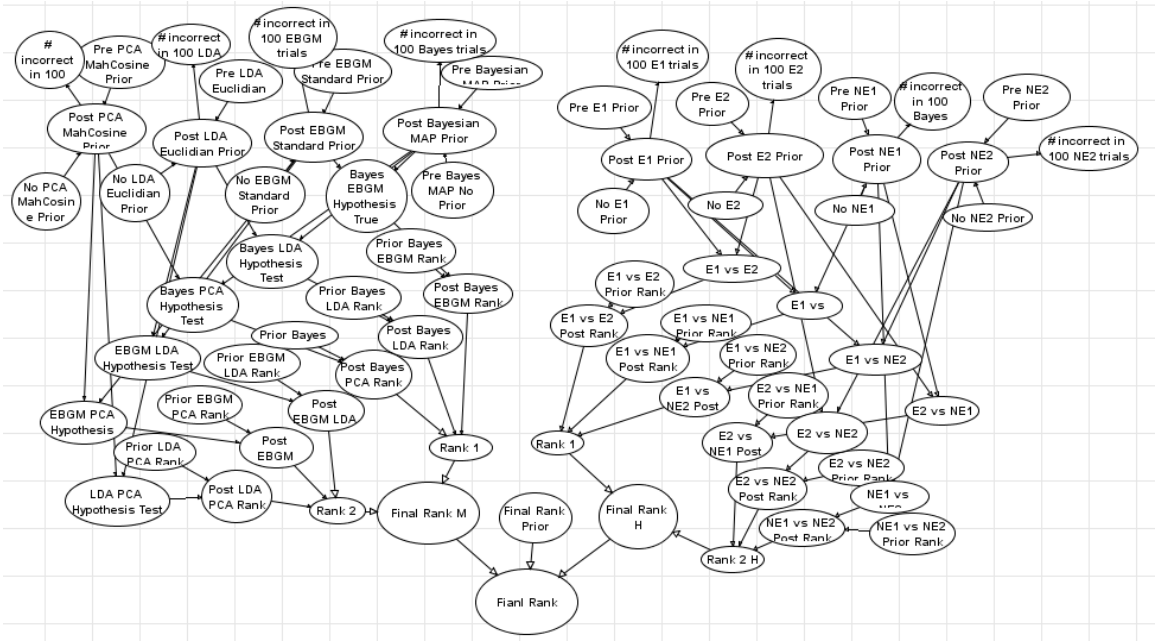

Figure 5-18. Hybrid Service Workflow BN.

The results of our experiments are twofold; First, we show the mean variance of the posterior marginal cumulated after running the hybrid services workflow BN for ten timestamps, Figure 5-19. A small variance demonstrates the mean of the posterior marginal results in minor changes from timestamp to timestamp, which translates to a high certainty of consistent performance. Subsequently, a large variance demonstrates a high certainty of inconsistent performance. To calculate the mean, we use α and β from the aforementioned beta distribution.

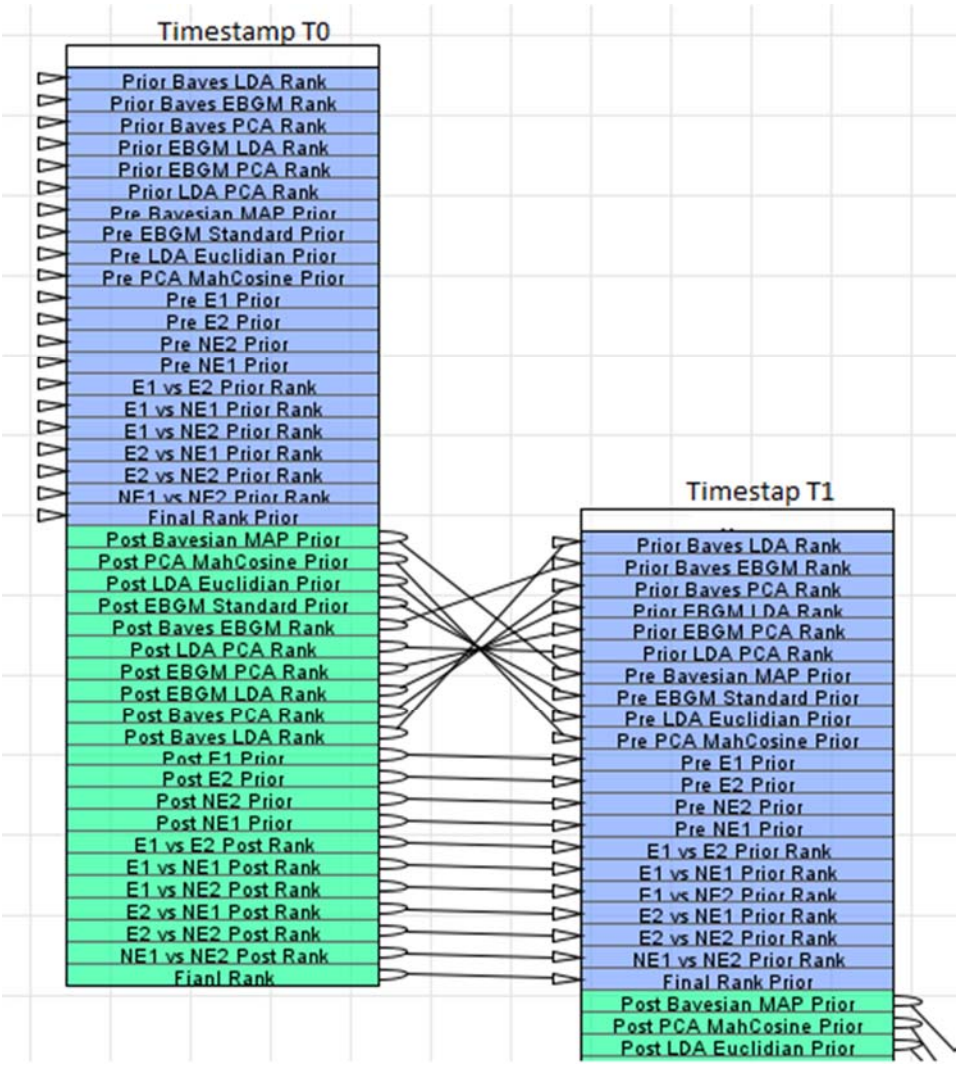

Figure 5-19. Node Mapping and Odds Propagation in Hybrid Service BN.

The four top performers for consistent performance listed from highest to lowest are BIEC, then E1, then PCA, and finally NE2. It should be noted that both PCA and NE2 had low initial odds but due to their performance we were able to detect their "unexpected" performance and update their odds accordingly.

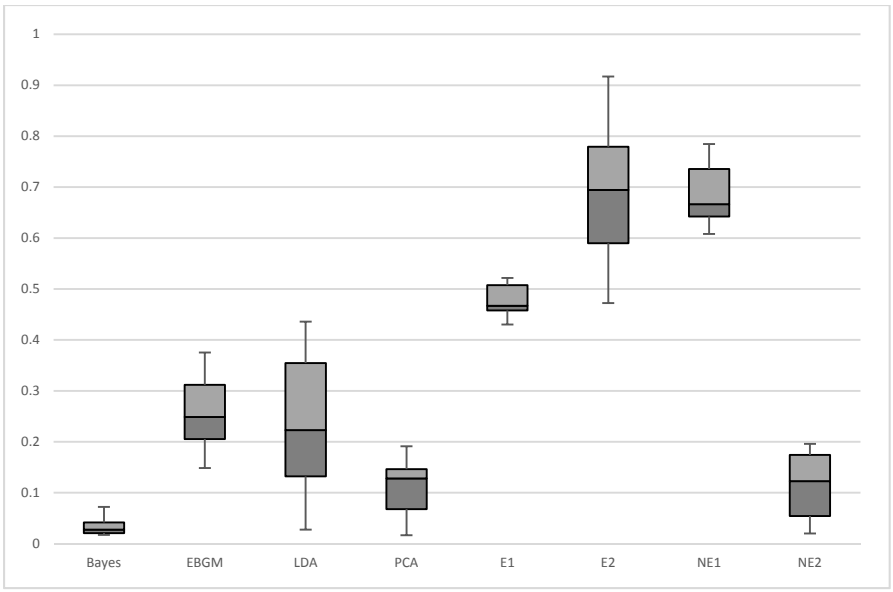

Figure 5-20. Mean-Variance for each Computing Element.

Secondly, we compare the posterior marginals for each CS to that of other CS, Figures 5-20 to 5-25, and likewise for the HCS, Figures 5-26 to 5-31. Each graph in Figures 5-20 to 5-31 represent a comparison of the performance of two CS where the x-axis represent timestamps and the y-axis posterior marginal. A CS with has a consistent higher posterior marginal than the other across timestamps is considered to be a higher performer.

The results show the BIEC algorithm, (Labeled as Bayes in the Figures), outperformed the other algorithms. We anticipated this outcome since the BIEC algorithm was initially ranked with a high prior marginal. Also, from the simulated results, BIEC had the fewest false responses. However, when we analyze the performance of the PCA algorithm, which was initially ranked the lowest among the MCS, it outperformed the LDA algorithm and matched the performance on average of the EBGM algorithm. Both the LDA and EBGM were initially ranked higher. However our model was able to adjust the rankings based on data-driven evidence.

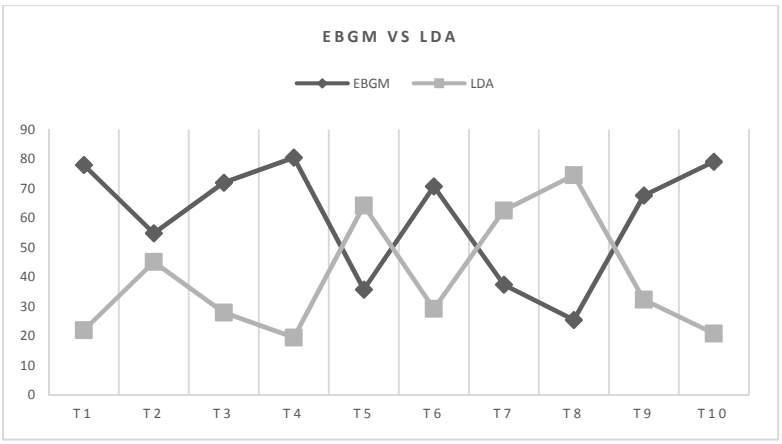

Figure 5-21. EBGM vs. LDA

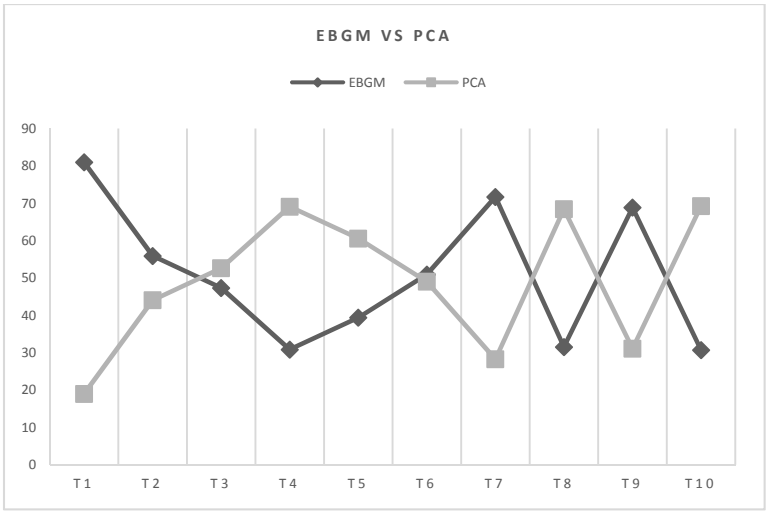

Figure 5-22. EBGM vs. PCA

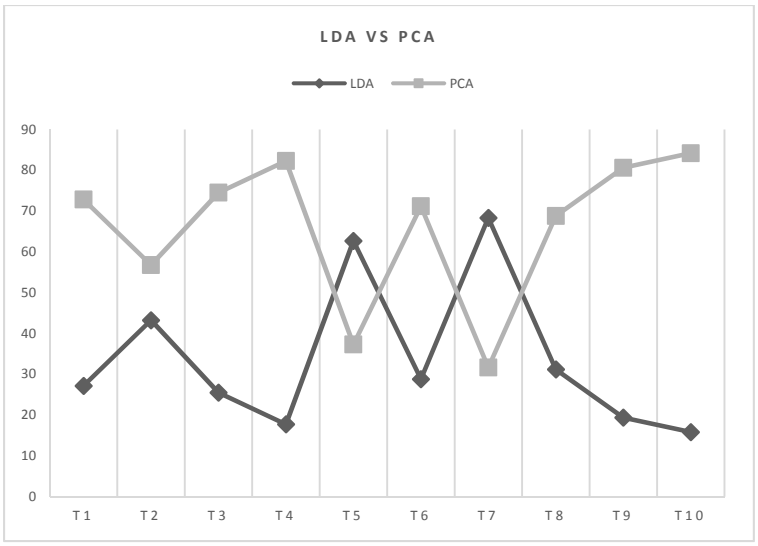

Figure 5-23. LDA vs. PCA

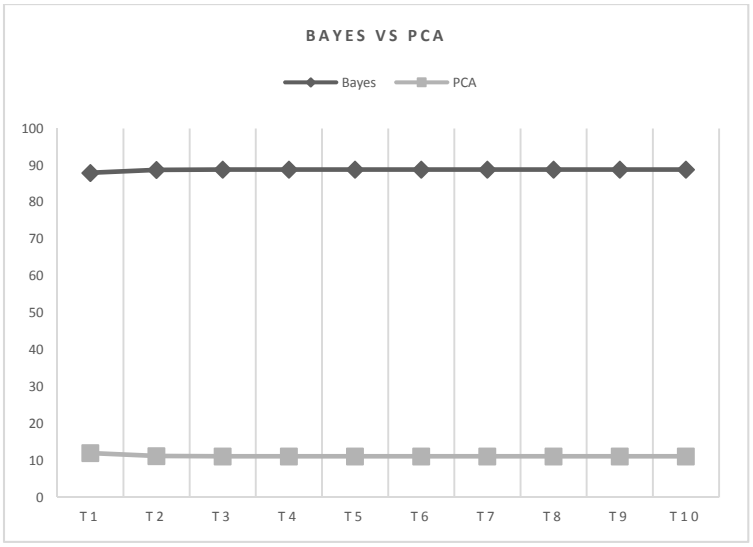

Figure 5-24. Bayes vs. PCA

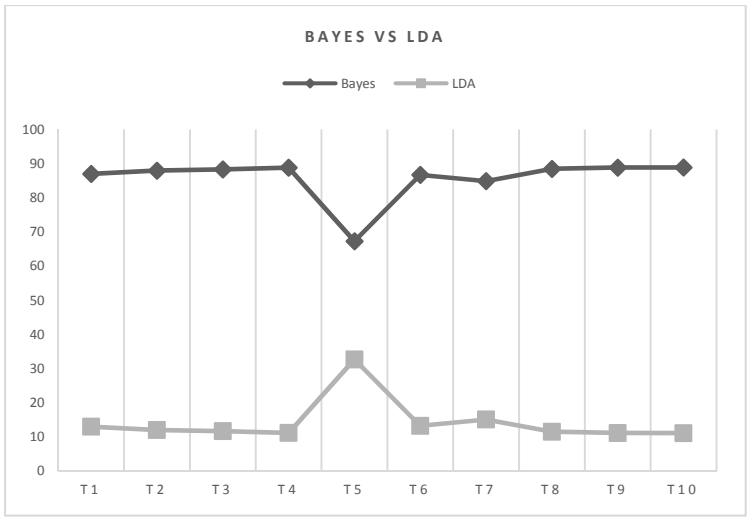

Figure 5-25. Bayes vs. LDA

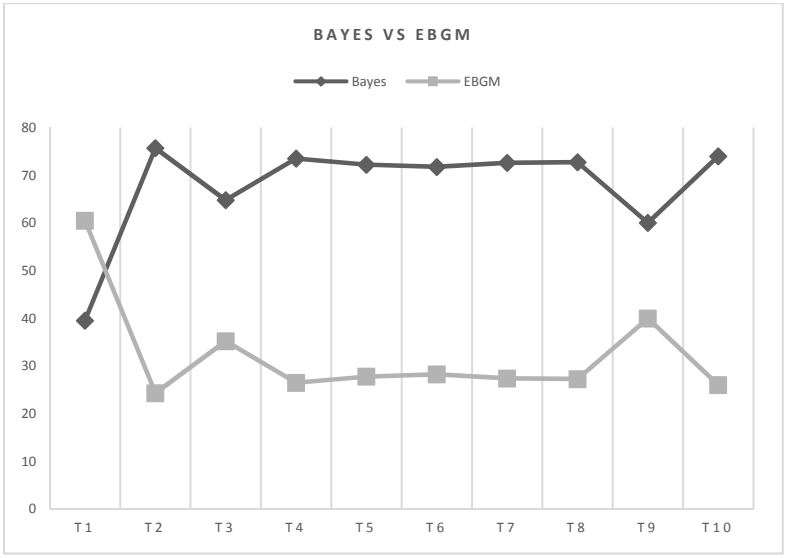

Figure 5-26. Bayes vs. EBGM

For the HCS, E1 and NE2 were initially ranked the highest and lowest respectively. As anticipated, E1 performed better than the other experts did. However, NE2 performed better than E2 and NE1. Similarly, our method was able to detect this due to data-driven evidence.

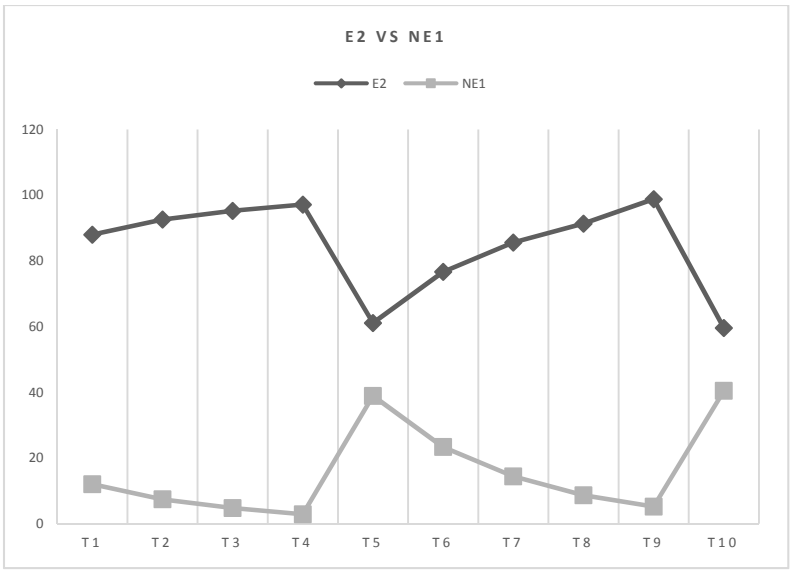

Figure 5-27. E2 vs. NE1

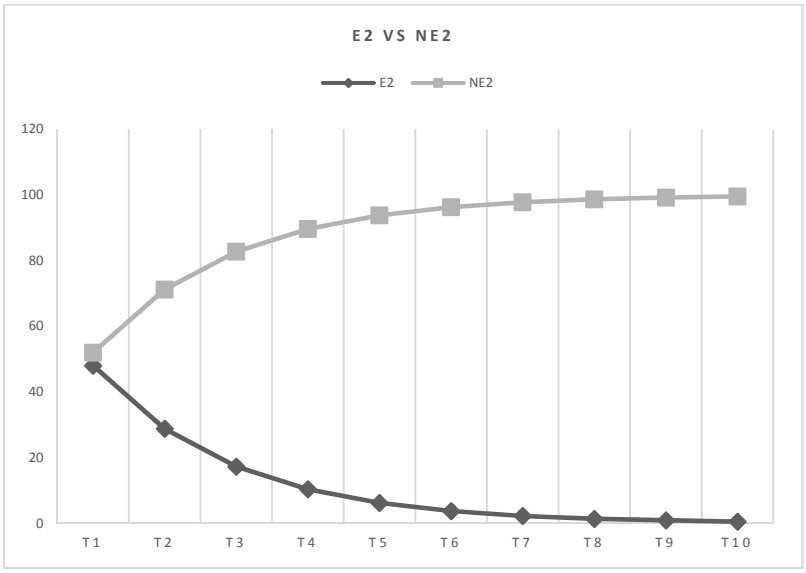

Figure 5-28. E2 vs. NE2

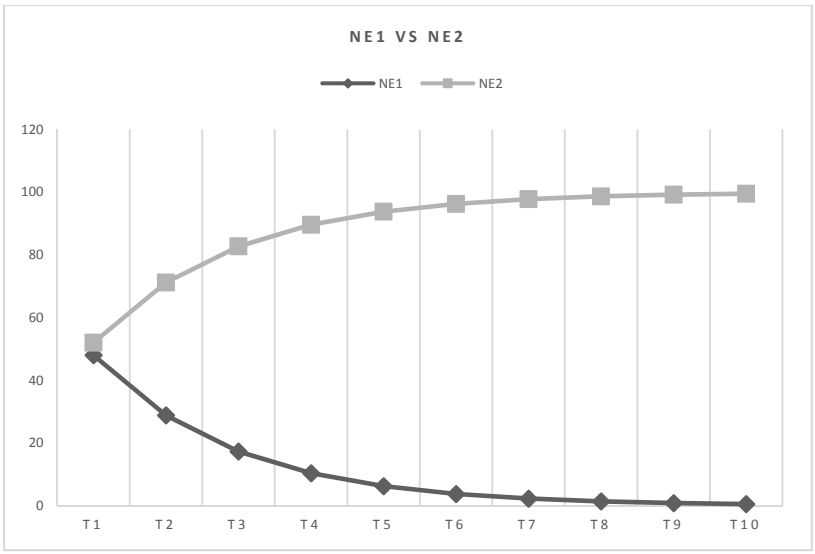

Figure 5-29. NE1 vs. NE2

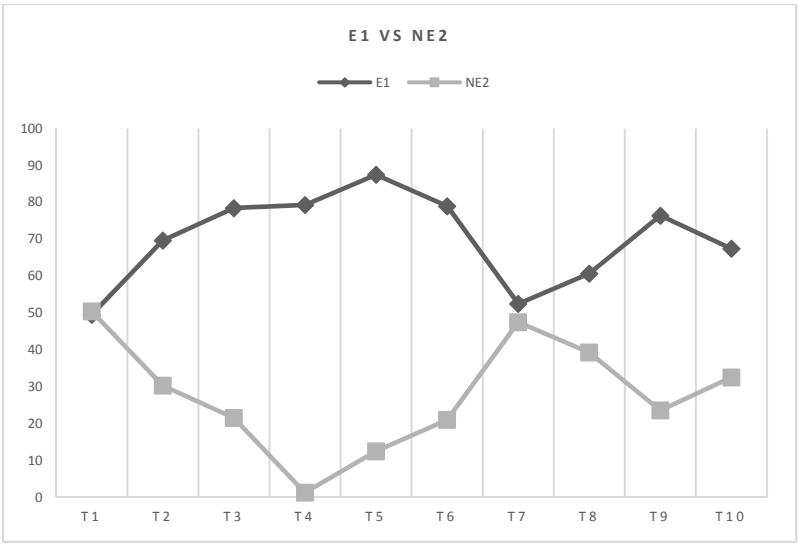

Figure 5-30. E1 vs. NE2

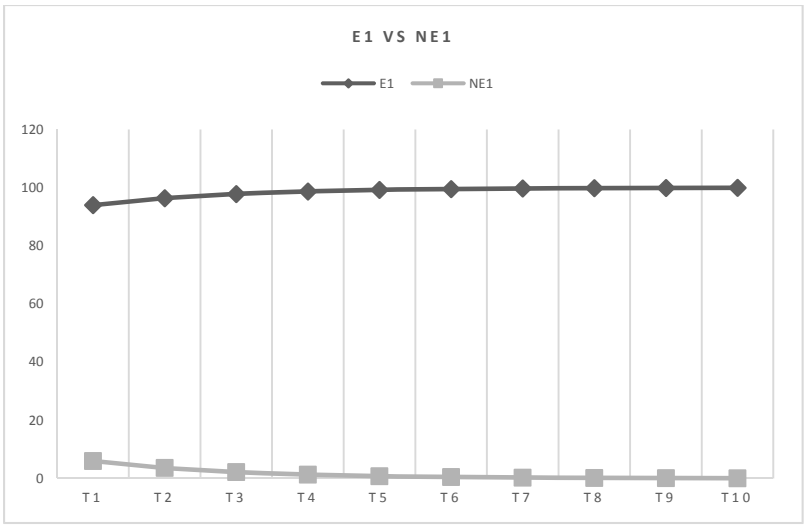

Figure 5-31. E1 vs. NE1

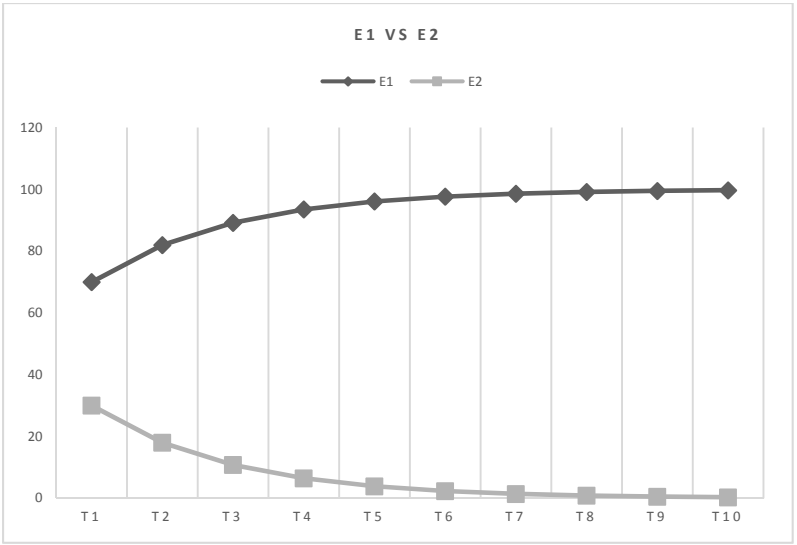

Figure 5-32. E1 vs. E2

5.4. DISCUSSION

This dissertation proposes two contributions: a principled methodology to capture workflow trends and an approach for decision support, prediction and workflow optimizations.

As a first step, we propose a workflow semantics approach that serves to label activities in the BPEL process according to its data transitions. For example, we postulate that there are variables within the workflow, which we labeled as defaulted variables that dynamically change during the BPEL execution. As a result, these variables play an important role in the output of the workflow. We further postulate that these changes, along with user trends, are captured within the data generated by the workflow. To capture the trends of both machine and users, we use a Bayesian network that provides an inferential view of the workflow data. In Chapter 5, we demonstrated the steps to convert workflow data and analyze the information from the perspective of a BN. Our results show that a Bayesian network is a principled approach to represent a workflow. This is a direct result of its graphical and probabilistic attribute. These attributes allowed us to learn the network probabilistic values directly from data. Evaluate the resulting network to ensure it reflects the likelihood of the data. Finally query the network to provide decision support and make predictions.

In general, our approach of using Bayesian networks demonstrated an effective method to model Web service workflows using Bayesian networks. In particular, a Bayesian network provides a *consistent theoretically solid mechanism for processing uncertain information*. Probability theory provides a consistent calculus for uncertain inference, meaning that the output of the workflow is always unambiguous. Given the input, all the alternative mechanisms for computing the output with the help of a Bayesian network model produce exactly the same answer.

Smoothness properties: Bayesian network models have been found to be very robust in the sense that small alterations in the model do not affect the performance of the system

dramatically. This means that maintaining and updating existing models is easy since the functioning of the system changes smoothly as the model is being modified. For sales and marketing workflow, this is a crucial characteristic, as these systems need to be able to follow market changes rapidly without complex and time-consuming re-modeling.

Flexible applicability: Bayesian networks model the problem domain as a whole by constructing a joint probability distribution over different combinations of the domain variables. This means that the same Bayesian network model can be used for solving both discriminative tasks (classification) and regression problems (configuration problems and prediction). Besides predictive purposes, Bayesian networks can also be used for explorative data mining tasks by examining the conditional distributions, dependencies and correlations found by the modeling process.

A theoretical framework for handling expert knowledge: In Bayesian modeling, expert domain knowledge can be coded as prior distributions, prior meaning that the probability distributions are defined before and independently of processing any possible sample data. This allows for combining expert knowledge with statistical data in a very practical way. Using suitable prior distributions, the priors can be given a semantically clear explanation in terms of the data (expert knowledge can be interpreted as an unseen dataset of the same form as the training data). This means that the experts will also be able to give an estimate of the weight or importance of their prior knowledge, compared to the available data.

A clear semantic interpretation of the model parameters: Unlike neural network models, which usually appear to the user as a *black box*, all the parameters in Bayesian networks have an understandable semantic interpretation. It is for this reason that Bayesian

networks can be constructed directly by using expert domain knowledge, without a time-consuming learning process. On the other hand, if machine learning techniques are used (with or without expert knowledge) for constructing Bayesian network models from sample data, the resulting model can be analyzed and explained in terms that are understandable to domain experts.

Different variable types: Probabilistic models can handle several different type variables at the same time whereas many alternative model technologies have been designed for some single specific type of variables (continuous, discrete, etc.). For these alternatives, working with several variable types requires some transformation operations, which in some cases may be the cause for unexpected results. From the probabilistic point of view, all the basic entities are distributions, which means that all the different variable types fall elegantly in the same unifying framework.

However, there are a few disadvantages worth noting. Firstly, the *discretization of continuous variables*. One can always discretize the continuous variables by partitioning their domain into some finite number of subsets, and by doing so transform the model to a discrete one. Discretization of continuous chance variables is equivalent to approximating a probability density function with mixtures of uniform distributions. However, discretization with a small number of states can lead to poor accuracy while discretization with a large number of states can lead to excessive computational effort.

Collecting and structuring expert knowledge. While Bayesian models are a useful way to model expert knowledge, it may prove difficult to solicit and capture knowledge from the experts in a form that can be converted into probability distributions. Firstly, many workflow domain experts may or may not be familiar with the sophistication of their data.

Domain experts, that understand the nuances of the data, may find it exceedingly difficult to provide any numbers without relying on the data. Secondly, domain experts may be used to classical statistical analyzes and feel uncertain when trying to think about their knowledge in terms of distributions or odds rather than point estimates and confidence intervals.

Available software packages. There are a number of commercial and free software packages are available for developing BN based models. Some popular ones are Analytica [156]; Netica [157]; Hugin [146]; AgenaRisk [155]and GeNie [158]. Information about some different software packages available for BNs is provided by Murphy [159]. In this dissertation we used SamIam , AgenaRisk, and various R packages. Each package has its strengths and disadvantages. For example, each package may implement a specific inference algorithm that has certain strengths and weakness such as dealing with either discrete or continues or both types of nodes. They may or may not perform either structural, parameter or both types of learning. In the workflow domain, we have found that the type of data is quite versatile and dynamic, and hence a suitable free package that can adequately and universally support research with the type of data this domain provides was not available.

Chapter 6
CONCLUSION AND FUTURE WORK

6.1. CONCLUSION

An increasing number of companies and organizations only implement their core business and outsource other application services. Thus, the ability to efficiently and effectively select and integrate inter-organizational and heterogeneous services is a significant step towards the development of Web service workflow applications. However, there is a strong requirement for methods that allow Web service workflows to be responsive to changes in a dynamic and uncertain environment to bridge the information gap caused by workflow data silos. In this dissertation, these two challenges are addressed by utilizing a probabilistic graphical network framework to provide data-derived inferential visibility of workflow semantics. The novel approach introduced a methodology for the transformation of BPEL processes to a normative expert system by providing an extendable and customizable abstraction for the BPEL process, which consisted of a knowledge base and an inference engine. In particular, a Bayesian Network served as the knowledge base to represent workflow semantics in a structured way. To support reasoning about events and decisions in the workflow's domain, we used graphical and probabilistic theory for the inference engine.

A Web service workflow represents a broad and complex domain with many random variables/messages. A workflow semantics was defined to represent each type of data transition with the workflow. The Bayesian network provided a graphical representation to encode dependence and independence relations among random workflow variables into a compact joint probability distribution. Conditional probability distributions within the

network specified the casual relations between Web services messages and was represented as a multiplicative factorization of the joint probability distribution known as the chain rule. The joint and conditional probability distribution was used to answer some probabilistic query often in the form of computing the probability of a random variable given some evidence. The qualitative component of the Bayesian network encoded the set of (conditional) dependence and independence statements among the set of random workflow variables and the quantitative component specified the strengths of dependence relations using probability theory.

The qualitative and quantities components provided efficient reasoning support under uncertainty in a given workflow domain and identified (optimal) decisions in the light of new knowledge to address workflow responsiveness and the information gap. Graphical and probabilistic theories were applied to both the qualitative and quantitative components respectively to compute the posterior marginal of workflow variables. The posterior marginal allowed for the efficient reasoning under uncertainty in a given workflow domain. To identified (optimal) decisions in the light of new knowledge, historical domain-specific functional data was leveraged to capture an initial snapshot of beliefs of the underlying phenomenon of the workflow. Ongoing run-time data was used to update beliefs providing run-time trends and just-in-time decision support.

6.2. FUTURE WORK

Due to limitations of the proposed work and to explore new directions, the following future work will be studied.

6.2.1. A PUBLICLY AVAILABLE HYBRID FRAMEWORK

Although many Bayesian Network (BN) applications are now in everyday use, BNs have not yet achieved mainstream penetration. The focus is mainly on the algorithm and theory design as opposed to implementation of practical, real-world problem solving and model building. Our goal is to provide an executable framework for the research community to test and develop Bayesian models that reflect real-world workflow situations.

From the perspective of data, Bayesian networks could be divided into three categories: discrete, continuous and hybrid. Discrete and continuous Bayesian networks contain only discrete and continuous random variables respectively, whereas mixed of hybrid contains both discrete and continuous data. In this work, we used a discrete Bayesian network. However, discrete networks are sometimes inadequate, since many important workflow domains have continuous attributes as well as discrete ones. An example of a Web service domain with continuous and discrete variables is a capital investment project where the outcome of an uncertain continuous variable, such as cash flows or customer demand, affects the probability that a business will invest (a discrete variable).

One can always discretize the continuous variables by partitioning their domain into some finite number of subsets, and by doing so transform the model into a discrete one. Discretization of continuous chance variables is equivalent to approximating a probability density function with mixtures of uniform distributions. However, discretization with a small number of states can lead to poor accuracy while discretization with a large number of states can result in excessive computational effort.

Discretization of continuous distributions can allow approximate inference in a hybrid Bayesian network without limitations on relationships among continuous and

discrete variables. On a very high level, the work on the inference problem can be divided into two main classes: exact inference and approximate inference. Exact inference algorithms are designed to give an exact answer to the probabilistic query. Approximate inference is intended to give an approximate answer to the probabilistic query, with the understanding that often the exact probability is not crucial. When exact inference is needed, then discretization is not an option

A commonly used type of hybrid Bayesian network is the conditional linear Gaussian (CLG) model [127]. In CLG models, the distribution of a continuous variable is a linear Gaussian function of its continuous parents. One limitation of CLG models is that discrete nodes cannot have continuous parents. Recent work introduced the Augmented CLG networks where discrete variables may depend on continuous parents [160]. However, there is no known publicly available implementation that can be used to implement Augmented CLG in the research community. Moving forward we propose creating an open source project for the Augmented CLG algorithm in the R programming language for the publicly avaibale Comprehensive R Archive Network (CRAN) [161] library.

REFERENCES

[1] R. S. Wurman, *Information Anxiety*, 1st edition. New York: Doubleday, 1989.

[2] W. Binder and H. Schuldt, "5th Workshop on Enhanced Web Service Technologies (WEWST 2010)," in *Proceedings of the 5th International Workshop on Enhanced Web Service Technologies*, New York, NY, USA, 2010, pp. 1–2.

[3] J. Ng, "The Personal Web," M. Chignell, J. R. Cordy, R. Kealey, J. Ng, and Y. Yesha, Eds. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 1–10.

[4] J. C. Mogul, "Clarifying the Fundamentals of HTTP," in *Proceedings of the 11th International Conference on World Wide Web*, New York, NY, USA, 2002, pp. 25–36.

[5] E. Zadok, J. M. Andersen, I. Badulescu, and J. Nieh, *A Comparison of Thin-Client Computing Architectures*. 2000.

[6] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing," *Commun. ACM*, vol. 46, pp. 25–28, 2003.

[7] "2013 IBM Annual Report." [Online]. Available: http://www.ibm.com/annualreport/2013/. [Accessed: 05-Jul-2015].

[8] J. Bishop, "The Data Deluge – How Software Engineering Can Help," in *Proceedings of the 2010 Asia Pacific Software Engineering Conference*, Washington, DC, USA, 2010, p. 2–.

[9] "Welcome to Apache™ Hadoop®!" [Online]. Available: https://hadoop.apache.org/. [Accessed: 10-May-2015].

[10] "HBase – Apache HBase™ Home." [Online]. Available: http://hbase.apache.org/. [Accessed: 10-May-2015].

[11] "MongoDB." [Online]. Available: https://www.mongodb.org/. [Accessed: 10-May-2015].

[12] "Differentiate. Don't Become Commoditized. - Company Founder." [Online]. Available: http://www.companyfounder.com/2011/09/differentiate-don%E2%80%99t-become-commoditized/. [Accessed: 15-Jun-2015].

[13] *Strength in Numbers: How Does Data-Driven Decisionmaking Affect Firm Performance?* .

[14] M. Bichier and K.-J. Lin, "Service-oriented computing," *Computer*, vol. 39, no. 3, pp. 99–101, Mar. 2006.

[15] "OASIS Web Services Business Process Execution Language (WSBPEL) TC | OASIS." [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel. [Accessed: 20-Feb-2014].

[16] M. B. Blake and M. N. Huhns, "Web-Scale Workflow: Integrating Distributed Services," *IEEE Internet Computing*, vol. 12, no. 1, pp. 55–59, 2008.

[17] J. Barjis, "The Importance of Business Process Modeling in Software Systems Design," *Sci Comput Program*, vol. 71, no. 1, pp. 73–87, Mar. 2008.

[18] "Extensible Markup Language (XML)." [Online]. Available: http://www.w3.org/XML/. [Accessed: 25-Apr-2015].

[19] "Web Service Definition Language (WSDL)." [Online]. Available: http://www.w3.org/TR/wsdl. [Accessed: 25-Apr-2015].

[20] M. Reichert and S. Rinderle, "On design principles for realizing adaptive service flows with BPEL," in *In Proceedings EMISA'06, Hamburg (Lecture Notes in Informatics (LNI), P-95*, 2006, pp. 133–146.

[21] A. Martens, "Analyzing Web Service based Business Processes," in *In Proc. Int'l Conf. on Fundamental Appr. to Software Eng. (FASE'05), LNCS 3442*, 2005.

[22] C. Ouyang, E. Verbeek, W. M. P. van der Aalst, S. Breutel, M. Dumas, and A. H. M. ter Hofstede, "Formal Semantics and Analysis of Control Flow in WS-BPEL," *Sci Comput Program*, vol. 67, no. 2–3, pp. 162–198, Jul. 2007.

[23] M. U. Reichert, S. B. Rinderle, and P. Dadam, "On the Modeling of Correct Service Flows with BPEL4WS," presented at the Proceedings Workshop Informationssysteme im E-Business und E-Government (EMISA 2004), Bonn, 2004, vol. 56, pp. 117–128.

[24] "2012 Digital 100 - Business Insider." [Online]. Available: http://www.businessinsider.com/2012-digital-100?op=1. [Accessed: 10-May-2015].

[25] "NOSQL Databases." [Online]. Available: http://nosql-database.org/. [Accessed: 18-Jun-2015].

[26] C. M. Judd and G. H. McClelland, *Data Analysis: A Model-Comparison Approach*. San Diego: Harcourt College Pub, 1989.

[27] P. Tambe, “Big Data Investment, Skills, and Firm Value,” Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 2294077, Jan. 2014.

[28] W. M. Bolstad, *Introduction to Bayesian Statistics, 2nd Edition*, 2nd edition. Hoboken, N.J: Wiley-Interscience, 2007.

[29] B. Srivastava and J. Koehler, “Web Service Composition - Current Solutions and Open Problems,” in *In: ICAPS 2003 Workshop on Planning for Web Services*, 2003, pp. 28–35.

[30] X. Liu, Y. Hui, W. Sun, and H. Liang, “Towards Service Composition Based on Mashup,” in *2007 IEEE Congress on Services*, 2007, pp. 332–339.

[31] N. Milanovic and M. Malek, “Current solutions for Web service composition,” *IEEE Internet Comput.*, vol. 8, no. 6, pp. 51–59, Nov. 2004.

[32] W. M. P. van der Aalst, *Challenges in Business Process Management: Verification of business processes using Petri nets*.

[33] X. Fu, T. Bultan, and J. Su, “Analysis of Interacting BPEL Web Services,” 2004, pp. 621–630.

[34] L. Bordeaux and G. Salaün, “Using Process Algebra for Web Services: Early Results and Perspectives,” in *Technologies for E-Services*, M.-C. Shan, U. Dayal, and M. Hsu, Eds. Springer Berlin Heidelberg, 2005, pp. 54–68.

[35] Z. Duan, A. Bernstein, P. Lewis, and S. Lu, “Semantics based verification and synthesis of BPEL4WS abstract processes,” in *IEEE International Conference on Web Services, 2004. Proceedings*, 2004, pp. 734–737.

[36] A. Wombacher, P. Fankhauser, and E. Neuhold, “Transforming BPEL into annotated deterministic finite state automata for service discovery,” in *IEEE International Conference on Web Services, 2004. Proceedings*, 2004, pp. 316–323.

[37] “CiteSeerX — Citation Query Corporation FlowPath Functional Specification.” [Online]. Available: http://citeseer.uark.edu:8080/citeseerx/showciting;jsessionid=4F7FAEECB6B9027A493F32B7894D8FAC?cid=3469099. [Accessed: 15-May-2015].

[38] C. Ellis, K. Keddara, and G. Rozenberg, “Dynamic Change Within Workflow Systems,” in *Proceedings of Conference on Organizational Computing Systems*, New York, NY, USA, 1995, pp. 10–21.

[39] M. Reichert and P. Dadam, “ADEPT flex - Supporting Dynamic Changes of Workflows Without Loosing Control,” *J. Intell. Inf. Syst.*, vol. 10, pp. 93–129, 1998.

[40] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M.-C. Shan, "Adaptive and Dynamic Service Composition in eFlow," in *Proceedings of the 12th International Conference on Advanced Information Systems Engineering*, London, UK, UK, 2000, pp. 13–31.

[41] S. A. McIlraith, T. C. Son, and H. Zeng, "Semantic Web services," *IEEE Intell. Syst.*, vol. 16, no. 2, pp. 46–53, Mar. 2001.

[42] "OWL-S: Semantic Markup for Web Services." [Online]. Available: http://www.w3.org/Submission/OWL-S/. [Accessed: 30-Jun-2015].

[43] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker, "Description Logic Programs: Combining Logic Programs with Description Logic," in *Proceedings of the 12th International Conference on World Wide Web*, New York, NY, USA, 2003, pp. 48–57.

[44] J. Mccarthy and P. J. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence," in *Machine Intelligence*, 1969, pp. 463–502.

[45] S. Narayanan and S. A. McIlraith, "Simulation, Verification and Automated Composition of Web Services," in *Proceedings of the 11th International Conference on World Wide Web*, New York, NY, USA, 2002, pp. 77–88.

[46] S. Mcilraith, "Adapting Golog for Composition of Semantic Web Services," 2002, pp. 482–493.

[47] B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid, "Composing Web Services on the Semantic Web," *VLDB J.*, vol. 12, no. 4, pp. 333–351, Nov. 2003.

[48] I. Vlahavas and D. Vrakas, Eds., *Intelligent Techniques for Planning*. Hershey, PA: Idea Group Pub, 2004.

[49] J. Koehler, G. Tirenni, and S. Kumaran, "From business process model to consistent implementation: a case for formal verification methods," in *Enterprise Distributed Object Computing Conference, 2002. EDOC '02. Proceedings. Sixth International*, 2002, pp. 96–106.

[50] W. Reisig and G. Rozenberg, Eds., *Lectures on Petri Nets II: Applications, Advances in Petri Nets, the Volumes Are Based on the Advanced Course on Petri Nets*. London, UK, UK: Springer-Verlag, 1998.

[51] K. Schmidt and C. Stahl, "A Petri net Semantic for BPEL4WS - Validation and Application," in *Universität Paderborn*, 2004, pp. 1–6.

[52] W. M. P. van der Aalst, “Three Good Reasons for Using a Petri-Net-Based Workflow Management System,” in *Information and Process Integration in Enterprises*, T. Wakayama, S. Kannapan, C. M. Khoong, S. Navathe, and J. Yates, Eds. Springer US, 1998, pp. 161–182.

[53] J. A. Bergstra, *Handbook of Process Algebra*. New York, NY, USA: Elsevier Science Inc., 2001.

[54] R. Milner, *Communication and Concurrency*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.

[55] T. Bolognesi and E. Brinksma, “Introduction to the ISO Specification Language LOTOS,” *Comput Netw ISDN Syst*, vol. 14, no. 1, pp. 25–59, Mar. 1987.

[56] R. Milner, *Communicating and Mobile Systems: The &pgr;-calculus*. New York, NY, USA: Cambridge University Press, 1999.

[57] J. Magee and J. Kramer, *Concurrency: State Models & Java Programs*. New York, NY, USA: John Wiley & Sons, Inc., 1999.

[58] H. Foster, S. Uchitel, J. Magee, and J. Kramer, “Model-based verification of Web service compositions,” in *18th IEEE International Conference on Automated Software Engineering, 2003. Proceedings*, 2003, pp. 152–161.

[59] H. Foster, S. Uchitel, J. Magee, and J. Kramer, “Compatibility verification for Web service choreography,” in *IEEE International Conference on Web Services, 2004. Proceedings*, 2004, pp. 738–741.

[60] H. Foster, S. Uchitel, J. Magee, J. Kramer, and M. Hu, “Using a rigorous approach for engineering Web service compositions: a case study,” in *2005 IEEE International Conference on Services Computing*, 2005, vol. 1, pp. 217–224 vol.1.

[61] A. Ferrara, “Web Services: A Process Algebra Approach,” in *Proceedings of the 2Nd International Conference on Service Oriented Computing*, New York, NY, USA, 2004, pp. 242–251.

[62] G. Salaün, A. Ferrara, and A. Chirichiello, “Negotiation Among Web Services Using LOTOS/CADP,” in *Web Services*, L.-J. (LJ) Zhang and M. Jeckle, Eds. Springer Berlin Heidelberg, 2004, pp. 198–212.

[63] J.-C. Fernandez, H. Garavel, A. Kerbrat, L. Mounier, R. Mateescu, and M. Sighireanu, *CADP - A Protocol Validation and Verification Toolbox*. 1996.

[64] E. Best, R. Devillers, and M. Koutny, “Petri nets, process algebras and concurrent programming languages,” in *Lectures on Petri Nets II: Applications*, W. Reisig and G. Rozenberg, Eds. Springer Berlin Heidelberg, 1998, pp. 1–84.

[65] E. Borger and R. F. Stark, *Abstract State Machines: A Method for High-Level System Design and Analysis*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003.

[66] R. Farahbod, U. Glässer, and M. Vajihollahi, “An Abstract Machine Architecture for Web Service Based Business Process Management,” in *Business Process Management Workshops*, C. J. Bussler and A. Haller, Eds. Springer Berlin Heidelberg, 2006, pp. 144–157.

[67] D. Fahland and W. Reisig, “ASM-based semantics for BPEL: The Negative Control Flow,” in *Proc. 12th International Workshop on Abstract State Machines*, 2005, pp. 131–151.

[68] W. Reisig, “Modeling- and Analysis Techniques for Web Services and Business Processes,” in *Formal Methods for Open Object-Based Distributed Systems*, M. Steffen and G. Zavattaro, Eds. Springer Berlin Heidelberg, 2005, pp. 243–258.

[69] W. van der Aalst, T. Weijters, and L. Maruster, “Workflow mining: discovering process models from event logs,” *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1128–1142, Sep. 2004.

[70] L. Wen, J. Wang, and J. Sun, “Detecting Implicit Dependencies Between Tasks from Event Logs,” in *Frontiers of WWW Research and Development - APWeb 2006*, X. Zhou, J. Li, H. T. Shen, M. Kitsuregawa, and Y. Zhang, Eds. Springer Berlin Heidelberg, 2006, pp. 591–603.

[71] A. J. M. M. Weijters and A. K. A. D. Medeiros, *Process Mining with the HeuristicsMiner Algorithm*. .

[72] G. Greco, A. Guzzo, L. Pontieri, and D. Saccà, “Mining Expressive Process Models by Clustering Workflow Traces,” in *Advances in Knowledge Discovery and Data Mining*, H. Dai, R. Srikant, and C. Zhang, Eds. Springer Berlin Heidelberg, 2004, pp. 52–62.

[73] B. F. van Dongen and W. M. P. van derAalst, “Multi-phase Process mining: Aggregating Instance Graphs into EPCs and Petri Nets,” in *PNCWB 2005 workshop*, 2005, pp. 35–58.

[74] A. K. A. de Medeiros, A. J. M. M. Weijters, and W. M. P. van der Aalst, “Genetic process mining: an experimental evaluation,” *Data Min. Knowl. Discov.*, vol. 14, no. 2, pp. 245–304, Jan. 2007.

[75] V. Rubin, B. F. V. Dongen, E. Kindler, and C. W. Günther, "Process Mining: A Two-Step Approach using Transition Systems and Regions," BPM Center Report BPM-06-30, BPM Center, 2006.

[76] B. F. V. Dongen, A. K. A. D. Medeiros, H. M. W. Verbeek, and A. J. M. M. Weijters, "The ProM framework: A new era in process mining tool support," in *In Proceedings 26 th Int'l Conf. on the Applications and Theory of Petri Nets (ICATPN'05), Miami, FL (LNCS 3536*, 2005, pp. 444–454.

[77] M. Lang, T. Bürkle, S. Laumann, and H.-U. Prokosch, "Process mining for clinical workflows: challenges and current limitations," *Stud. Health Technol. Inform.*, vol. 136, pp. 229–234, 2008.

[78] A. H. M. ter Hofstede, M. E. Orlowska, and J. Rajapakse, "Verification problems in conceptual workflow specifications," in *Conceptual Modeling — ER '96*, B. Thalheim, Ed. Springer Berlin Heidelberg, 1996, pp. 73–88.

[79] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services*, 2004 edition. Berlin ; New York: Springer, 2003.

[80] K. Nygaard, "Basic Concepts in Object Oriented Programming," in *Proceedings of the 1986 SIGPLAN Workshop on Object-oriented Programming*, New York, NY, USA, 1986, pp. 128–132.

[81] P. Naur and B. Randell, Eds., *Software Engineering*. Scientific Affairs Division, NATO, 1969.

[82] O. Zimmermann, V. Doubrovski, J. Grundler, and K. Hogg, "Service-oriented Architecture and Business Process Choreography in an Order Management Scenario: Rationale, Concepts, Lessons Learned," in *Companion to the 20th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, New York, NY, USA, 2005, pp. 301–312.

[83] "SOAP Specifications." [Online]. Available: http://www.w3.org/TR/soap/. [Accessed: 25-Apr-2015].

[84] H. Zhao and P. Doshi, "Towards Automated RESTful Web Service Composition," in *IEEE International Conference on Web Services, 2009. ICWS 2009*, 2009, pp. 189–196.

[85] C. D. Rosso, "Performance Analysis Framework for Large Software-intensive Systems with a Message Passing Paradigm," in *Proceedings of the 2005 ACM Symposium on Applied Computing*, New York, NY, USA, 2005, pp. 885–889.

[86] "DCE -- OpenDCE -- Portal." [Online]. Available: http://www.opengroup.org/dce/. [Accessed: 03-May-2015].

[87] Y. Liu and D. . Hoang, “OSI Remote Procedure Call: Standardization Issues, Design and Implementation,” *Comput Commun*, vol. 20, no. 6, pp. 462–474, Jul. 1997.

[88] E. Curry, “Message-Oriented Middleware,” in *Middleware for Communications*, Q. H.hmoud, Ed. John Wiley & Sons, Ltd, 2004, pp. 1–28.

[89] “Welcome to MQSeries.net - Serving IBM WebSphere MQ Professionals Worldwide.” [Online]. Available: http://mqseries.net/. [Accessed: 03-May-2015].

[90] “Message Queuing (MSMQ).” [Online]. Available: https://msdn.microsoft.com/en-us/library/ms711472%28v=vs.85%29.aspx. [Accessed: 03-May-2015].

[91] “ORB.” [Online]. Available: http://www.omg.org/gettingstarted/orb_basics.htm. [Accessed: 03-May-2015].

[92] F. E. Redmond, *Dcom: Microsoft Distributed Component Object Model with Cdrom*, 1st ed. Foster City, CA, USA: IDG Books Worldwide, Inc., 1997.

[93] D. J. Armstrong, “The Quarks of Object-oriented Development,” *Commun ACM*, vol. 49, no. 2, pp. 123–128, Feb. 2006.

[94] “Learn about Java Technology.” [Online]. Available: https://www.java.com/en/about/. [Accessed: 21-Jun-2015].

[95] “CCM.” [Online]. Available: http://www.omg.org/spec/CCM/. [Accessed: 03-May-2015].

[96] S. Denninger and I. Peters, *Enterprise Java Beans 2.1*. APress L. P., 2003.

[97] “[MS-COM]: Component Object Model Plus (COM+) Protocol.” [Online]. Available: https://msdn.microsoft.com/en-us/library/cc225390.aspx. [Accessed: 03-May-2015].

[98] C. Nagel, *Enterprise Services with the .NET Framework: Developing Distributed Business Solutions with .NET Enterprise Services (Microsoft Net Development Series)*. Addison-Wesley Professional, 2005.

[99] “UDDI | Online community for the Universal Description, Discovery, and Integration.” [Online]. Available: http://uddi.xml.org/. [Accessed: 25-Apr-2015].

[100] K. J. Ma, “Web services: what’s real and what’s not?,” *IT Prof.*, vol. 7, no. 2, pp. 14–21, Mar. 2005.

[101] M. B. Blake, A. L. Sliva, M. z. Muehlen, and J. V. Nickerson, "Binding Now or Binding Later: The Performance of UDDI Registries," in *40th Annual Hawaii International Conference on System Sciences, 2007. HICSS 2007*, 2007, p. 171c–171c.

[102] "ebXML - Enabling A Global Electronic Market." [Online]. Available: http://www.ebxml.org/. [Accessed: 25-Apr-2015].

[103] "Architectural Styles and the Design of Network-based Software Architectures." [Online]. Available: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm. [Accessed: 25-Apr-2015].

[104] Y. Liu, Q. Wang, M. Zhuang, and Y. Zhu, "Reengineering Legacy Systems with RESTful Web Service," in *Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International*, 2008, pp. 785–790.

[105] P. Wehner, C. Piberger, and D. Göhringer, "Using JSON to manage communication between services in the Internet of Things," in *2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, 2014, pp. 1–4.

[106] H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker, "Modeling and Composing Service-Based Nd Reference Process-Based Multi-enterprise Processes," in *Proceedings of the 12th International Conference on Advanced Information Systems Engineering*, London, UK, UK, 2000, pp. 247–263.

[107] "Estimated-Regression Planning for Interactions with Web Services." [Online]. Available: http://www.aaai.org/Library/AIPS/2002/aips02-021.php. [Accessed: 03-May-2015].

[108] E. Sirin, J. Hendler, and B. Parsia, "Semi-automatic Composition of Web Services using Semantic Descriptions," in *In Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS 2003*, 2002, pp. 17–24.

[109] R. J. Waldinger, "Web Agents Cooperating Deductively," in *Proceedings of the First International Workshop on Formal Approaches to Agent-Based Systems-Revised Papers*, London, UK, UK, 2001, pp. 250–262.

[110] F. Casati, M. Sayal, and M.-C. Shan, "Developing E-Services for Composing E-Services," in *Advanced Information Systems Engineering*, K. R. Dittrich, A. Geppert, and M. C. Norrie, Eds. Springer Berlin Heidelberg, 2001, pp. 171–186.

[111] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klien, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovoc, and S. Weerawarana, "Business Process Execution Language for Web Services Version 1.1," *Business Process Execution Language for Web Services Version 1.1*, 05-May-2003. [Online]. Available: http://msdn.microsoft.com/en-us/library/ee251594(v=bts.10).aspx. [Accessed: 04-Nov-2014].

[112] "Cover Pages: Web Services Flow Language (WSFL)." [Online]. Available: http://xml.coverpages.org/wsfl.html. [Accessed: 21-Jun-2015].

[113] "XLANG/s Language." [Online]. Available: https://msdn.microsoft.com/en-us/library/aa577463.aspx. [Accessed: 21-Jun-2015].

[114] "Web Services Addressing (WS-Addressing)." [Online]. Available: http://www.w3.org/Submission/ws-addressing/. [Accessed: 28-Apr-2015].

[115] "Apache ODE – Apache ODE™." [Online]. Available: http://ode.apache.org/. [Accessed: 28-Apr-2015].

[116] O. Kopp, K. Görlach, D. Karastoyanova, F. Leymann, M. Reiter, D. Schumm, M. Sonntag, S. Strauch, T. Unger, M. Wieland, and R. Khalaf, "A Classification of BPEL Extensions," *J. Syst. Integr.*, vol. 2, no. 4, pp. 3–28, Oct. 2011.

[117] W. M. C. Specification, *Workflow Management Coalition Terminology & Glossary*. Workflow Management Coalition Specification, 1999.

[118] D. Clarke, I. Saleh, and M. B. Blake, "Modelling Service Workflow Outcomes by Assessing the Underlying Message Flows," in *WETICE Conference (WETICE), 2014 IEEE 23rd International*, 2014, pp. 9–14.

[119] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA: MIT Press, 2009.

[120] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[121] D. Edwards, *Introduction to Graphical Modelling*. New York: Springer, 2000.

[122] M. Taboga, *Lectures on Probability Theory and Mathematical Statistics - 2nd Edition*. s.l.: CreateSpace Independent Publishing Platform, 2012.

[123] A. P. Dawid, "Conditional Independence for Statistical Operations," *Ann. Stat.*, vol. 8, no. 3, pp. 598–617, May 1980.

[124] J. Hammersley and P. Clifford, "Markov field on finite graphs and lattices," 1971.

[125] S. L. Lauritzen, A. P. Dawid, B. N. Larsen, and H.-G. Leimer, "Independence properties of directed markov fields," *Networks*, vol. 20, no. 5, pp. 491–505, Aug. 1990.

[126] S. L. Lauritzen, *Graphical Models*. Oxford; New York: Clarendon Press ; Oxford University Press, 1996.

[127] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter, *Probabilistic Networks and Expert Systems: Exact Computational Methods for Bayesian Networks*, 1st ed. Springer Publishing Company, Incorporated, 2007.

[128] S. A. Andersson, D. Madigan, and M. D. Perlman, "An Alternative Markov Property for Chain Graphs," *ArXiv13023553 Cs*, Feb. 2013.

[129] R. R. Bouckaert and M. Studený, "Chain graphs: Semantics and expressiveness," in *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, C. Froidevaux and J. Kohlas, Eds. Springer Berlin Heidelberg, 1995, pp. 69–76.

[130] E. T. Jaynes, *Probability Theory: The Logic of Science*, 1 edition. Cambridge, UK ; New York, NY: Cambridge University Press, 2003.

[131] J. R. Pierce, *An Introduction to Information Theory: Symbols, Signals and Noise*, Subsequent edition. New York: Dover Publications, 1980.

[132] G. Shafer, "Jeffrey's Rule of Conditioning," *Philos. Sci.*, vol. 48, no. 3, pp. 337–362, Sep. 1981.

[133] T. D. Nielsen and F. V. JENSEN, *Bayesian Networks and Decision Graphs*, 2nd edition. New York: Springer, 2007.

[134] D. A. Schum and S. Starace, *The Evidential Foundations of Probabilistic Reasoning*, 1 edition. Evanston, Ill: Northwestern University Press, 2001.

[135] S. L. Lauritzen, "Some modern applications of graphical models," presented at the Oxford University Press, 2003.

[136] "RITA | BTS | Title from h2." [Online]. Available: http://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp. [Accessed: 12-Jun-2015].

[137] H. Chan and A. Darwiche, "On the Robustness of Most Probable Explanations," *ArXiv12066819 Cs*, Jun. 2012.

[138] A. Darwiche and J. D. Park, "Complexity Results and Approximation Strategies for MAP Explanations," *ArXiv11070024 Cs*, Jun. 2011.

[139] N. L. Zhang and D. Poole, *A Simple Approach to Bayesian Network Computations*. 1994.

[140] S. L. Lauritzen, "The EM Algorithm for Graphical Association Models with Missing Data," *Comput Stat Data Anal*, vol. 19, no. 2, pp. 191–201, Feb. 1995.

[141] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *Ann. Math. Stat.*, vol. 22, no. 1, pp. 79–86, Mar. 1951.

[142] G. Schwarz, "Estimating the Dimension of a Model," *Ann. Stat.*, vol. 6, no. 2, pp. 461–464, 1978.

[143] T. Fawcett, "An Introduction to ROC Analysis," *Pattern Recogn Lett*, vol. 27, no. 8, pp. 861–874, Jun. 2006.

[144] H. Chan and A. Darwiche, *A Distance Measure for Bounding Probabilistic Belief Change*. 2003.

[145] H. Chan, "Sensitivity Analysis of Probabilistic Graphical Models," University of California at Los Angeles, Los Angeles, CA, USA, 2005.

[146] K. Kim, W. Javed, C. Williams, N. Elmqvist, and P. Irani, "Hugin: A Framework for Awareness and Coordination in Mixed-presence Collaborative Information Visualization," in *ACM International Conference on Interactive Tabletops and Surfaces*, New York, NY, USA, 2010, pp. 231–240.

[147] N. Friedman, "The Bayesian Structural EM Algorithm," *ArXiv13017373 Cs Stat*, Jan. 2013.

[148] E. Estellés-Arolas and F. González-Ladrón-De-Guevara, "Towards an Integrated Crowdsourcing Definition," *J Inf Sci*, vol. 38, no. 2, pp. 189–200, Apr. 2012.

[149] J. R. Beveridge, G. H. Givens, P. J. Phillips, and B. A. Draper, "Factors that influence algorithm performance in the Face Recognition Grand Challenge," *Comput. Vis. Image Underst.*, vol. 113, no. 6, pp. 750–762, Jun. 2009.

[150] J. Jarrett, I. Saleh, M. B. Blake, R. Malcolm, S. Thorpe, and T. Grandison, "Combining human and machine computing elements for analysis via crowdsourcing," in *2014 International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, 2014, pp. 312–321.

[151] D. Kahneman, P. Slovic, and A. Tversky, Eds., *Judgment Under Uncertainty: Heuristics and Biases*, 1 edition. Cambridge ; New York: Cambridge University Press, 1982.

[152] "The Skew Normal and Related Families | Statistical theory and methods," *Cambridge University Press*. [Online]. Available: http://www.cambridge.org/us/academic/subjects/statistics-probability/statistical-theory-and-methods/skew-normal-and-related-families?format=HB. [Accessed: 23-Jan-2015].

[153] O. Bangsø and P.-H. Wuillemin, "Top-down Construction and Repetitive Structures Representation in Bayesian Networks," in Proceedings of the Thirteenth International Florida Artificial Intelligence Research Society Conference, 2000, pp. 282–286.

[154] D. Koller and A. Pfeffer, "Object-Oriented Bayesian Networks," 1997, pp. 302–313.

[155] N. Fenton and M. Neil, *Risk Assessment and Decision Analysis with Bayesian Networks*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2012.

[156] "Lumina | Business Intelligence Tools, Enterprise Risk Management & Decision Support Software | Analytica - Lumina Decision Systems." [Online]. Available: http://www.lumina.com/. [Accessed: 05-Jul-2015].

[157] "Norsys - Netica Application." [Online]. Available: https://www.norsys.com/netica.html. [Accessed: 05-Jul-2015].

[158] "GeNIe and SMILE - Home." [Online]. Available: https://dslpitt.org/genie/. [Accessed: 05-Jul-2015].

[159] "Software Packages for Graphical Models." [Online]. Available: http://www.cs.ubc.ca/~murphyk/Software/bnsoft.html. [Accessed: 05-Jul-2015].

[160] U. Lerner, E. Segal, and D. Koller, "Exact Inference in Networks with Discrete Children of Continuous Parents," *ArXiv13012289 Cs*, Jan. 2013.

[161] "The Comprehensive R Archive Network." [Online]. Available: https://cran.r-project.org/. [Accessed: 28-Jul-2015].