Spring 5-1-2013

Database Environmental Change Impact Prediction for Human-driven Tuning in Real-time (DECIPHER)

Monish Sharma
Dakota State University

Database Environmental Change Impact Prediction for Human-driven Tuning in Real-time (DECIPHER)

A graduate project submitted to Dakota State University in partial fulfillment of the requirements for the degree of

Doctor of Science

In

Information Systems

May, 2013

By

Monish Sharma

Advisor

Dr. Surendra Sarnikar

Project Committee:

Dr. Amit Deokar

Dr. Stephen Krebsbach

Dr. Maureen Murphy

PROJECT APPROVAL FORM

We certify that we have read this project and that, in our opinion, it is satisfactory in scope and quality as a project for the degree of Doctor of Science in Information Systems.

Student Name: Monish Sharma

Project Title: Database Environmental Change Impact Prediction for Human-driven Tuning in Real-time (DECIPHER)

Faculty supervisor:	Dr. Surendra Sarnikar	Date: _____________
Committee member:	Dr. Amit Deokar	Date: _____________
Committee member:	Dr. Stephen Krebsbach	Date: _____________
Committee member:	Dr. Maureen Murphy	Date: _____________

DAKOTA STATE dsu UNIVERSITY

DISSERTATION APPROVAL FORM

We certify that we have read this project and that, in our opinion, it is satisfactory in scope and quality as a project for the degree of Doctor of Science in Information Systems.

Student Name: Monish Sharma

Project Title: Database Environmental Change Impact Prediction for Human-driven Tuning in Real-time (DECIPHER)

Faculty supervisor: [illegible] Date: 2/28/2013

Dr. Surendra Sarnikar

Committee member: [illegible] Date: 2/28/2013

Dr. Amit Deokar

Committee member: [illegible] Date: 2/28/2013

Dr. Stephen Krebsbach

Committee member: [illegible] Date: 3-12-13

Dr. Maureen Murphy

ACKNOWLEDGMENT

This dissertation would not have been possible without the contributions of so many people in so many different ways. I would like to extend my appreciation especially to the following.

Thank God for the knowledge, courage and determination that he has bestowed upon me during this research, and indeed, throughout my life.

I could not have done this without the love, encouragement and support of my family members, who allowed me to remain, focused on the important things in life. I would like to thank my parents for instilling within me the desire to keep learning and their unconditional love that fuelled my inspiration and became my driving force.

I would like to thank my wife Neelam and my son Vicranth from bottom of my heart for their unequivocal personal support and great patience at all times. I immensely thank Neelam for single handedly managing the family responsibilities during times when I had to focus on the my doctoral studies and also Vicranth for his understanding that his dad could not play with him or take him to his friend's birthday parties or sporting events because he needs to work on his doctoral thesis. I am truly grateful and blessed for having you both in my life for which my mere expression of thanks does not suffice.

I am also immensely thankful to my advisor, Dr. Surendra Sarnikar and my committee members, Dr. Amit Deokar, Dr. Stephen Krebsbach and Dr. Maureen Murphy for their guidance, insights and encouragement throughout this process.

ABSTRACT

Organizations in today’s rapidly evolving digital economy are relying more than ever on their database systems for critical decision-making functions. As a result, speedy and timely availability of the information from these systems is one of key factors crucial to organizational survival. Operating these database systems at high performance levels under highly-integrated, dynamic and complex environments is a knowledge-intensive and an error-prone human-driven task. Although there have been several developments in the area of autonomous performance tuning, such approaches are of limited use because they do not include a holistic view of the problem space and the environment under which they operate. Specifically, these approaches largely ignore the impact and the extent of organization-specific environmental changes on the performance of their database systems. This research addresses these issues by proposing: 1. A holistic autonomic tuning knowledge model that extends the existing autonomic tuning reference model by incorporating the organization-specific environmental change impact knowledge. 2. A theory based framework called “DECIPHER” that that not only acquires this knowledge component but does so in a proactive fashion. This framework predicts the potential impact of environmental changes and its dependencies by mining the historical change information stored within the existing organizational incident management data stores. 3. A new change pattern recurrence metric to identify the contexts in which change impact prediction algorithms will be useful and to help identify the best subset of data to use for change impact prediction model building.

DECLARATION

I hereby certify that this project constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the project describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

Monish Sharma

Monish Sharma

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This chapter presents a detailed discussion on the background of the research problem and the objectives of this research. This chapter begins with an in-depth review of the background of the research problem and then discusses the key factors that were critical to the formation of research objectives and then concludes with a high level overview of the structure and flow of this document.

Background of the Problem

Database systems are one of the critical backbone infrastructure components of modern organizations. In today's digital economy, an increasing number of organizations are relying on their database systems for their critical decision-making functions (Power and Sharda, 2009). As a result, speedy and timely availability of the information from database systems is one of key factors crucial to organizational survival (Conway, Vesset, and Earl, 2009). This rapidly evolving digital economy has also led organizations to constantly strive to maximize the utilization of their Information Technology (IT) assets while reducing their operating costs and the total cost of ownership. One initiative that has been very successful in this regard is the server virtualization. Using virtualization technologies, more and more organizations are using their computing resources as a utility. This setup is typically referred to as "Private Clouds" under the cloud computing paradigm (Mell and Grance, 2009). Database systems are no exception to this. Private clouds consisting of databases are typically referred to as "Private Database Clouds"(Curino et al., 2011).

Another artifact of this constantly changing digital age is the rate at which organizations undergo change. These organization changes are fuelled by factors like mergers, acquisitions, explosive data growth, changing competitive landscape and long-term investments (McKendrick, 2011). As a result, the Information Technology (IT) environments within the organizations are becoming highly-integrated, dynamic and more and more complex (Böhm et al., 2010; Corp, 2005). According to a recent Forrester research report, organizations make up to 500 changes per month to their IT infrastructure (Forrester, 2007). When virtualization initiatives are added to this mix, it further exacerbates this situation and can even lead to severe manageability issues (Kotsovinos, 2011). These virtualization related factors coupled with speedy and timely requirement of database information pose new performance challenges for the database systems (McKendrick, 2011; Telford et al., 2003).

Operating database systems at high performance levels under complex, dynamic and dense environments such as private database clouds, requires the database administrators (DBAs) to frequently conduct performance tuning or optimizations (Rabinovitch, 2009; Schallehn, 2010; Telford et al., 2003). Database performance tuning or optimization is a very broad term and has several perspectives and definitions. In this dissertation, we will use the Sasha (1992) definition of database performance tuning since it is a holistic and realistic description of the task – *"Database tuning is the activity of making a database application run more quickly. More quickly usually means higher throughput, though it may mean lower response time for some applications. To make a system run more quickly, the database tuner may have to change the way applications are constructed, the data structures and parameters of a database system, the configuration of the operating system, or the hardware*" (Shasha, 1992).

Typically, the database tuner in most organizations is a human (Elnaffar, Powley, Benoit, and Martin, 2003; Rabinovitch, 2009). The tuning of database by a human is referred to as manual database tuning or human-driven database performance tuning.

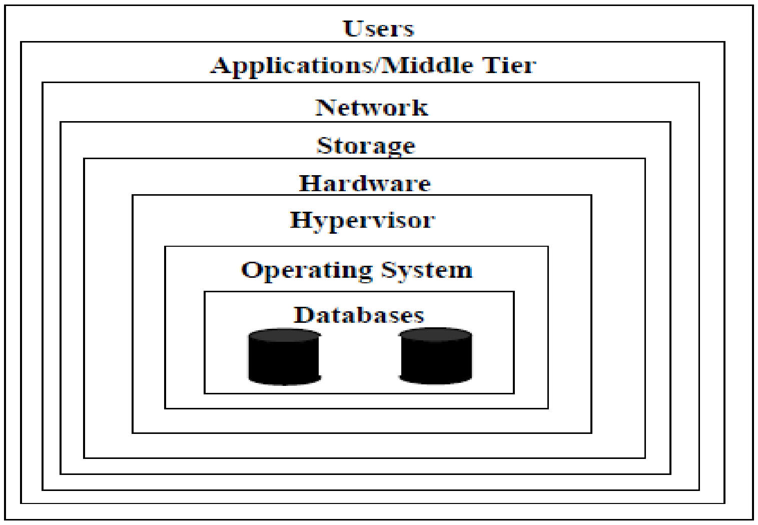


Figure 1. Typical Organizational Database Environment Stack

As evident from the above definition, the performance tuning is a goal-oriented task. These goals are largely organization-specific and typically part of the organization's service level agreements (SLAs), e.g., *the order management database application should process 1000 orders in one minute.* The above definition also highlights the reactive aspect of tuning, the complexity involved with this task and also the various factors that come into play in a typical database environment. A typical database environment, as shown in Figure 1, has several layers (Schallehn, 2010; Shasha, 1992). Private database clouds are one such example of a database environment. The database environment stack represents the IT components including the databases that are required for database application(s) to fully function. A typical modern database environment, as shown in Figure 1, has following major components (Schallehn, 2010; Shasha, 1992):

a) Users: These are the end users that use the database either directly or via a database application. Some of examples of this component are data-entry operators, system analysts, developers etc.

b) Application/Middle Tier: This consists of queries, Data Manipulation Language (DML), database packages/stored procedures or application interfaces. Some of the examples of this component are Enterprise Resource Planning (ERP), Order entry, reporting application etc.

c) Database: This consists of the database management system (DBMS) software components and the data that it manages. Examples – Oracle RDBMS, IBM DB2, MS SQL Server etc.

d) Operating System: This consists of software components that manage system resources and execution of programs and the processes. Examples – Solaris, Linux, AIX etc.

e) Network: This consists of networking components such as interconnects, LAN, WAN that support communication between different components within the database environment.

f) Storage: This consists of components that physically store the data, backups, and archived data. Examples – hard disks, flash, tapes etc.

g) Hypervisor: This layer includes server virtualization kernels that virtualize system resources. This is the key layer for private database clouds. Examples: Vmware, Xen, hyperv etc.

h) System Hardware: This consists of components such as CPU, memory, system bus etc.

Given the high density of IT environments, especially in a private database cloud setting, an issue at any of these layers of the database environment stack has a high potential of causing impact to other layers within the stack. Since organizations have large number of database systems within their complex and highly-integrated environments, the impact and the extent of environmental changes on the performance of its databases becomes significant and far-reaching.

Consider the following two scenarios:

1) As part of the quarterly patching policy, a system administrator patches the operating system on a database virtual machine (VM) of a database private cloud on a planned maintenance window on a Sunday morning. On Monday morning, users experience severe performance degradation with some of their analytical queries. The DBA working on the issue sees huge waits on logical reads on the database. Based on the recommendations of their tuning tools and automatic advisors, the DBA tweaks the database configuration parameters and even reboots the VM. After few painful hours of trying several options, the problem was narrowed down to a buggy OS patch that was applied to the VM. A workaround provided by the OS vendor fixed the issue.

2) The storage team upgraded the microcode of the SAN storage used by a business intelligence (BI) VM cluster's storage repository on a planned maintenance window on a Saturday afternoon. On Sunday evening, scheduled reports using the database on one of the VM's were running almost 3 times slower than usual. Based on the data gathered by the performance monitoring tools and through tracing, the DBA's found out that the physical reads on the database were very slow. DBA's started adding indexes on the tables used by the report queries. This somewhat helped but created new performance issues with some other queries. By Tuesday, the problem was traced back to the SAN microcode upgrade. The microcode was downgraded to fix the issue.

Had the DBAs known about the potential impact and extent of these environmental changes before they were implemented, they could have made better decisions to mitigate the risks posed by these changes. In the case of first scenario, DBAs could have asked for a full load test on QA VM server or a clone of the production VM with the new OS patch so that more realistic testing would have been possible. In the case of the second scenario, the DBAs could have prepared themselves for switching to a standby database that used a different SAN storage. Incorrect diagnosis and troubleshooting is expensive and error-prone. Also, these events end up repeating themselves across time and systems. Furthermore, the human-driven performance tuning task is repetitive, expensive, time-consuming and error-prone (Gil et al., 2002; Oliveira et al., 2006; Wiese and Rabinovitch, 2009).

Although, there have been several developments in the area of autonomous as well as semi-autonomous performance tuning research, they are limited in their use because they do not holistically understand the problem space and the environment under which they operate. These semi-autonomous and autonomous approaches adopt a narrow focus towards the organizational database environment stack by focusing primarily on the database layer within the environment stack. Furthermore, these approaches largely ignore the impact and the extent of organization-specific environmental changes on the components of the stack. Predicting the potential impact of environmental changes and knowing its extent before they are executed can help human as well as autonomic tuners in proactively mitigating the risks posed by them. So, how to accurately predict the potential impact and the extent of environmental changes before they are even implemented or executed in an organizational database environment stack?

Research Objectives

The objective of this project is to address the aforementioned problems:

1. By proposing a holistic autonomic tuning knowledge model that extends the existing autonomic tuning knowledge reference model by incorporating the organization-specific environmental change impact knowledge.

2. A theory based framework called "DECIPHER" that that not only acquires this knowledge component but does so in a proactive fashion. This framework predicts the potential impact of environmental changes and its dependencies by mining the historical change information stored within the existing organizational incident management data stores.

3. A new change pattern recurrence metric to identify the contexts in which change impact prediction algorithms will be useful and to help identify the best subset of data to use for change impact prediction model building.

The next six chapters provide the necessary background materials for this project. In Chapter 2 an in-depth review of the existing approaches to the database tuning problem is presented. This chapter also discusses the limitations of these approaches by adopting a knowledge management perspective towards the tuning knowledge. Chapter 3 discusses the research methodology used for the DECIPHER. Chapter 4 presents the relevant theoretical foundations for DECIPHER. This chapter also discusses the functional design factors based on the identified limitations with existing approaches that were covered in the Chapter 2. Chapter 5 presents a detailed discussion of the implementation and evaluation of DECIPHER using a real-world incident management system. This chapter discusses in detail the algorithms and steps used for DECIPHER implementation and the questions that were used for validating the accuracy of DECIPHER's prediction. Chapter 6 presents the DECIPHER evaluation results that demonstrate its accuracy as well as metrics for identifying conditions under which the system will perform effectively. Finally, Chapter 7 presents an overview of the contributions of this project and a discussion on the possible future directions for DECIPHER.

CHAPTER 2

LITERATURE REVIEW

This chapter presents an in-depth review of the existing approaches to the database tuning problems. It begins with a review of the issues pertaining to the human-driven database performance tuning from various perspectives. This chapter continues with an effort of to formalize the existing tuning approaches and solutions by adopting a knowledge management perspective towards the tuning knowledge. This chapter concludes with a discussion on the missing knowledge component required for effective database performance tuning in modern IT organizational environments such as private database clouds that are highly –integrated and complex.

Database performance tuning is one of the most significant, time-consuming and repetitive tasks performed by the database administrators (DBAs) in order to meet the organization-specific performance goals (Belknap, Dageville, Dias, and Yagoub, 2009; Boughton, Martin, Powley, and Horman, 2006; Charvet, 2003; DBTA, 2009; Embarcadero-Technologies, 2010; Oliveira et al., 2006; Wiese, Rabinovitch, Reichert, and Arenswald, 2008). The DBA’s that are able to perform such tuning successfully, efficiently and consistently are expensive and increasingly harder to find (Chaudhuri and Weikum, 2006; Krayzman, 2005; Schallehn, 2010; Sullivan, Seltzer, and Pfeffer, 2004; Wiese et al., 2008). Furthermore, this task can also be error prone which can introduce system unpredictability or even lead to system unavailability (Oliveira et al., 2006). Since organizations typically have large number of database systems, the tuning task consumes most of the DBA’s time, preventing them from focussing on strategic and long-term value adding organizational initiatives (DBTA, 2009; Embarcadero-Technologies, 2010; Oliveira et al., 2006) .

More and more organizations are embracing cloud computing technologies in the form of private clouds to address their evolving business needs and reduce their operating costs.

But, organizational Information Technology (IT) environments in today's rapidly evolving digital economy undergo several changes fueled by factors like mergers, acquisitions, explosive data growth, changing competitive landscape and long-term investments. As a result, the private cloud environments within the organizations are becoming highly-integrated, complex and very dynamic. Given the high server density of such database environments, the potential impact of IT environmental changes to the systems becomes significant and far-reaching. Human-driven database performance tuning under such environments further exacerbates the aforementioned issues (Kotsovinos, 2011).

In order to address the aforementioned issues of human-driven database performance tuning, the focus adopted by existing research efforts can be broadly classified into autonomous and semi-autonomous tuning approaches. These approaches are proposed as potential solutions towards reducing or eliminating the need for human-driven performance tuning from a maintenance, administration and resource consumption perspective (Chaudhuri and Narasayya, 2007; Kephart and Chess, 2003; Shasha, 1992; Wiese et al., 2008).

Autonomous Tuning Approaches

At a very high level, these approaches can be classified based on their integration with the database and the temporal nature (how and when) of their tuning decision (Chaudhuri and Weikum, 2006). This paper assumes a database as a relational database system that is designed to function under all types of workloads. There are several specialized database technologies and architectures that are designed for specific performance requirements that are not considered in this paper. For more information on such technologies/architectures, see Stonebraker et al. (2007) and Stonebraker (2010). Autonomous tuning approaches can be broadly categorized into Tradeoff elimination-based (Vengurlekar et al., 2008), Feedback-based (Herodotos Herodotou, 2010), Exploration-based (Sullivan et al., 2004), Model-based (Chaudhuri and Weikum, 2006) and Hardware-based (Chaudhuri and Weikum, 2006; Herodotos Herodotou, 2010; Krayzman, 2005; Schallehn, 2010; Shasha, 1992; Sullivan et al., 2004). A high level summary of these approaches are shown in Table 1 below.

Table 1. Summary of Existing Autonomic Tuning Approaches

Tuning	Pros and Cons	References
Tradeoff Elimination-Based	Pros: One-size-fits-all approach; Very Flexible. Cons: Sacrificing Optimal Performance for Flexibility; close integration to database internals.	Vengurlekar et al., (2008); Chaudhuri and Weikum (2006); Schallehn (2010)
Feedback-Based	Pros: Control-loop; pay-as-you-go approach; Quick adaptability to unseen or changing workload situations. Cons: Time-consuming; can introduce runtime unpredictability.	Chaudhuri and Narasayya, (2007); Wiese and Rabinovitch (2009)**;** Sullivan et al., (2004); Kephart and Chess (2003); Elnaffar et al., (2003)
Exploration-Based	Pros: Proactive; less runtime overhead. Cons: Time-consuming with large solution search space; solutions cannot be generalized across different database workloads.	Sullivan et al., (2004); Ziauddin et al., (2008); Markl et al. (2003)**;** Lee and Zait (2008); Brown et al.,(1996)
Model-Based	Pros: Statistical or Probabilistic models to predict optimal parameters for different workloads; can tune several parameters or knobs. Cons:	Sullivan et al., (2004); Chaudhuri and Weikum (2006)**;** Schallehn (2010)

	Needs sufficient training data for accurate prediction; difficult to model for a complex system.	
Hardware-Based	Minimal tuning; Minimal change to the database objects or application code. Cons: Expensive; unable to handle all types of performance issues.	Krayzman (2005); Mueller and Teubner(2009); Bigus et al.,(2000)

Tradeoff elimination-based approaches are based on the principle that if a policy or high level parameter or knob provides near optimal results (sweet-spot) under unseen or changing workloads then its low level knobs or parameters can be eliminated (Chaudhuri and Weikum, 2006; Schallehn, 2010). The advantage of this approach is its one-size-fits-all approach (Vengurlekar, Vallath, and Long, 2008). The disadvantage is the sacrificing of optimal performance and also addition of some overhead at the expense of flexibility (Chaudhuri and Weikum, 2006). Furthermore, this approach requires detailed understanding of the low level parameters and their sensitivities (Chaudhuri and Weikum, 2006). Typically these approaches are closely integrated to the database internals (Vengurlekar et al., 2008).

Feedback-based methodologies largely employ exploitation or control-loop or pay-as-you-go approaches towards performance tuning. Such methodologies follow a step-wise performance tuning approach wherein one parameter or knob or a policy is changed at a time based on some pre-defined threshold value (Rabinovitch and Wiese, 2007; Sullivan et al., 2004). Adaptability to workload changes is a key feature of such approaches (Elnaffar et al., 2003; Kephart and Chess, 2003). These approaches typically use a feedback or control loop. Such models aim to provide the autonomic managers within a database with the localized and internal knowledge about its environment in order to make better tuning decisions. These approaches are also online in nature , i.e., tuning is performed continuously (Schallehn, 2010).

These approaches have advantages like quick adaptability to unseen or changing workload situations without the need of much prior training (Brown, Carey, and Livny, 1996; Chaudhuri and Narasayya, 2007; Chaudhuri, Narasayya, and Ramamurthy, 2008; Markl, Lohman, and Raman, 2003; Sullivan et al., 2004). The disadvantage of such approaches is that it cannot effectively handle issues that might require tuning of multiple parameters or knobs simultaneously to resolve a performance issue (Sullivan et al., 2004). Furthermore, this architecture has a runtime overhead in situations where several iterations are needed to find an optimal solution (Sullivan et al., 2004). Moreover, in such situations the system usually does not know when an optimal situation or critical value has been reached (Herodotos Herodotou, 2010; A. W. Lee and Zait, 2008; Sullivan et al., 2004). The feedback –based approaches could also introduce runtime unpredictability (Herodotos Herodotou, 2010; Ziauddin, Das, Su, Zhu, and Yagoub, 2008). These approaches are also closely integrated to the database system (Chaudhuri and Weikum, 2006; Markl et al., 2003).

Exploration-based methodologies utilize explorative or comparison-based approaches wherein comparisons can be made proactively or even reactively with past measurements of parameters or knobs in order to reach an optimal value using an empirical or experimental exploration process (Herodotos Herodotou, 2010; Sullivan et al., 2004). These approaches are static in nature , i.e., tuning is not performed continuously but initiated by the database system (Schallehn, 2010). The advantage of such approaches are that they can have less runtime overhead as the exploration or comparison process can be done off hours or on an experimental or sandboxed environment (Herodotos Herodotou, 2010; Sullivan et al., 2004; Ziauddin et al., 2008). Furthermore, this approach can avoid runtime unpredictability (Ziauddin et al., 2008). The disadvantages of such approaches are that the exploration or search process can be very time-consuming in situations where the search space of potential solutions is very large (Sullivan et al., 2004). Furthermore, the solutions in this approach cannot be generalized for all workload situations, especially the unseen ones (Sullivan et al., 2004). In this approach the decision-making and execution of the tuning decision can be de-coupled with the database system and can also be supported by external tools (Chaudhuri and Weikum, 2006; Schallehn, 2010).

Model-based methodologies usually employ approaches that use probabilistic or statistical models that can predict the database system's performance under various workload situations (Sullivan et al., 2004). These approaches are also static in nature , i.e., not performed continuously but initiated by the database system (Schallehn, 2010). The advantages of such approaches are that they have low runtime overhead since they do not actually need to test the solution (Sullivan et al., 2004). Furthermore, these models can effectively handle issues that require tuning of multiple parameters or knobs simultaneously to resolve a performance issue (Sullivan et al., 2004). The disadvantage of such approaches are that they need sufficient training data for accurate prediction (Sullivan et al., 2004). Moreover, data collection process to train the model can have runtime overhead (Sullivan et al., 2004). Also, in this approach the decision-making and execution of the decision can be de-coupled with the database system and can also be supported by external tools (Chaudhuri and Weikum, 2006; Schallehn, 2010). Model building of a complex system can also be a challenge with this approach (Sullivan et al., 2004).

Hardware-based methodologies employ solutions that involve hardware upgrades or hardware accelerators to improve the performance of a database system (Chaudhuri and Weikum, 2006; Krayzman, 2005; Mueller and Teubner, 2009). Advantages of these type of approaches are that these can provide more system resources to a performance problem without having to change the database objects or application code (Krayzman, 2005; Mueller and Teubner, 2009). The disadvantages of such approaches are higher costs and inability to handle all types of performance issues (Bigus, Hellerstein, Jayram, and Squillante, 2000; Krayzman, 2005).

Semi-Autonomous Tuning Approaches

There are very few research efforts that focus on combining autonomous tuning approaches with the human knowledge. Sullivan, et al., (2004) research proposes a probabilistic reasoning approach as part of a model-based tuning approach to automate software tuning in general. The author in this research proposes that the domain experts with detailed knowledge of internal workings of a system construct initial models for inter-dependent low level system functions in order to attain the desired performance goal. These models can be trained under various workloads to automatically handle tuning of various knobs to achieve the desired tuning goals. Such an approach can be a very challenging task to do for today's database systems given their internal complexity. Also, this approach solely focuses on tweaking or tuning of so called knobs or parameters and may not work for performance issues that either do not have tunable knobs or may require tuning that is applicable to other components within a database environment, e.g., application, database, operating system, network, and storage and system hardware.

Rabinovitch (2009) research formalizes the DBA's database-specific tuning knowledge into textual information called "tuning plans" and saves them in a best practice repository. Policies are then used to activate or deactivate these plans to address the performance problem as part of feedback-based tuning methodology. Other approaches in this category focus on the human database tuner user either reviewing the solutions provided by the autonomous approaches or providing higher level workload-specific goals or policies (Herodotos Herodotou, 2010; Ziauddin, Das, Su, Zhu, and Yagoub, 2008).

Limitations with Existing Approaches

Since performance tuning is a knowledge-intensive task, the component based reference tuning knowledge model proposed by Wiese and Rabinovitch (2009) can be used to formalize the existing autonomous and semi-autonomous tuning approaches and help us better understand the limitations of these approaches. Furthermore, this model adopts a generic view of the database system making it effective to formalize across various database technologies and architectures.

This model lays out the knowledge components required for successful database performance tuning in an environment under control. In this layered model, shown in Figure 2 below, each knowledge component builds on top of the each other. The most general knowledge is at the bottom and the very specific knowledge is at the top. Figure 2 is an adaptation of Wiese and Rabinovitch (2009) autonomic tuning knowledge reference model.

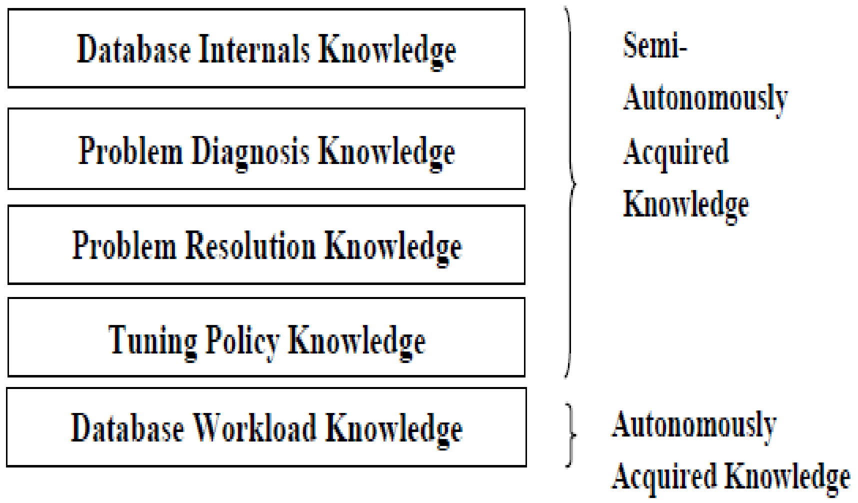

Database Internals Knowledge
Problem Diagnosis Knowledge
Problem Resolution Knowledge
Tuning Policy Knowledge
Database Workload Knowledge
Semi-Autonomously Acquired Knowledge
Autonomously Acquired Knowledge

Figure 2. Autonomic Tuning Knowledge Reference Model

This autonomic tuning knowledge reference model divided into two parts – Object level that represents the environment under control and meta-level that represents the knowledge components. The autonomic tuning knowledge reference model, as shown in Figure 2, has following components (Wiese and Rabinovitch, 2009):

1) **Database Workload Knowledge:** The foundational knowledge component of this model is the database workload knowledge. This model considers workload changes as the only source of changes in a database environment. In fact, this knowledge component is considered as the “surrounding and influencing environment” for the database system. The model also views this knowledge component as non-modifiable and is constantly changing, e.g., online transaction processing or batch processing or a hybrid of these two. This knowledge component is the most general compared to others and can be autonomously obtained through myriad of approaches and tools. Furthermore, many modern database systems have the basic instrumentation in their kernels to capture and process their workload (Markl et al., 2003; Shasha, 1992).

2) **Tuning Policy Knowledge:** This knowledge component builds on top of the workload knowledge component and refers to the specific knowledge of resources that need to be monitored or changed along with their specific thresholds and user-defined performance goals. Even in a fully autonomic system, the DBAs would be required to define or update new policies that control the behavior of autonomous managers (Herodotos Herodotou, 2010). Hence this knowledge component falls under semi-autonomously acquired knowledge category.

3) **Problem Resolution Knowledge:** This knowledge component builds on top of the tuning policy knowledge component and refers to the database specific procedural knowledge regarding the actions needed to resolve the performance problem. This knowledge component can be viewed as a recorded or codified reaction to a particular performance problem that happened in past or is current or may occur in future. Designing such a plan or action may need human intervention depending upon the nature of the problem (Ziauddin et al., 2008).

4) **Problem Diagnosis Knowledge:** This knowledge component builds on top of the problem resolution knowledge component and refers to best practices used by DBAs to diagnose or troubleshoot performance problems. This knowledge component can be considered analogous to the standard operating procedures that are designed and maintained by experts (Wiese et al., 2008).

5) **Database Internal Knowledge:** This knowledge component builds on top of the problem diagnosis knowledge component and refers to the expert knowledge of internal workings of a database system components and understanding of their inter-dependencies and inter-reactions. This is the most specific form of knowledge as it requires understanding of the underlying database technology, configuration, and hierarchy of system components, their behavior, construction and their complex interdependent cause-effect relationships. In most cases, this knowledge component is acquired through a human expert (Sullivan et al., 2004).

Based on this understanding of the tuning reference knowledge model, the gaps within the existing research literature can be broadly classified into the following areas:

a) **Narrow Focus of the Database Environment Stack:** Existing tuning approaches do not account for the constantly changing and highly-integrated nature of today's database environments such as private database clouds. Existing approaches focus primarily on the database layer of the database environment. As highlighted in Figure 1, the database environment has several layers and changes to any of these layers can have an effect on the performance of the databases. Hence we need to consider a more holistic view of the database environment.

b) **Lack of Organization Specific Focus**: Every organization has its own unique database environment stack, its specific change cycles and service level requirements. Hence we need to consider an approach that adapts to these organization-specific requirements.

c) **Workload-specific focus**: As evident from Figure 2 above, the foundational knowledge component for the existing autonomous and semi-autonomous tuning approaches is the workload knowledge. These approaches do not account for changes that organizational database environments typically go through that have an effect on the performance of its database systems, e.g., the addition of disks of different speeds to an existing storage system on a database server could result in hotspots for database reads resulting in performance degradation, or an operating system patch upgrade causes memory issues that in turn have an adverse impact on the database's performance, or a firewall network change causes network latency resulting in connection slowness, or timeouts for database applications. Hence, it is crucial that we consider the knowledge of impact and extent of environmental changes within the stack besides just the workload changes.

d) **Lack of Focus on Proactive Problem Resolution Knowledge**: Most of the autonomous as well as semi-autonomous tuning approaches adopt a reactive approach towards acquiring problem resolution knowledge. In organizational settings, following factors typically come into play in human-driven performance situations that influences the tuning task and its outcome:

   1) Aggressive tuning deadlines: Speedy and timely requirement of information from the database systems dictates the aggressive deadlines for the tuning tasks. As a result, this task becomes cognitively taxing for the DBAs.

   2) Performance tuning costs: These are the opportunity costs that an organization incurs as a result of the performance problem. This is also responsible for aggressive deadlines and making the tuning a cognitively taxing task for the DBAs.

   3) Environmental change impact uncertainty: This is the result of complexity and density of the database environment stack as shown in figure 1. This

uncertainty makes the troubleshooting of the performance problem a challenging task.

Performance tuning under such factors could result in incorrect diagnosis and error-prone tuning (Endsley, 1995; Oliveira et al., 2006). Hence, acquiring the problem resolution knowledge specific to the environment, i.e., the knowledge of impact and extent of environmental changes proactively , i.e., before the changes are even implemented could minimize the impact of some of these adverse effects.

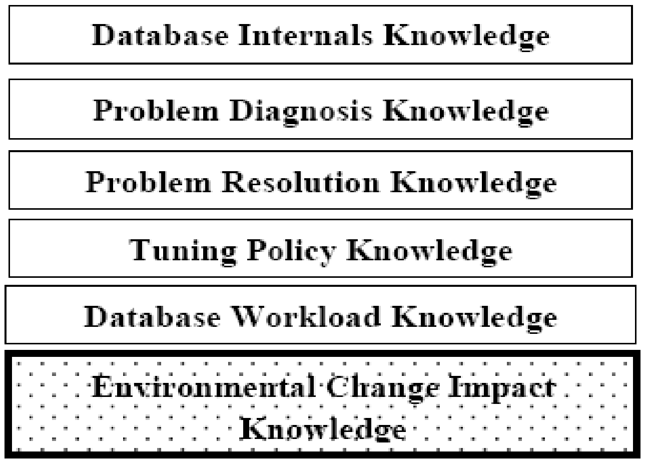

Database Internals Knowledge
Problem Diagnosis Knowledge
Problem Resolution Knowledge
Tuning Policy Knowledge
Database Workload Knowledge
Environmental Change Impact Knowledge

Figure 3. Holistic Tuning Knowledge Reference Model

Based on these limitations the tuning knowledge reference model proposed by Wiese and Rabinovitch (2009) can be extended to incorporate the Environment Change Impact Knowledge (ECIK) as shown in grid pattern in Figure 3 above. This knowledge component is referred to as the impact and extent of environmental changes. This extended knowledge model holistically represents the problem space and environment for today's organizational database environmental stacks.

CHAPTER 3

RESEARCH METHODOLOGY

This chapter discusses the research methodology used for the DECIPHER. It begins with a discussion on Design Science research methodology and the reason for using it for the DECIPHER. This chapter then continues the discussion presenting the Design Science steps for DECIPHER that lays the foundation for the design, development and evaluation of DECIPHER. It concludes with a diagram that summarizes the overall Design Science approach adopted for DECIPHER.

Design Science Research Methodology

This research utilizes the Design Science research methodology since it is a fundamentally problem-solving paradigm aimed at designing artifacts that solve identified organizational problems (Hevner, March, Park, and Ram, 2004; Peffers, Tuunanen, Rothenberger, and Chatterjee, 2007). Also, since the motivation for this research also originated from the observation of the problems related to human-driven database performance tuning at author’s workplace, a problem-centered approach was taken towards research design (Blakey and Atkins, 2008; Peffers et al., 2007). Using this research methodology enables us to exploit the design process as an opportunity for learning and further advancing our understanding of the problem (Blakey and Atkins, 2008).

In the design science research methodology, design is both a process and a product. This research methodology is characterized by two fundamental research activities – build and evaluate. Build activity refers to building of the DECIPHER framework that address the aforementioned problem and the evaluate activity refers to its evaluation of the framework with respect to its ability in addressing the identified problems (Blakey and Atkins, 2008; Hevner et al., 2004). Following are the steps that were taken during the course of this research (Blakey and Atkins, 2008; Peffers et al., 2007):

a) **Problem Identification and Motivation**: A detailed and extensive analysis of database performance tuning literature on the current solutions, approaches and future trends was conducted. This highlighted the problems and challenges that the DBA's are facing within the organizations and also reinforced the author's observation of these problems at his workplace. The knowledge gained from this step was fundamental to the extension of the existing tuning knowledge reference model and also artifact design process and also in ensuring the justification of the potential value of the proposed solution. The research question that came out of this step is - *How to accurately predict the potential impact and the extent of environmental changes before they are even implemented or executed in an organizational database environment stack?*

b) **Objectives of the Solution**: The objective of the solution is to provide a capability to accurately predict the potential impact of environmental changes and identify the extent of the impact before the changes are even implemented or executed from all layers of the database environment stack

c) **Design and Development**: This entails theory based design of framework called "DECIPHER" (Database Environmental Change Impact Prediction in Human-driven Tuning in Real-time) that not only acquires this knowledge component but does so in a proactive fashion. This framework predicts the potential impact of environmental changes and identifies its dependencies by mining the historical change information stored within the existing organizational incident management data stores. The development process involved implemention of the data processing algorithms to prepare the unstructured incident management data, the implementation of predictive text mining and similarity matching algorithims. Also, a new change pattern recurrence metric is developed to identify the contexts in which change impact prediction algorithms will be useful and to help identify the best subset of data to use for change impact prediction model building.

d) **Evaluation:** The evaluation involved a prototypical implementation and validation of the DECIPHER framework against a real-world organizational incident management data store. This step involves validating the accuracy of the prediction of the environmental change impact and also the accuracy of identification of the extent of impact of the dependencies that unimplemented change have on other factors within the database environment stack.

Figure 4 is an instantiation of Peffers, et al., (2007) design approach.

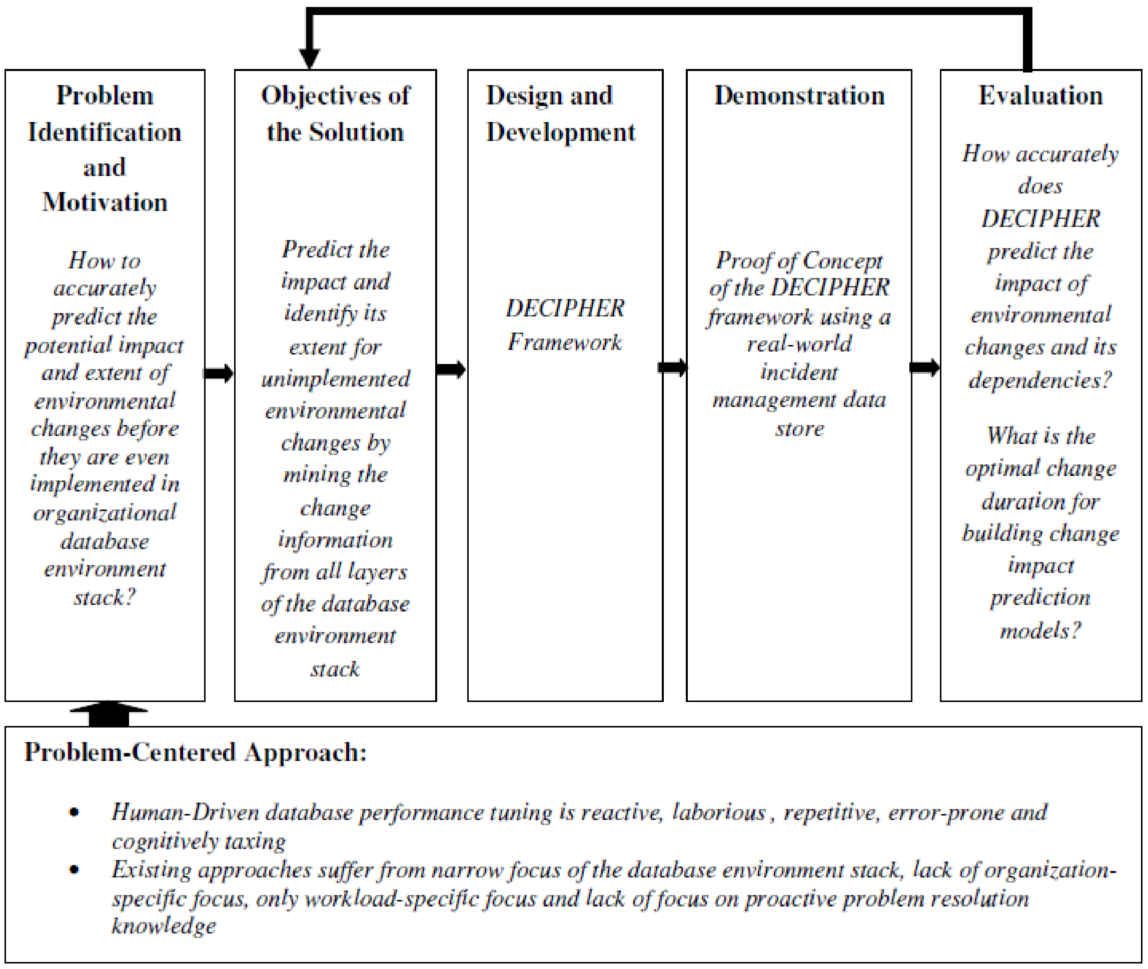

Problem Identification and Motivation
How to accurately predict the potential impact and extent of environmental changes before they are even implemented in organizational database environment stack?
Objectives of the Solution
Predict the impact and identify its extent for unimplemented environmental changes by mining the change information from all layers of the database environment stack
Design and Development
DECIPHER Framework
Demonstration
Proof of Concept of the DECIPHER framework using a real-world incident management data store
Evaluation
How accurately does DECIPHER predict the impact of environmental changes and its dependencies?
What is the optimal change duration for building change impact prediction models?
Problem-Centered Approach:
Human-Driven database performance tuning is reactive, laborious, repetitive, error-prone and cognitively taxing
Existing approaches suffer from narrow focus of the database environment stack, lack of organization-specific focus, only workload-specific focus and lack of focus on proactive problem resolution knowledge

Figure 4. Research Methodology for DECIPHER

Research Data

The implementation of DECIPHER uses a real-world incident management system called "Request Tracker" (RT) used by a medium sized organization and Oracle data miner as the text-mining mining tool. This medium-sized organization has close to 5000 employees worldwide and on average experiences 200-300 IT environmental changes per month. These environmental changes range for simple changes like resetting password for users that have forgotten their passwords to complex changes like upgrading a critical software system or migrating systems from one data center to another. A single unit of work is typically captured in the form of a "Ticket" within an incident management system. Appendix B shows the webpage with the fields for a typical change management ticket. Changes within a ticket of an Incident Change Management system have the following fields that capture the environmental change information:

1) Ticket #
2) Queue
3) Ticket Subject (Text)
4) Change Purpose (Text)
5) Creator of the ticket
6) Owner of the ticket.
7) Approver of the ticket.
8) Last Update User
9) Start Date
10) End Date
11) Classification (Regulatory Compliance)
12) Line of Business
13) Work-plan for the change (Text).
14) Impact (High or Medium or Low).

15) Impact Description (Text)

16) Back-out plan (Text).

17) General Comments/Notes (Text)

A screenshot of RT's user interface for the organization under study is shown in Appendix B. Impact field has the typical change management impact values , i.e., "Low", "Medium" and "High". The change information within the incident management system is stored in unstructured or textual format. For this research, tickets from 2008 -2011 are considered. This duration resulted in 11,118 unique change tickets and the average word count per ticket after the linguistic preprocessing stage of DECIPHER is ~ 40. The breakdown of the number of distinct words for the above change ticket's text fields is shown in Table 2. The distinct word count does not include the words that are on the stop list.

Table 2. Distinct Words for Text Ticket Fields for 2008-2011

Ticket Field	Distinct Word Count	Average Word Count
Ticket Subject	3086	5
Change Purpose	3638	8
Workplan	3059	7
Backout Plan	1882	2
Impact Description	3008	5
General Comments	2324	2

The incident management data stores has environmental changes from all support team workflow queues that maintain a specific component of the IT environment, e.g., hardware, network, operating system etc as shown in Figure 1 so that change information from all the layers of the environment is taken into account. This collection of change information is referred in this research as Change Information Corpus. This is shown in Figure 5 below. Most incident management systems have a specific field that is used to enter perceived impact. Usually, it has values like “Low”, “Medium”, and “High”. Table 3 lists the ticket count breakdown by queues within the incident management system.

Table 3. Ticket Counts by Queue for 2008-2011

Queue	Ticket Count
Users	1536
Apps	1252
Network	2081
Storage	1408
Hardware	1379
Operating System	1214
Database	2248

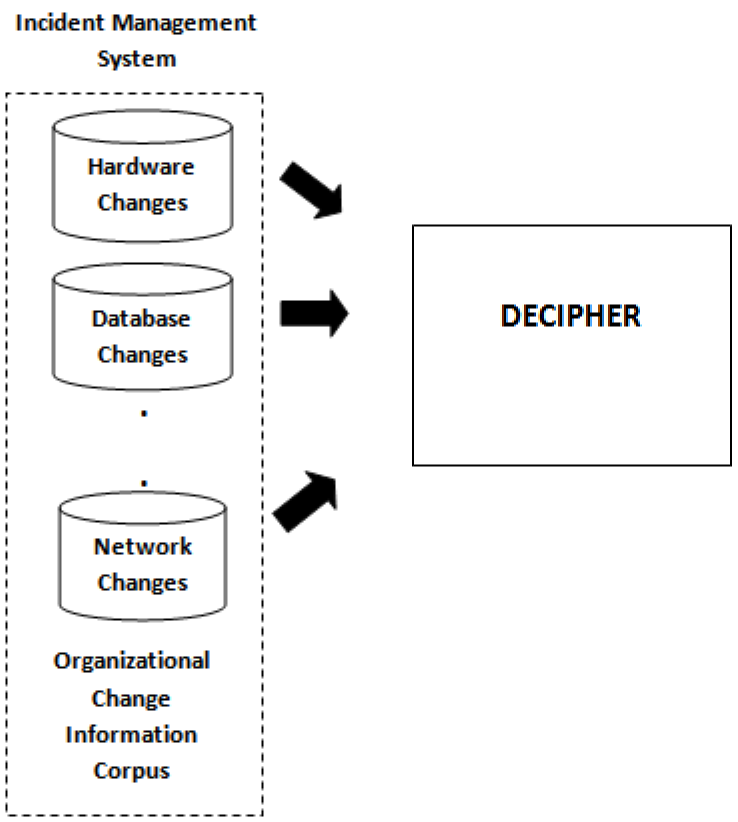

Incident Management System
Hardware Changes
Database Changes
Network Changes
Organizational Change Information Corpus
DECIPHER

Figure 5. Organizational Change Information Corpus

CHAPTER 4

THEORY AND DESIGN

This chapter presents the relevant theoretical foundations for DECIPHER and an in-depth discussion of its design. This chapter has two main sections – DECIPHER theoretical foundation and DECIPHER design. DECIPHER theory section begins with a discussion of the autonomic tuning reference architecture that highlights the role and the importance of knowledge management in database performance tuning domain. This section then continues with an explanation on the incident management systems and their role as the source for acquiring the Environmental Change Impact Knowledge. The DECIPHER design section presents a detailed explanation of the DECIPHER design architecture, including its component and concludes with a diagram on the DECIPHER's process flow.

Theoretical Underpinnings of DECIPHER

Since database performance tuning is a knowledge intensive task, a knowledge management perspective is well-suited approach for the proposed solution. The knowledge driven autonomic reference architecture proposed by Kephart and Chess (2003) provides an ideal holistic theoretical model for database performance tuning process. This process applies to both autonomous as well human-driven tuning approaches. This model is inspired by the biological autonomous human nervous system(Bell, 2004).

According to this architecture, the knowledge is the central component and should provide a common and shared understanding of the environment and problem space (Bell, 2004; Miller, 2005). The two main components of this architecture are the managed element (ME) and the autonomic Manager (AM). The autonomic manager is a unit that employs the autonomic functionality for a dedicated autonomic system management function. For example, in the database system, the query optimizer can be an autonomic manager responsible for database query optimization function (Markl et al., 2003). The managed element can be a software or hardware resource. Typically, one or more AM's exists within an autonomic system.

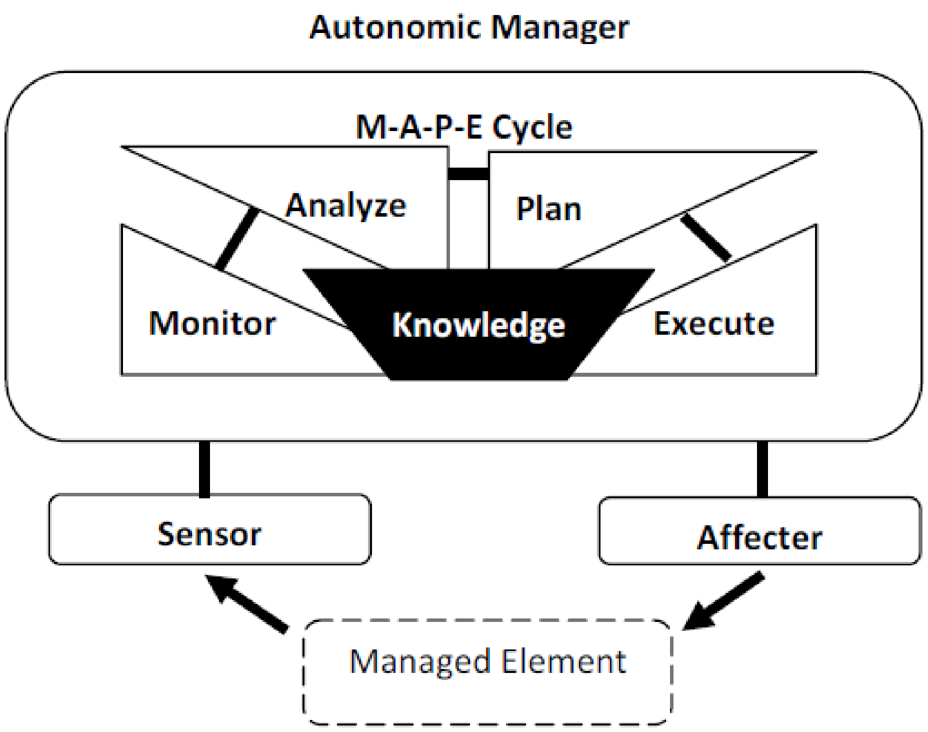


Figure 6. Knowledge-Driven Autonomic Reference Architecture

The high level architecture is shown in Figure 6. This diagram is an adaptation of the autonomic computing architecture proposed by Kephart and Chess (2003). As shown in Figure 5, the sensor collects and retrieves information about the current state of the environment stack then compares it with expectations that are held in knowledge base. The required action is executed by the effecter.

In case of human-driven tuning, the DBA can be viewed as the AM. According to the tuning reference architecture shown in the Figure 6, there are four knowledge functions that come into play (Corp, 2005; Kephart and Chess, 2003):

1) *Monitor*: The monitor function is responsible for collecting the event details from the managed element and organizing them as symptoms using the knowledge base. Specifically, the monitor function aggregates, correlates and filters the information.

2) *Analyze*: The analyze function utilizes the knowledge base in order to analyze the event symptoms presented by the previous monitor function by correlating and modeling the complex situations to better understand the problem space and the environment. This is the key knowledge-intensive step for the MAPE.

3) *Plan*: After the event is identified and analyzed, the plan function structures the actions using the knowledge base that are needed to achieve the goal and objectives.

4) *Execute*: This function is responsible for changing the behavior of the managed element via the effectors using the knowledge base.

Design of DECIPHER

In the context of database performance tuning knowledge components, let us understand how this MAPE cycle works. Figure 7 below shows the instantiation of the autonomic architecture proposed by Kephart and Chess (2003) to include the complete database environment stack from Figure 1 and the extended knowledge model from Figure 3.

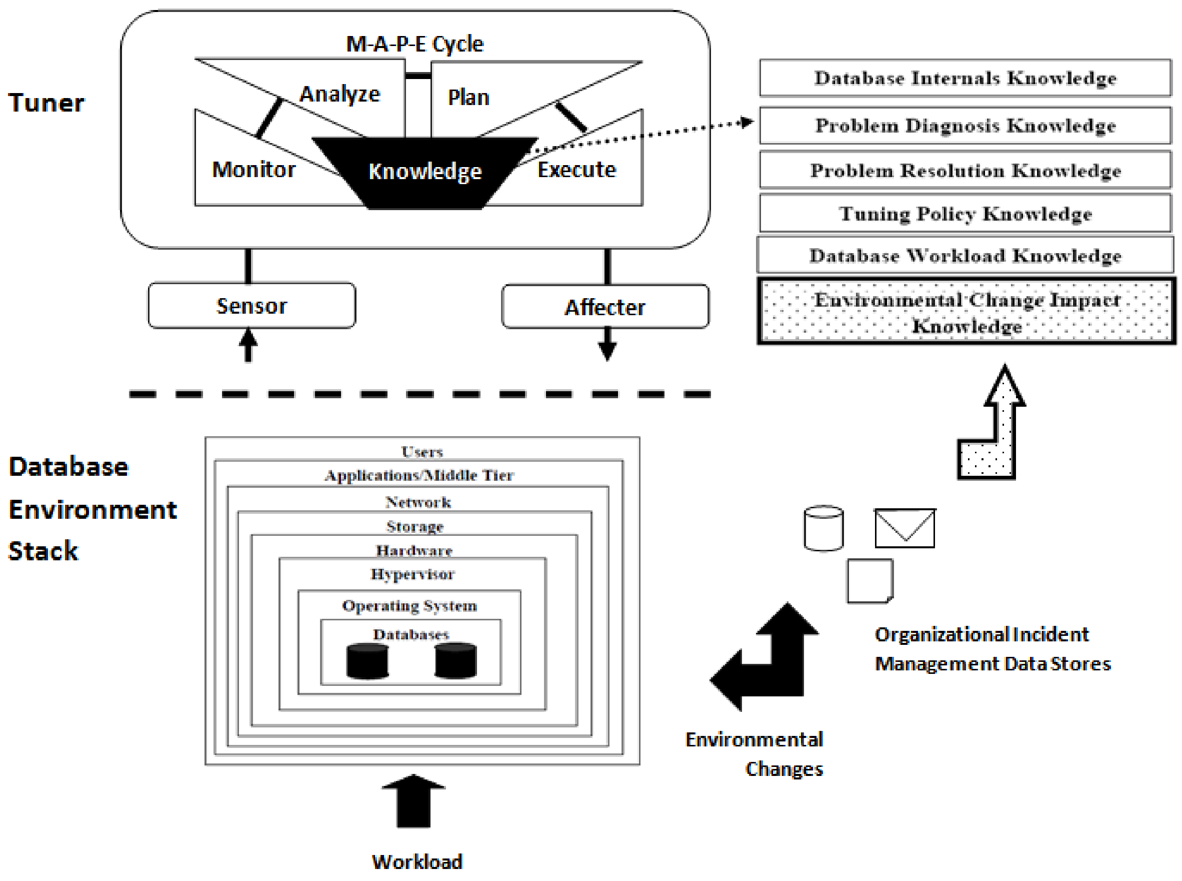

Tuner
M-A-P-E Cycle
Analyze
Plan
Monitor
Knowledge
Execute
Sensor
Affecter
Database Internals Knowledge
Problem Diagnosis Knowledge
Problem Resolution Knowledge
Tuning Policy Knowledge
Database Workload Knowledge
Environmental Change Impact Knowledge
Database Environment Stack
Users
Applications/Middle Tier
Network
Storage
Hardware
Hypervisor
Operating System
Databases
Organizational Incident Management Data Stores
Environmental Changes
Workload

Figure 7. Reactive Knowledge Driven Autonomic Tuning Architecture

The tuning process can be described using the four MAPE knowledge processes as following (Bell, 2004; Kephart and Chess, 2003; Miller, 2005; Wiese and Rabinovitch, 2009):

5) **Monitor:** The step involves identifying the existing performance problem. This step utilizes the environmental change impact, database workload and tuning policy knowledge components.

6) **Analyze:** This step is the key knowledge intensive step of M-A-P-E. It involves troubleshooting or diagnosing the performance problem, e.g., enabling query tracing for poor performing application queries. This step utilizes the environmental change impact, database workload, problem diagnosis, tuning policy and database internals

knowledge components. This step is also responsible for predicting the impact of changes (workload as well as environmental) and the dependencies of the impact on other environmental factors from a database performance perspective (Bell, 2004; Wiese and Rabinovitch, 2009).

7) **Plan:** This step involves coming up with a plan for actions that need to be taken to fix the performance problem, e.g., adding indexes. This step utilizes the environmental change impact, database workload, tuning policy, database internals and problem resolution knowledge components.

8) **Execute:** This step involves the execution of the plan to carry out the actions. This step utilizes the problem resolution knowledge component.

Most organizations use some or other form of an organizational incident management data store (help desk system or trouble ticketing system) to manage the changes to their Information Technology (IT) environment, including the database environments (Hass, 2003). Changes to any production IT environment component, e.g., operating system, hardware, and database, referred to as change management process, are facilitated by these systems (Conradi and Westfechtel, 1998). Furthermore, the increase of regulatory compliance needs (, e.g., Sarbanes-Oxley) have also pushed for a wider adoption of change management processes and tools for achieving better traceability(Chen, Kurtz, and Lee, 2009).

Typically, the organizational incident management data stores have vast amount of information about the changes to the organizational IT environment in unstructured form like notes or comments, e.g., "*added more memory to a database server to increase system performance" or "changed kernel parameters for the operating system to fix swapping issue".* Before a change is executed in production environments, they go through some form of formal or informal approval process. The approver for these change requests is typically the business owner or the stakeholder responsible for the target system. Change requests typically include information about the change purpose, date of their execution, the change work plan, perceived impact and a plan for reversing the changes if they result in any issues. This process is depicted in Figure 8 below.

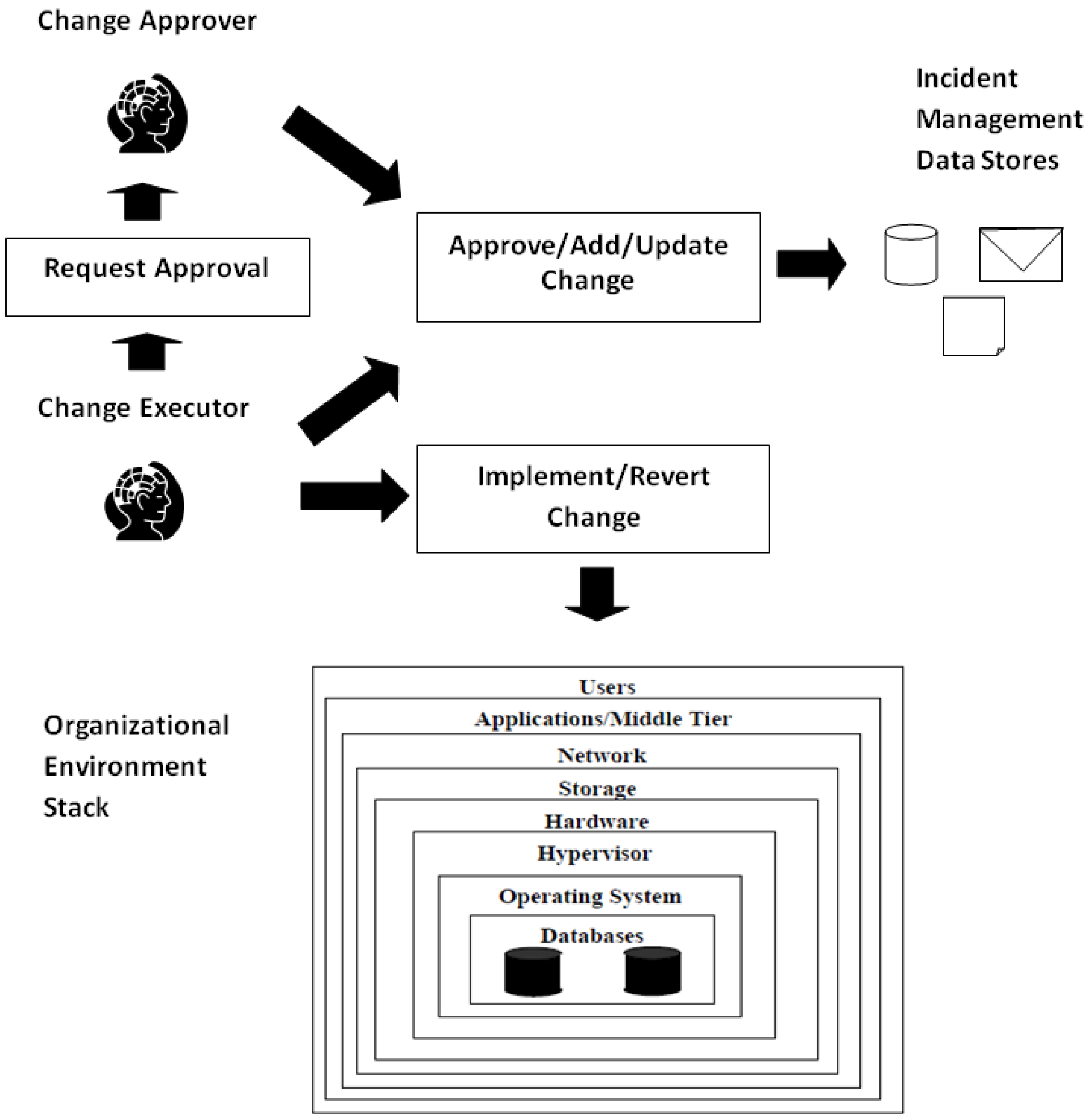


Figure 8. High Level Change Management Process

To highlight the contributions of this research, Figure 9 below shows the DECIPHER framework along with the database environment stack. This figure depicts the extension of the tuning knowledge reference model to include the environment change impact knowledge that is proactively extracted from the change information stored within organizational incident management data stores.

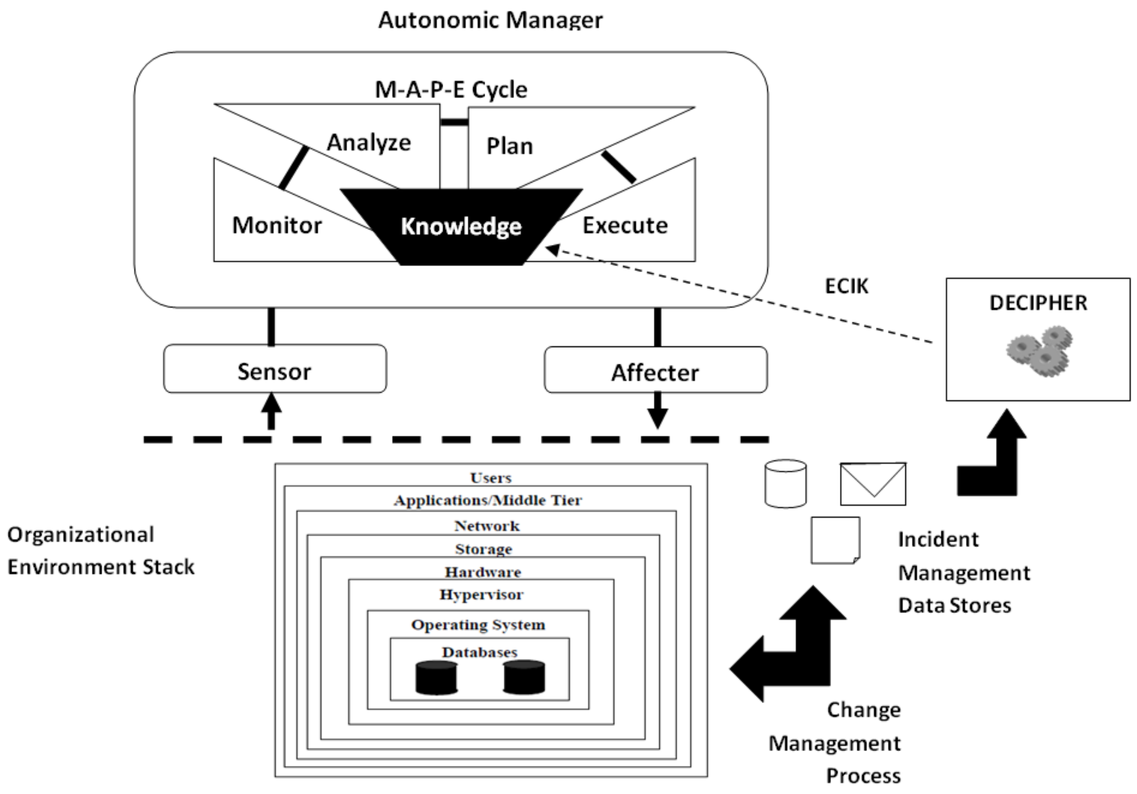


Figure 9. Overview of DECIPHER

At the core of DECIPHER are two major modules – Impact Prediction Module (IPM) and Impact Extent Identification Module (IEIM). A high level DECIPHER architecture is shown in Figure 10 below.

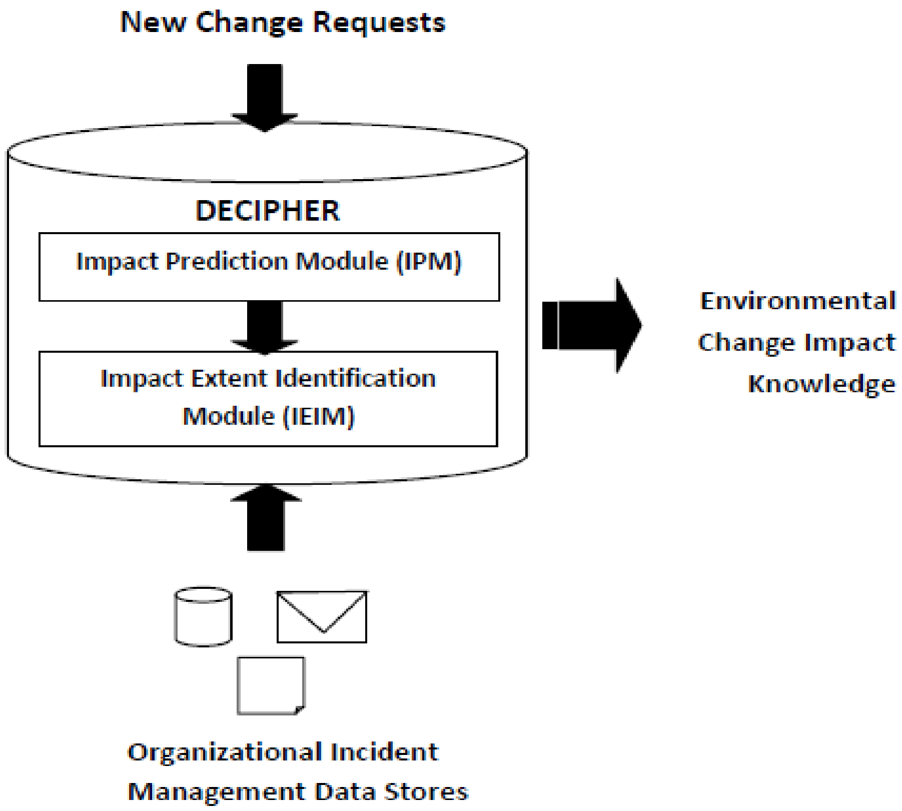

New Change Requests
DECIPHER
Impact Prediction Module (IPM)
Impact Extent Identification Module (IEIM)
Environmental Change Impact Knowledge
Organizational Incident Management Data Stores

Figure 10. High Level DECIPHER Architecture

Let us look at each of these modules in detail.

Impact Prediction Module (IPM)

The core function of IPM is predicting the potential impact for environmental changes that have not yet been implemented in real-time. More than 85% of data within enterprises is stored in unstructured formats like web pages, emails, spreadsheets, digital images and videos (Guduru, 2006). Organizational incident management data stores such as help-desk or a trouble ticketing systems are such types of unstructured or textual data stores that store rich information on changes that organizations undergo. Unfortunately, mining of such types of stores efficiently and accurately has always been a challenge. The process of extraction of previously unknown and potential useful knowledge from large unstructured textual collection is typically referred to as Text Mining (Dumais, 1998; Landau et al., 1998). Text-mining differs from traditional data mining in two major ways. Two such main differentiators are (Guduru, 2006; Landau et al., 1998) :

1) Special linguistic pre-processing is required to extract key terms from the textual collection. Furthermore, text mining process are also required to handle word ambiguities such as spelling mistakes, pronouns, synonyms, acronyms etc.

2) Several of the existing data mining algorithms do not work on textual or unstructured data that typically high dimensionality.

Impact Prediction Module (IPM) design at a high level involves two main steps – Data Extraction and Term-Extraction. Data Extraction step involves use of open source Extraction, Transformation and Load (ETL) tools that extract unstructured change data from various data sources without imposing rigid data format restrictions. Furthermore, these tools offer out-of-the-box data quality and profiling features that makes it easy to implement DECIPHER under various organizational settings. Specifically, this research uses TALEND tool that achieves this task using intuitive graphical interface (Majchrzak, Jansen, and Kuchen, 2011). The textual change information from historical organizational incident management data stores for all environment support workflows data stores, e.g., hardware, network, operating system etc. so that change information from all the layers of the environment is taken into account for IPM. This collection of change information is referred in this research as Change Information Corpus.

Most incident management systems have a specific field that is used by the change executor to enter perceived impact. Usually, it has values like “Low”, “Medium”, and “High”. IPM considers this impact field is dependent or target attribute and all other fields as Independent or predictor variables. Once the data is extracted, IPM’s term-extraction process performs the linguistic pre-processing to extract key terms. This process involves tasks such as removing stop-words, stemming and term-weighting. The terms from this process are fed to the predictive text mining algorithms to create an impact prediction model. This predictive model can then be used to score new changes. This processing is explained in detail in the IPM implementation section in the next chapter.

Another key design requirement for IPM is scoring performance, i.e., how fast IPM can predict the potential impact of new environmental changes. Modern organizations undergo a very large amount of IT environmental changes (Forrester, 2007). Furthermore, since text mining process is a computationally expensive and a time-consuming approach, it needs to produce accurate results in short period of time in order to be practically feasible under environments with very high numbers of changes. From a design perspective, this implies that the IPM algorithms and the method of processing large amounts of textual data needs to be scalable as well as efficient. This design requirement is met by using in-database text mining architecture. In-database mining avoids the traditional data or text mining step of moving of data between the source system and compute environment. Furthermore, using the massively parallel architectures of database engine, and its advanced memory management techniques, DECIPHER's text-mining algorithms can be processed efficiently and closer to the data (Inchiosa, 2011; Oracle, 2011a). Figure 11 below shows the high-level IPM process flow.

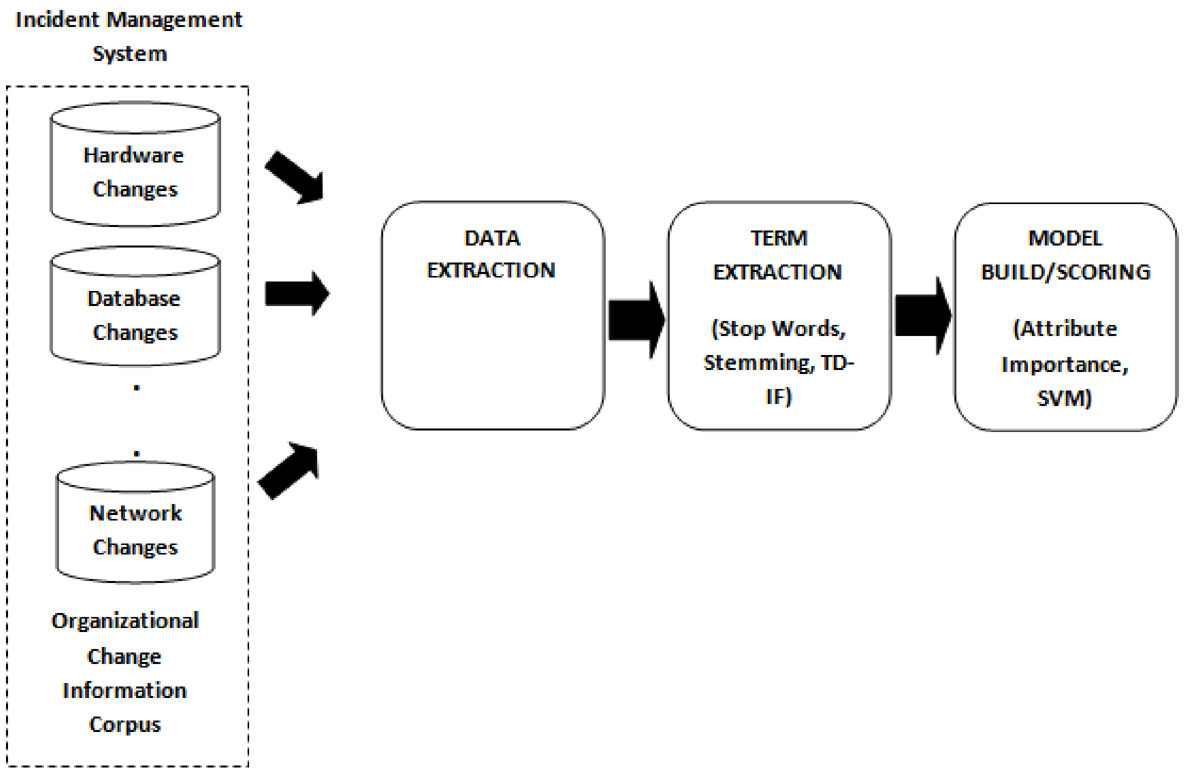

Incident Management System
Hardware Changes
Database Changes
Network Changes
Organizational Change Information Corpus
DATA EXTRACTION
TERM EXTRACTION
(Stop Words, Stemming, TD-IF)
MODEL BUILD/SCORING
(Attribute Importance, SVM)

Figure 11. High-Level IPM Process Flow

Impact Extent Identification Module (IEIM)

The core function of IEIM is that of identifying the potential extent of impact of change factors that are not yet implemented. In order to achieve this, relevant change features need to be extracted from change terms. IEIM uses feature extraction algorithm on the change terms from the IPM processing to create feature sets consisting of semantically related features (Solka, 2008). This is described in detail in the IEIM implementation section in the next chapter. These feature sets are saved in a repository.

In order to identify the potential extent of impact of changes factors, the unimplemented change tickets first undergo data and term extraction process as explained above. These new change terms are then scored for impact prediction by IPM. The change factors with high and medium predicted impact are matched for similarity against the factors from the saved feature sets. The feature set that has change factors with the high values of similarity match with new change factors represent the feature set with other potential change factors that have dependencies with the unimplemented change factors. In other words, this feature set has change factors that represent the potential extent of the impact of unimplemented change factors. For example, a new or unimplemented ticket that has change factor for a software application called ERP_HR would be match with historical change factors stored in IED. The feature set that has highest match with this factor will be returned along with its semantically related change factors like applications such as time_management or performance_management that would be dependent on ERP_HR. This knowledge of the potential impact and the dependent factors represent the Environmental Change Impact Knowledge (ECIK). Similar to IPM, scoring performance of IEIM is also important, i.e., how fast IEIM can identify the potential dependent change factors based on the new unimplemented environmental changes. From a design perspective, this implies that the IEIM algorithms and the method of processing large amounts of textual data need to be scalable as well as efficient. This design requirement is met by using in-database feature extraction and matching architecture. In-database feature extraction avoids the step of moving of data between the source system and compute environment. Furthermore, using the massively parallel architectures of database engine, and its advanced memory management techniques, DECIPHER’s feature extraction and matching algorithms can be processed efficiently and closer to the data (Inchiosa, 2011; Oracle, 2011a). Figure 12 below shows the high-level IEIM process flow.

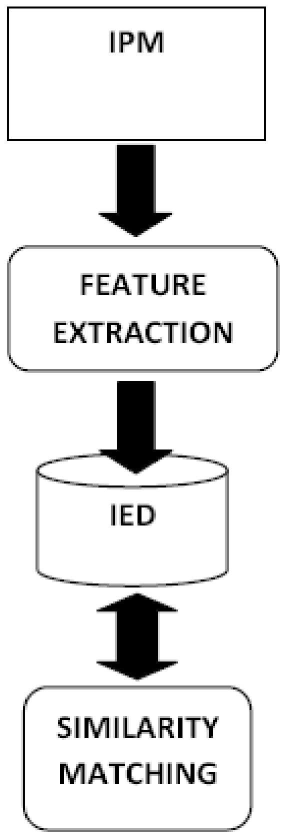


Figure 12. High-Level IEIM Process Flow

In order to summarize, the core design of DECIPHER framework is influenced by functional requirements listed in Table 4 below.

Table 4. Overview of DECIPHER Functional Design Requirements

Limitations with Existing Solutions	Solution Objectives	Theory	Features/ Functionality
1) Existing approaches have no capability of knowing the environmental changes to the organizational database environment stack 2) Existing approaches consider database layer as the only source for changes within the entire organizational database environment stack 3) Existing approaches do	1) The proposed solution needs to have the capability of knowing environmental changes to the organizational database environment stack 2) The proposed solution needs to consider all layers (, e.g., hardware, network, operating system etc.) as source for changes within organizational database environmental stack	1) The tuner's knowledge base is the central component and should provide a common and shared understanding of the whole environment Kephart and Chess (2003) and Wiese and Rabinovitch (2009) 2) The tuner knowledge base should also provide a common and shared	1) IPM's data extraction step extracts environment change information stored within the organizational incident management data stores 2) IPM's data extraction step extracts change information from all organizational environment work flows stores (, e.g., hardware, network, operating

not have capability of knowing the impact of an environmental change 4) Existing approaches do not have capability of knowing the extent of the impact of an environmental change	3) The proposed solution needs to have the capability of knowing the potential impact of an environmental change 4) The proposed solution needs to have the capability to identify the potential extent of an environmental change impact	understanding of problem space (, e.g., organization-level) Kephart and Chess (2003) and Wiese and Rabinovitch (2009) 3) The analyze step of M-A-P-E is responsible for predicting the impact of changes to the environment Kephart and Chess (2003) and Miller (2005) 4) The analyze step of M-A-P-E is responsible for predicting the dependencies	system etc.) stored within the organizational incident management data stores. 3) IPM predicts the potential impact of unimplemented environmental changes based on the historical change information stored within the incident management data stores. 4) IEIM identifies the potential extent or the dependencies for unimplemented changes

	of the impact of changes to the environment Kephart and Chess (2003) and Miller (2005)

A high Level process flow for DECIPHER is shown in Figure 13.

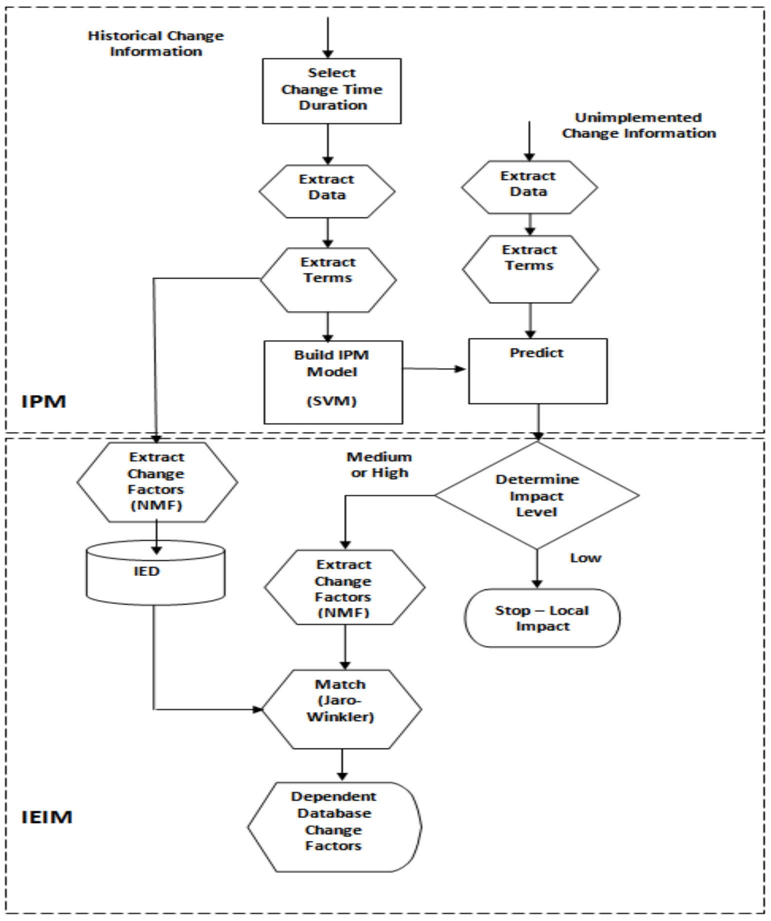

Historical Change Information
Select Change Time Duration
Unimplemented Change Information
Extract Data
Extract Data
Extract Terms
Extract Terms
Build IPM Model (SVM)
Predict
IPM
Extract Change Factors (NMF)
Medium or High
Determine Impact Level
Low
IED
Extract Change Factors (NMF)
Stop – Local Impact
Match (Jaro-Winkler)
IEIM
Dependent Database Change Factors

Figure 13. High Level DECIPHER Process Flow

As seen from the above DECIPHER process flow diagram, there are two streams of processing. On the left hand side, we have the IPM model build and IED build steps. This process stream is responsible extracting the change information from the historical change information corpus based on a specific change time duration and building an IPM impact prediction model after the change information goes through the term extraction process. These terms then go through the feature extraction process. The features along with their feature sets get stored in an Impact Extent Database (IED). This database also stores features that are specific to the database queue.

The right hand side processing stream in Figure 13 shows the scoring and matching features of DECIPHER. In this stream, the new unimplemented change requests undergo the same term extraction processing before scoring. During scoring stage, the IPM model built as part of left hand side processing stream is used to score the new terms. Once these are scored, a decision is made based on the predicted level of impact. For the change factors or terms that have the predicted impact of “Low” are ignored because they represent a localized impact. If the IPM model prediction is “Medium” or “High” then additional processing is done.

The change factors or terms that have “Medium” or “High” predicted impact level undergo feature extraction process. The extracted features are probabilistically matched with factors stored within IED to determine which feature sets have the maximum match. The feature set that has the maximum match with the new features are returned by the matching process along with their features ranked in order of their coefficients. This feature set contains the factors that are dependent on the new implemented change factors. From this feature set, the features that belong to the database queue are returned. These represent the database dependent change factors.

Change Pattern Reoccurrence

This research proposes a new change pattern recurrence metric to identify the contexts in which change impact prediction and matching algorithms will be useful and to help identify the best subset of data to use for change impact prediction model building, matching and scoring.

As shown in Figure 13, there are two streams of processing that represent two different data sets. If the left hand side processing represents the historical data set C_j and the right had side processing represents the new data set C_i then the pattern reoccurrence can be calculated as:

$$\frac{1}{n}\sum_{i=1}^{n} Similarity\left(C_i^{t+1}, C_j^{t}\right)\text{), where } C_j^t \text{ is such that } Similarity\left(C_i^{t+1}, C_j^{t}\right) \geq\ Similarity\left(C_i^{t+1}, C_k^{t}\right) \text{ for } k = 1 \ldots m$$

In the above metric, n is the number of NMF feature sets and m is the number of unique change factors from C_j. The similarity calculation is part of the similarity matching processing shown in Figure 13. This pattern reoccurrence values are stored in IED. The metric can be used for two things:

1) Understanding the extent of reoccurrence of environmental changes over period of time
2) Selection of the optimal change time duration as represented by the “Select Change Time Duration” step in the Figure 13. Selecting the optimal change time duration will help in reducing the data set sizes thereby expediting the data processing, model building, and model scoring and matching processes. This is very crucial in IT environments that undergo large amount of IT environmental changes.

CHAPTER 5

IMPLEMENTATION AND VALIDATION

This chapter presents a detailed discussion of the implementation and evaluation of DECIPHER using a real-world incident management system in a medium sized organization. This chapter is divided into two main sections – DECIPHER implementation and DECIPHER validation. The implementation section begins with the background information on the incident management system used by a medium sized organization and the text mining tool to implement DECIPHER. This chapter then continues with an in-depth discussion on the algorithms, their parameters and steps used by the DECIPHER modules. The validation section of this chapter begins with a discussion on the data sets for DECIPHER modules and the questions that were used for validating the accuracy of DECIPHER's prediction.

DECIPHER Implementation

In this section we will review the implementation details for DECIPHER. The implementation of DECIPHER uses a real-world incident management system called "Request Tracker" used by a medium sized organization and Oracle data miner as the text-mining mining tool. This organization has close to 5000 employees worldwide and on average experiences 200-300 IT environmental changes per month.

Request Tracker

Request Tracker (RT) is an open source web-based incident and a workflow management system that has been widely used by several organizations ranging from fortune 500 companies to government agencies under various implementations like bug-tracking, help-desk system, change management etc. (Practical, 2011). Its simplicity, extensibility and ease of customization with general public license (GPL) make it an ideal and powerful issue tracking tool for many organizations.

A single unit of work is captured as a “Ticket” within Request tracker. Change Management system’s typically have the following fields that capture the change information:

1) Ticket #
2) Queue
3) Ticket Subject (Text)
4) Change Purpose (Text)
5) Creator of the ticket
6) Owner of the ticket.
7) Approver of the ticket.
8) Last Update User
9) Start Date
10) End Date
11) Classification (Regulatory Compliance)
12) Line of Business
13) Work-plan for the change (Text).

14) Impact (High or Medium or Low).

15) Impact Description (Text)

16) Back-out plan (Text).

17) General Comments/Notes (Text)

A screenshot of RT’s change request user interface for the organization under study is shown in Appendix B. Impact field has the typical change management impact values , i.e., “Low”, “Medium” and “High”. From the DECIPHER model building perspective, the impact field is dependent or target attribute. Every ticket has a unique identifier that is represented by the Ticket # field. As part of the implementation, this field is referred as case_id. All other variables are Independent or predictor variables. Request tracker also has the ability to connect to any leading database for storing the change information. In this implementation, the relational store that has the RT information is Oracle RDBMS. Given that the change information data was already available within an Oracle database, using Oracle data miner was the natural and convenient choice.

Oracle Data Miner

Oracle Data Miner is a free User Interface (UI) extension to Oracle’s free Integrated Development Environment (IDE) – SQL Developer. The screen shots of Data Miner/SQL Developer are shown in Appendix D section. The strength of the tool is its ability in providing knowledge discovery using native SQL functions right inside the database. Furthermore, Oracle data miner provides powerful algorithms for both structured and unstructured data. DECIPHER’s text mining algorithms and its pre-processing steps are implemented using Oracle Data Miner tool. The reasons for choosing this tool are two-fold:

1) The Request Tracker's database used by the organization in this study is an Oracle database which makes it convenient and efficient for the implementation because the data does not have to be moved from the source to the compute environment.

2) Oracle data miner provides a powerful suite of battle-tested text-mining algorithms that support in-database knowledge discovery and out-of-the-box scalability (Oracle, 2011a). One of the design requirements for IPM and IEIM is the need for high performance model building and scoring. Oracle data miner's in-database text mining architecture helps in that regard. Oracle's in-database mining avoids the traditional data or text mining step of moving of data between the source system and compute environment. Furthermore, using the massively parallel architectures of Oracle's database engine, and its advanced memory management techniques, DECIPHER's text-mining algorithms can be processed efficiently and closer to the data (Inchiosa, 2011; Oracle, 2011a). Furthermore, Oracle's grid database architectures provide the flexibility for organizations to scale out based on their business needs (Hamm and Burleson, 2006; Serpa, Roncero, Costa, and Ebecken, 2008).

Oracle data miner has a workflow-driven UI and in a workflow, each element is represented by a graphical icon called "node". Each node has a specific function along with its properties. These nodes when linked together form a modeling process to solve a specific data mining problem. Nodes and links can be simply dragged and dropped from the component palette. The component palette is shown in the Appendix D. Next, let us review the implementation details of DECIPHER modules – Impact Prediction Module (IPM) and Impact Extent Identification Module (IEIM).

Impact Prediction Module (IPM):

At a high level, the IPM implementation involves three major tasks – Data Extraction Term Extraction, and Model Building/Scoring. This is shown in Figure 10. Data Extraction step involves extraction of the textual change information from historical organizational incident management data stores for all environment support workflows data stores that represent the layers with the database environment stack so that change information from all the layers of the environment are taken into account. Request Tracker has an impact field with values - “Low”, “Medium”, and “High”. IPM considers this impact field as dependent or target attribute and all other fields as Independent or predictor variables. Organizational change information corpus may consist of several fields. Having too much information can negatively affect IPM’s performance and accuracy. Some of the fields within the change information corpus may not provide meaningful information to the model building process and can act as a noise factor increasing the size of the model and the amount of resources needed to build and score the model. A feature selection process addresses this issue.

For the IPM implementation, attribute importance function is used for feature selection. Attribute importance function ranks attributes in the change information corpus according to their significance in predicting the target which is the impact field (Campos, Stengard, and Milenova, 2005). Oracle data miner uses Minimum Description Length (MDL) to implement the attribute importance function. MDL assumes that a simple as well as a compact representation of data is best and likely explanation of data (Rissanen, 2004). The attribute importance settings and output for IPM for some of the Request Tracker fields is shown in Table 5.

Table 5. IPM Attribute Importance Results for 2008-2011

Rank	Ticket Field	Importance
1	Creator	0.2250
2	Owner	0.1843
3	Last Update User	0.1741
4	Workplan	0.0716
5	Change Purpose	0.0683
6	Start Date	0.0681
7	End Date	0.0634

8	Backout Plan	0.0533
9	Ticket Subject	0.0519
10	Approver	0.0393
11	Impact Description	0.0310
12	General Comments	0.0281
13	Ticket #	0.0227
14	Queue	0.0087
15	Classification	0
16	Line of Business	0

The output of attribute importance has the following two main indicators (Campos et al., 2005):

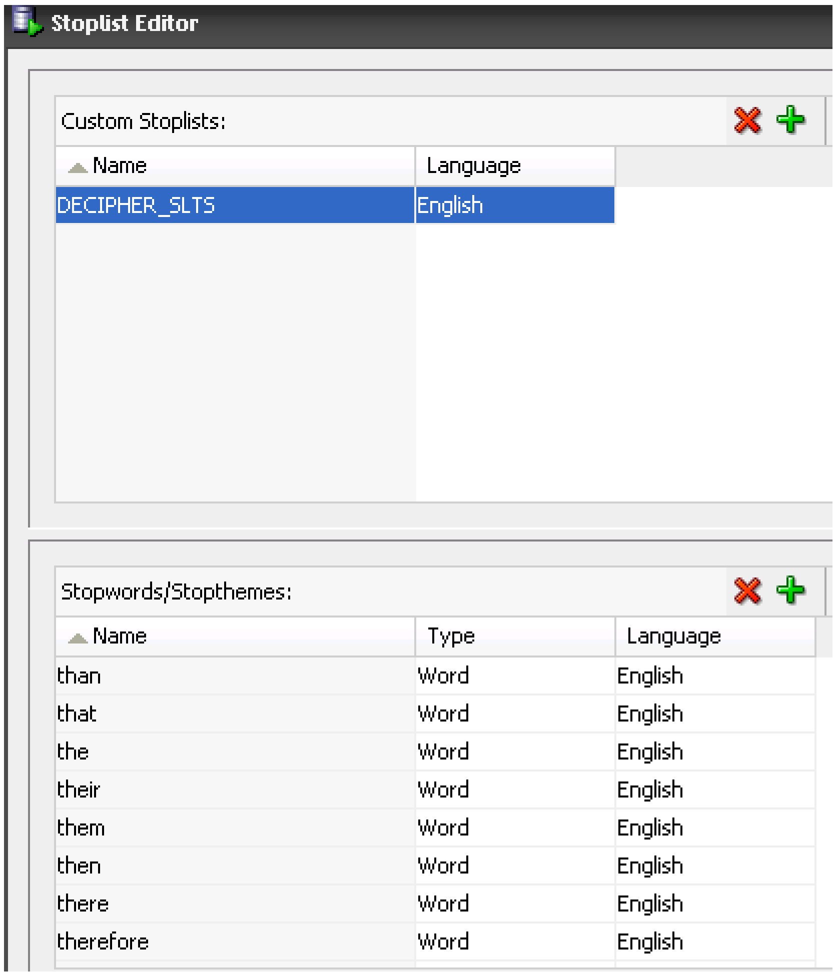
1) Measure of explanatory power: This indicates how useful the attribute is to determining the value of the explained column. This is shown by the “Importance” column in Table 4. Values range from 0 to 1. Higher values indicate greater explanatory power.

2) Measure of relative importance: This indicates the attributes relative importance compared to other attributes.

For the IPM implementation, attributes with importance greater than zero are only selected. Negative values show presence of noise and hence fields that have importance of zero or less indicate non-significant contribution need to be removed for IPM’s term extraction process. Based on the results in Table 5, classification and line of business fields are removed from further processing since they have insignificant influence to the target attribute. Also, as seen from the Table 5, the top three important attributes based on the rank and the importance column are the creator, owner and the last update user fields in the change ticket. The creator field is the creator of the change ticket and the owner is the person who owns this task and the last update user who lasts updates the ticket before closing it. Typically these three are the same person but in some cases like when the actual impact of the change was found to be different from the perceived impact, the ticket might be updated by a different person who is responsible of executing that particular change.

Having these fields at the top three positions, implies that knowing the person who creates, executes and updates the ticket last, helps determine the level of the impact. Typically, the creator of a change ticket is also the person who implements the plan for executing the change, including the understanding of the perceived impact and the steps needed to roll back the change if needed. Since the creator of the ticket is at the top position, it shows that the person who creates the ticket is comparatively most helpful in determining the impact level. For example, a more experienced DBA will typically be handling complex changes while a less experienced DBA might work on systems that are less critical to the organization. The workplan field describes the step by step plan for the change. This implies that knowing the terms that describe the plan for the change helps in knowing what the impact might be. For example, this workplan field might lay out how one of the critical applications, say, ERP_HR is being taken down for an upgrade. Similarly, the terms used in change purpose that describe the motive behind the change helps understand the impact level.

In this implementation of IPM, each Request Tracker ticket’s unstructured fields are transformed into a vector space model (term vectors or environmental change factor vectors) using the term extraction process. The term extraction process using Oracle data miner leverages stop lists, stemming and term weighting. Stop lists contain words, called as stop-words, are the common words that are found in change tickets within Request tracker, e.g., “and”, “the”, “or”. Using Oracle text technology, Oracle data miner creates a stop list that can be easily enhanced to make the IPM’s term extraction process context-aware under various organizational settings (Grivolla, 2005). Figure 14 shows the screen shot of Oracle Data Miner’s stop list editor.



Custom Stoplists:

Name	Language
DECIPHER_SLTS	English

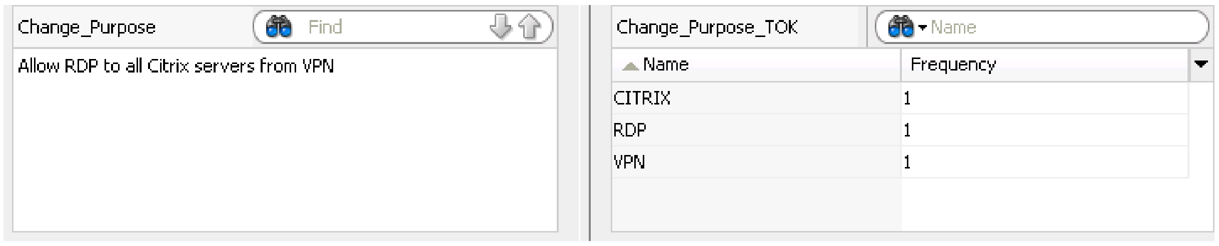
Stopwords/Stopthemes:

Name	Type	Language
than	Word	English
that	Word	English
the	Word	English
their	Word	English
them	Word	English
then	Word	English
there	Word	English
therefore	Word	English

Figure 14. IPM Stop List Editor

Oracle data miner also uses a term-weighting technique to count how many times a term is used within a ticket. Specifically, Oracle data miner uses term frequency–inverse document frequency (TF-IDF) measure. This measure shows how important a term is to a change ticket with respect to the change information corpus.

Figure 15, shows a screen shot of TDIF output for one of the fields called “Change Purpose” within the Request tracker.



Change_Purpose

Allow RDP to all Citrix servers from VPN

Change_Purpose_TOK

Name	Frequency
CITRIX	1
RDP	1
VPN	1

Figure 15. IPM Term weighting Output Example

Once IPM's term extraction processing is done, the unstructured fields within Request tracker are stored within the database. This transformed text is now ready to be used as any other attribute in the building, testing, and scoring of models. IPM's function form can be represented as

$$F(x_1,x_2\ldots x_n) = IP$$

where $x_1,x_2,\ldots x_n$ are the terms that are fed into the IPM after the term extraction process and IP is the prediction impact with values "Low", "Medium" and "High".

IPM's input context can be represented as:

$$\text{IPM Input Context} = (T, TN)$$

where T is the finite set of historical change tickets (, e.g., last 4 years) within an organizational change information corpus that have already been executed and is represented as:

$$T= \{t_1,t_2,\ldots, t_n\}.$$

TN is finite set of new change tickets that are captured in real-time but not yet implemented and is represented as:

$$TN = \{tn_1, tn_2, \ldots, tn_n\}.$$

The change information within the tickets is stored in unstructured format. Every new ticket has a perceived Impact value (Low, Medium, and High) that is entered by the change executor. Once the ticket is executed and if its impact is found to be different than what was perceived, the change requester updates the ticket with the actual impact value.

IPM’s model building and scoring tasks is implemented using Oracle data miner’s support vector machine (SVM) classification algorithm. These are supervised or directed learning algorithms that work with both structured and unstructured fields of Request tracker. These set of algorithms assign the items in the corpus to the target classes. In our implementation the change impact is the target attribute for the classification algorithm with classes – Low, Medium and High. All other fields are independent attributes. Since we have three possible values for our target attribute, we will use multi-class classification algorithms.

In IPM’s model building and scoring, Support Vector Machines (SVM) classifier is used. In the model build, the classifier finds the relationships between the values of the independent attributes and the values of the target attribute (Oracle, 2011a). SVM is a powerful algorithm based on statistical learning theory (Milenova, Yarmus, and Campos, 2005). SVM also has strong regularization properties, especially on complex problems like posed by unstructured data types (Guduru, 2006; Oracle, 2011a). Regularization means the generalization of the classification model to the new change data.

SVM models have a similar functional form to radial basis functions and neural networks but compared to these approaches, SVM models have strong theoretical approach to regularization which is the key to IPM’s effectiveness to predict impact of new environmental changes (Dumais, 1998; Milenova et al., 2005). Furthermore, one of the strengths of SVM that is important from the IPM perspective is the number of attributes that it can handle without negatively affecting the performance. Organizational change information corpus can have large number of fields based on their environmental complexity. SVM performs well on data sets involving many attributes despite the fact that there might be few cases available to train the model (Milenova et al., 2005; Oracle, 2011a). The SVM scalability is dependent on IPM's compute environment.

For creating a predictive model, the unstructured text data in T is extracted into an attribute vector and is represented as:

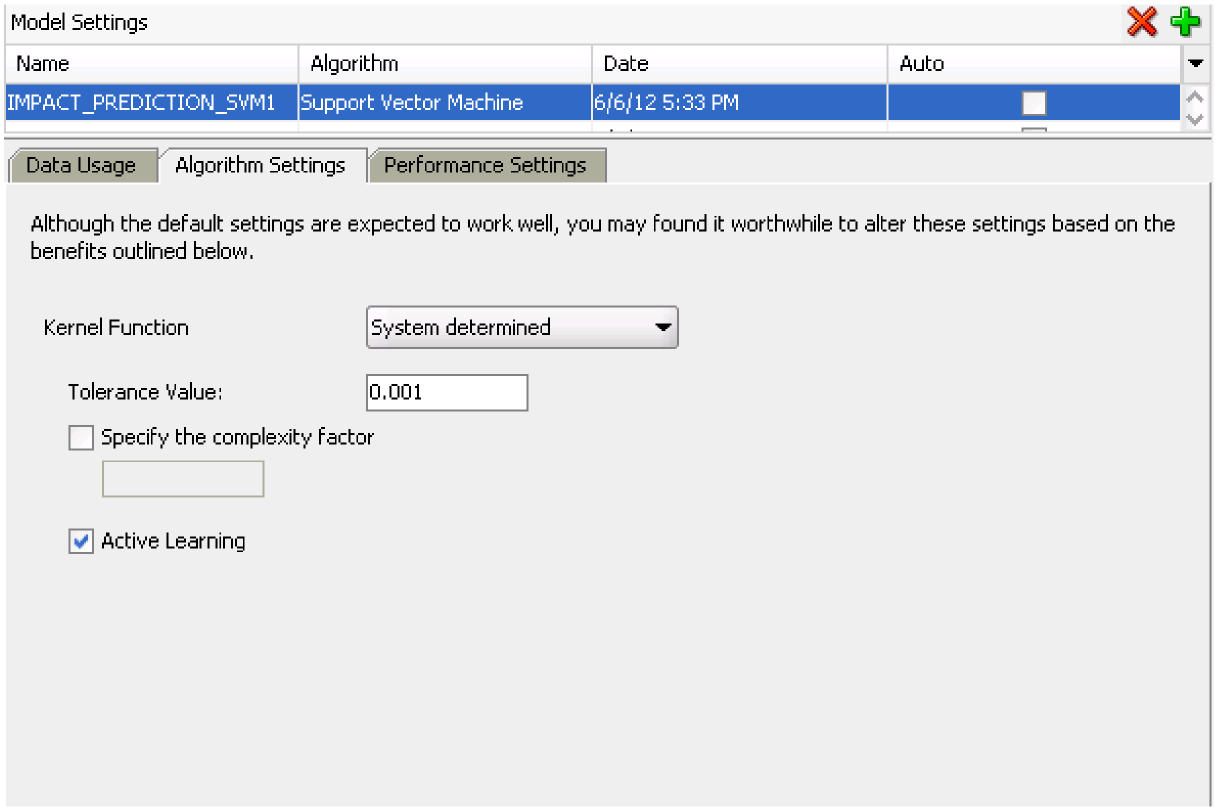
$$X = \{x_1, x_2 \ldots x_n\}$$

where X belongs to an n-dimensional space R^n where $x_1, x_2, \ldots x_n$ are components of vector X. Output of SVM is given by:

$$f_i = b + \sum_{j=1}^{m} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i)$$

where *fi* is the distance of each point to the decision hyper plane defined by setting $f_i = 0$; b is the intercept; α_j is the Lagrangian multiplier for the jth training data record x_j; and y_j is the corresponding target value (±1) (Milenova et al., 2005). K is a kernel function that can be linear or non-linear. In case of non-linear kernel, then the above equation defines a linear equation on a new set of attributes that can be as many as the number of rows in the training data, making SVM very powerful (Milenova et al., 2005). The input attributes with non-zero α_j are called support vectors. In case of linear kernel, the above equation is simplified wherein the decision hyper plane is defined in terms of input attribute coefficients alone.

The process of learning in SVM is basically estimating the values of α_j which is achieved by solving a quadratic optimization problem. For real-time scoring performance, SVM's active learning helps by optimizing the selection of a subset of the support vectors (using target stratified sampling) that maintain accuracy while enhancing the speed of the model. Furthermore, it increases performance and reduces the size of the kernel thereby improving its scalability (Milenova et al., 2005). Also, active learning forces the SVM algorithm to restrict learning to the most informative examples and not to attempt to use the entire body of data (Oracle, 2011a). Oracle data miner's SVM algorithm settings screenshot is shown in Figure 16.


Model Settings

Name	Algorithm	Date	Auto
IMPACT_PREDICTION_SVM1	Support Vector Machine	6/6/12 5:33 PM	☐

Although the default settings are expected to work well, you may found it worthwhile to alter these settings based on the benefits outlined below.

Kernel Function: System determined

Tolerance Value: 0.001

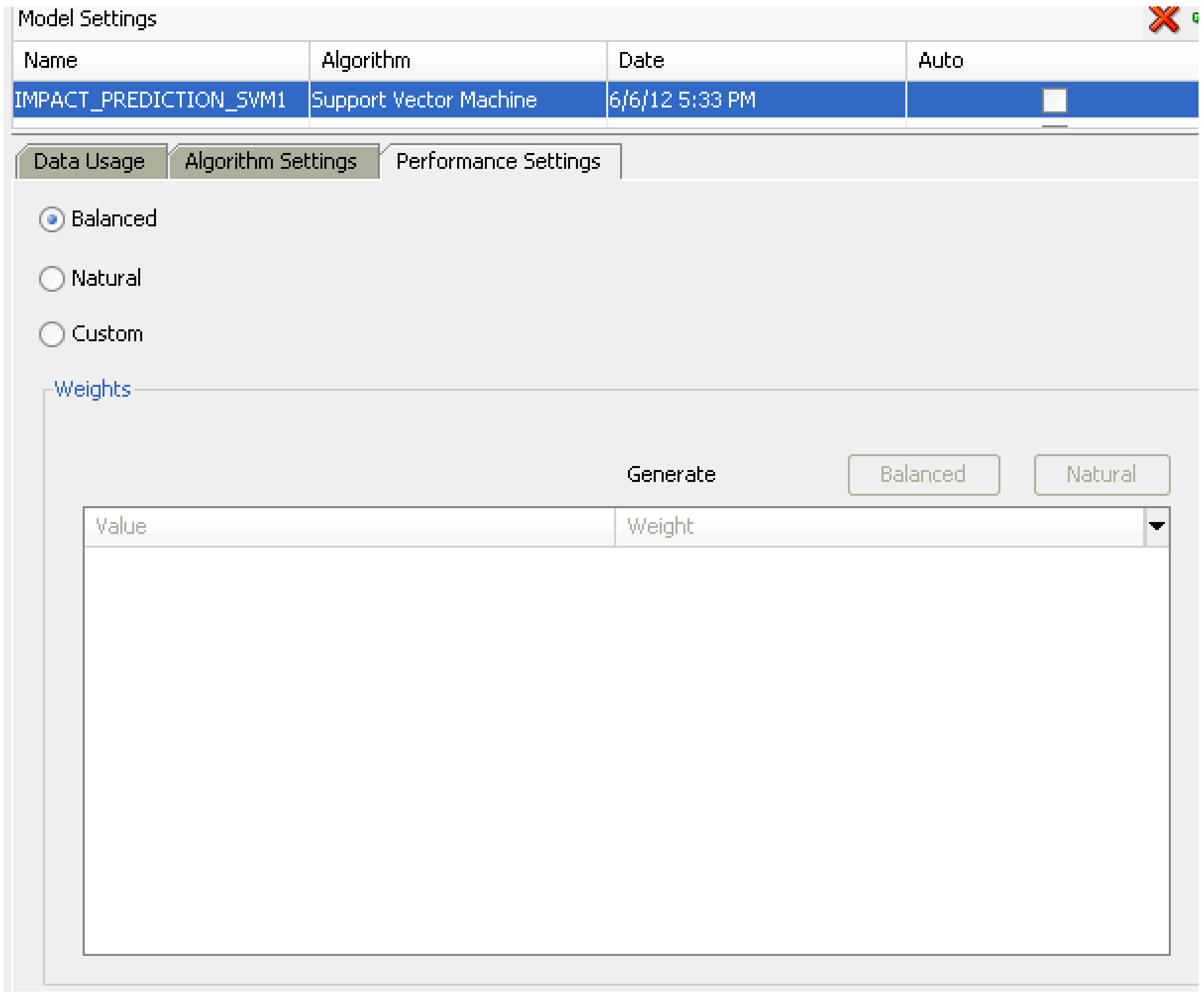
☐ Specify the complexity factor

☑ Active Learning

Figure 16. Oracle Data Miner SVM Algorithm Settings

The convergence tolerance value is the maximum size of a convergence criterion violation such that the model is considered to have converged (Guduru, 2006). It is a user-defined function that specifies for a SVM model to be considered as converged. Lower tolerance values results in a more accurate classification model at the expense of longer processing time (Milenova et al., 2005).

Complexity factor setting for SVM decides the trade-off between minimizing model error on training data and minimizing the complexity of the model (Dumais, 1998; Guduru, 2006). If the SVM model is very complex than it fits the noise present in the training data and on the other hand a SVM model that is very simple under-fits the training data (Guduru, 2006; Milenova et al., 2005). A large value of the complexity factor leads to high penalty on errors and a small value leads to low penalty on errors that can lead to under-fit. For IPM implementation, no value is specified, which implies that the value of complexity factor is automatically determined by the system. This is the default and the recommended approach for Oracle data miner (Oracle, 2011a). The same approach is adopted for kernel function for SVM.


Model Settings

Name	Algorithm	Date	Auto
IMPACT_PREDICTION_SVM1	Support Vector Machine	6/6/12 5:33 PM	☐

Data Usage | Algorithm Settings | Performance Settings

(●) Balanced
( ) Natural
( ) Custom

Weights

Generate [Balanced] [Natural]

Value	Weight

Figure 17. Oracle Data Miner SVM Performance Settings

Figure 17 shows the performance setting screen of Oracle Data Miner's SVM algorithm. Basically, there are three settings – Balanced, Natural and Custom. Balanced is the default setting that attempts to achieve best overall accuracy across all values of impact attribute. Under this setting, the model build process is biased using the weight values that provide extra weight to impact attribute values that occur less frequently. Natural performance setting builds the model without any biasing and under this setting the model uses it natural view of the data to build an accurate model. The downside of this setting is that impact attribute values that are rare will not be predicted as frequently compared to the model that was built with balanced setting. Custom performance setting for SVM model allows the user to enter a set of weights for each impact attribute values.

IPM's Output Context is represented as:

$$IPM\ Output\ Context = (SF, IP)$$

where *SF* is finite set of change terms scored in real-time using the built SVM model and is represented as:

$$SF = \{sf_1, sf_2, \ldots, sf_n\}.$$

IP is the predicted impact of the terms and is represented as:

$$IP = \{low, medium, high\}.$$

Impact Extent Identification Module Implementation

IEIM implementation, as shown in Figure 12, involves two major steps - Feature Extraction and Similarity Matching. Feature Extraction step implements Non-negative Matrix Factorization (NMF) algorithm to extract features from the term vector that was created as part of the data transformation stage of IPM. NMF is found to very effective in text mining domains compared to other feature extraction algorithms (Guduru, 2006; D. Lee and Seung, 1999).

NMF algorithm decomposes a text data matrix A_{mn} where columns are tickets and rows are terms , into the product of two lower rank matrices W_{mk} and H_{kn} represented by the below equation (Guduru, 2006). To prevent cancellation, NMF expects A_{mn} and H_{kn} have non-negative entries. NMF algorithm employs an iterative procedure to modify the initial values of W_{mk} and H_{kn} so that the product approaches A_{mn} (Guduru, 2006; D. Lee and Seung, 1999). The procedure terminates based on error convergence value or when the specified number of iterations is reached. Each user-defined feature after NMF decomposition is a linear combination of the original attribute set and has non-negative coefficients.

The matrix decomposition can be represented as:

$$A_{mn} = W_{mk} \ x \ H_{kn}.$$

where

A_{mn} : (mxn) matrix: m nonnegative values of n text tickets,

W_{mk} : (mxk) matrix: k columns of W feature vectors,

H_{kn} : (kxn) matrix: each column of H is called weight column.

Matrix A represent the change information corpus such that A_{ij} is the number of times the ith word appears in the jth ticket (Guduru, 2006). Oracle data miner's NMF implementation is based on the multiplicative update algorithm by Lee and Seung (1999) wherein the algorithm iteratively updates the factorization based on an objective function (Guduru, 2006; Wild, Curry, and Dougherty, 2003). The general objective function is to minimize the Euclidean distance between each column of matrix A_{mn} and its approximation $A_{mn} \sim W_{mk} \ x \ H_{kn}$ (Guduru, 2006). The objective function is shown as below (Guduru, 2006; Wild et al., 2003):

$$\Theta_E(W,H) \equiv \sum_{j=1}^{n} \left\| a_j - Wh_j \right\|_2^2 = \left\| A - WH \right\|_F^2 \equiv \sum_{i=1}^{m} \sum_{j=1}^{n} \left(A_{ij} - \sum_{l=1}^{k} W_{il} H_{lj} \right)^2$$

The following multiplicative update rules are used for monotonic convergence (Guduru, 2006; D. Lee and Seung, 1999):

$$H_{aj} \leftarrow H_{aj} \frac{\left[W^T \mathrm{A}\right]_{aj}}{\left[W^T W H\right]_{aj}}$$

$$W_{ia} \leftarrow W_{ia} \frac{\left[A H^T\right]_{ja}}{\left[W H H^T\right]_{ja}}$$

The number of feature vectors *k* is user-defined and decided the accuracy of the approximation (Guduru, 2006; D. Lee and Seung, 1999; Oracle, 2011a). The feature sets along with the features (change factors) and their NMF coefficients are saved in the Impact Extent Database (IED).

Input Context of IEIM is represented as

Input Context = (X, CPI)

where CPI is the classified impact for historical change factors by IPM and is represented as*:*

CPI = {medium, high}.

CPI impacts with "Low" values are ignored for IEIM analysis because they represent localized change impact. X belongs to an n-dimensional space R^n where $x_1, x_2, \ldots x_n$ are components of vector X from IPM. Output context at this stage of IEIM is represented as:

Output Context = (CF, FEID, CO).

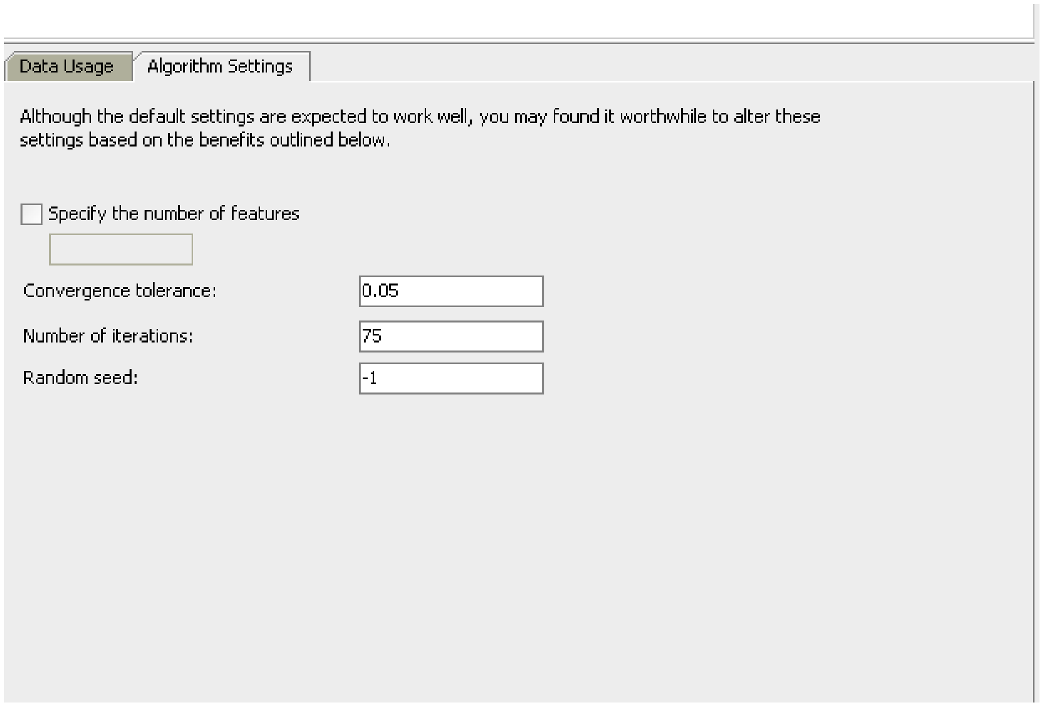
where CF is finite set of change factors that are extracted using NMF; $CF \subseteq X$ and is represented as:

$$CF = \{cf_1, cf_2, \ldots, cf_n\}.$$

FEID is the NMF feature set ID that the *CF* is a member of and *CO* are the NMF coefficients for *CF* represented as:

$$CO = \{c0_1, c0_2, \ldots, c0_n\}.$$

Figure 18 shows the Oracle data miner's NMF algorithm settings



Although the default settings are expected to work well, you may found it worthwhile to alter these settings based on the benefits outlined below.

☐ Specify the number of features

Convergence tolerance:	0.05
Number of iterations:	75
Random seed:	-1

Figure 18. Oracle Data Miner NMF Algorithm Settings

The number of features setting shows the number of feature vectors or *k* for the IEIM's NMF implementation. As part of the IEIM implementation, default value for this is selected. This implies not specifying any value for this setting as shown in the Figure above. In this case, Oracle data NMF algorithm automatically determines the number of features based on computation complexity and the distribution of change factors in each feature set (Oracle, 2011a). Convergence tolerance setting indicates the minimum value. Number of iterations specifies the maximum number of iterations for the NMF algorithm. Random seed is the random seed for the sample. The default value of -1 is used for the IEIM implementation.

The similarity matching step of IEIM probabilistically matches the unimplemented changes with the change factors stored in IED using the Jaro-Winkler distance similarity algorithm (Winkler and Nov, 2006) . The Jaro-Winkler is measure of similarity between strings. In case of IEIM, the similarity is measured between the change factors stored in a repository referred to as Impact Extent Database (IED) with the change factors extracted from the unimplemented change tickets.

Similarity functions such as Jaro-Winkler map a pair of strings *s* and *t* to a real number *r,* where a larger value of r indicates greater similarity. Jaro-Winkler metric is given by (Cohen, Ravikumar, and Fienberg, 2003; Winkler and Nov, 2006):

$$\mathit{Jaro\text{-}Winkler}(s,t) = \mathit{Jaro}(s,t) + \frac{P'}{10} \cdot (1 - \mathit{Jaro}(s,t))$$

where,

$$\mathit{Jaro}(s,t) = \frac{1}{3} \cdot \left(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - T_{s',t'}}{|s'|} \right)$$

where,

$s' = a'_1 \ldots a'_K$ are the characters in string *s* that are common with *t* and

$t' = b'_1 \ldots b'_L$ are the characters in string t that are common with *s*

P = *length* of the longest common prefix of *s* and *t and*

$P' = \max(P, 4)$

IEIM uses Oracle's implementation of Jaro-Winkler to achieve in-database matching. This ensures that matching can leverage database system's massive parallel processing and efficient memory management capabilities. Oracle Jaro-Winkler is a function call using Oracle's native PL/SQL programming language (Oracle, 2011b). The syntax of this function is shown below:

```
UTL_MATCH.JARO_WINKLER_SIMILARITY (
                s1 IN VARCHAR2,
                s2 IN VARCHAR2)
                RETURN PLS_INTEGER;
```

S1 and S2 are the strings that serve as the input to this function. An example use of this function is as following (Oracle, 2011b). In this example, the function compares two strings "shackleford" and "shackelford" and returns a score 0 (no match) and 100 (perfect match).

```
SELECT UTL_MATCH.JARO_WINKLER_SIMILARITY('shackleford', 'shackelford')
FROM DUAL;
--------------
        returns 98
```

IEIM implementation of this function call is within a procedural context. A PL/SQL block loops through all the change factors that are stored in IED and compare the factors with change factors from the unimplemented change ticket to find a similarity match. Score of more than 90% is considered for IEIM.

The features set whose change factors have maximum number of matches with the new features are returned along with the change features ranked by their coefficients. The change factors within this feature set, excluding the ones that matched, identify the factors that are dependent on the new changes.

Input Context at this step is represented as:

Input Context = (CF, FEID, CO) and *IEIM Output context = (DCF, FEID, CO)*

where *DCF is represented as:*

$$DCF = \{dcf_1, dcf_2, \ldots, dcf_n\}.$$

DCF is finite set of database dependent change factors that are extracted using NMF for the database queue that has maximum matched change factors with the new change factors such that $DCF \subseteq CF$.

Using in-database feature extraction and similarity matching algorithms, IEIM scalability is achieved. Also, similar to IPM, the optimal change time duration will help in the performance of feature extraction algorithm as well as similarity matching algorithms.

DECIPHER Validation

Impact Prediction Module (IPM) Validation: The core function of IPM is predicting the potential impact for environmental changes that have not yet been implemented in real-time. Before IPM can score new environmental changes, the accuracy of IPM needs to be validated for accuracy. This section validates the IPM accuracy by answering the following two questions:

1) How accurately does IPM predict the impact of environmental changes?

2) How does SVM's prediction accuracy compare with another classifier for impact prediction?

In order to validate these two points for IPM, cases or tickets from Request Tracker (RT) filed between 2008 -2011 were used to train and test the IPM model. 60% of the data was used for training the model and 40% for testing the model in a random fashion(Bramer, 2007). Text mining is a computationally expensive process. The following server and software configuration was used for this testing:

1) Sun SPARC V490

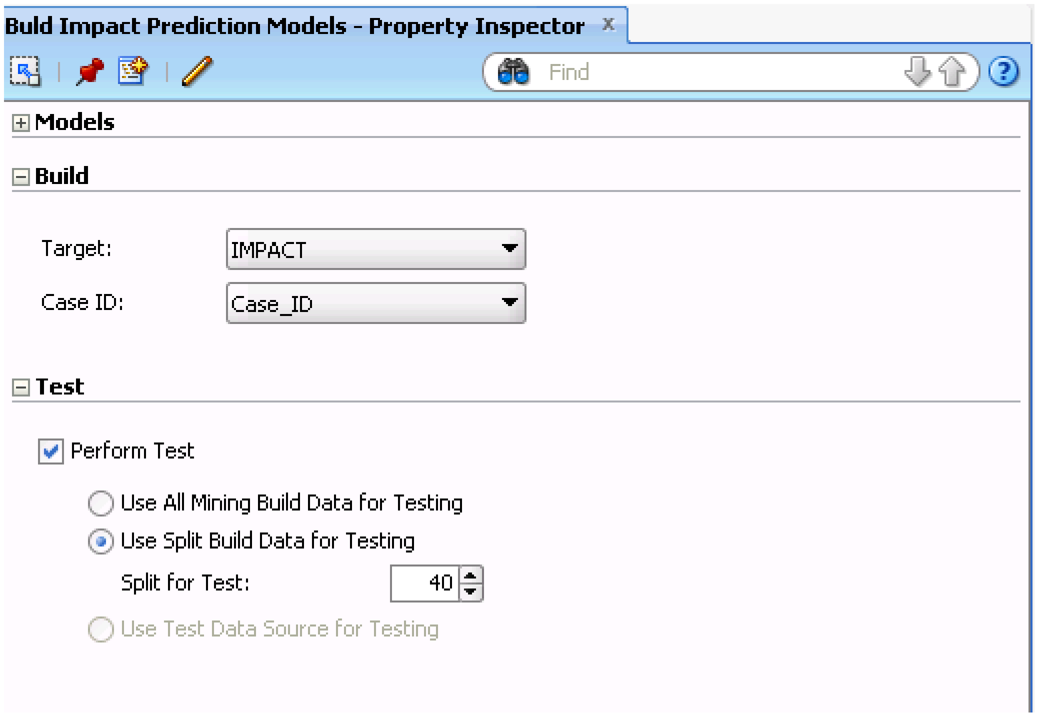
2) 8 CPU cores

3) Solaris 10 Operating system

4) 32g RAM

5) Oracle RDBMS 11.2.0.3

The tickets were considered from the RT change information corpus, i.e., RT change management database across different organizational teams. This is achieved by extracting the tickets from all Request Tracker workflow queues (Practical, 2011). This enables us to use the changes from all layers, shown in Figure 1, for the analysis. Figure 19 show the Oracle data miner's SVM model train/test setting as part of IPM Model building. As seen in this Figure, Target is the IMPACT column from Request Tracker and Case ID is Ticket # field from Request Tracker.


Buld Impact Prediction Models - Property Inspector

Models

Build

Target: IMPACT

Case ID: Case_ID

Test

[✓] Perform Test

( ) Use All Mining Build Data for Testing

(●) Use Split Build Data for Testing

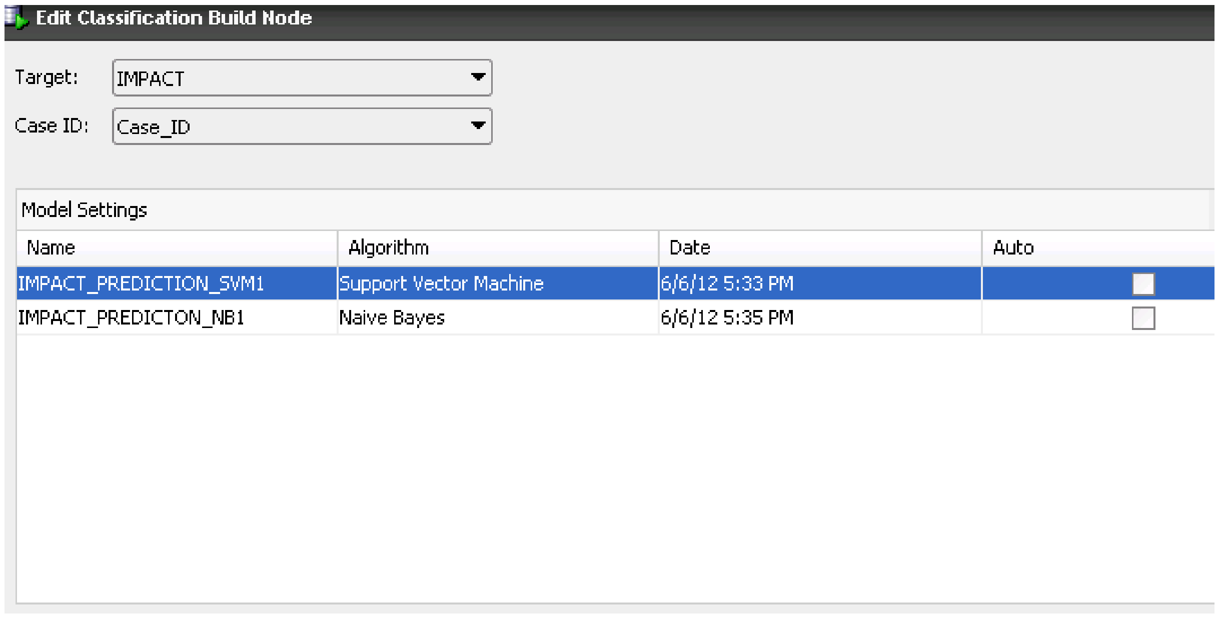
Split for Test: 40

( ) Use Test Data Source for Testing

Figure 19. Oracle Data Miner IPM Model Train/Test Settings

In order to validate SVM's prediction accuracy, its results will be compared with Naïve Bayes classifier. Oracle data miner makes it relatively easy to do this. In order to compare multiple algorithms, during classification model build process, Oracle data miner provides a way to select various models. Figure 20 shows the IPM's Classification build node.

If multiple models are selected then Oracle data miner, executes them during the build process which includes the model test/train process as well for those models.


Edit Classification Build Node

Target: IMPACT

Case ID: Case_ID

Model Settings

Name	Algorithm	Date	Auto
IMPACT_PREDICTION_SVM1	Support Vector Machine	6/6/12 5:33 PM	☐
IMPACT_PREDICTON_NB1	Naive Bayes	6/6/12 5:35 PM	☐

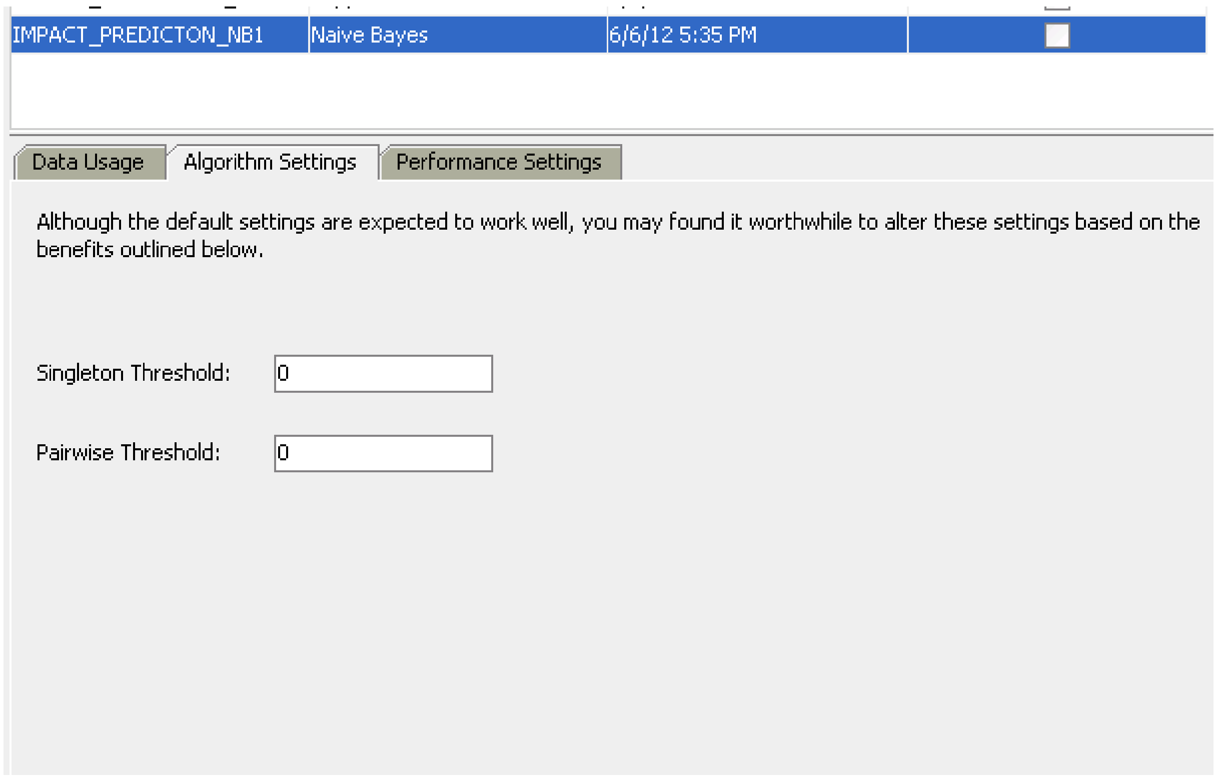
Figure 20. IPM Multiple Model Selection

Naïve Bayes is founded on the principle of conditional probabilities. This algorithm uses the Bayes' Theorem which basically computes the probability by counting the frequency of values and the combinations of the values in the data (Bayes and Price, 1763; Oracle, 2011a). Bayes Theorem evaluates the probability of an event occurrence given the probability of another even that has occurred in the past. Bayes Theorem can be represented as following (Bayes and Price, 1763; Oracle, 2011a):

$$\text{Prob (B given A)} = \text{Prob (A and B) / Prob (A)}$$

where, B is the dependent event and A represents the event that has occurred in the past.

Oracle Data Miner Naïve Bayes algorithm settings is shown in the Figure 21. There are two settings – Singleton Threshold and Pair wise Threshold. Singleton threshold setting specifies the minimum percentage of singleton occurrences required for the inclusion of a predictor in the model. Pair wise Threshold specifies the minimum percentage of pair wise occurrences required for the inclusion of a predictor in the model. For IPM validation, both these settings are at default values of 0.



IMPACT_PREDICTON_NB1	Naive Bayes	6/6/12 5:35 PM	

Although the default settings are expected to work well, you may found it worthwhile to alter these settings based on the benefits outlined below.

Singleton Threshold: 0

Pairwise Threshold: 0

Figure 21. Oracle Data Miner Naïve Bayes Algorithm Settings

Impact Extent Identification Module (IEIM) Validation: The core function of IEIM is that of identifying the potential extent of impact of change factors that are not yet implemented. The validation for IEIM involves validating the accuracy of the identification of potential extent of change factors. This involves, answering the following two questions:

1. Do patterns (change factors) reoccur in future years, and to what extent?

2. What is the time interval across which pattern (change factor) reoccurrence is the maximum and how does this impact prediction accuracy?

For IEIM model validation, two data sets will be created across various time intervals. The pattern reoccurrence will be measured using the change pattern reoccurrence metric explained in chapter 4. The pseudo code of pattern reoccurrence metric is described in Appendix C. This pattern reoccurrence metric will help us answer the above two questions. Also, this metric will help us in identifying the contexts in which change impact prediction and matching algorithms will be useful and to help identify the best subset of data to use for change impact prediction model building, scoring and matching.

Answering both these questions requires data sets that are across time intervals. Following data sets are used for IEIM validation:

a) 2009 (Q1) with 2010 (Q1)

b) 2009 (Q1 and Q2) with 2010 (Q1 and Q2)

c) 2009 (full year) with 2010 (full year)

d) 2008 and 2009 (full years) with 2010 (full year).

In order to validate the similarity matching accuracy, Jaro-Winkler's matching will be compared to that of Levenshtein. Oracle databases' native Levenshtein function call will be used for this (Levenshtein, 1966). Levenshtein distance between two strings a and b is represented as (Levenshtein, 1966):

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} 0 & , i = j = 0 \\ i & , j = 0 \text{ and } i > 0 \\ j & , i = 0 \text{ and } j > 0 \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j)+1 \\ \operatorname{lev}_{a,b}(i,j-1)+1 \\ \operatorname{lev}_{a,b}(i-1,j-1)+[a_i \neq b_j] \end{cases} & , \text{ else} \end{cases}$$

Similar to Jaro-Winkler, Levenshtein matching also leverages the database system's massive parallel processing and efficient memory management capabilities. Oracle Levenshtein is a function call using Oracle's native PL/SQL programming language (Oracle, 2011b). The syntax of this function is shown below:

```
UTL_MATCH.EDIT_DISTANCE_SIMILARITY (
s1 IN VARCHAR2,
s2 IN VARCHAR2)
RETURN PLS_INTEGER;
```

S1 and S2 are the strings that serve as the input to this function. An example use of this function is as following (Oracle, 2011b). In this example, the function compares two strings "shackleford" and "shackelford" and returns a score 0 (no match) and 100 (perfect match).

```
SELECT UTL_MATCH.EDIT_DISTANCE_SIMILARITY('shackleford', 'shackelford')
FROM DUAL;
--------------
    returns 82
```

CHAPTER 6

RESULTS AND DISCUSSION

This chapter presents the DECIPHER results that demonstrate its accuracy as well as metrics that identify conditions under which the system will perform effectively. This chapter starts with a discussion on DECIPHER's IPM results based on the questions that were covered in the previous chapter. The chapter then continues with the IEIM results also based on the questions covered in the previous chapter. This chapter concludes with a discussion on the impact of IEIM results on the overall DECIPHER accuracy and on the performance of its model building, scoring and matching tasks.

IPM Results

The core function of IPM is predicting the potential impact for environmental changes that have not yet been implemented in real-time. This section discusses the results of IPM with respect to the following two questions. The goal behind these questions is to validate the prediction accuracy of IPM:

1) How accurately does the IPM predict the impact of environmental changes?

2) How does SVM's prediction accuracy compare with another classifier for impact prediction?

These questions are answered using the Oracle data miner's performance and accuracy metrics. In order to answer these two questions for IPM, cases or tickets from Request Tracker (RT) filed between 2008 -2011 were used to train and test the IPM model. This duration resulted in 11,118 unique tickets. 60% of the data was used for training the model and 40% for testing the model (Bramer, 2007). Based on attribute importance process (discussed in the implementation chapter) only independent attributes or change ticket fields with greater than zero explanatory power is selected. This ensures that only attributes that are useful in explaining the impact are selected. These independent attributes are:

1) Ticket #
2) Creator
3) Owner
4) Last Update User
5) Workplan
6) Change Purpose
7) Start Date
8) End Date
9) Backout Plan
10) Impact Description
11) Ticket Subject
12) Approver
13) General Comments
14) Queue

Let us look at the IPM's prediction performance and accuracy metrics in comparison to Naïve Bayes algorithm. Following metrics are used to answer the above two questions for IPM. Additional screenshots from Oracle data miner's IPM for some of these metrics are shown in the Appendix D.

a) Predictive Confidence

b) Confusion Matrix

IPM Predictive Confidence: Predictive confidence provides an estimate of the overall goodness of the model. This indicates how much better the predictions made by the tested model are than predictions made by a naive model (Oracle, 2011a). The naive model always predicts the mean for numerical targets and the mode for categorical targets. The following formula defines Predictive Confidence (Oracle, 2011a):

Predictive Confidence = MAX ((1-((error of model)/(error of naive model))), 0)

If predictive confidence is 0, the model's predictions are no better than predictions made using the naive model. If predictive confidence is 1, the predictions are perfect. IPM's Support vector machine (SVM) model is ~ 50% better than naïve model and 13% better than naïve bayes (NB) model. This is shown in Figure 22 below.

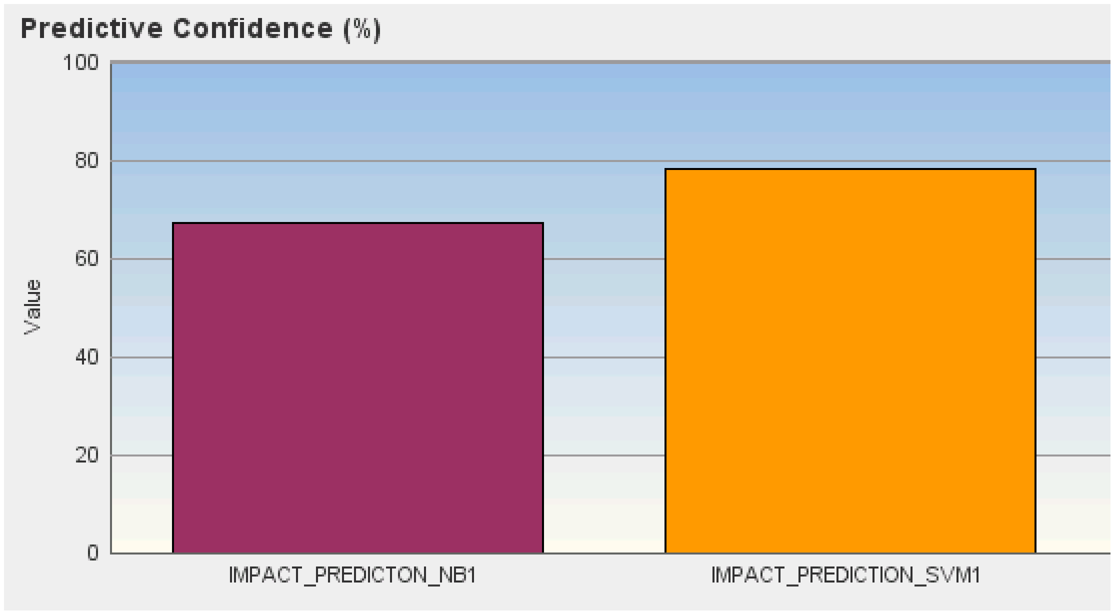


Figure 22. IPM's Predictive Confidence (4 years of Change Data)

The model build time for SVM and Naïve Bayes is shown in Table 6.

Table 6. IPM Predictive Confidence Metrics (4 years of Change Data)

Models	Predictive Confidence %	Model Build and Test Time
Naïve Bayes	66.18	30 minutes and 10 seconds
Support Vector Machines	80.75	18 minutes and 15 seconds

IPM Performance Matrix (Confusion Matrix): This measures the probability of the model to predict incorrect and correct values and also indicates the types of errors that the model is likely to make. IPM's SVM model has identified over 80% accurate predictions. The performance matrix is calculated by applying the model to a hold-out sample (the test set, created during the split step in a Classification activity) taken from the build data. This is shown in Table 7 below. The confusion matrix by impact level is shown in Appendix E.

Table 7. IPM Confusion Matrix (4 years of Change Data)

Models	Correct Predictions %	Correct Prediction Count	Total Count
Naïve Bayes	86.67	3811	4397
Support Vector Machines	92.01	4046	4397

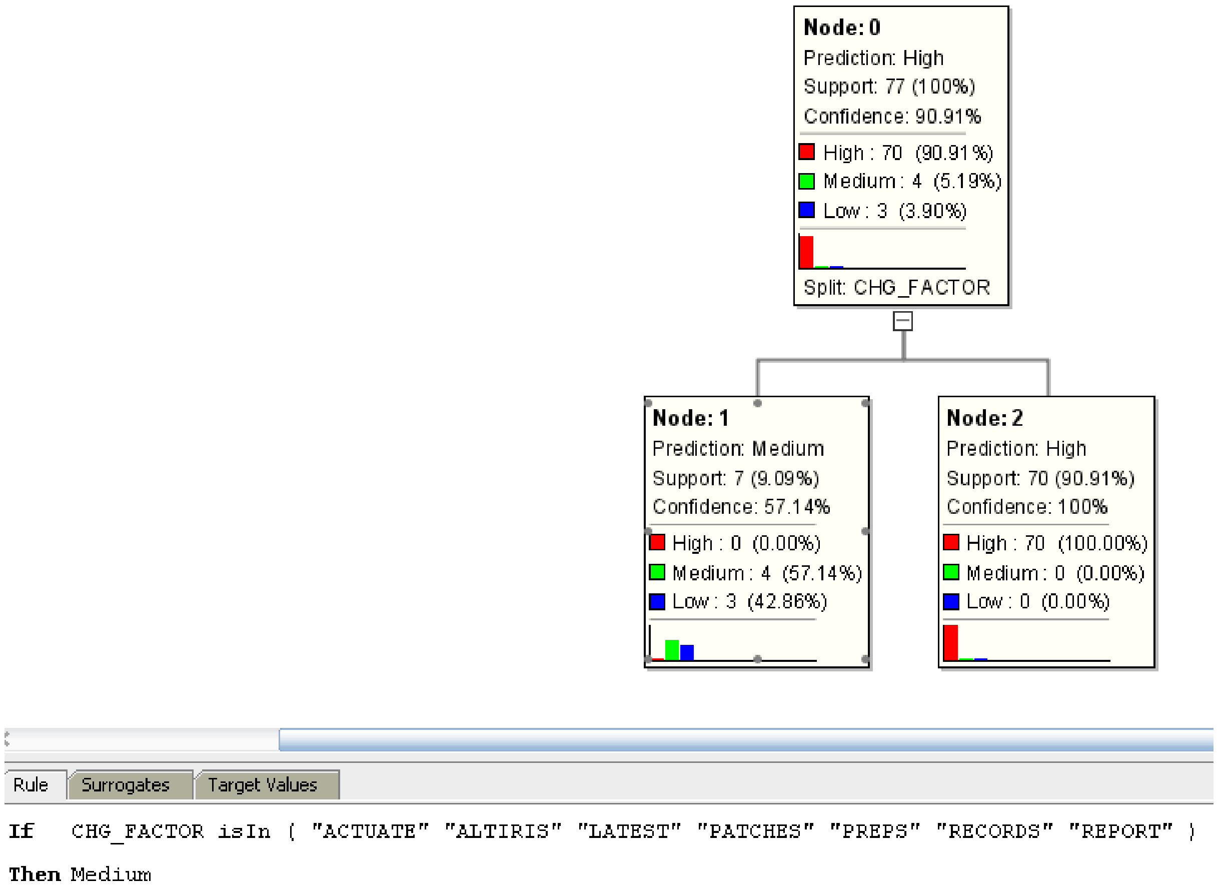
While the above tests and metrics are geared towards validating the IPM prediction accuracy, the IPM results can also provide us with further insights regarding the change terms. Oracle data miner provides detailed information on the SVM coefficients for IPM. These coefficients show the statistical significance or importance for the IPM terms in reference to the impact category. This is similar to the attribute importance model that provides the importance of attributes or change ticket fields in reference to impact category as seen in Table 5 in the previous chapter. The SVM coefficients provide insights into the relationship of IPM terms with respect to the impact categories.

Table 8. Top 5 Dominant Key Terms with High Impact

Ticket Field	Change Terms	SVM Coefficients
Workplan	Roms5.5	0.68
Workplan	Tincup	0.49
Workplan	CV	0.42
Change Purpose	Trinidad	0.37
Change Purpose	Globalscape	0.31

The table 8 shows the top 5 dominant change terms for the “High” impact category and their corresponding SVM coefficients sorted in descending order. The first factor “Roms5.5” refers to a software product for one of the mission critical applications for the organization under study. This implies that a change ticket involving this product has been found on several high impact tickets across time intervals and is likely to reoccur in future. In other words, changes to this application have a high and far-reaching impact to the organization and needs to have a thorough risk mitigation plan before any environmental component related to this software undergoes change in the future. Similarly, factors “CV” and “Globalscape” refer to client-facing applications that have similar impact across time intervals and have potential of high impact to the organization. Factors “Trinidad” and “Tincup” refer to server names that host shared and critical supply chain processes.

In order to gain further insights into the characteristics of the dominant terms that predict a common target, decision trees algorithm can be used. Decision trees are similar to Naïve Bayes in the sense that they too are based on conditional probabilities but decision trees also provide rules. Decision tree rules are conditional statements that provide model transparency by explaining the inner workings of a model (Quinlan, 1986).



```
If   CHG_FACTOR isIn ( "ACTUATE" "ALTIRIS" "LATEST" "PATCHES" "PREPS" "RECORDS" "REPORT" )
Then Medium
```

Figure 23. Decision Tree for Dominant IPM Key Terms

Figure 23 shows the decision tree for the dominant terms. In other words, it shows the profile of the dominant terms. The decision tree was built using the dominant terms (SVM coefficients > 0.1) for high and medium impact category based on the SVM coefficients (Oracle, 2011a). This resulted in 77 terms. The inputs to the decision tree algorithm are these terms, their coefficients and the impact level that the terms belong to. The selected section of the Figure 23 shows a decision rule for Node 1 that explains the prediction for medium impact category and the terms that predict that target. The decision rule describes the terms that have been associated to medium impact tickets across time intervals. The terms listed in Node 1 in the above figure point to reporting applications changes for the organization. These terms are coming from workplan, change purpose and backout plan ticket fields. This rule implies that if the new changes that involve the reporting applications would be classified with potential impact of category medium. These insights help in understanding the classification behavior of the IPM and also understand the terms that influence the impact prediction. Now that we have a better understanding of the terms and its relationship with the impact levels, let us look at how we can understand the extent of the impact.

IPM results can also help us better understand the attribute importance model findings presented in the Table 5 in Chapter 5. Based on that table, we found that creator and owner fields were the top two attributes that have comparatively most effect on the target attribute. But does this imply that these attributes by themselves are enough to predict the impact? Table 9 below presents the findings of IPM model built using only creator and owner fields for the four years of change data.

Table 9. IPM Confusion Matrix using Creator and Owner attributes only

Models	**Correct Predictions %**	**Correct Prediction Count**	**Total Count**
Naïve Bayes	58.56	2575	4397
Support Vector Machines	70.59	3104	4397

Based on the results in Table 9 we can see that percentage of correct predictions has dropped significantly compared to Table 7 which was for an IPM model that included all the attribute fields in the model building process. The confusion matrix by impact level is shown in Appendix F. The results highlight that the other change ticket fields listed in Table 5 are also important to IPM’s prediction capability. To further understand this, let us look at Table 10 below that presents the top 5 dominant terms for Low impact category. This table shows the user identification numbers for creators and owners of change ticket as the dominant terms. This implies that a change ticket involving these users has been found on several low impact tickets across time intervals and is likely to reoccur in future. The findings from Table 10 along with the findings from Table 8 and Figure 23, highlight that terms coming from unstructured text fields such as workplan and change purpose are important for high and medium impact tickets since most of the top terms are coming from those ticket fields. Also based on Table 10 and from the attribute importance model results of Table 5, it implies that knowing the creators and owners for change tickets is more likely helpful in determining low impact tickets but terms found within unstructured text ticket fields have comparatively more likely in determining high and medium impact category tickets. This is important because from an IPM perspective, only high and medium impact terms are passed on for IEIM processing. Low impacts terms represent local impact and hence do not undergo IEIM processing.

Table 10. Top 5 Dominant Key Terms with Low Impact

Ticket Field	Change Terms	SVM Coefficients
Creator	1345	0.61
Creator	112	0.51
Owner	38	0.50
Creator	442	0.53
Owner	1981	0.55

IEIM Results

The core function of IEIM is that of identifying the potential extent of impact of change factors that are not yet implemented. In this section, we will look at the results for IEIM with respect to answering the following two questions:

1. Do patterns (change factors) reoccur in future years, and to what extent?

2. What is the time interval across which pattern (change factor) reoccurrence is the maximum and how does this impact prediction accuracy?

For IEIM model validation, two data sets were created across various time intervals. The data set details are shown in Appendix G:

a) 2009 (Q1) with 2010 (Q1)

b) 2009 (Q1 and Q2) with 2010 (Q1 and Q2)

c) 2009 (full year) with 2010 (full year)

d) 2008 and 2009 (full years) with 2010 (full year).

The matching accuracy for IEIM's Jaro-Winkler's matching was compared to that of Levenshtein matching algorithm.

IEIM Pattern Reoccurrence: This helps us understand if the patterns (change factors) reoccur in future years as explained in Chapter 4. This metric is represented as below:

$\frac{1}{n}\sum_{i=1}^{n} Similarity(C_i^{t+1}, C_j^t))$, where C_j^t is such that $Similarity(C_i^{t+1}, C_j^t) \geq Similarity(C_i^{t+1}, C_k^t)$ for k = 1...m

where, the new feature set is C_i (, e.g., 2010 Q1 dataset) with older feature set C_j (, e.g., 2009 Q1 dataset). The IED details and build times are shown in Table 11 across time intervals. The numbers of NMF feature sets are automatically determined by the NMF algorithm.

Table 11. IED Details

IED Data Set	NMF Feature Sets	Number of Change Factors	IED Build Time
2009 Quarter 1	25	2234	3 minutes and 11 seconds
2009 Quarter1 and 2	49	3408	5 minutes and 2 seconds
2009 - full year	101	5053	8 minutes and 17 seconds
2008 and 2009 Full years	204	7972	16 minutes and 49 seconds

Table 12. IEIM Accuracy Results

Old Data Set	New Data Set	IEIM Pattern Reoccurrence Jaro-Winkler Algorithm	IEIM Pattern Reoccurrence Levenshtein Algorithm	Runtime
2009 Quarter 1	2010 Quarter 1	33.33	29.64	5 minutes and 33 seconds
2009 Quarter1 and 2	2010 Quarter1 and 2	78.21	74.90	6 minutes and 20 seconds
2009 - full year	**2010 - full year**	**84.16**	**79.55**	**11 minutes and 17 seconds**
2008 and 2009 Full years	2010 - full year	83.66	80.11	17 minutes and 45 seconds

The results in Table 12 show the IEIM accuracy results. These results highlight that there is significant change factor reoccurrence between the old and new change data sets. These results reaffirm our core assumption that change patterns reoccur year over year.

IEIM Prediction Accuracy and Optimal Change Duration: This helps us understand the optimum change duration or time interval across which change factors reoccur. Based on the results in Table 12, the optimum change duration is a full complete year. It also evident from these results is that going beyond the year for old data does not yield significant benefits.

These findings can help us to identify the best subset of data to use for change impact prediction model building. In other words, we can use these findings to select the optimal change time duration as represented by the "Select Change Time Duration" step in the Figure 13. Selecting the optimal change time duration will help in reducing the data set sizes thereby expediting the data processing, model building, and model scoring and matching processes. This is very crucial in IT environments that undergo large amount of changes. Next, we will use the data sets based on change time duration similar to what for IEIM and run IPM accuracy and performance metrics to evaluate the impact of change data time duration on the IPM prediction and accuracy.

The results in Table 13-15 shows the effect of using change time duration based data similar to what was used for IEIM evaluation, for IPM in order to find out the impact of change data time duration on the IPM's prediction accuracy as well as its model build and test times. For this evaluation, 2010 change data is divided into sets similar to IEIM evaluations, i.e., 2010 quarter 1, 2010 quarter 1 and 2, and 2010 full year and finally 2009 and 2010 combined for build the IPM prediction models.

Table 13. IPM Predictive Confidence Summary

Models	**Predictive Confidence % (2 yrs Change Data)**	**Predictive Confidence % (1 yr Change Data)**	**Predictive Confidence % (Q1 and Q2 Change Data)**	**Predictive Confidence % (Q1 Change Data)**
Naïve Bayes	62.86	66.92	75.21	70.07
Support Vector Machines	74.88	78.04	85.52	90

Table 14. IPM Correct Predictions Summary

Models	**Correct Predictions (2 yrs Change Data)**	**Correct Predictions (1 yr Change Data)**	**Correct Predictions (Q1 and Q2 Change Data)**	**Correct Predictions (Q1 Change Data)**
Naïve Bayes	86.70	87.56	87.93	86.16
Support Vector Machines	92.27	95.79	93.88	94.86

Table 15. IPM Build and Test Time Summary

Models	Model Build and Test Time (2 yrs Change Data)	Model Build and Test Time (1 yr Change Data)	Model Build and Test Time (Q1 and Q2 Change Data)	Model Build and Test Time (Q1 Change Data)
Naïve Bayes	15 minutes and 56 seconds	8 minutes 32 seconds	4 minutes and 25 seconds	2 minutes and 30 seconds
Support Vector Machines	11 minutes and 36 seconds	6 minutes 8 seconds	3 minutes and 44 seconds	2 minutes and 26 seconds

From Tables 13- 15, it is evident that the one year of change data for IPM model build and testing has no negative impact on accuracy but significantly reduces the model build and test time compared to 4 years of change data, while providing the maximum IEIM change pattern reoccurrence compared to other quarterly data sets. The confusion matrix by impact for the one year change date is shown in Appendix I. This change time duration implies that the potential risks posed by environmental changes can be accurately and quickly surfaced and addressed in a constantly changing organizational IT environment with yearly change information from the change information corpus without having to store large amounts of historical change information.

IEIM's pattern reoccurrence metric can also be used to highlight the importance of including all layers of environments as part of the change information corpus. Table 16 show the IEIM results if we only use the database workflow queue in the change information corpus and ignore other workflow queues (network, hardware, operating system etc.). The Table 16 results show a significant drop in pattern reoccurrence values across time intervals for this data set.

Table 16. IEIM Accuracy Metrics with Only Database Workflow Queue

Old Data Set	New Data Set	IEIM Pattern Reoccurrence Jaro-Winkler Algorithm	IEIM Pattern Reoccurrence Levenshtein Algorithm	Runtime
2009 Quarter 1	2010 Quarter 1	29.55	26.67	4 minutes and 19 seconds
2009 Quarter1 and 2	2010 Quarter1 and 2	32.66	21.16	5 minutes and 47 seconds
2009 - full year	**2010 - full year**	**66.16**	**57.34**	**8 minutes and 1 second**
2008 and 2009 Full years	2010 - full year	64.11	59.32	11 minutes and 15 seconds

In order to better the reason behind significant drop of pattern reoccurrence, two different sets of tests are conducted using the change data sets described in Appendix G. The first test builds an IPM model using change data from all queues and scores it against the data from only database queue. The results of this test are shown in Table 17 across the time intervals. The second test builds an IPM model using change data from only database queue and scores it against the data for the database queue across time intervals. The results of this test are shown in Table 18. Scoring is the process of running the built model (after it goes through test-train phase) on a different data set. Prediction Accuracy of a model is defined as number of correct predictions by the model divided by the total number of records in the scoring data set.

Table 17. IPM Prediction Score Accuracy- All Queues model on DB Queue

Models	Prediction Accuracy % (2 yrs Change Data)	Prediction Accuracy % (1 yr Change Data)	Prediction Accuracy % (Q1 and Q2 Change Data)	Prediction Accuracy % (Q1 Change Data)
Naïve Bayes	60.21	67.43	73.74	68
Support Vector Machines	73.89	77.33	83.45	92.45

Table 18. IPM Prediction Score Accuracy- DB Queue model on DB Queue

Models	Prediction Accuracy % (2 yrs Change Data)	Prediction Accuracy % (1 yr Change Data)	Prediction Accuracy % (Q1 and Q2 Change Data)	Prediction Accuracy % (Q1 Change Data)
Naïve Bayes	49.12	50.11	54.22	59.01
Support Vector Machines	56.54	60.16	65.81	70.47

The Table 17 shows that the IPM prediction accuracy for model built using all queues change information has higher accuracy than model built with only the database queue change information as seen from the findings from Table 18. In order to better understand these results, three different cases were considered from one year change data test. The confusion matrix by impact for both these IPM models for one year of change data is shown in Appendix J. For sake of this discussion, high and medium impact are treated as same because IPM stage passes the terms for both of these impact levels to the IEIM processing. IPM only blocks low impact terms because they represent local impact. Having said this, the false negatives are the ones that we need to review. False negatives are the tickets that have the impact of high/medium but were scored as low by the IPM models. These are important cases to review because we do not want a high/medium impact ticket to be scored as a low impact and not make it to the IEIM processing. The first ticket is where the IPM model built only on database queue changes got a false negative but the IPM model built using all queue data got correct prediction. For the sake of this discussion we will call this ticket as ticket 1. The second case is where IPM model built using all queue got false negative but the model built using just the database queue got the impact prediction right. This situation did not exist across time intervals. Most of the false negatives for the IPM model built using all queues model was also found to be false negative for the IPM model built using just the database queue. The final case is where both models got a false negative. We will call this ticket as ticket 2.

Ticket 1 that was reviewed was for updating data for certain customers within a database. The update was a multiple step non-atomic process that modified several customer demographics. Furthermore, in order to expedite this process, this update was executed across all the clients concurrently. This process had to be completed within 4 hours per the customer service level agreements. This process was found hung after 40 minutes of processing. The DBA filed a ticket with the storage team because the process was found to be waiting on the storage system. After few hours of investigation, the storage team found that a SAN to SAN copy change that was executed around the same time as the update process was the culprit. The SAN copy process was aborted until further investigation. The database update had to be terminated and the database had to be recovered in order to get it back into a consistent state before re-starting the process again. This whole incident added additional 3 hours to the total time and ended up violating the customer Service Level Agreements (SLAs). The ticket 1’s impact before it was closed was set to high but the IPM built using just the database queue scored it as low. The ticket 1 did not have any of this problem history because that history was mentioned in the storage ticket with the storage queue. All, the ticket 1 had was a mention of the storage queue ticket number as a comment in its impact description text field. Further review of this storage queue ticket showed that this issue was a known issue within the storage team and had happened in the past too. The IPM model built only on the database queue changes did not have the visibility of this problem history. The IPM model built using change information from all queues was able to score the impact correctly for ticket 1.

Based on the review of ticket 1 and also the findings from Table 17 and Table 18 it seems that significant change factors that impacts the database workflow queue are coming from other queues. A closer look at some of the key terms in Ticket 1 shows that it has “Trinidad” server name as one of the terms which from Table 8 tell us is part of the high impact category of the IPM model with SVM coefficient of 0.37. This is the server where the update process was running. Besides this term, ticket 1 also had terms for 3 customer names that were also part of the high category IPM model with SVM coefficients of 0.31, 0.3 and 0.22. Two of these customers have stringent SLAs for the update process. This finding reinforces the core point of this research that in increasingly integrated and dynamic IT environments, the impact of changes from all layers within the environment need to be taken into account. Adopting a narrow view of the environment does not accurately enable the human database tuners to understand the extent or reach of the impact of environment changes which may lead to incorrect diagnosis and error-prone tuning efforts.

Ticket 2 that was reviewed was for modifying the profile and preferences for customers that are no longer active within the database system. The change information within this ticket did not have much information about what the change was or the plan for executing the change. The ticket had comments like “Replicate the steps from previous release of this app…” and “Following the same steps that I did on QA…”. These comments might make sense to the same person who has prior knowledge about this change or similar change done on a test system but for a DBA that has no background information on it might not understand this. The DBA who created and executed the changes in ticket 2 did not explicitly mention the steps he was planning to execute in ticket 2. For ticket 2, the DBA during his testing of the change on QA system found few database packages to become invalid which caused other applications to fail. Without mentioning any of this information he vaguely referenced his prior knowledge but made the impact of the ticket as high. Both IPM models got the prediction for ticket 2 as low.

Based on the review of ticket 2, it seems that if a change executor is not explicit about the change in his or her ticket, it most likely will be scored incorrectly by IPM, irrespective of what type of change data goes into building the model. This highlights a process issue within the change management process. The observations from ticket 2 point to the process failure on two levels – first is the lack of explicit change information documentation on part of the change executor and second at the change approver level for approving such incomplete changes that neither document the plan or scope of the change.

Scoring Example of an Unimplemented Change Ticket

The core function of DECIPHER is that of predicting the potential impact and extent for environmental changes that have not yet been implemented in real-time. This aspect of DECIPHER is called "Scoring of the New Environmental Changes". In order to understand the scoring function of DECIPHER, let us consider an actual unimplemented incident ticket from the organization's change information corpus.

The network engineering team needs to update the organizations firewall rule policy to add a new system to the database environment. In order to get ready to do this change, the network engineer goes through the organizational change management process as depicted in Figure 7. As part of the change management process, the network engineer creates a new incident ticket with the following information. This ticket is submitted for the approval process.

a) Ticket #: 14356

b) Ticket Subject (Text): Update firewall rule policy

c) Change Purpose (Text): We need to update the firewall rule policy in order to add new forecasting BI server to sales portal.

d) Owner of the ticket: Dylan

e) Approver of the ticket: Susan

f) Work-plan for the change (Text): The new firewall rule for salesportal_dmz is as below: 123.0.xx…

g) Perceived Impact (High or Medium or Low): Low

h) Back-out plan (Text): Revert the rule

i) General Comments/Notes (Text): N/A

This new unimplemented change information is fed to DECIPHER in real-time. DECIPHER's IPM and IEIM modules process this information. Tables 19 and 20 show the results of the DECIPHER scoring for ticket 14356:

Table 19. DECIPHER IPM Scoring Results

Predicted Impact	**Predictive Confidence**
High	91.23%

Table 20. DECIPHER IEIM Scoring Results

Database Dependent Change Factors	NMF Coefficient
Sales_recos_db	0.13
Connection	0.12
Hang	0.12
Port	0.11
1521	0.11
…	…

Based on the findings from Table 19 and 20, we can see that the perceived impact by the network engineer for the ticket 14356 was set to “Low” implying a localized impact of his change, but DECIPHER’s IPM scored it with a predicted impact of “High” and with a high probability of 91.23%. Also, the IEIM scoring shows the sales_recos_db and hang are dependent change factors based on the factors in the 14536 ticket. This implies that if the 14536 ticket were to be implemented then there is a high probability of a potential hang with connections for the sales_recos_db on port 1521.

Incorrect diagnosis and troubleshooting is expensive and error-prone. Having this knowledge ahead of time would allow the DBA to mitigate the potential risk of connection hangs by either enabling a different temporary port or even delaying this change request. This case shows how using the environmental change impact knowledge; the DBA can proactively mitigate the risks. This research makes an important assumption that based on the knowledge of the potential impact and extent of IT environmental changes, the DBAs can effectively use it to ensure that such changes do not negatively impact the organizational database systems.

In general any manual work is prone to human errors and database performance tuning is no exception. DECIPHER in its current form can be a very effective tool for DBA's in enhancing their explicit tuning knowledge sources like best practice repositories or designing policies that can automatically trigger a reaction based on an event. The next chapter discusses how DECIPHER can be leveraged by DBAs with existing frameworks to effectively and safely tune their database systems.

DECIPHER Generalizability

Effectively and safely managing changes in a highly-integrated database environment is a challenge for modern organizations. Predicting the potential impact of environmental changes and its extent before they are implemented can help in mitigating these risks and reducing their associated costs proactively for organizations (Forrester, 2007). DECIPHER is built with this goal in mind. Coming to the generalizability aspect, the architecture of DECIPHER is designed in a way that its implementation under different organizational settings can be achieved with minimal modifications.

Most organizations use some or other form of an organizational incident management data store (help desk system or trouble ticketing system or basic email) to manage the changes to their Information Technology (IT) environment, including the database environments (Hass, 2003). Changes to any production IT environment component, e.g., operating system, hardware, and database are facilitated by these systems (Conradi and Westfechtel, 1998). Furthermore, the increase of regulatory compliance needs (, e.g., Sarbanes-Oxley) have also pushed for a wider adoption of change management processes and tools for achieving better traceability(Chen et al., 2009).

The first processing step of DECIPHER which is the IPM’s data Extraction step, as shown in Figure 11, is responsible for extracting the unstructured change information from any type of organizational incident management data stores like help desk system or trouble ticketing system or even basic email. This process within IPM ensures that DECIPHER can get change information from various data sources to build its change information corpus without imposing rigid data collection requirements. This data extraction step involves use of open source Extraction, Transformation and Load (ETL) tool that extracts unstructured change data from various data sources without rigid data format restrictions (Majchrzak, Jansen, and Kuchen, 2011). Furthermore, such type of ETL tools offer out-of-the-box data quality and profiling features that makes it easy to implement DECIPHER under various organizational settings (Majchrzak, Jansen, and Kuchen, 2011). Specifically, this research uses TALEND ETL tool that achieves this task using intuitive graphical interface (Majchrzak, Jansen, and Kuchen, 2011). Once the data is extracted from the organizational incident management data stores, other downstream processing steps for DECIPHER remains unchanged. Organizations can also enhance the stop list processing of IPM term extraction process, as explained in Chapter 5 to make DECIPHER context-aware by incorporating organization-specific terms that might add undesirable noise into the DECIPHER’s IED repository.

Organizational IT environments undergo changes that are influenced by many external factors such as mergers, acquisitions, explosive data growth, changing competitive landscape and long-term investments. So from a generalizability perspective, DECIPHER also needs to adapt to new change information that deal with new situations and technologies. In order to maintain the accuracy and effectiveness of DECIPHER under such changing circumstances, DECIPHER’s IPM model needs to get periodically updated or refreshed or challenged. One effective approach of managing the IPM Model refresh or update is by using an adaptive control to model management. This is often referred to as Champion/Challenger or test and learn process (Shyam Varan 2007; Taylor 2010). This is covered in detail in the future work chapter. Basically, the Champion/Challenger process involves building several models using the historical change information. The model that has better accuracy based on metrics like predictive confidence, confusion matrix and lift is picked as the champion. If the challenger outperforms the current champion IPM model based on some user-defined threshold then the challenger becomes the curent champion. This process can be performed iteratively every year to ensure that the impact and extent prediction remains accurate throughout the life cycle of DECIPHER and without having the end-user(DBA) of this system to understand the naunces of the various underlying models and their parameters.

As mentioned earlier, managing environmental changes in a highly-integrated database environment is a challenge for modern organizations. Since DECIPHER is designed with this goal in mind, it may not be well-suited for organizations that do not have a complex and dense database environment stacks. Furthermore, DECIPHER may not work for organizations that do not have a standardized change management process. Absence of such a process would result in inadequate or inferior change management data which would negatively impact the DECIPHER's ability to accurately predict and impact and extent of environmental changes.

CHAPTER 7

CONCLUSION AND FUTURE WORK

This chapter presents an overview of the contributions of this project and a discussion on the possible future directions for DECIPHER. This chapter begins by summarizing the contributions of this research with respect to the identified issues. The chapter then continues with the discussion on the two possible areas future work for the DECIPHER framework. The chapter concludes with an example of how DECIPHER could be used with some of the existing autonomic tuning frameworks.

Conclusion

More and more organizations are embracing technologies such as cloud computing in the form of private clouds to address their evolving business needs and reduce their operating costs. But, as organizational Information Technology (IT) environments in today's rapidly evolving digital economy undergo several changes, the database environments within the organizations are becoming highly-integrated, complex and very dynamic.

Given the high server density of cloud environments, the potential impact of IT environmental changes to the performance of database systems becomes significant and far-reaching. As a result, human-driven performance tuning is needed to addresses these issues. Human-driven performance tuning is expensive, error-prone and time-consuming. With organizational IT environments undergoing large number of changes, there is a strong need for a solution that can provide fast and accurate decision-support to the DBAs in database performance tuning situations.

Although there have been several developments in the area of self-managing systems, these approaches are rather limited in their use, especially in the cloud computing domain because they do not include a holistic view of the problem space and the environment under which they operate. Specifically, these approaches largely ignore the impact and the extent of IT environmental changes on its systems.

Effectively and safely managing changes in highly-integrated database environments such as private database clouds is a challenge. Predicting the potential impact of environmental changes and its extent before they are implemented can help in mitigating these risks and reducing their associated operating costs proactively.

This research addresses these relevant issues by proposing a novel framework that predictively acquires the knowledge of impact and extent of environmental change in database environments. The contributions of this research are significant in the following aspects:

1) This research proposes a holistic autonomic tuning knowledge model that extends the existing autonomic tuning reference model by incorporating the organization-specific environmental change impact knowledge.

2) This research also presents a theory based framework called “DECIPHER” that that not only acquires this knowledge component but does so in a proactive fashion. This framework predicts the potential impact of environmental changes and its dependencies by mining the historical change information stored within the existing organizational incident management data stores.

3) In addition to demonstrating the validity of the system using a real-world change management system, this research also presents a new pattern recurrence metric to identify the contexts in which prediction algorithms will useful and helps identify the best subset of data to use for model building.

Future Work

Future work for DECIPHER can be broadly classified in two main areas:

1. DECIPHER IPM Model Management
2. Enhancement of Autonomous Tuning Managers

DECIPHER IPM Model Management: The life-cycle for DECIPHER's IPM model can be broken down into three major phases:

1. Model Build
2. Model Deployment
3. Model Management

Figure 25 shows the Model Build phase as well as the Model deployment phase for DECIPHER's IPM Model. IPM Model Build phase refers to the development and building of DECIPHER's IPM model from the historical change information corpus. The IPM Model Deployment or Scoring phase refers to the applying the built IPM model on the unimplemented change information.

Organizational IT environements undergo changes that are influenced by many external factors such as mergers, acquisitions, explosive data growth, changing competitive landscape and long-term investments. As a result, the change information corpus has new change information that deal with new situations and technologies. In order to maintain the accuracy and effectiveness of DECIPHER under such changing circumstances, the IPM model needs to get periodically updated or refreshed or challenged. This aspect is referred in here as IPM Model Management.

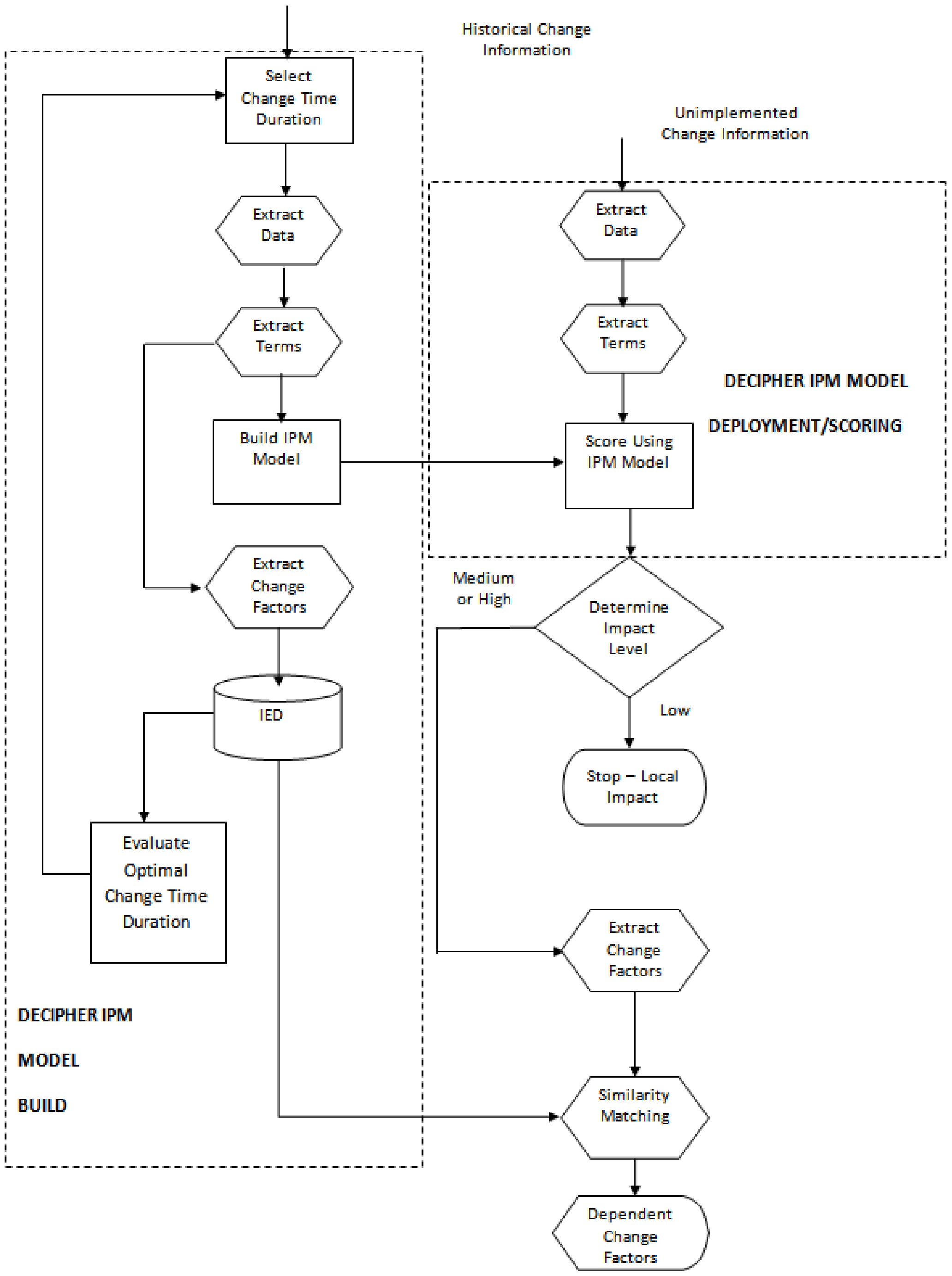


Figure 25. DECIPHER IPM Model Build and Deployment Phase

One effective approach of managing the IPM Model refresh or update is by using an adaptive control to model management. This is often referred to as Champion/Challenger or test and learn process (Shyam Varan, 2007; Taylor, 2010). The Champion/Challenger process involves building several models using the historical change information. The model that has better accuracy based on metrics like predictive confidence and confusion matrix is picked as the champion. During the IPM Model build phase, we picked SVM over NB because it was compartively more accurate. In this SVM can be seen as the champion model and NB as the challenger model. This process can be done with several models to create a challenger model list.

With SVM in production as the champion model, the challenger models can be periodically be executed using a small percentage of the new change information. In this small acid test, if one of the challengers have better accuracy and performance metrics, then it can be picked to be run against larger percentage of the new change information (Shyam Varan, 2007). If the challenger outperforms the current champion IPM model based on some user-defined threshold then the challenger becomes the curent champion. This process is performed iteratively to ensure that the impact and extent prediction remains accurate throughout the life cycle of DECIPHER.

Enhancement of Autonomous Tuning Managers:

Effectively and safely managing changes in highly-integrated database environments such as private database cloud environments is a challenge. Predicting the potential impact of environmental changes and its extent before they are implemented can help in mitigating these risks proactively. DECIPHER in its current form can be an effective tool for DBA’s in this regard. However, manual intervention by a human tuner is prone to errors.

One possible approach to address this concern would be to use the environmental change impact knowledge component by the DBAs to design policies that can be leveraged by exisiting policy based feedback or control mechansims to automatically self-regulate the autonomic database tuners before the environmental change are even implemented in an anticipation of a need. This type of architecture can be referred to as an autonomous predictive performance tuning.

Since one of the major goals for organizations adopting private clouds is to reduce the operating costs, having an autonomous predictive performance tuning framework that can self-regulate before a change is even implemented can be very beneficial to minimize the costs associated with an undesired change that negatively effects the performance or availability of the systems within the private database cloud environments.

A high level architecture of a potential autonomous predictive performance tuning framework is shown in Figure 26 below. The Environmental Change Impact Knowledge (ECIK) can be used in conjunction with existing policy based frameworks to control the autonomic managers (Russell, Morgan, and Chron, 2003; Wiese et al., 2008). On such framework is the Automatic Tuning Expert (ATE) that uses best practice databases of database tuning plans that get picked up autonomic tuning manager based on predefined policies (Wiese et al., 2008). Such policy based frameworks use a policy database to automatically trigger a reaction based on a predefined performance situation or an event(Wiese et al., 2008). ECIK from the predictive change management framework can be used to lookup an existing policy or define a new policy that can be applied before a change is implemented in order to self-manage the autonomic manager in an anticipation of a need.

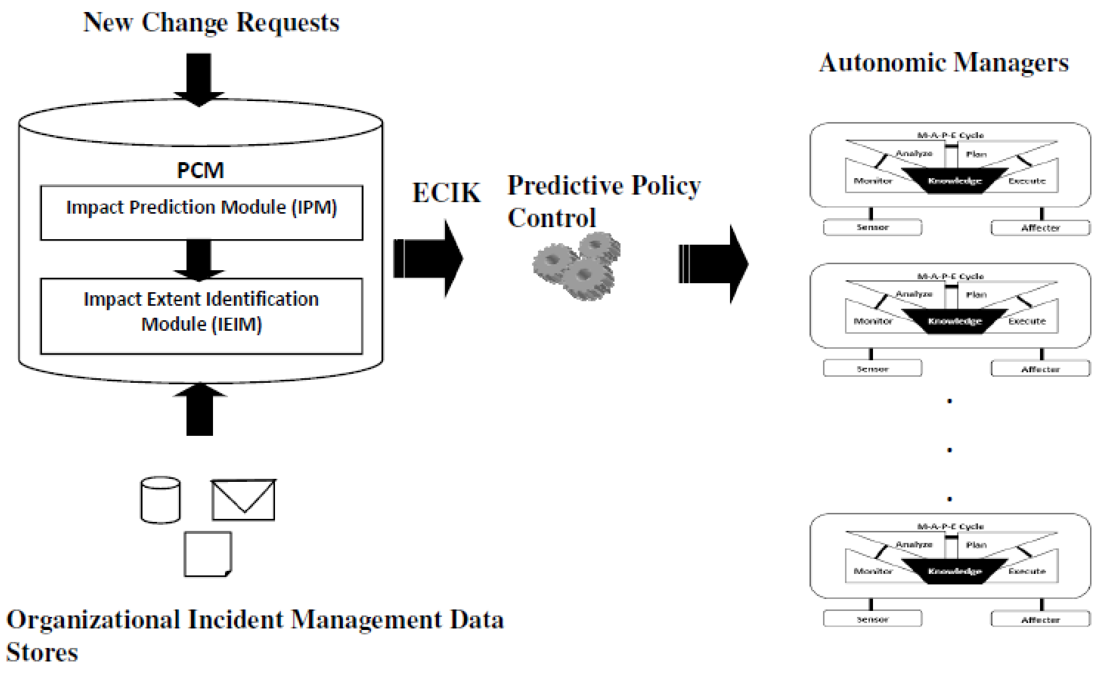

New Change Requests
PCM
Impact Prediction Module (IPM)
Impact Extent Identification Module (IEIM)
ECIK
Predictive Policy Control
Autonomic Managers
M-A-P-E Cycle
Analyze
Plan
Monitor
Knowledge
Execute
Sensor
Affecter
Organizational Incident Management Data Stores

Figure 26. Autonomous Predictive Performance Tuning Framework

REFERENCES

Bayes, M., and Price, M. (1763). An Essay towards Solving a Problem in the Doctrine of Chances. By the Late Rev. Mr. Bayes, communicated by Mr. Price, in a letter to John Canton, M. A. and F. R. S. *Philosophical Transactions (1683-1775)*. doi: citeulike-article-id:2306393

Belknap, P., Dageville, B., Dias, K., and Yagoub, K. (2009). *Self-Tuning for SQL Performance in Oracle Database 11g*. Paper presented at the Proceedings of the 2009 IEEE International Conference on Data Engineering.

Bell, J. (2004). Understand the autonomic manager concept. http://www.ibm.com/developerworks/autonomic/library/ac-amconcept/index.html

Bigus, J., Hellerstein, J., Jayram, T., and Squillante, M. (2000). *AutoTune: A Generic Agent for Automated Performance Tuning*. Paper presented at the Practical Application of Intelligent Agents and Multi Agent Technology.

Blakey, J. P., and Atkins, C., Crump, B. J. (2008). *Using Design Research to Improve Data Modelling Performance among Novice End Users*. Paper presented at the *19th Australasian Conference on Information Systems*, Christchurch, NZ.

Boughton, H., Martin, P., Powley, W., and Horman, R. (2006). *Workload Class Importance Policy in Autonomic Database Management Systems*. Paper presented at the Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks.

Bramer, M. (2007). *Principles of Data Mining (Undergraduate Topics in Computer Science)*: Springer.

Brown, K. P., Carey, M. J., and Livny, M. (1996). *Goal-oriented buffer management revisited*. Paper presented at the Proceedings of the 1996 ACM SIGMOD international conference on Management of data, Montreal, Quebec, Canada.

Böhm, M., Nominacher, B., Fähling, J., Leimeister, J. M., Yetton, P., and Krcmar, H. (2010). *IT Challenges in MandA Transactions – The IT Carve-Out View on Divestments.*

Paper presented at the Proceedings of 31st International Conference on Information Systems (ICIS).

Campos, M. M., Stengard, P. J., and Milenova, B. L. (2005). *Data-Centric Automated Data Mining*. Paper presented at the Proceedings of the Fourth International Conference on Machine Learning and Applications.

Charvet, F. P., Ashish. (2003). Database Performance Study: University of Missouri - St. Louis.

Chaudhuri, S., and Narasayya, V. (2007). *Self-tuning database systems: a decade of progress*. Paper presented at the Proceedings of the 33rd international conference on Very large data bases, Vienna, Austria.

Chaudhuri, S., Narasayya, V., and Ramamurthy, R. (2008). A pay-as-you-go framework for query execution feedback. *Proc. VLDB Endow., 1*(1), 1141-1152. doi: 10.1145/1453856.1453977

Chaudhuri, S., and Weikum, G. (2006). *Foundations of automated database tuning*. Paper presented at the Proceedings of the 32nd international conference on Very large data bases, Seoul, Korea.

Chen, J. Q., Kurtz, J. M., and Lee, O. F. (2009). Information management and regulatory compliance\and\#58; a case analysis. *Int. J. Inf. Syst. Chang. Manage., 4*(1), 42-56. doi: 10.1504/ijiscm.2009.030050

Cohen, W., Ravikumar, P., and Fienberg, S. (2003). A comparison of string distance metrics for name-matching tasks.

Conradi, R., and Westfechtel, B. (1998). Version models for software configuration management. *ACM Computing Surveys, 30*(2), 232-282. doi: citeulike-article-id:1658340

Conway, s., Vesset, D., and Earl, J. (2009). Rasing the Bar on Business Analytics: Innovation Powered by Grid: SAS Institue.

Corp, I. (2005). An architectural blueprint for autonomic computing. doi: citeulike-article-id:821263

Curino, C., Jones, E., Popa, R., Malviya, N., Wu, E., Madden, S., . . . Zeldovich, N. (2011). *Relational Cloud: a Database Service for the cloud.* Paper presented at the In Conference on Innovative Data Systems Research (CIDR).

DBTA. (2009). Surveying the Role of Today's DBA. http://www.dbta.com/Articles/Editorial/News-Flashes/GoldenGate-Survey-Highlights-Top-Database-Management-Challenges-in-the-Year-Ahead-54316.aspx

Dumais, S. (1998). Using {SVMs} for Text Categorization. *IEEE Intelligent Systems, 13(4)*. doi: citeulike-article-id:619385

Elnaffar, S., Powley, W., Benoit, D., and Martin, P. (2003). *Today's DBMSs: How autonomic are they?* Paper presented at the Proceedings of the 14th International Workshop on Database and Expert Systems Applications.

Embarcadero-Technologies. (2010). Database Trends Survey Report. Retrieved from www.embarcadero.com website: http://www.embarcadero.com/reports/database-trends-survey

Endsley, M. R. (1995). Toward a Theory of Situation Awareness in Dynamic-Systems. *Human Factors, 37*(1), 32-64.

Forrester, R. (2007). CMDB: GET Ready, Get Set, Go.

Gil, P., Arlat, J., Madeira, H., Crouzet, Y., Jarboui, T., Kanoun, K., . . . Gil, D. (2002). Fault representativeness. *Deliverable ETIE2 of Dependability Benchmarking Project, IST-2000, 25245*.

Grivolla, J. (2005). *Using Oracle\and\#174; for natural language document retrieval an automatic query reformulation approach*. Paper presented at the Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, Salvador, Brazil.

Guduru, N. (2006). *Text mining with support vector machines and non-negative matrix factorization algorithms*: University of Rhode Island.

Hamm, C., and Burleson, D. K. (2006). *Oracle Data Mining: Mining Gold from Your Warehouse (Oracle In-Focus series)*: Rampant TechPress.

Hass, G. (2003). *Configuration Management Principles and Practice*: Addison-Wesley Longman Publishing Co., Inc.

Herodotos Herodotou, S. B. (2010). Xplus: A SQL-Tuning-Aware Query Optimizer. *VLDB'10*.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS Q., 28*(1), 75-105.

Inchiosa, M. E. (2011). *Accelerating large-scale data mining using in-database analytics*. Paper presented at the Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, San Diego, California, USA.

Kephart, J. O., and Chess, D. M. (2003). The Vision of Autonomic Computing. *Computer, 36*(1), 41-50. doi: 10.1109/mc.2003.1160055

Kotsovinos. (2011). Virtualization: Blessing or Curse? *Communications of the {ACM, 54*(1), 61-65. doi: citeulike-article-id:10265743

Krayzman, E. (2005). *development of self-tuning and autonomic databases and latest achievements:*. Paper presented at the 21st Computer Science Seminar SE2-T4-1.

Landau, D., Feldman, R., Aumann, Y., Fresko, M., Lindell, Y., Liphstat, O., and Zamir, O. (1998). *TextVis: An Integrated Visual Environment for Text Mining*. Paper presented at the Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery.

Lee, A. W., and Zait, M. (2008). Closing the query processing loop in Oracle 11g. *Proc. VLDB Endow., 1*(2), 1368-1378. doi: http://doi.acm.org/10.1145/1454159.1454178

Lee, D., and Seung, S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature, 401*(6755), 788-791. doi: citeulike-article-id:83540

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals (Vol. 10, pp. 707-710).

Majchrzak, T. A., Jansen, T., and Kuchen, H. (2011). *Efficiency evaluation of open source ETL tools*. Paper presented at the Proceedings of the 2011 ACM Symposium on Applied Computing, TaiChung, Taiwan.

Markl, V., Lohman, G. M., and Raman, V. (2003). LEO: An autonomic query optimizer for DB2. *IBM Syst. J., 42*(1), 98-106. doi: 10.1147/sj.421.0098

McKendrick, J. (2011). The Petabyte Challenge: 2011 IOUG Database Growth Survey.

Mell, P., and Grance, T. (2009). The NIST definition of cloud computing. *National Institute of Standards and Technology (NIST)*. doi: citeulike-article-id:8789668

Milenova, B., Yarmus, J., and Campos, M. (2005). *SVM in oracle database 10g: removing the barriers to widespread adoption of support vector machines.* Paper presented at the VLDB '05: Proceedings of the 31st international conference on Very large data bases, Trondheim, Norway.

Miller, B. (2005). The autonomic computing edge: The role of knowledge in autonomic systems. doi: citeulike-article-id:335721

Mueller, R., and Teubner, J. (2009). *FPGA: what's in it for a database?* Paper presented at the Proceedings of the 35th SIGMOD international conference on Management of data, Providence, Rhode Island, USA.

Oliveira, F., Nagaraja, K., Bachwani, R., Bianchini, R., Martin, R. P., and Nguyen, T. D. (2006). *Understanding and validating database system administration*. Paper presented at the Proceedings of the annual conference on USENIX '06 Annual Technical Conference, Boston, MA.

Oracle. (2011a). Oracle Data Mining Documentation. http://www.oracle.com/technetwork/database/options/odm/documentation/datamining-091203.html

Oracle. (2011b). Oracle PL/SQL Packages and Types Reference.

Peffers, K., Tuunanen, T., Rothenberger, M., and Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *J. Manage. Inf. Syst., 24*(3), 45-77. doi: 10.2753/mis0742-1222240302

Power, D. J., and Sharda, R. (2009). Decision Support Systems

Springer Handbook of Automation. In S. Y. Nof (Ed.), (pp. 1539-1548): Springer Berlin Heidelberg.

Practical, B. (2011). Request Tracker. from http://bestpractical.com/rt/

Quinlan, J. R. (1986). Induction of Decision Trees. *Mach. Learn., 1*(1), 81-106. doi: 10.1023/a:1022643204877

Rabinovitch, G. (2009). *Policy-Based Coordination of Best-Practice Oriented Autonomic Database Tuning*. Paper presented at the Proceedings of the 2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns.

Rabinovitch, G., and Wiese, D. (2007). *Non-linear Optimization of Performance Functions for Autonomic Database Performance Tuning*. Paper presented at the Proceedings of the Third International Conference on Autonomic and Autonomous Systems.

Rissanen, J. (2004). Minimum Description Length Principle *Encyclopedia of Statistical Sciences*: John Wiley and Sons, Inc.

Russell, L. W., Morgan, S. P., and Chron, E. G. (2003). Clockwork: A new movement in autonomic systems. *IBM Syst. J., 42*(1), 77-84. doi: 10.1147/sj.421.0077

Schallehn, E. (2010). Database Tuning and Self-Tuning.

Serpa, A. A., Roncero, V. G., Costa, M. C., and Ebecken, N. F. (2008). Text Mining Grid Services for Multiple Environments. In Jos, M. Palma, R. A. Patrick, D. Michel, M. Marta, Jo and L. o Correia (Eds.), *High Performance Computing for Computational Science - VECPAR 2008* (pp. 576-587): Springer-Verlag.

Shasha, D. E. (1992). *Database tuning: a principled approach*: Prentice-Hall, Inc.

Shyam Varan, N. (2007). Champion-challenger based predictive model selection. *CORD Conference Proceedings*, 254-254. doi: 10.1109/SECON.2007.342897

Solka, J. (2008). Text Data Mining: Theory and Methods. *Statistics Surveys, 2*. doi: citeulike-article-id:3021047

Sullivan, D. G., Seltzer, M. I., and Pfeffer, A. (2004). Using probabilistic reasoning to automate software tuning. *SIGMETRICS Perform. Eval. Rev., 32*(1), 404-405. doi: 10.1145/1012888.1005739

Taylor, J. (2010). Operational Analytics: Putting Analytics to work in Operational Systems. http://www.oracle.com/us/products/applications/hyperion/operational-analytics-report-081829.pdf

Telford, R., Horman, R., Lightstone, S., Markov, N., O'Connell, S., and Lohman, G. (2003). Usability and design considerations for an autonomic relational database management system. *IBM Syst. J., 42*(4), 568-581. doi: 10.1147/sj.424.0568

Vengurlekar, N., Vallath, M., and Long, R. (2008). *Oracle Automatic Storage Management: Under-the-Hood \\and Practical Deployment Guide*: McGraw-Hill, Inc.

Wiese, D., and Rabinovitch, G. (2009). *Knowledge Management in Autonomic Database Performance Tuning*. Paper presented at the Proceedings of the 2009 Fifth International Conference on Autonomic and Autonomous Systems.

Wiese, D., Rabinovitch, G., Reichert, M., and Arenswald, S. (2008). *Autonomic tuning expert: a framework for best-practice oriented autonomic database tuning*. Paper presented at the Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds, Ontario, Canada.

Wild, S., Curry, J., and Dougherty, A. (2003). *Motivating non-negative matrix factorizations.* Paper presented at the SIAM Conference on Applied Linear Algebra.

Winkler, W., and Nov, P. (2006). Overview of record linkage and current research directions.

Ziauddin, M., Das, D., Su, H., Zhu, Y., and Yagoub, K. (2008). Optimizer plan change management: improved stability and performance in Oracle 11g. *Proc. VLDB Endow., 1*(2), 1346-1355. doi: 10.1145/1454159.1454175

APPENDICES

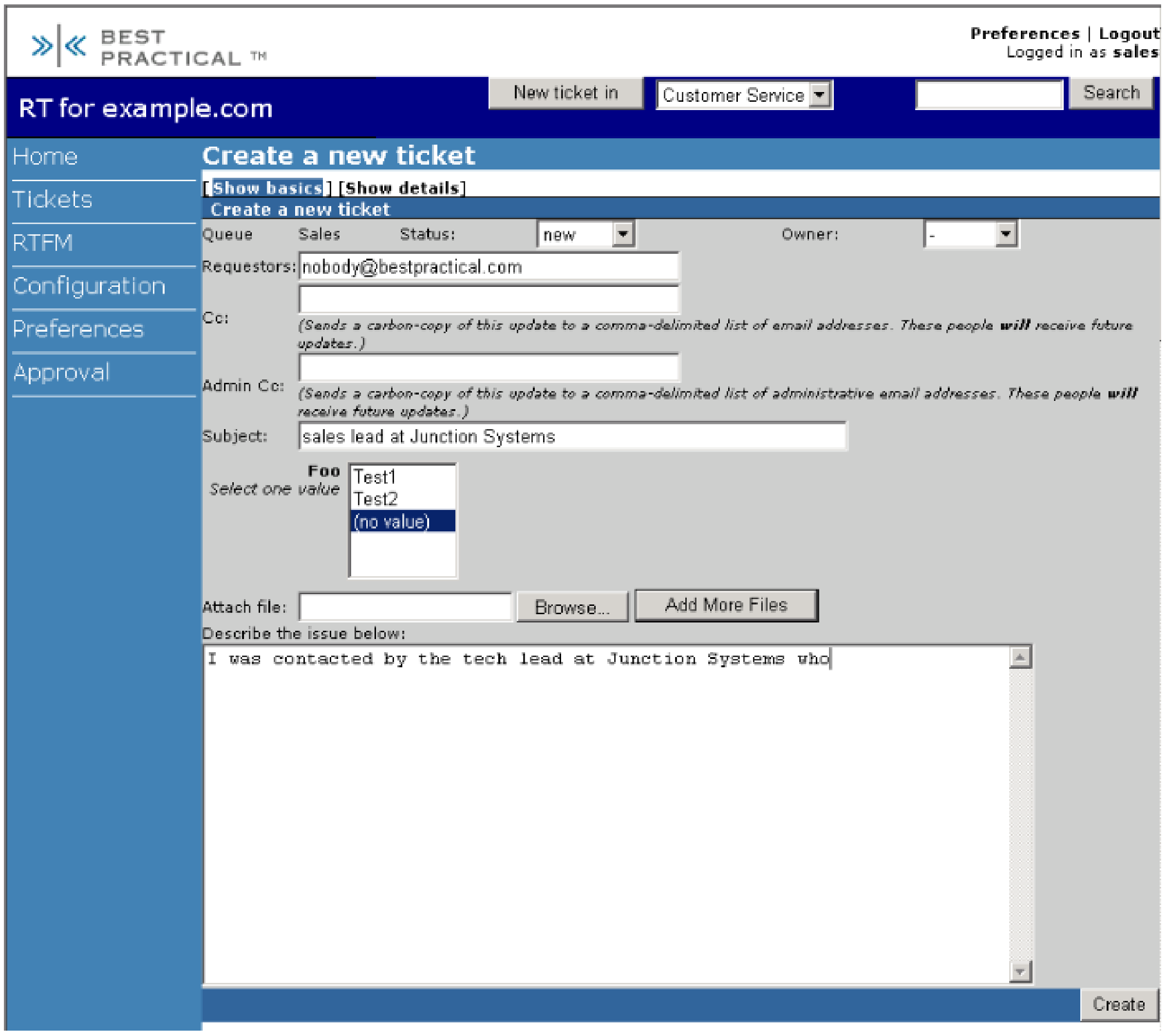
APPENDIX A: GLOSSARY

Acronym	Definition
CDB	Change Management Database
CDSS	Cognitive Decision Support System
CF	change factors that are extracted using NMF
CIK	Change Impact Knowledge
CO	NMF coefficients
CPI	Classified impact for historical terms by IPM
DBA	Database Administrator
DCF	Dependent change factors that are part of the NMF Feature identifications
DECIPHER	Database Environmental Change Impact Predictive-analysis for Human-driven Tuning Efforts in Real-time
DML	Data Manipulation Language
DSS	Decision Support System
DW	Data Warehouse
ECIK	Environmental Change Impact Knowledge
ETL	Extraction Transformation Loading
ERP	Enterprise Resource Planning
FEID	NMF Feature Identification
IED	Impact Extent Database
IEIM	Impact Extent Identification Module
IP	Predicted Impact
IPM	Impact Prediction Module
IT	Information Technology
GPL	General Public License
MAPE	Monitor Analyze Plan Execute
MDL	Minimum Descriptor Length
NMF	Non-Negative Matrix Factorization
OLTP	Online Transaction Processing
RT	Request Tracker
SA	Situation Assessment

SAW	Situation Awareness
SF	Change factors scored in real-time
SLA	Service Level Agreement
SVM	Support Vector Machine
TF-IDF	Term Frequency-Inverse Document Frequency
WCIK	Workload Change Impact Knowledge

APPENDIX B: REQUEST TRACKER SCREENSHOTS

1) RT Main Screen



Queue Sales Status: new Owner: -

Requestors: nobody@bestpractical.com

Cc:

(Sends a carbon-copy of this update to a comma-delimited list of email addresses. These people **will** receive future updates.)

Admin Cc:

(Sends a carbon-copy of this update to a comma-delimited list of administrative email addresses. These people **will** receive future updates.)

Subject: sales lead at Junction Systems

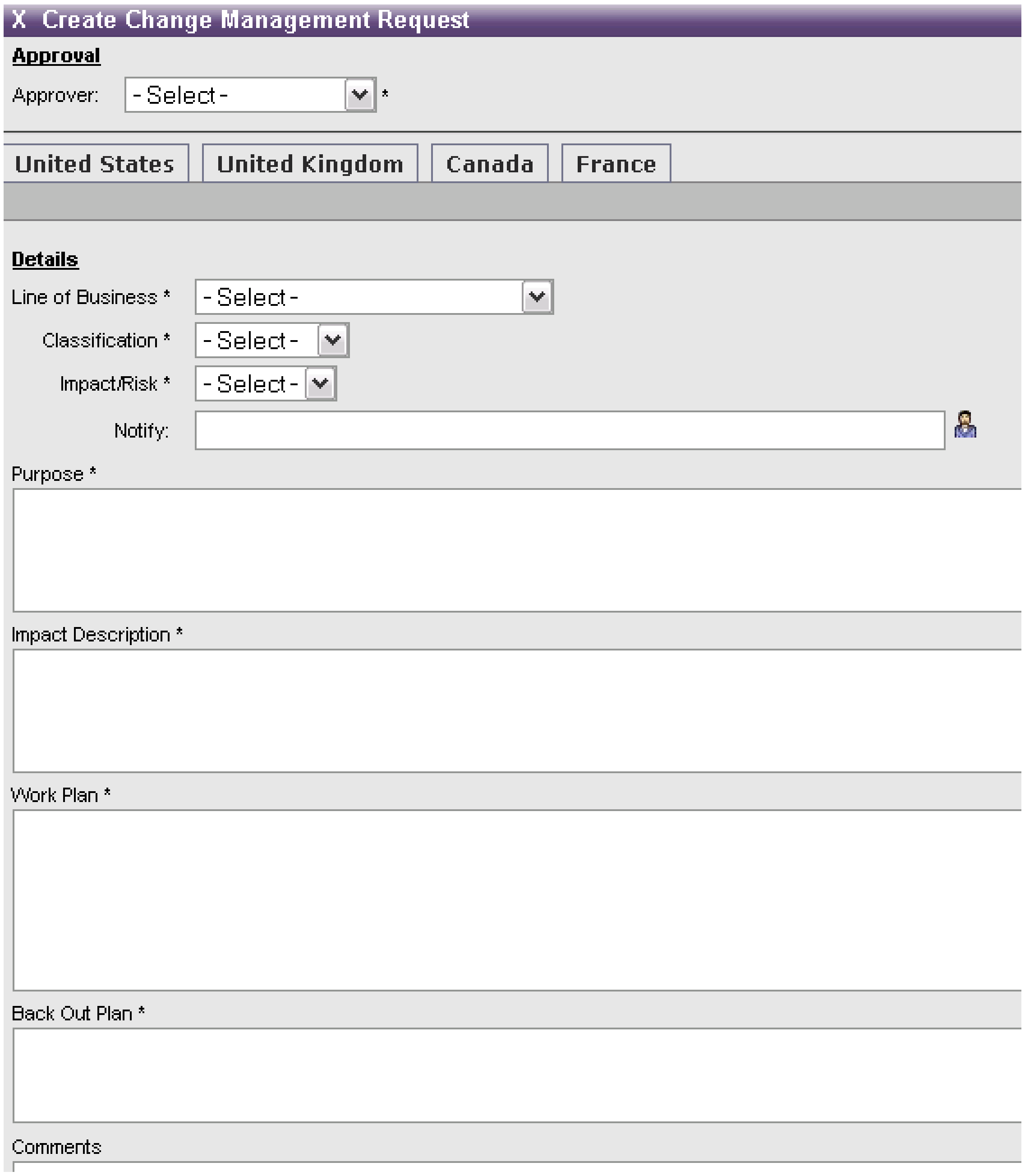
Foo *Select one value*: Test1 / Test2 / (no value)

Attach file:

Describe the issue below:

```
I was contacted by the tech lead at Junction Systems who
```

2) Change Request Creation Web Form



X Create Change Management Request

Approval

Approver: - Select - *

United States | United Kingdom | Canada | France

Details

Line of Business * - Select -

Classification * - Select -

Impact/Risk * - Select -

Notify:

Purpose *

Impact Description *

Work Plan *

Back Out Plan *

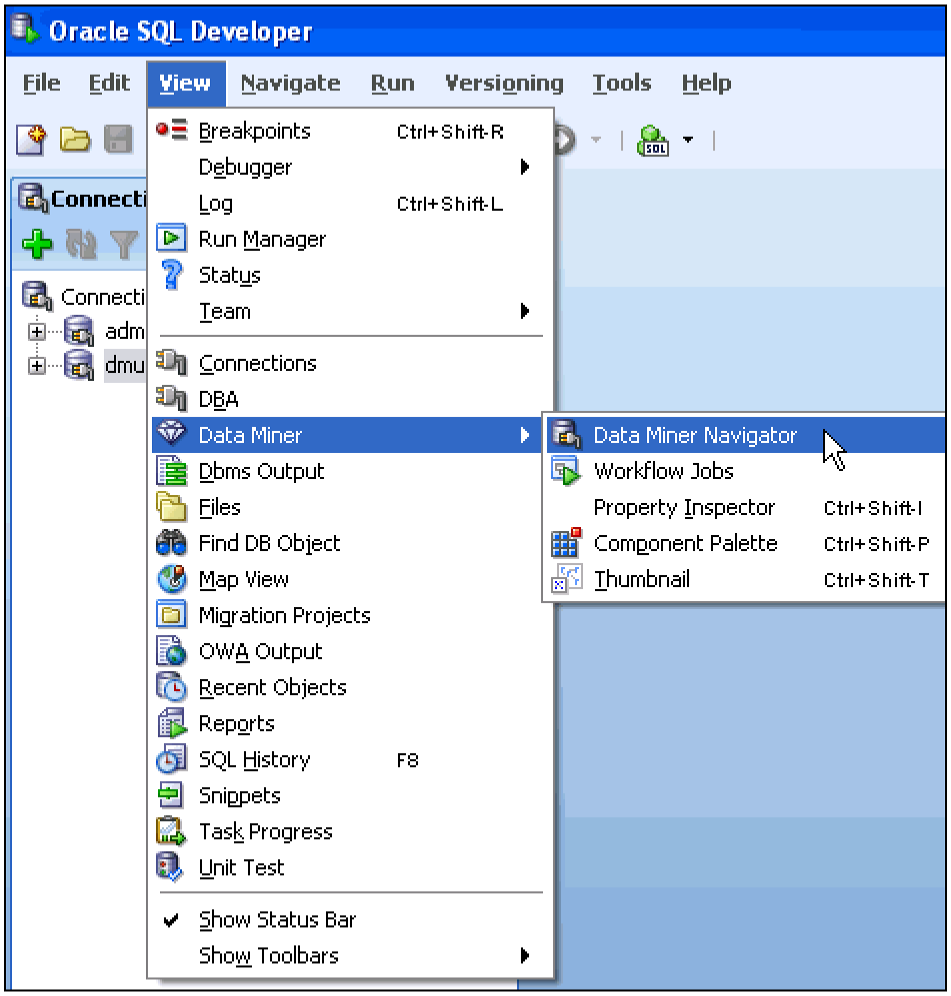
Comments

APPENDIX C: PATTERN REOCCURRENCE PSEUDOCODE

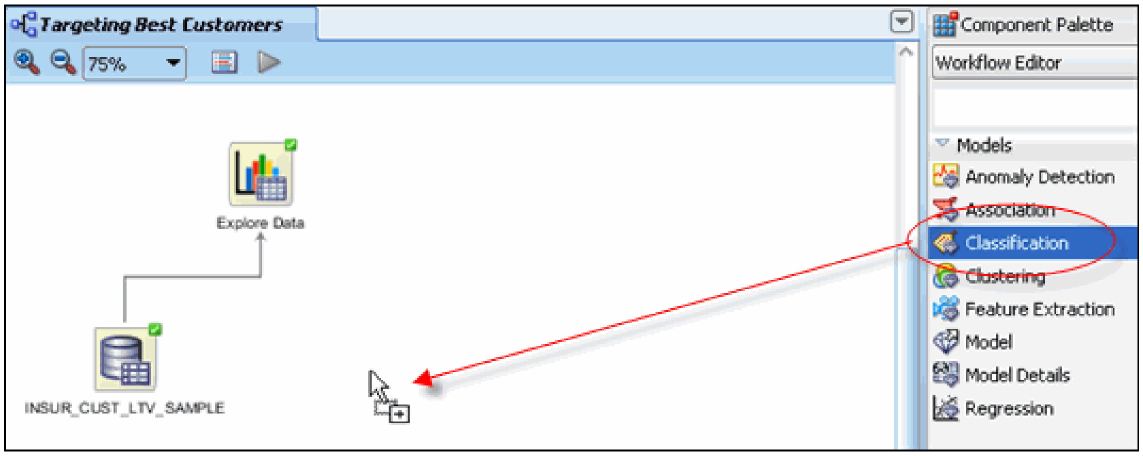
```
for each unimplemented change factor grouped by NMF feature id and sorted by NMF
coefficient in descending order
loop
  for each IED change factor sorted by NMF coefficient in descending order
  loop
     match factors using the jaro-winkler similarity function;
     get the score of the match;
     if the match score greater than or equal to the user-defined match score threshold
     then
       save the unimplemented change factor and the score in a temporary match result
       table;
  end inner loop;
end outer loop;
for each feature id  loop
  get max similarity value;
  save the max similarity values of each feature set in a temporary table;
end loop;
get average of the max similarity;
return the result;
end;
```

APPENDIX D: ORACLE DATA MINER SCREENSHOTS

1. Oracle SQL Developer Main Screen

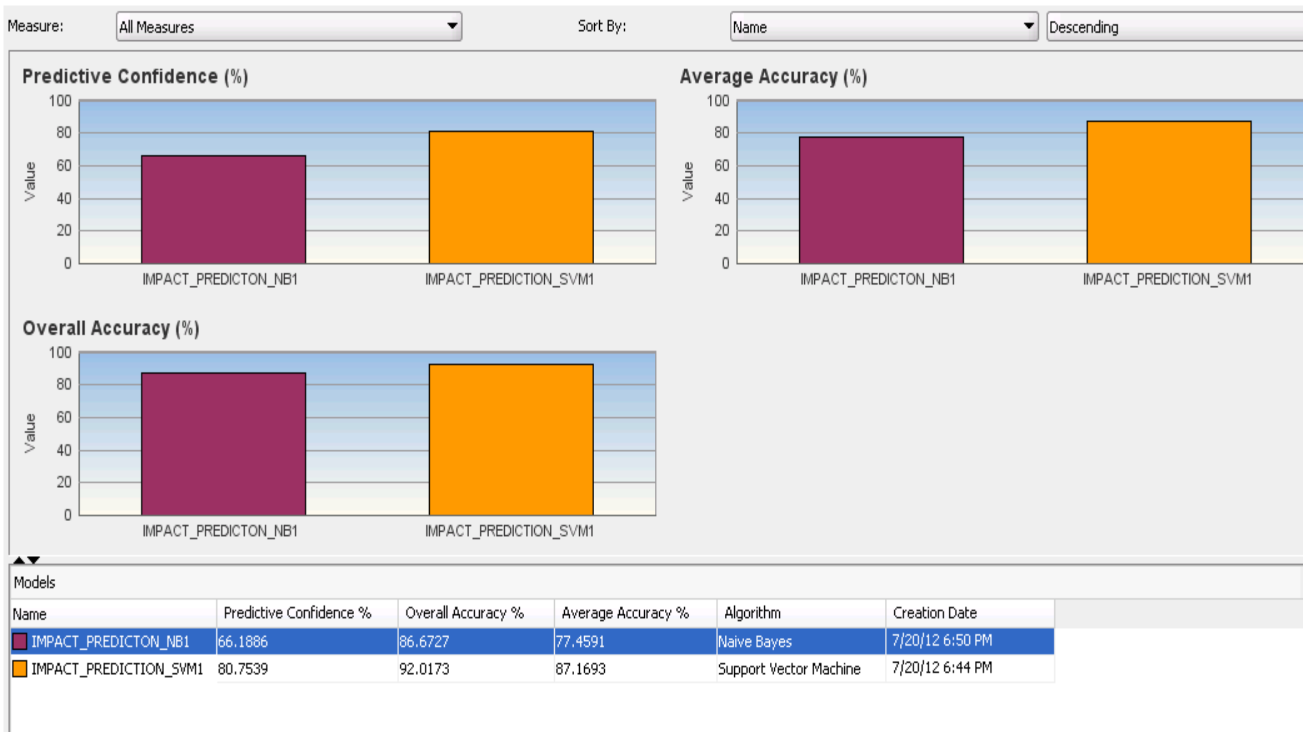


2. Oracle Data Miner Component Palette



APPENDIX E: ORACLE DATA MINER IPM SCREENSHOTS

IPM Overall Performance Measures (4 years of change data):



Name	Predictive Confidence %	Overall Accuracy %	Average Accuracy %	Algorithm	Creation Date
🟥 IMPACT_PREDICTON_NB1	66.1886	86.6727	77.4591	Naive Bayes	7/20/12 6:50 PM
🟧 IMPACT_PREDICTION_SVM1	80.7539	92.0173	87.1693	Support Vector Machine	7/20/12 6:44 PM

Confusion Matrix by Impact level for 4 years of change data:

SVM IPM Model (4 years of Change Data):

Impact	Correct Predictions %	Correct Prediction Count	Total Count
High	80	68	85
Medium	87.18	1177	1350
Low	94.56	2801	2962

NB IPM Model (4 years of Change Data):

Impact	Correct Predictions %	Correct Prediction Count	Total Count
High	67.05	57	85
Medium	71.25	962	1350
Low	94.42	2792	2962

APPENDIX F: CONFUSION MATRIX BY IMPACT FOR CREATOR-OWNER MODELS

SVM IPM Model (4 years of Change Data):

Impact	Correct Predictions %	Correct Prediction Count	Total Count
High	11.76	10	85
Medium	37.85	511	1350
Low	87.20	2583	2962

NB IPM Model (4 years of Change Data):

Impact	Correct Predictions %	Correct Prediction Count	Total Count
High	9.41	8	85
Medium	23.18	313	1350
Low	76.09	2254	2962

APPENDIX G: OLD AND NEW CHANGE DATA SET DETAILS

Old Change Data Set	New Change Data Set
2009 Quarter 1	2010 Quarter 1
2009 Quarter1 and 2	2010 Quarter1 and 2
2009 - full year	**2010 - full year**
2008 and 2009 Full years	2010 - full year

APPENDIX I: CONFUSION MATRIX BY IMPACT FOR ONE YEAR DATA

SVM IPM Model (1 year of Change Data):

Impact	Correct Predictions %	Correct Prediction Count	Total Count
High	92.85	26	28
Medium	93.41	908	972
Low	96.89	2062	2128

NB IPM Model (1 year of Change Data):

Impact	Correct Predictions %	Correct Prediction Count	Total Count
High	67.85	19	28
Medium	73.86	718	972
Low	94.07	2002	2128

APPENDIX J: CONFUSION MATRIX BY IMPACT FOR DB AND ALL QUEUE MODELS

ALL Queue IPM model built using 2009 full year Change Data scored against 2010 DB Queue Change Data:

SVM:

Impact	Correct Predictions %	Correct Prediction Count	Total Count
High	86.95	20	23
Medium	87.77	201	229
Low	72.86	419	575

NB:

Impact	Correct Predictions %	Correct Prediction Count	Total Count
High	73.91	17	23
Medium	79.91	183	229
Low	62.26	358	575

DB Queue IPM model built using only 2009 DB Change Data scored against 2010 DB Queue Change Data:

SVM:

Impact	Correct Predictions %	Correct Prediction Count	Total Count
High	47.82	11	23
Medium	50.21	115	229
Low	64.69	372	575

NB:

Impact	Correct Predictions %	Correct Prediction Count	Total Count
High	34.78	8	23
Medium	43.66	100	229
Low	53.39	307	575