Towards a Training-Oriented Adaptive Decision Guidance and Support System

Farhana H. Zulkernine[#1], Patrick Martin[#2], Sima Soltani[#3], Wendy Powley[#4], Serge Mankovskii[*5], Mark Addleman[ǂ6]

[#]*School of Computing, Queen's University, Kingston, ON, Canada*

[1]farhana@cs.queensu.ca, [2]martin@cs.queensu.ca, [3]soltani@cs.queensu.ca, [4]wendy@cs.queensu.ca

[*]*CA Labs, CA Technologies, Toronto, ON, Canada*

[5]serge.mankovskii@ca.com

[ǂ]*CA Technologies, CA, USA*

[6]mark.addleman@ca.com

***Abstract*—Information systems today have become incredibly complex and span multiple organizational networks, database and applications servers and on to the external Internet cloud resources. Consequently strategic approaches are needed to troubleshoot system failures by first identifying the component causing the failure, and thereby, further investigating the cause of the failure to solve the problem. Information regarding past troubleshooting strategies can be used to provide guidance for solving similar problems. We present a framework, DSDAware (Decision Support for Database Administrators using Warehouse-as-a-service) for developing a Decision Guidance and Support System (DGSS). The framework dynamically extracts knowledge from various correlated data sources containing systems related data and from the problem solving procedures of the human experts. The knowledge is used in a strategic problem solving approach to train new administrators by guiding them through the troubleshooting process using an interactive interface, and to offer a decision support service to the Web community. Our work specifically focuses on z/OS Mainframe DB2 database (DB) problems where the inherent complexity of the system makes troubleshooting a challenging task. The diminishing population of mainframe DB administrators (DBA) asserts the need for a DGSS for the new DBAs. The research applies text and data mining techniques for knowledge extraction, a rule-based system for knowledge representation and problem categorization, and a case-based system for providing decision support.**

I. Introduction

Decision Support Systems (DSS) and Decision Guidance Systems (DGS) have long been an active field of research striving to facilitate decision making tasks in various application domains [7][18][21]. Decisions are guided by knowledge, which is extracted from data using many different techniques. Evolution in technology has lead to automated and fast data collection using many different monitoring tools. However, extracting useful knowledge from this data is an important ongoing research problem. A major application of the knowledge is to generate further knowledge by finding correlations and drawing inferences, and to provide decision support. DSSs have come a long way and many organizational and commercial DSSs are now available [1][20]. Ongoing research focuses on combining human and machine intelligence and dynamic extraction of knowledge from machine interactions with the human experts [6].

An important application of the DSSs is troubleshooting system failures, which has been somewhat less addressed by the research community partly because software developers already provide tools and support for troubleshooting [5][7]. It has, however, become a very challenging task with the evolution of large complex multi-component systems and more intelligent support is required. For example, failure to access some data can be caused by a part of the application software, the network, the operating system, or the database in the back end. Modern systems are generally equipped with many different types of monitors that report messages, alerts and values of various system parameters [3][11]. Discovering the important information about the problem from all this data poses another problem. Often problem reports provide very little or misleading information about the actual cause of the problem. Administrators need to have knowledge of the system architecture and its various components including the various monitoring and system tools to efficiently investigate and solve problems. Prior knowledge and expertise can, therefore, be an asset in such endeavour since it helps to follow the same route of a previously reported problem of similar type [15].

Typically each system component comes with manuals and tools for troubleshooting and solving problems specific to that component [1][14]. However, often the challenge lies in identifying the component causing the problem due to the dependencies among the various components. A decision support framework that provides a strategic guidance for solving problems in complex multi-component systems can be used to train the new system administrators as well as help guarantee the quality of performance.

The state-of-the-art DSSs and DGSs apply many different automated and semi-automated techniques for knowledge acquisition, and provisioning of the decision support functionality [1][6][7][18][21]. Most of these approaches address a specific application or problem domain [6][19], which influence the techniques chosen for knowledge acquisition, representation and communication [1]. For example, DSSs exist today for medical diagnostics [19], management decision making [20], and geographic systems [6]. However, very few of these systems provide guidance for strategic decision making i.e., to explore probable options and

interactively guide the user to arrive at a decision specifically for systems management.

We propose a generic strategic problem solving approach for systems management, and a training-oriented interactive and adaptive framework, DSDAware (Decision Support for Database Administrators using Warehouse-as-a-service), for Decision Support and Guidance Systems (DGSS). It applies offline data mining to extract generic rules from the historical data and an interactive interface to dynamically extract knowledge from human experts as they perform investigation on both historical and real-time data. The knowledge thus acquired is stored as different data models, and as rules and case-bases (CB), which are used to provide training and guidance to new administrators during their problem solving process. The framework also provides an interface for warehouse-as-a-service to share and exchange knowledge with external users outside the organizational boundary. The research focuses on z/OS Mainframe DB2 [1] database management system (DBMS) due to the scarcity of experts in this domain, and the inherent complexity of troubleshooting these systems. Our DSDAware framework gathers data in a data warehouse, extracts knowledge into a knowledge base for providing decision support, and allows sharing of the knowledge as a service to the Web community.

The next section presents the related work in this area. An overview of our strategic problem solving approach is discussed in Section 3. Section 4 presents the DSDAware framework. The work-in-progress is discussed in Section 5. Finally, Section 6 summarizes our contributions and draws the conclusion.

II. RELATED WORK

DSSs and DGSs have been interesting topics of research for a number of decades and their applications are widespread; from clinical systems, geographic information systems, systems and risk management to Web-based negotiations and business intelligence [12][14][18][19][20][21]. DSSs can be very effective in automating the regular routine tasks, and thereby, relieving the operator to struggle to remember the common rules. These systems were built originally as expert systems by extracting knowledge from human experts, which proved to be time consuming and difficult [8]. With automated data collection techniques, now the effort is being directed towards applying automated or semi-automated knowledge extraction techniques to build efficient DSSs [9][23]. Where human judgement and cognitions are important, DGSS are implemented to provide guidance to human experts given the knowledge extracted from analysed data [14].

We focus on a DGSS for z/OS Mainframe DB2 database administration. Database administrators (DBAs) face extreme challenges due to the requirement to understand a system architecture that supports both legacy batch jobs and distributed networked applications [2]. It also uses time sharing option for parallel execution of hundreds of critical transactions in both data sharing and non data sharing modes. Previous work on DSSs in the area of systems management addresses extracting error diagnostics from single log files of applications [17][22], and events from various system components [10].

Many commercial tools also exist, which assist in the management on mainframe systems [11] such as ABEND-AID and File-AID from Compuware [5], Log Analyzer from IBM [14], and a suite of management tools from CA Technologies [3]. However, they do not provide strategic step-by-step guidance for problem solving but rather help to detect, debug or undo specific errors. Most of these tools do not provide a platform independent service interface like our DSDAware framework.

Carneiro *et al.* [4] propose the DBSitter approach that uses a multi-agent framework to collect data on the operational status of a DBMS. The data is analysed by a reasoning agent and the problems are either corrected automatically or presented to the DBA with a set of possible options. Although the concept is similar to our approach, the DBSitter requires an elaborative agent framework for data collection in a predefined format from very specific sources that simplifies the knowledge extraction process. The application is designed for problem solution rather than guided problem investigation. Musen [16] asserts the importance of using ontology in intelligent systems for rule-based problem detection but does not provide complete system architecture as to how such systems may be implemented. We propose a similar idea of defining an ontology for correlating different data sources for error detection using data and text mining techniques.

Research on error detection has spanned into several directions such as event driven root cause analysis techniques for software systems [10], and data and log mining techniques to identify data that indicates an exception situation [17] [22][23]. Xu *et al.* [22] apply source code analysis with information retrieval applied to only console logs to identify problem features. The features are further analysed using principal component analysis anomaly detection technique to learn and to detect operational problems. Instead of using time windows, the authors use message variables to group a set of correlated messages. Finally, a decision tree visualization approach is used to display the anomalies.

Mantaras *et al.* [15] focus on a CB reasoning (CBR) approach in general to solve problems based on previous knowledge of similar problems. Guo *et al.* [9] present an approach to acquiring cases for the CB using data mining techniques and reducing the number of attributes used to define a case. A similar approach is investigated in our work since there can be numerous parameters in the data collected by the various monitoring tools and it is important to identify a set of parameters that can distinctly identify the problem. Our framework applies a hybrid approach using rule-based and CBR to design an adaptive training-oriented DGSS.

III. OVERVIEW OF OUR APPROACH

Troubleshooting systems management issues on complex multi-component legacy systems is a challenging task. It requires:

- Critical knowledge of the concerned problem domain.

- A good strategy for troubleshooting so that other systems are not affected.
- Knowledge of the complete system architecture to understand the cause of the problem.

In this section we discuss the problem domain of troubleshooting z/OS Mainframe DB2 DBMS, its challenges, and our hypotheses behind the design of the DSDAware based on our interviews with the DBAs in a large organization [1]. Since there are diminishing numbers of experts in this domain and the new DBAs have to go through a steep learning curve for understanding the different features and components of the system, there is a crucial need for a DGSS in this area.

In this paper, we propose a preliminary version of our DSDAware framework that would:

- Build a history of previous problem cases in a case-base indexed by the problem features.
- Provide an interactive interface to extract knowledge from the expert DBAs while they solve the problems.
- Extract knowledge automatically from various distributed systems management data sources in the organization using text and data mining techniques.
- Use the knowledge for guiding the new DBAs in investigating and solving DB problems.
- Provide a warehouse-as-a-service interface for external users to inquire and contribute knowledge.

We first discuss the problem domain and then present an overview of our approach.

A. *The z/OS Mainframe DB2 Challenges*

DB2 on z/OS has primarily two different types of system setup: *non-data sharing* and *data sharing* [1]. Each setup is used for different types of applications such as legacy COBOL batch jobs or distributed Web-based applications with varying workloads and user groups. A mainframe system typically has multiple logical partitions (LPAR) having different storage settings, each hosting one or more instances or subsystems of DB2. In a data sharing setup, multiple DB2 instances share the same data set through a data connector. So, when a problem is notified the DBA has to check which subsystem, SQL thread, and data set are involved. Datasets are stored in table and index spaces as VSAM (Virtual Storage Access Method) linear datasets with specific naming conventions. The DBA should understand the multipart naming convention to detect which particular dataset is giving the problem.

The z/OS Mainframe DB2 system comprises several other sub-components for information (IMS) and transaction management (CICS) [1][5], which use different types of DB connections, queries and update operations with different priority levels. Knowledge of the application type and these components helps the DBA to troubleshoot a problem efficiently. In addition, a Job Entry Subsystem (JES) is typically used with a Time Sharing Option (TSO) on the z/OS to receive, schedule and manage jobs in the job queue and their outputs [1][5]. The JES master log file contains messages logged by different system components involved in the execution of the jobs. Therefore, it is an important source of information for the DBAs to investigate conflicts among multiple components, and systems resources. The DB2 Log manager uses other log files for DBMS transactions only, which are used for data backup and recovery [2].

There are also a number of tools with which the DBAs must be familiar [3][5][14]. These tools are used to configure and monitor the system, investigate problems, and execute commands. As a result, troubleshooting on z/OS for DB2 can be very challenging. Often problems in other components including the network and the application servers cause failure in data access and the report gets forwarded to the DBAs. Therefore, DBAs need to understand the system architecture, applications, and the user groups to identify, address and resolve DB2 related problems.

B. *The Problem Solving Life Cycle*

We conducted an on-site study of the problem solving strategy followed by an expert mainframe DBA at a large organization. We observed the troubleshooting processes for a number of different system problems, which were reported to the DBAs from multiple sources. The study also included interviewing DBAs working at different levels such as looking after systems related issues for standalone legacy software or web applications to specification of requirements for software products to be used for DBMS. The study revealed the following facts:

- Problems are reported in a variety of ways and formats, for example, through help desk forms, events or alerts generated by monitoring tools, by auto-generated or forwarded emails containing log data or by error message.
- The information in the problem report may indicate a very different problem than the actual problem.
- Common investigations are made to retrieve further information based on the data given in the problem report.
- The investigation and problem solving strategy is largely influenced by the available tools and the comfort level of the DBA with using those tools.

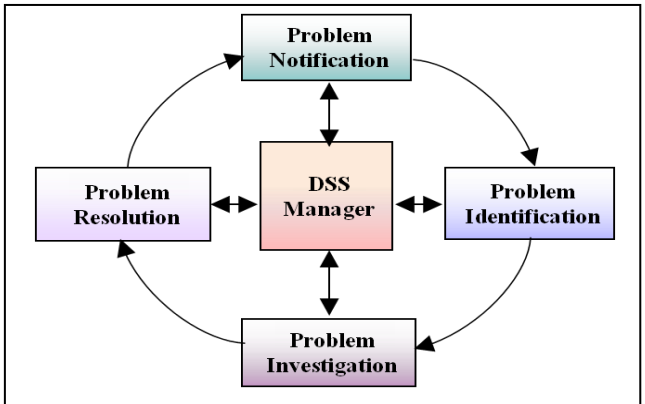

Fig. 1 Problem solving life cycle for large complex software systems

- The domain knowledge of the DBA about the system architecture also governs the path that the DBA follows for problem investigation and resolution.
- The level of expertise gathered from dealing with similar problems in the past helps to ascertain the cause of the problem and quickly apply an effective solution.

Based on the above observations, the troubleshooting process of large complex software systems can be conceived of having generally four main steps as shown in Fig. 1.

1) Problem Notification: This is the step that receives the problem report in different ways and formats. Depending on the apparent cause of the problem, the report gets forwarded to the appropriate administrator, for example, the DBA, network, application, or z/OS systems administrator. The problem notification messages often resemble the messages that are found in the JES log file. As explained before, it is an important source for problem investigations because different system components log messages in that file. If other than DSN errors were logged at the time a problem was reported there may not be a DB problem.

Fig. 2 shows an example of the JES master log file. Each message is preceded by a date field. The error codes starting with DSN indicate database errors (STC codes indicate Started Task Control codes used by JES). DSN codes ending with 'E' means an error and 'I' means information. However, many times information messages lead to the cause of the problem. For example, the last three messages in Fig. 2 help detecting the cause of a resource conflict problem, which was reported as a log in problem for the application users. A problem report can contain any one of these messages.

```
21.00.25  STC09048 IEF695I  START  SS11MSTR WITH JOBNAME  SS11MSTR IS ASSIGNED
          TO USER  MADPRD  , GROUP OMVSGRP
21.00.25  STC09048 $HASP373  SS11MSTR STARTED
21.00.25  STC09048 ACF9CCCD USERID MADPRD  IS ASSIGNED TO THIS JOB - SS11MSTR
21.00.25  STC09048 IEF403I  SS11MSTR - STARTED - TIME=21.00.25
03.58.02  STC09048 DSNJ110E  !SS11 LAST COPY 1 ACTIVE LOG DATA SET IS 90
          PERCENT FULL
03.58.05  STC09048 DSNJ110E  !SS11 LAST COPY 1 ACTIVE LOG DATA SET IS 95
          PERCENT FULL
03.58.10  STC09048 DSNJ111E  !SS11 OUT OF SPACE IN ACTIVE LOG DATA SETS
13.28.15  STC10799 DSNI031I  !SS11  DSNILKES - LOCK ESCALATION HAS  337
  337       OCCURRED FOR
  337       RESOURCE NAME = CCM30.CHAN1AVS
  337       LOCK STATE =  X
  337       PLAN NAME : PACKAGE NAME = DISTSERV : SYSLH200
  337       COLLECTION-ID = NULLID
  337       STATEMENT NUMBER = 000001
  337       CORRELATION-ID = db2java1
  337       CONNECTION-ID = SERVER
  337       LUW-ID = ODCA1AD7.O744.C7F340EE13E9
  337       THREAD-INFO = ABCPROD : usilap533a.org.co : ABCPROD :
  337       db2java1appl
13.31.55  STC10799 DSNT376I  !SS11  PLAN=DISTSERV WITH  670
  670       CORRELATION-ID=db2java1
  670       CONNECTION-ID=SERVER
  670       LUW-ID=ODCA1AD7.FAC8.C7F17F909299=141851
  670       THREAD-INFO=ABCPROD: usilap533a.org.co: ABCPROD: db2java1appl
  670           IS TIMED OUT. ONE HOLDER OF THE RESOURCE IS PLAN=DISTSERV
  670       WITH
  670           CORRELATION-ID=db2java1
  670           CONNECTION-ID=SERVER
  670           LUW-ID=ODCA1AD7.O744.C7F340EE13E9=2777
  670       THREAD-INFO=ABCPROD: usilap533a.org.co: ABCPROD: db2java1appl
  670           ON MEMBER SS11
13.31.56  STC10799 DSNT501I  !SS11  DSNILMCL RESOURCE UNAVAILABLE  671
  671       CORRELATION-ID=db2java1
  671       CONNECTION-ID=SERVER
  671       LUW-ID=ODCA1AD7.FAC8.C7F17F909299=141851
  671       REASON 00C9008E
  671       TYPE 00000210
  671       NAME CCM30.CONF1X1W.00000001
```

Fig. 2 Example error messages in the JES master log file

2) Problem Identification: This step identifies the type of problem through some regular checks given the information or error message in the problem report in the previous step. These checks are basically done to simply verify if the reported problem is a DBMS problem, and if it should be further investigated by the DBA. In this step, a DBA typically searches for related information given the key attributes in the error notification. For example, if the DBA gets a notification of DSNJ110E error message for the SS11 subsystem (as shown in Fig. 2) then further information should be extracted about that subsystem. By the error code (ends with an E), it is known to be a critical DB error. For the DSNI031I Lock Escalation message, the THREAD-INFO parameter gives information about the user ID, workstation ID and the application ID and the LUW-ID identifies the logical unit of work ID or a specific DB command. Further information about the dataset (RESOURCE NAME), the job (PLAN NAME), the specific part of its execution (PACKAGE NAME), and the specific statement (STATEMENT NUMBER) are obtained in this step. The DBA also checks the monitoring applications for alert reports, and the current status of the application or DB command, and the datasets referred to in the error message. The JES log messages are checked as well to find more details about the associated errors if any were reported within a bounded time frame of the reported problem. This information is used at this stage for problem identification and for further investigation in the next step.

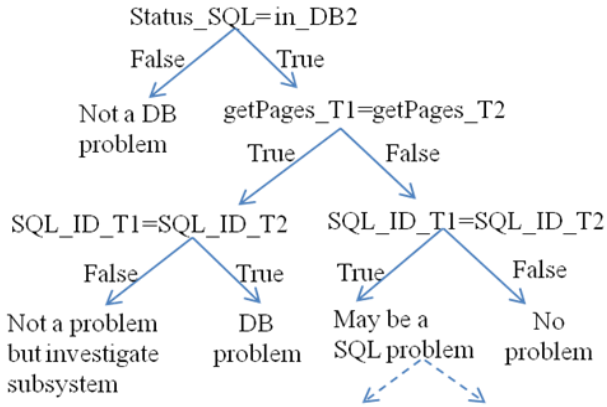

Fig. 3 Problem solving life cycle for large complex software systems

Fig. 3 shows a decision tree of rules (based on our interview), which the DBA follows to identify an SQL problem. If a data access error is reported, the DBA generally checks the current status of the related SQL command identified by LUW-ID in the problem report. The DBA uses one of the tools to check if the SQL statement is still accessing the DB2 data sources (STATUS_SQL=in_DB2), and if so, whether the number of pages being retrieved is changing (getPages_T*n* denotes pages retrieved at time T*n*, and T2>T1).

If both getPages and the SQL statement (SQL_ID_T*n* denotes the SQL statement being executed at time T*n*) are the same as before, then there may be some problem with that statement that needs further investigation, and hence, indicates a possible DBMS problem. But if both getPages and SQL_ID are changing then the problem may be elsewhere and a message should be forwarded to the application server admins.

3) Problem Investigation: Once a problem is identified as a DBMS problem, it needs to be further investigated to find the actual cause of the problem so that it can be addressed accordingly. This is the most complex step because of the numerous ways problems can be investigated based on the strategy followed by a DBA, the tools used, and the information available at each step. Two different problem investigation scenarios are discussed below.

For example, an error is reported for an application that the users are receiving data access errors. The problem report is forwarded to the concerned DBA. After a few hours the problem is automatically solved by the DBMS system. However, an investigation is initiated to find the cause of the problem. Based on the error report, the DBA checks the current application status, the data sources, and the submitted job history and no anomaly is detected. The DBA also checks the log messages and alerts. The alerts show a large number of DSNI031I Lock Escalation messages as shown in Fig. 2. Further investigation of the JES log file reveals that the concerned application has a DSNT376I TIME OUT error because the resource it is trying to access is held by another application, which caused the lock escalation problem. Typically TIME OUT errors are followed by DSNT501I RESOURCE UNAVAILABLE messages as shown by the last message in Fig. 2. Thus the DBA confirms that the lock escalation caused the application error. The interesting observation is that although there are many lock escalation messages in the log file, only the ones that have resource conflicts with the reported application cause the application failure. Further investigation on the lock escalation problem checks the history to see if a similar error happened before, how frequently, which application caused it, what were the data sets, and what is the configuration setting for the threshold when the lock escalation occurs. These thresholds can vary greatly for different problem symptoms and system configurations. For example, a threshold of 1000 locks to trigger lock escalation may be good for most systems (if more than 1000 locks are acquired by an application, lock escalation occurs) but for other more complex systems it may be 20,000.

Another scenario is a SQL performance problem, which is difficult to investigate. Possible causes may be problems in acquiring necessary locks, ill structured SQL (unable to take advantage of an index for using inequality checks), absence of an index, bad scheduling of an analytical query during business hours, or insufficient system resources. All the probable causes have to be investigated strategically to detect the actual cause of the problem and to finally find a solution.

It is, therefore, a very challenging problem to extract all the knowledge as illustrated above to perform the checks automatically. The results of the investigation actions are non-deterministic i.e., new information may be retrieved or not, and the discovery from one action can call for a revised course of investigation. Also the same investigation actions are often made for many different problems.

4) Problem Resolution: This is the final step of the problem solving life cycle. Through the investigation steps, the DBA gets closer to the solution by collecting more information about the problem state, and thereby, narrowing down the probable causes of the error. Once enough information (beyond some predefined threshold which can be set by the minimum number of features used to identify problem cases) is obtained, the DBA applies expert knowledge, or consults fellow experts to find if a similar situation occurred in the past, and consults manuals to find a solution to the problem.

C. *Hypotheses based on the above Observations*

From our observations of the problem solving life cycle on a z/OS Mainframe DB2 syste, we propose several hypotheses as explained below about the technologies to apply and the approaches to follow to provide decision guidance and support for all four problem solving steps. Considering the difficulties and challenges at each step of the life cycle, we propose a hybrid approach to building our decision support framework, DSDAware, using rule based systems (RBS), case based systems (CBS), and interactive interfaces to guide the new DBAs through the troubleshooting process supported by data, knowledge and case repositories.

1) Hypothesis for Problem Notification: The report on the problem is processed automatically to note the time, priority, and implications of the problem and store the detailed information or message in the warehouse within the DSDAware framework for later processing.

2) Hypothesis for Problem Identification: Our hypothesis for this step is that since a set of regular checks are performed to identify a problem as a DB problem; the knowledge required for this step can be stored as decision rules and executed using a rule based system. Most of this information is available in the DB2 manuals [1] and information such as the error codes and the key terms can be stored in the data warehouse within the framework. The key terms are used to search for, and retrieve the corresponding values from the problem report, log messages and other data sources. All this information helps to define the problem features. Fig. 3 shows a decision tree representation of the rules which are structured as: if <condition> then <action>. Some of these rules are also retrieved from interviewing the DBA experts and by applying text and association rule mining techniques on the data sets. The proposed rule based approach helps extract necessary information automatically for the investigation step, and thereby, relieves the DBA of the regular trivial tasks.

3) Hypothesis for Problem Investigation: This is the most complex step and its efficiency largely influences the time to reach the ultimate goal of problem resolution. It cannot be automated reliably due to the difficulty in extracting and

representing all the analytical knowledge needed at this step, and in executing the highly analytical and cognitive tasks.

Knowledge about previous experiences can lead to an efficient problem resolution strategy. Therefore, we use Case Based Reasoning (CBR) with machine learning techniques to acquire knowledge about the system, reports on previous problems, and the problem solving strategy followed by the expert DBAs. When a problem is reported, the case base (CB) is searched based on the problem features. Case information is dynamically collected during the troubleshooting process and used for new case definition and adaptation.The challenges in this approach are:

- What parameters should be used to define the features of a problem? There are hundreds of parameters in the monitored data that are collected by the monitoring tools [11]. Furthermore, the set of parameters varies depending on the specific problem case.
- What data structure is most suitable for representing the CB considering that we may have insufficient information to define a case initially? This would also affect the case search and adaptation.
- How to efficiently search the CB. Not all problem data are available in the initial step. Searching the CB with insufficient information returns a large set of probable cases.
- How to manage and maintain the CB with new knowledge being acquired constantly.
- How to store the knowledge of step by step investigation.
- How to adapt the cases as new knowledge is acquired dynamically using the framework.

We also propose to combine the CBR and machine learning techniques with a specially designed interactive problem solving interface similar to that shown in Fig. 4, which would record the investigation steps. In this example interface, the DBA can select one of the attributes or keys (such as LUW-ID) from the message to query further information, select one of the options, or enter a custom query or command. This feature can be extended to support integration of different DB tools to facilate execution of the commands.

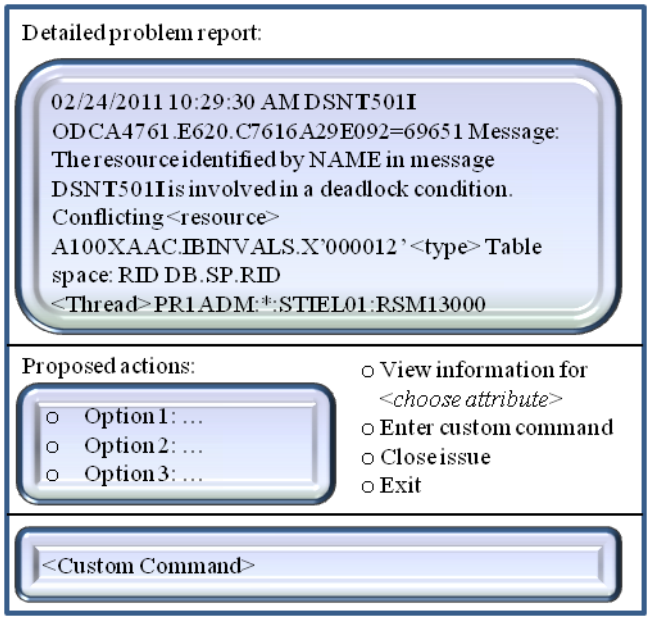

Fig. 4 The specially designed interactive screen in the DSDAware framework for knowledge extraction and providing guidance

Although we can record the data from the investigation steps, the challenge is to deduce the logic behind the sequence of actions, which may be values of certain parameters that the DBA observes and some reasoning that the expert applies to draw some inferences. Particularly, if the DBA backtracks or executes some out of context queries then we will get knowledge that may be misleading. However, we believe that once we get some trace data, it can be analyzed to reach a better hypothesis. At this stage, we propose to trace all the queries, DB commands and actions that the DBA executes and capture the screen snapshots that the DBA goes through.

We plan to store the above mentioned actions and trace data separately from the CB. Multiple cases can propose the same problem solving action for the next step. Action details include a description, type of operation (execute command to apply a change in the system or investigate a value to make inference), and possible implications. Separation of the actions will allow for easier case adaptation, and display of a set of options for the actions during provisioning of decision support. Action id, type and the weight of the action for a case are only stored in the case detail.

Based on the choice of action of the DBA, the decision model and the CB are adapted to retain new knowledge or adapt the decision support model by changing the weights of the suggested options.

4) Hypothesis for Problem Resolution: The CB as discussed above is used to search for probable solutions. We propose to use similar interactive interface as shown in Fig. 4 to display a list of solution options to select from in the order of their weights. The weight for a solution is computed from a number of statistics such as its applicability to the problem, past success and preference records. An option to officially close the issue would indicate that the problem has been resolved.

IV. The DSDAware Framework

We propose the DSDAware framework to provide decision guidance and support to the DBAs in investigating and troubleshooting z/OS Mainframe DB2 DBMS problems. The framework:

- Helps DBAs with the four steps of the problem solving life cycle.
- Extracts knowledge
  - Offline from the historical data collected in a data warehouse from various correlated distributed systems management data sources using machine learning techniques,
  - From querying and analyzing real time data using the attributes reported in the error message, and

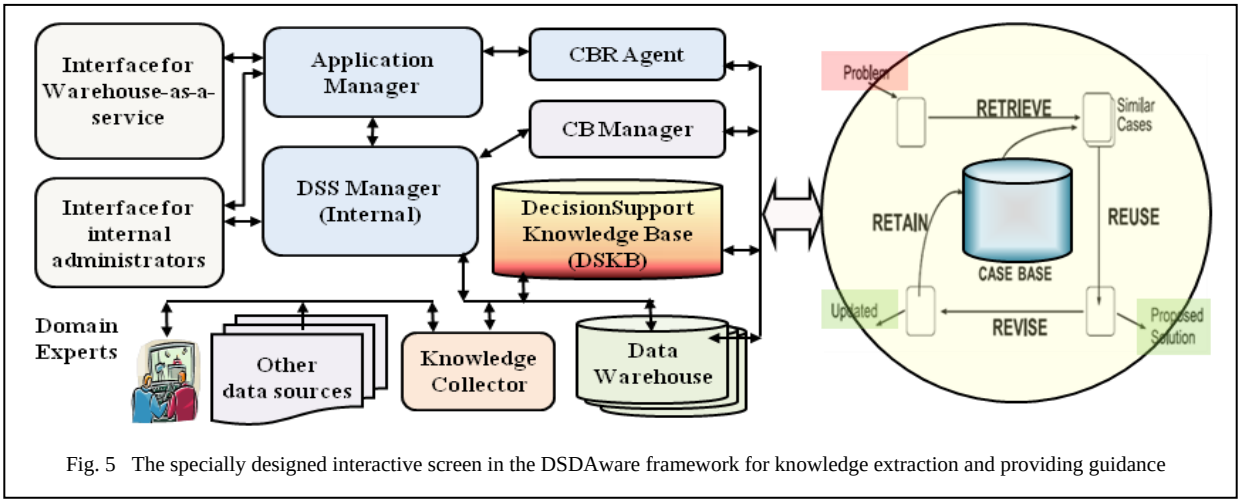

Fig. 5 The specially designed interactive screen in the DSDAware framework for knowledge extraction and providing guidance

- From the problem solving strategies of the expert DBAs during the problem solving procedure using a specially designed interactive interface.

- Builds a history of problem cases in a CB.
- Provides training and decision support to new DBAs in troubleshooting problems using the interactive interface.
- Provides a service interface for external users to query the warehouse for problem features and solutions from the associated CB.

A. *Description of the Framework*

Fig. 5 shows our DSDAware framework. There are two interfaces for communicating with the users. The application interface is used by the internal users of the organizations such as the DBAs, system administrators, and application developers (as shown in Fig. 4). The *warehouse-as-a-service* interface provides decision support services to the external Web users as shown in Fig. 6. In the service interface, the users can search for a solution by entering a text description such as "resource unavailable" with additional features such as "reason = 00C9008E". The lower pane of the screen displays solution based on the warehouse data, CB and the internal knowledge base if the user searches for a solution. Otherwise, it allows users to contribute solution information like the Internet forums. The information can be rated based on a contributor's profile and the acceptability of the solution to the DBAs later in the troubleshooting process.

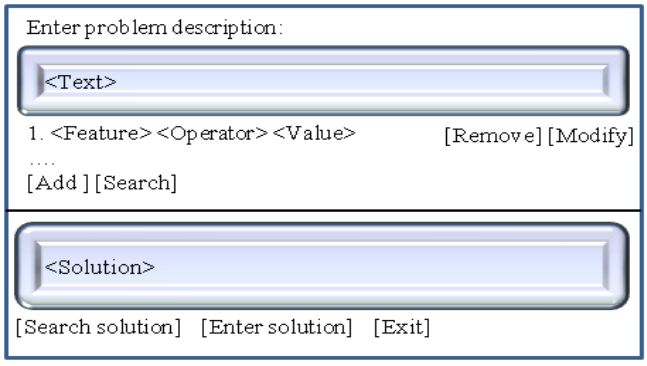

Fig. 6 Interactive Warehouse-as-a-Service interface

The *Application Manager* communicates with the users in two ways, i.e., processes the request, and furnishes the requested information in proper format and style with the help of the DSS manager.

The *DSS Manager* uses decision models to provide decision support information. Internal programmers can communicate with the DSS manager for configuration and administration purposes. The decision models use knowledge and data from the *Decision Support Knowledge Base (DSKB)*, the *Data Warehouse*, and the *Case Base (CB)*. During the problem solving process, the DSS manager communicates with the CB manager as guided by the decision models. It also communicates with the knowledge collector, the data warehouse and the DSKB for storing and retrieving necessary information.

The *Knowledge Collector* mainly extracts, preprocesses, and analyzes data from domain experts, various accessible data sources containing both historical and real time data, the DSS manager during the decision support process. It uses the existing knowledge to process the data and generate new knowledge and information, which is stored in the appropriate storage i.e., the data warehouse or the DSKB.

We store data and knowledge separately for better management, maintenance and provisioning.

- The data warehouse contains aggregated and analyzed data, related metadata and real time data from various monitoring tools, log files, user profile information, system information, and operational manuals.
- The DSKB contains inferred data, rules, and learned knowledge, which is the critical knowledge used to provide decision support and create and update decision models.
- The CB is the repository of historical problem cases indexed by a set of specified problem features.

The *CB Agent* is used only to retrieve case information whereas the *CB Manager* oversees the life cycle of the CB system (CBS). The data and knowledge from the different storages are used by the CB manager to define and update the

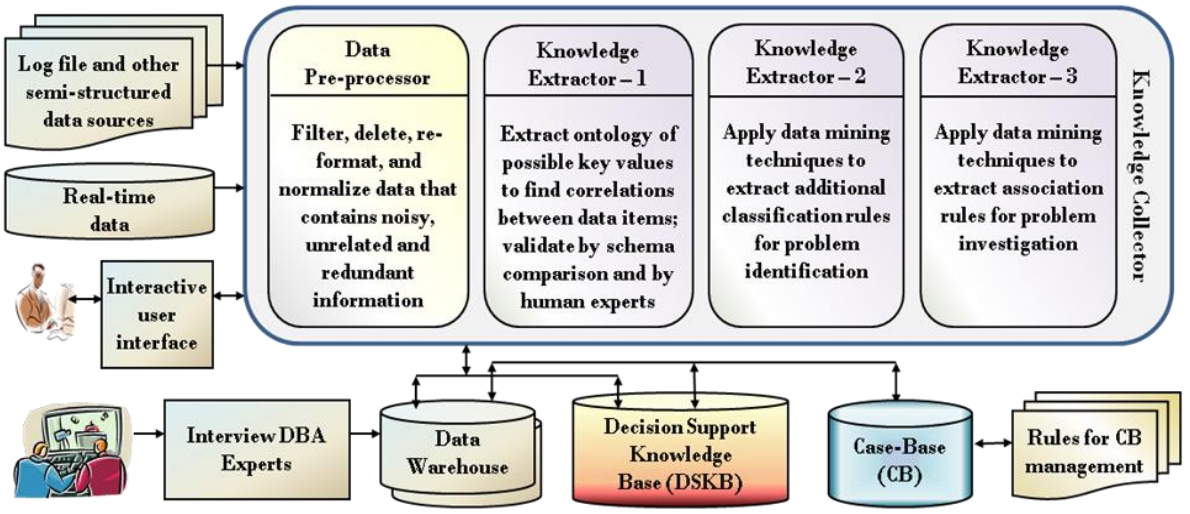

Fig. 7 The knowledge extraction and storage framework

CB. The CBS life cycle consists of four steps: case *retrieval*, *reuse*, *revise* and *retain*. Each of these steps in the CBS poses interesting challenges and represents an important area of research. Depending on the information domain, different data structures are used to design the CBS. *Case retrieval* focuses on organization and indexing of the information in the CBS. The *reuse* step dictates how the case information can be used to solve the problem. If the information does not properly match the problem or object state, the *revise* step is used to adapt the retrieved case to make it applicable to the current problem. The *retain* step is used to create new cases and store the adapted cases in the CBS.

When a request arrives at the DSDAware framework, the application manager directs the request to either the CBR agent for a quick inquiry, or to the DSS manager to execute the decision support process, which provides step-by-step guidance to solve the problem. It then communicates the responses back to the users through the proper interface.

B. *Knowledge Extraction and Storage Framework*

The knowledge collection and storage framework in Fig. 7 shows how the knowledge collector extracts information and knowledge and stores them in the DSDAware framework. By interviewing the experts, knowledge is obtained manually and stored in the data warehouse using one of the interfaces and the predefined data structures. The *Data Pre-processor* uses a semi-automated approach to clean, transform and organize the data. *Knowledge Extractors* (KE) 1, 2 and 3 perform multilevel knowledge extraction. KE-1 first extracts the key or attributes that have different synonyms in different data sources using text mining techniques in order to create an ontology. An ontology is essential to find a correlation among the different data sources, for example the log messages and the monitored parameters. Domain knowledge, for example, the log file structure and the message formats from the DB2 manuals, is necessary at this stage [1].

Applications are created to extract the key and value pairs from massive data sets. KE-2 applies data mining techniques to find frequent word groups to use as problem categories. Error codes are used in combination with the word group to define problem categories. KE-3 applies association rule mining techniques to find co-occurance of messages containing the same attribute values, which indicates possible causal relationships between the messages within a bounded and sliding time window.

The CB is connected directly to the data warehouse and the DSKB. The knowledge required for the CB manager for case management are stored separately as shown in Fig. 7 and defined by human administrators. Other information such as step-by-step action details and screen captures or other data collected during the troubleshooting process are stored in the data warehouse.

C. *An Example Case Template*

Fig. 8 shows an example case template. The high level problem category is useful to organize cases in a distributed manner. The other attributes are also used for case organization and retrieval. The error code, message, and system configuration attributes are self-explanatory. A case can contain information about how to solve a problem if the attributes reasonably identify the cause of the problem. Otherwise, a case may contain information about what may be the

Problem Attributes
▪ Problem category ▪ Error code ▪ Error message ▪ Case category ▪ System configuration ▪ Feature values [{feature, op, value}, …]
List of Actions
{Action id, type, weight} …
Inferences
Recent updates Number of applications Other rating

Fig. 8 Example of a case template

next step for investigating the problem further. There can also be multiple ways that a problem can be investigated or resolved. The case category attribute is used to identify the type of actions contained in the case and the list of actions are the possible suggested moves that can be made from this stage. The feature values are the monitored parameters. An expression of the same format as shown in Fig. 6 is used to match the feature values. For example, a feature defined by {lock_escalation_threshold, LT, 1000) indicates that the threshold value at which a lock escalation is triggered is less than 1000 for the observed system. Actions can be of two types: one that makes some modification in the system, the other that simply queries some value and makes some inference. A weight is associated with an action to indicate the importance of this action at this state. The inferences help in explaining the actions taken to solve a problem, rating the importance of a case based on the number of times it has been applied and other information such as user or system feedback about the effectiveness of the suggested actions.

D. *Warehouse-as-a-service*

It is important to specify a Service Level Agreement (SLA) for provisioning the warehouse service. Table 1 shows a set of SLA parameters that can be used to uphold the quality of service. For information services, *reliability* should indicate some measure of confidence for the information provided by the service. Separate guidelines are provided with the service description to state the reliability measures. For increased reliability more analysis may be required and the response time may be higher. Other SLA parameters such as the *response time* and *availability* are the standard ones. Additional SLA parameters such as *number of users* and *queries*, and *domain coverage* (types of configurations) are specified to define the business model to compute price based on the above user selected options.

TABLE I
EXAMPLE SLA MODEL FOR THE WAREHOUSE-AS-A-SERVICE

SLA Parameters	**Business model**		
	Gold	**Silver**	**Bronze**
Reliability	95% (confidence)	90%	80%
Availability	99%	97%	95%
Response Time	3 min. (max)	1 min. (max)	15 sec. (max)
Domain coverage	DB2 on z/OS and UNIX	DB2 on z/OS only	No data sharing
Number of users	50	30	10
Number of queries/day	up to 20 parallel queries	up to 10	up to 10

V. WORK IN PROGRESS

We are currently working in parallel on the knowledge collector, the storage structures, and the CBS.

A. *The Knowledge Collector*

For the knowledge collector, we are working on the following aspects.

1) Extraction of Key Words: We are working on automated extraction of key words and their synonyms from the log files and the various DB schemas in order to be able to retrieve related values from the various distributed data sources in the system. For example, a job is referred to using 'Plan', 'Plan name', 'Job', and 'Jobname' in different messages and monitored data. We are applying frequent word mining techniques [22][17] combined with domain knowledge to semi-automatically:

- Build a vocabulary of key words
- Extract the corresponding values of the keys from the problem report and associated set of log messages
- Build a list of problem categories from the list of error IDs and mining of groups of words that form message templates

2) Extraction of Patterns of Associated Messages: Based on our study, for a number of problem cases a group of messages generally appear together. We are applying association rule mining techniques to the temporal data in the JES master log file to discover such patterns of co-occurring messages [23]. Most real error messages are infrequent, which pose additional challenges in mining error related information.

B. *Storage Structures for Knowledge and Data*

We are working on definition of schemas for the warehouse and the DSKB to store the data and knowledge.

C. *The CBS*

For the CBS, we are exploring different data structures for the case repository and the different case retrieval techniques [6][15]. We are also examining the use of data mining techniques for finding the set of parameters that can be used to define a case [13].

VI. CONCLUSIONS

DSS has evolved to DGSS to both provide guidance and extract knowledge from human experts [18]. The approach and techniques applied in designing a DGSS largely depends on the domain specific data and the type of decision support that needs to be provided. In this research we address the domain of administration of DB2 DBMSs on z/OS Mainframe systems. Administration of legacy mainframe DB2 systems is becoming increasingly more challenging due to the scarcity of expert DBAs, the steep learning curve, and the increased systems complexity to support parallel data transaction requests from highly distributed applications. Although many different tools exist for log analysis, generating problem alerts, querying system data and executing DB commands, none of the tools and the DSSs in this domain provide strategic decision support and guidance while extracting problem features [3][5][14].

In this paper, we present our study of the troubleshooting process of a z/OS Mainframe DB2 DBMS in a large organization, which supports hundreds of transaction requests from many different distributed Web applications as well as legacy COBOL batch jobs. Based on the observations, we define a strategic problem solving life cycle that consists of four steps: Problem Notification, Identification, Investigation, and Resolution. For better understanding, the strategic problem solving life cycle is illustrated in the light of example problem scenarios. We, thereby, propose our hypotheses regarding the technologies and approaches needed for designing a training oriented adaptive decision guidance and support framework.

We propose our DSDAware framework that:

- Enables automated and semi-automated knowledge extraction both offline and during the problem solving process,
- Applies the system knowledge and case history of previous problem solving strategies to train the less experienced DBAs by guiding them in troubleshooting z/OS Mainframe DB2 problems, and
- Provides a warehouse-as-a-service interface for external users to access the knowledge and get decision support.

Our ongoing work focuses on the knowledge extraction and storage framework and the design of the CBS.

In a data rich society, we are now in grave need of learned and inferred knowledge that is conveniently accessible as required through platform independent Web-based service interfaces. Information and data are stored in a distributed manner in a variety of formats some of which may be accessible independently by many different tools. However, we get more information that often needs to be further searched, filtered and processed to suit specific needs. The DSDAware framework would support the next generation DGSS that connects multiple information sources with a goal of providing reliable knowledge through an interactive interface to the global Web users while training and providing strategic decision support to the internal users.

ACKNOWLEDGMENT

We want to express special thanks to Troy Coleman and Jeff Drake at CA Technologies for their feedback and support to conduct this research.

REFERENCES

[1] D., Arnott, and G. Pervan. A Critical Analysis of Decision Support Systems Research. *Journal of Information Technology*, vol. 20(2), pp. 67-87, 2005.

[2] P., Bruni, J., Iczkovits, H., Mynhardt; P., Zerbini. *DB2 9 for z/OS and Storage Management*, IBM Redbooks, 2010.

[3] CA Database Management Solutions for DB2 for z/OS. At: http://www.ca.com/files/learningpaths/db2_tools_learning_path_216908.pdf.

[4] A., Carneiro, R., Passos, R., Belian, T., Costa, P. Tedesco, and A., Salgado. DBSitter: An Intelligent Tool for Database Administration. *DEXA 2004, LNCS 3180*, F. Galindo et al. (Eds.), pp. 171-180, Springer-Verlag Berlin Heidelberg, 2004.

[5] Compuware ABEND-AID for Mainframe DB2. At http://www.compuware.com/mainframe-solutions/r/19752_AAforDB2_fs_c.pdf

[6] W. Dilla, and P. Steinbart. Using Information Display Characteristics to Provide Decision Guidance in a Choice Task under Conditions of Strict Uncertainty. *Journal of Information Systems*, vol.19 (2), pp. 29, 2005.

[7] S. Eom, S. Lee, E. Kim and C. Somarajan. A Survey of Decision Support System Applications (1988-1994). *Journal of the Operational Research Society*, vol. 49, pp. 109-120, Operational Research Society Ltd., 1998.

[8] L. Frenzel, and E. Turban. *Expert Systems and Applied Artificial Intelligence*, Prentice Hall Professional Technical Reference, 1992.

[9] H. Guo, X. Zhou, and Y. Zhu. Knowledge Acquisition based on Rough Set and Data Mining. *Int. Conf. on Future BioMedical Information Engineering (FBIE'09)*, pp. 126-128, IEEE, 2009.

[10] J., Hellerstein, S., Ma, and C., Perng. Discovering Actionable Patterns in Event Data. *IBM System Journal*, vol. 41(3), 2002.

[11] G., Henderson. z/OS Performance Monitoring Tools Shoot-Out: ASG, BMC, CA, Rocket, ASG, whitepaper, 2011.

[12] J., Horak, and J., Owsinski. Transcat Project and Prototype of DSS. In proc. of *10th EC GI & GIS Workshop on ESDI: State of the Art*, Warsaw, Poland, 2004.

[13] C., Hsu, and Y., Huang. Case Mining from Raw Data for Case Library Construction. In proc. of *9th Joint Conference on Information Sciences (JCIS*), 2006.

[14] IBM, DB2 Log Analysis Tool for z/OS. At http://www-01.ibm.com/software/data/db2imstools/db2tools/db2lat/.

[15] R., Mantaras, D., Mcsherry, D., Bridge, D., Leake, B., Smyth, S., Craw, B., Faltings, M., Maher, M., Cox, K., Forbus, M., Keane, A., Aamodt and I., Watson. Retrieval, Reuse, Revision and Retention in Case-Based Reasoning. In proc. of *The Knowledge Engineering Review*, vol. 20(3), pp.215–240, Cambridge University Press, 2006.

[16] M., Musen. Modern Architectures for Intelligent Systems: Reusable Ontologies and Problem-Solving Methods. In *AMIA Annual Symposium*, (Ed.) C.G. Chute, pp.46-52, 1998.

[17] W., Peng, T., Li, and S., Ma. Mining Logs Files for Data-Driven System Management, *ACM SIGKDD Explorations Newsletter - Natural language processing and text mining*, vol. 7(1), ACM New York, NY, USA, 2005.

[18] D., Power. Decision Support Systems: From the Past to the Future. In proc. of the *Americas Conference on Information Systems*, pp. 2025-2031, New York, NY, 2004.

[19] M., Romano, and R., Stafford. Electronic Health Records and Clinical Decision Support Systems: Impact on National Ambulatory Care Quality. *Archives of Internal Medicine*, vol.171(10), pp.897-903, 2011.

[20] L., Tai, W., Yu, and X., Feng, 2000. DSS and Business Strategic Decision Making. In proc. of *the 3rd World Congress on Intelligent Control and Automation*, Hefei , China, vol.3, pp. 1962 - 1965, 2000.

[21] E., Turban. Implementing Decision Support Systems: a Survey. In proc. of *IEEE Int. Conf. on Systems, Man, and Cybernetics*, vol.4, pp. 2540–2545, Beijing, 1996.

[22] W., Xu, L., Huang, A., Fox, D., Patterson, and M., Jordan. Detecting Large-Scale System Problems by Mining Console Logs. In proc. of *SOSP'09,* Big Sky, Montana, USA, ACM, 2009.

[23] S., Zhang and X., Wu. Fundamentals of Association Rules in Data Mining and Knowledge Discovery, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1(2), John Wiley & Sons, 2011.