An Architecture for Multi-Dimensional Temporal Abstraction Supporting Decision Making in Oil-Well Drilling

Odd Erik Gundersen and Frode Sørmo

Abstract We have developed an online decision support system that advice drilling engineers online based on data streams of real-time rig site measurements. This is achieved by combining multi-dimensional abstraction for recognizing symptoms and case based reasoning. Case-based reasoning compares the current situation in the well with past situations stored in the case base that contains advices for how to solve similar problems. The architecture is described in detail, and an example is presented in depth as well as results of commercial deployment.

Keywords case-based reasoning · real-data stream · oil-will drilling · integrated decision support

1 Introduction

Real-time data stream is available in a multitude of domains. Typical examples include health care in which patients are monitored continuously, stock market trading in which stock prices are recorded over time and aviation control. In this paper we will focus on the oil well drilling domain. Currently, over 3,800 oil-well drilling rigs are operating world-wide [11], and real-time data is collected from around 75 % of these according to domain experts. Examples of parameters that are measured in real-time are stand-pipe pressure, total amount of mud pumped through the drill-string, drill-string rotations per minute and the torque used by the motor to

O. E. Gundersen (✉) · F. Sørmo
Verdande Technology AS, 7041 Trondheim, Norway
e-mail: odderik@verdandetechnology.com
URL: http://www.verdandetechnology.com

O. E. Gundersen
Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway

rotate the drill string. Although a vast amount of data is produced, there is still great unrealized potential in its use. Today, the data is primarily used for visual display, statistical analysis of performance and threshold alarming, and there exists a few systems that update physical models in real-time to get a better overview of the drilling environment [21], but these are not in wide-spread use. Monitoring is usually done manually by drilling engineers on the rig site, or in real-time operation centers that gather data from multiple rigs to a central location [5]. Typically, drilling engineers interpret these data continuously in shifts lasting up to 12 h by monitoring data graphs. Identifying situational events like symptoms of developing problems by manually inspecting data graphs is error prone, as it is a tiring task and it is hard to keep all previously identified symptoms in mind at all times.

Problematic oil-well drilling situations typically develop over time with symptoms that are recognizable by drilling experts. However, the symptoms behaves differently based on the context of the operation, like for instance with depth or the drilling tools used. Thus, threshold alarms, which are known to produce many false positive alarms, do not work well as users tend to ignore them if they go off often without any reason. Automatic analysis of the data from operations support drilling engineers by allowing them to shift the cognitive task from identifying situational elements to predicting the future status of the drilling situation and thus increasing the situational awareness helping drilling engineers to making better decisions [9].

Our main goal is to develop an online decision support system that provides relevant information to drilling engineers during oil well drilling operations using data streams of real-time rig site measurements and case-based reasoning. Case-based reasoning (CBR) is a methodology for reusing past experiences by comparing them to a current problem and then solves the current problem using the past experiences [2]. Experiences are stored in a case base as cases comprised of a problem description and a problem solution. A new experience is compared to the past cases in the case base. Then, the problem solutions of the most similar past cases are used to solve new experience. If no applicable cases are contained in the case base, the new experience can be retained along with the proper solution, which allows the system to learn by gradually building the case base.

In this paper we present an architecture that supports the main goal. The main component in the architecture is an agent system that coordinates a set of agents in which each agent has a specific task related to enhance the understanding of the current situation. Tasks include recognizing different aspects of the current operation, such as which activity is performed and symptoms of problems, in addition to projecting the future based on the recognized symptoms. In order to recognize different aspects of the operation multi-dimensional temporal abstraction is performed while CBR is utilized to project the future state of the operation and provide support in selecting the most appropriate remedial actions. Temporal abstraction is a term used when transferring data stream over time from a low level quantitative form to a high level qualitative form incorporating domain knowledge and context information [20], and hence avoiding simplistic threshold alarms that do not take sound reasons for parameters to change into account.

The architecture is implemented in DrillEdge, a commercial decision support system that is used by several large oil companies. In DrillEdge, different aspects of the current situation of the oil well drilling operation are visualized to alleviate the cognitive load of the drilling engineers monitoring the situation. Past problematic oil-well drilling situations and experiences related to how the drilling problem could be avoided are captured as cases, which are brought to the attention of the user when the current situation becomes similar to a past situation.

The rest of this paper is structured as follows. Some related work will be investigated in Sect. 2. The architecture of the system will be presented in depth in Sect. 3. In Sect. 5 selected results are presented, and finally in Sect. 6 we will conclude with some final remarks and future work.

2 Related Work

Cately et al. discuss trends and challenges of multi-dimensional temporal abstractions and data mining of medical time series in [6]. While they restrict their discussion to medical time series, several of the trends apply to the domain of oil well drilling as well. They identify six trends. Firstly, they note that data analysis methods must support multi-dimensional, high frequency real-time data. Multi-dimensional correlations between parameters exist in the oil well drilling domain, but while down hole tools record high frequency data, these are not communicated to the surface in real-time and can therefore not be used when supporting decisions made by humans. Another trend is that medical data mining systems and temporal abstractions should be applied to real-world clinical data. This clearly applies to real-world oil well drilling data as it often contains faulty readings and not only some occasional data points, but for long periods of time as the measuring tools are in though conditions. Furthermore, they require null-hypothesis testing of current frameworks. This also applies for oil well drilling data, but as far as we know, no work has been done in relation to this. Then the authors argue that data mining methods should be used to explore complex patterns that might hide parameter correlations from the users, which is true in the oil domain too. Some work has been done, but much is left to do, as oil well drilling companies are secretive about their data. Data mining algorithms will push the knowledge bases towards synthesized knowledge bases, that is knowledge derived through a data mining processes and not domain experts. This will apply to the oil well drilling domain when data mining methods get more widespread, but currently most knowledge is derived from domain experts. Finally, they identify a need for automatically transferring knowledge between data mining and temporal abstraction mechanisms, which is what we seek to achieve with the combination of pattern matching and CBR.

In [19], Stacey et al. presents an architecture for multi-dimensional temporal abstraction and its application to support neonatal intensive care. The goal was to go beyond threshold alarms causing many false positive alarms by introducing domain knowledge in form of an ontology and rules. The architecture was based on previously performed research in business event monitoring and performance measurement [13]. However, while the previous work focused on correlating workflow events, the neonatal intensive care research performs temporal abstraction on low level quantitative data from patient data streams logging parameters like blood pressure and oxygen saturation.

The temporal abstraction is performed by Patient Agents that execute rules on the patient data streams and fire alerts (events) if the conditions of the rules apply. Rules can be made manually by domain experts like doctors or mined automatically by an Analytical Processor. Such rules are supported as Smart Alarms in DrillEdge. However, because of the complexity of symptoms, rules will typically generate too many false alarms and thus we use Graph and Symptom Agents that are based on heuristic mathematical models designed by pattern matching experts in cooperation with oil-well drilling experts.

The alerts that are fired if a rule applies are received by an active ontology which triggers an alarm to the users of the system. The ontology is a central knowledge base which stores the rules and agent responses and allows temporal abstraction across multiple patient's data. A similar learning of patterns across operations is found in DrillEdge by introducing new cases representing new situations and experiences in the case base. While the active ontology performs this cross-correlating automatically, DrillEdge requires an intervention by the user to identify when to capture a case. The roots of the ontological reasoning in DrillEdge is from the Creek [1], a knowledge intensive CBR system in which ontologies are an integral part. Compared to Creek, the ontological reasoning mechanisms have been simplified in DrillEdge and the ontology is primarily used for quality assuring the terminology used by the system.

Montani et al. present an architecture in which a CBR system configures and processes temporal abstractions produced from raw time-series data in [14, 17]. The problem description part of cases capture the context knowledge about the time series interpretation while the problem solution part configure temporal abstraction trends and states that are used to monitor the time-series data. So, the CBR system configures the temporal abstractions using context information in order to monitor and evaluate patients undergoing hemodialysis to provide medical personnel with an assessment of the situation. In comparison, DrillEdge provide drilling engineers with an assessment of the current situation of an oil-well drilling operation by relating the temporal abstractions (events) with the context.

Another domain with large amounts of time-series data is in the financial market. Barbosa and Belo present an architecture for currency trading in [4], which will forecast the direction of the currency value the next six hours, decide how much to trade for and whether to trade. The architecture has three parts: An agent system which forecasts the direction of several currencies, a CBR system that decides how much to trade for based on previous experiences and a rule-based system that decides whether to trade or not. Each agent in the agent-based system forecasts the value of one currency by using an ensemble of data mining algorithms. The data mining algorithms perform the temporal abstractions which is the input to the CBR system. Then based on how coherent the data mining algorithms was in the decision that the agent made, the CBR system decides how much to trade. Feedback on the accuracy of the forecasting is used provided to the CBR system to learn whether to trust the decision in similar situations later. Thus, the system is autonomous and can learn from its own experiences in contrast to DrillEdge which requires manual intervention to capture a new case. The architecture was later used to implement an agent task force for stock trading [3].

3 The Architecture

The architecture is designed with online decision support and modularity in mind. From a process view, the architecture has four layers, which are: (1) *Data acquisition*, (2) *Data interpretation*, (3) *Decision support* and (4) *Visualization*. These four layers are implemented by (1) *Input Data Sources*, (2) *Data Analysis Agents*, (3) *Case-Based Reasoning* and (4) *Symptoms and Case Radar* GUI elements, as illustrated in part *a* of Fig. 1. In a decision support system, the reasons for providing a recommendation must be transparent for a user to trust it, in particular when the users are experts in their field. Case-based reasoning systems provide such insight because it forms its advice based on a small (typically 1–5) set of similar cases, which include context and descritpion of the previous situation in a form natural to an expert. This allows the expert to verify that these cases are indeed similar to the current situation, and use the experience from them. The system is modular in that the system can grow with both symptoms and experiences without having to change what is already learned by the system. New agents recognizing new symptoms can be easily added and so can new cases.

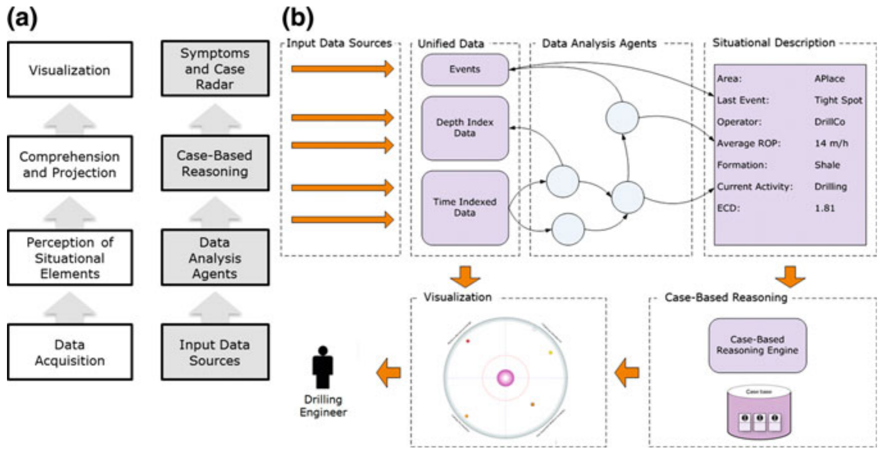

Fig. 1 **a** Layers in the architecture to the *left* and their implementations on the *right*. **b** Data flow oriented representation of the DrillEdge architecture

3.1 Data Flow

Part *b* of Fig. 1 shows how data flows through the system to the user. The data is gathered mainly from *Input Data Sources* such as streaming XML sources that use the oil well drilling industry standard called WITSML and the operators setting up and configuring operations that are to be monitored by DrillEdge. The different types of data that is stored in the system as *Unified Data* are time indexed data, depth indexed data and events, which are the multi-dimensional temporal abstraction that are computed by the *Data Analysis Agents*. In addition, the system stores contextual information describing the drilling operation. A structured *Situational Description* that describes the current state in the oil well is generated and compared by the *Case-Based Reasoning* engine to past cases with situational descriptions in the same structured format stored in the case base. The most similar cases are *Visualized* to the Drilling Engineer on the Case Radar as color coded dots according to their severity.

3.2 Agent Platform

The agent platform serves as an interface between the data and the agents in addition to control the execution of the agents. Agents depend both on the data streams and each other, and they are executed at every time step which is defined by the arrival of new data. Agents are executed in such a way that dependencies are maintained—i.e. an agent that depends on the output of another agent is always executed after the agent it depends on.

This agent architecture differs from the typical agent architecture found in the agent literature [15]. First, they do not communicate through a meta-level agent communication language like FIPA ACL [16] or KQML [10], but through a common data structure. Second, agent coordination is performed through a pre-defined partial ordering of dependencies based on requirements specified when coding the agents, and not through a multi-agent coordination strategy like negotiation [12]. Thirdly, the agents are not autonomous in the way that they run in separate threads or machines, but have their execution controlled by the agent platform. However, the agents are autonomous in that they control themselves when to generate a new symptom; they do not necessarily produce new values each time step.

Hence the architecture is similar to a blackboard system [7] in which the agents act as *Knowledge Sources* that communicate indirectly and anonymously through the unified data that act as the *Blackboard*. The agent platform resembles the *Control Component* that controls which agents to execute. This is illustrated in Fig. 2. Such an interpretation of an agent system is not uncommon, as is shown in Sect. 2 Related Work.

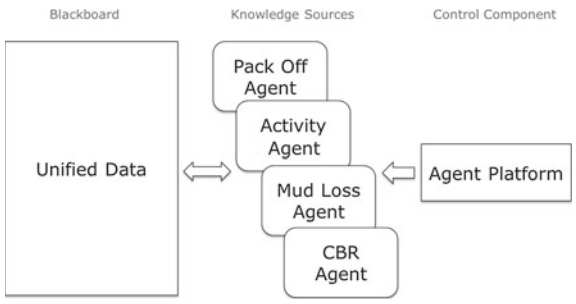

Fig. 2 The architecture can be interpreted as a blackboard system where the agents act as knowledge sources, the unified data can be thought of as the blackboard and the agent platform acts as the control component, controlling execution of the agents

We distinguish between three different types of data analysis agents in Drill-Edge -*Graph agents*, *Symptom agents* and the *CBR agent*. All types of agents can use all data available to the system to perform their tasks. Some agents only use parameter values stored in the time or depth indexes while others use context information too. The agent types and an example of dependence relations are illustrated in Fig. 3.

Graph Agents: In general, graph agents produce one or more new data value for every time step and store these in dedicated columns in the depth and/or time indexes. Graph agents are not necessarily dependent on any other agents, but some are. Mainly they produce values that are required by the other agents.

Symptom Agents: Symptom agents perform the multi-dimensional temporal abstraction by monitoring a set of predefined parameters and generate events if symptoms of problems are detected. The multi-dimensional temporal abstraction is implemented as fuzzy rules. An example of a symptom is when gravel starts packing off around the drill-bit. In the worst case, this can lead to the drill bit getting stuck in the ground. The symptom agent recognizing such pack-off tendencies perform multi dimensional temporal abstraction on several parameters, such as the amount of mud pumped in to the drill-string, standpipe pressure, torque and the weight on the drill-bit. In addition, it relies on context parameters like the dimension of the wellbore, casing depth and whether a mud motor is used when drilling. It is also dependent on the output from two graph agents, one that

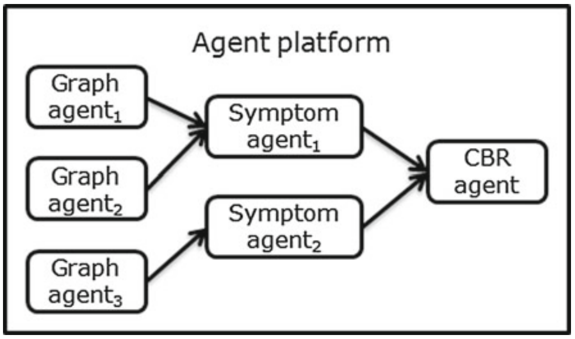

Fig. 3 The agent platform showing the three different agent types: graph agents, symptom agents and the CBR agent

identifies whether the drill-bit is in the open hole or in the casing and the agent that recognizes which drilling activity is performed.

When a symptom is recognized by a symptom agent, an event is produced, containing the time and depth where it occured, along with information about what type of event was observed and the severity of the event. Events serve as input to the CBR agent, but they are also visualized in the depth and time user interface view so that the end user can inspect them together with the parameters used to generate them. This is done to enhance the situation awareness of the drilling engineers monitoring the oil-well drilling operation and provides user transparency to the system.

CBR Agent: The CBR agent contains a CBR system and captures the current situation continuously and compares it to cases stored in the case base. The current situation is represented by a data structure containing both events and contextual information. Events are stored in the case as depth and time sequences sorted on the distance from the drill-bit and distance from the current time respectively. Problematic situations typically occur when symptoms cluster together—either on depth or in time or both. By storing the events both in time and depth sequences, the system can recognize when a case is similar along on or the other, using sequence similarity measures [18]. Context information describing the circumstances that the symptoms occurred in is also important. For instance, some problems only occur or is more likely to occur in particular geological formations.

3.3 Case-Based Reasoning Cycle

Figure 4 illustrates the CBR process in DrillEdge. Real-time data (1) is interpreted by graph and symptom agents and events are generated (2). Events together with other data are captured to form the input cases specifying the current situation (3). The case base is searched and past cases are retrieved from the case base (4). The past cases are plotted on the Case Radar to alert and advice the users (5). New cases can be captured and the retained case (6) is learned when stored in the case base.

Capturing and learning a new case is not done in real-time as quite a lot of experience is required to capture proper cases, and a new case should be tested on similar wells and the advice vetted. This means cases are typically captured after the operation is over, rather than during the monitoring.

3.4 Visualization

The results of the analysis are visualized in different ways, and the two most important ways are the overview screen and the time view. Users can verify or

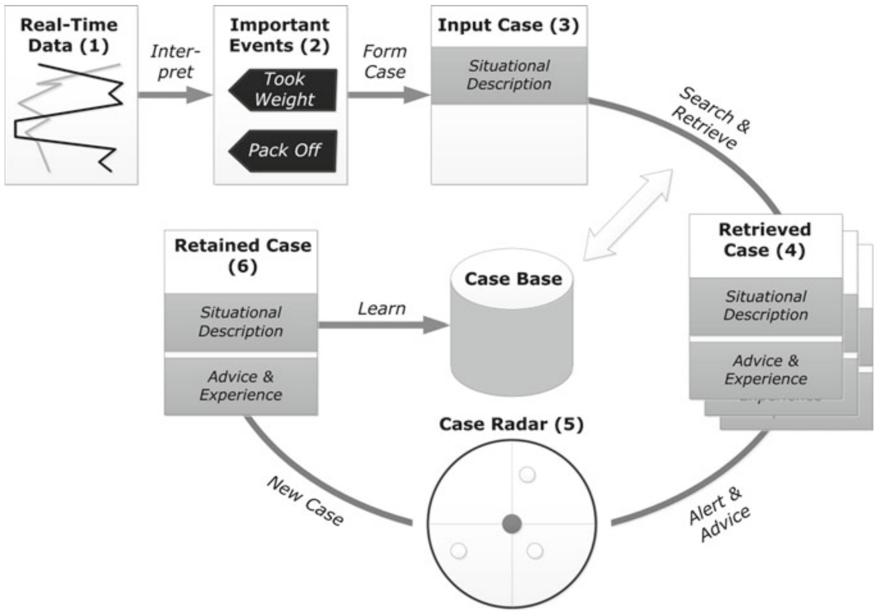

Fig. 4 Case-based reasoning cycle: from real-time data to a retained case

falsify the correctness of events by checking the parameter values around the time where the events are fired, and this is the main use for the time view. An screen capture of the time view is shown in Fig. 5.

Figure 6 shows a screen capture from the DrillEdge Client that visualizes the current state of an oil well drilling operation. The depth column with the drill-bit at the bottom of the wellbore and pack off events distributed along the drill string is

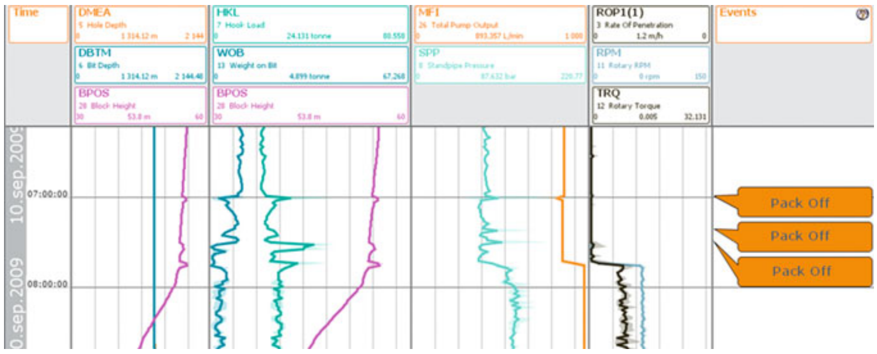

Fig. 5 Screen capture from the DrillEdge client showing the time view with parameters and events distributed over time

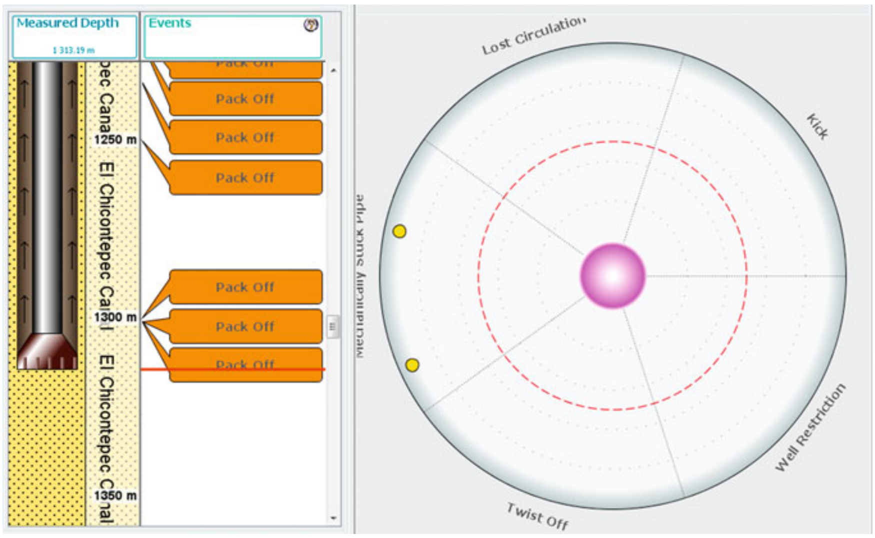

Fig. 6 Screen capture from the DrillEdge client showing the overview screen with the depth column containing events distributed according to depth and the Case Radar. Sections in the Case Radar specify problem type and case color indicate severity

showed to the left, and the Case Radar is depicted with two cases indicating a stuck pipe situation to the right.

The CBR agent searches for and retrieves cases from the case base and compares them to the current case. The comparison results in a similarity score for each case in the case base, describing how similar it is to the current case. All past cases with a degree of similarity above a given threshold are visualized on a GUI element, the Case Radar, to alert and advice the user of past historic cases that are similar to the current situation. The sectors in the Case Radar indicate different problem types that are covered by the cases in the case base. The closer a case is to the center of the Case Radar, the more similar the current situation is to the past situation represented by the case. Currently, the threshold for the Case Radar is 50 % similarity, so all cases on the Case Radar are more than fifty percent similar to the current situation. So, if a case enters the Case Radar, the current situation is getting similar to the situation captured in the past case. Thus it can be inferred that the current situation develops into a similar problematic situation and mitigating actions must be taken. By investigating the similar situations the user is advised on what others did and what should have been done in similar situations in the past. A new case is made by the drilling engineer if the current situation is not covered by any cases stored in the case base or if different or more specific advice apply to this situation. The system learns when new cases are added to the case base. The result of the case comparison is written by a graph agent to the time table so that the users can monitor the behavior of cases over time in the time view.

3.5 Client Server Architecture

The DrillEdge server is a distributed system that can be deployed on virtual server parks like the Amazon Elastic Computing Cloud (ECC) or VMWare clusters as well as physical servers. The client is a Java application that is installed from a web page using Java Web Start. All data interpretation is done on the server cluster, so the main task of the client is to visualize the information analyzed by the server.

The server is implemented as a set of services, and some services, like the license and management services, provide administrative functions and only one of each of these are required in the cluster. The license service ensures that the cluster runs with a valid license while the management service enables power users to set up and configure operations and administrators to administrate users. The operation service analyses oil-well drilling operations and communicates with DrillEdge clients through a front end. Each oil-well drilling operation monitored by the system has its own dedicated operation service.

Figure 7 illustrates the client server architecture. The box stippled with gray lines contain a server cluster deployed on the Amazon ECC while the gray boxes illustrate physical or virtual machines that run services. Hence, the server cluster is comprised of four machines in which two of them are running four operations, one machine is running the management and license server while the last one runs the front end service that communicates with the clients through HTTPS.

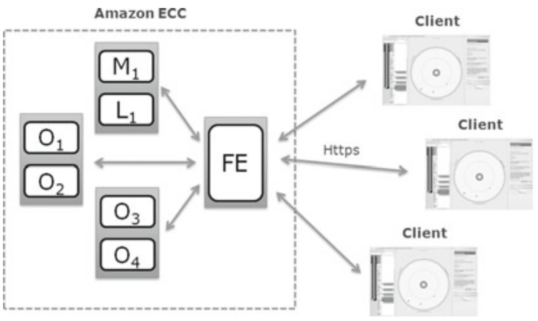

Fig. 7 The client server architecture: *O* indicates a operation service while *L* and *M* refers to license and management services respectively. *FE* is the front end and the subscripts indicate different services. The *gray*, stippled line confines the server cluster and the *gray boxes* indicate physical machines on the Amazon ECC

3.6 Operation Service

The main task of an operation service is to analyze the data streams from one concrete oil well drilling rig that has been assigned to the service. These data streams are measured on the rig, then aggregated and made available through a web service-based industry standard called WITSML [8]. The operation service is responsible for the data acquisition, and it has to communicate the results to the clients that users monitor the operations through. The agent platform is the most important part of the operation service.

Operation services are set up and configured by operators, which typically are drilling engineers responsible of ensuring that the service is set up correctly. The time streams update frequency ranges from a measurement per second to once per minute. However, the lower the data rate, the less detailed patterns can be recognized. DrillEdge achieves best results when the range between data points is shorter than ten seconds, as internal tests have shown that the results degrade significantly with lower frequencies. Depth data is measured by down hole tools, which collect measurements at high frequencies that are sampled and communicated to the surface at a rate of up to 40 bits per second. In addition to the streamed measurement data from the rig, the system requires context information that can be added to the operation service by operators. Context information includes static parameters like rig type and the configuration of the drilling tools, and the expected geology that is to be drilled through. When setting up the operation, the case base will have to be configured by the operators too.

4 Architecture and Decision Support Example

In this section, we will present an example showing how the architecture supports giving complex advices. Only three agents are required to illustrate how advices can be given for two completely different situations. The two different situations will be investigated in detail.

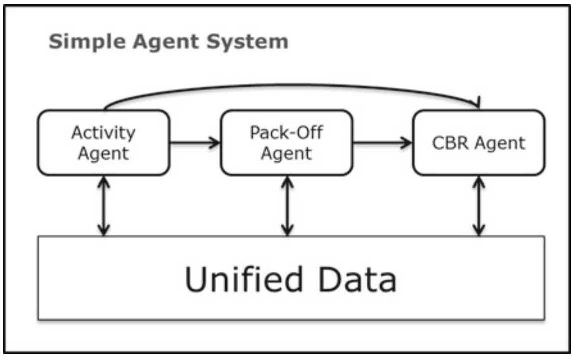

Fig. 8 The figure shows a simple agent system with one graph agent, the *Activity Agent*, one symptom agent, the *Pack Off Agent*, and the *CBR Agent*, which compares cases. Dependencies are illustrated by *arrows*. All the agents both read and write to the unified data

4.1 Agent System Configuration

The example system is illustrated in Fig. 8. It contains four components, and these are three agents that both read and write to the fourth component, unified data. Arrows indicate data flow, which also defines the dependencies. Three agents are implemented in the system: One graph agent, one symptom agent and the CBR agent. The four components of the system are described in detail below:

The Unified Data The unified data contains three types of data: Real-time parameters, symptoms, and context parameters. The following real-time parameters are used by the system:

- **Hole depth**: The depth of the hole that has been drilled. When this increase the hole is being drilled.
- **Bit depth**: Bit depth indicates where in the drilled hole the drill bit is located at a given point in time. If the bit depth is the same as the hole depth, the drill bit is at the bottom of the hole.
- **Rotation per minute**: Rotations per minute indicates whether the drillbit is rotating. If it is zero, then the drillbit is not rotating.
- **Mud Flow**: The amount of mud pumped into the well is indicated by mud flow.
- **Pressure**: Pressure indicates the pressure in the bottom of the well, and it depends on the amount of the mud pumped into the hole, which is measured by mud flow, and the length of the hole.

Only one symptom is recognized in the example system, which is pack off, and only one context parameter is provided, which is the formation type at a given depth.

The Activity Agent The Activity agent recognizes which activity that is performed on the rig. In this example, the activity code agent recognize two different types of activity. These are *drilling* and *tripping*. Drilling is when the drill bit is rotating at the bottom of the hole, while mud is flowing (to clean the cuttings of the drilling process out of the hole). Tripping is when the drill bit is moving into or out of the hole.

Algorithm 1 Activity Agent

```
$t \leftarrow getCurrentTime()$
$d \leftarrow getBitDepth(t)$
$d_1 \leftarrow getBitDepth(t-1)$
$h \leftarrow getHoleDepth(t)$
$m \leftarrow getMudFlow(t)$
$r \leftarrow getRotationPerMinute(t)$
if $d = h \wedge m > 0 \wedge r > 0$ then return isDrilling
end if
if $d \neq h \wedge d_1 - d \neq 0$ then return isTripping
end if
```

Algorithm 1 specify how drilling and tripping can be recognized. The current activity is recognized as drilling if the drill bit is at the bottom of the hole while mud is flowing and there is rotation. When the drill bit is not at the bottom of the hole, but is moving, the activity is recognized as tripping.

The Pack Off Agent Algorithm 2 lists a simplified pseudo code for the *PackOff* symptom agent, as presented in 3.2. The PackOff symptom agent generates a new pack off event at time t and depth d if the mudflow has been stable while the pressure has increased the last 60 seconds when drilling. The parameters $\boldsymbol{m}$ and $\boldsymbol{p}$ are vectors containing all values over the last 60 s of the mudflow and pressure parameters respectively. The methods *isStable*() and *isIncreasing*() can be implemented using regression analysis on moving windows of values that, for example, contains all values of a parameter over the last 60 s, such as $\boldsymbol{m}$ and $\boldsymbol{p}$.

Algorithm 2 PackOff Symptom Agent

```
$t \leftarrow getCurrentTime()$
$d \leftarrow getBitDepth(t)$
$t_0 \leftarrow t - 60s$
$\boldsymbol{m} \leftarrow mudflow(t_0, t)$
$\boldsymbol{p} \leftarrow pressure(t_0, t)$
if $isDrilling \wedge isStable(\boldsymbol{m}) \wedge isIncreasing(\boldsymbol{p})$ then
    return PackOff(t,d)
end if
```

The CBR Agent Data that is read by the CBR agent from the unified data are symptoms, activity and formation, which is a type of context information, while it writes the computed similarity for each of the cases in the case base. The CBR agent have a case base of two cases, which are illustrated in Fig. 9. *Case 1: Mechanically Stuck Pipe* captures a past situation in which several pack-offs were experienced and the outcome was that the drill string got stuck in the hole. The case description, which is used by the CBR agent to compare cases, contains a sequence of pack off events distributed over time and the activity, which was drilling at the time of the situation. When similar situations are experienced, the outcome might be the same. The action recommended by the experts capturing this case was to circulate—pump mud without drilling, to clean the cuttings out of the hole. The results of the case matching are written to the unified data in order to store the case matching history for later use.

Case 2: Severe Overpull describes a situation in which the drill bit was tripping into a section of the well that were not cleaned well enough when drilling. The case description contains the events distributed over depth close to the drill bit as well as the activity, which was tripping, and the formation, which was shale. When similar situations are experienced, tripping slowly is advice in order to avoid a severe overpull.

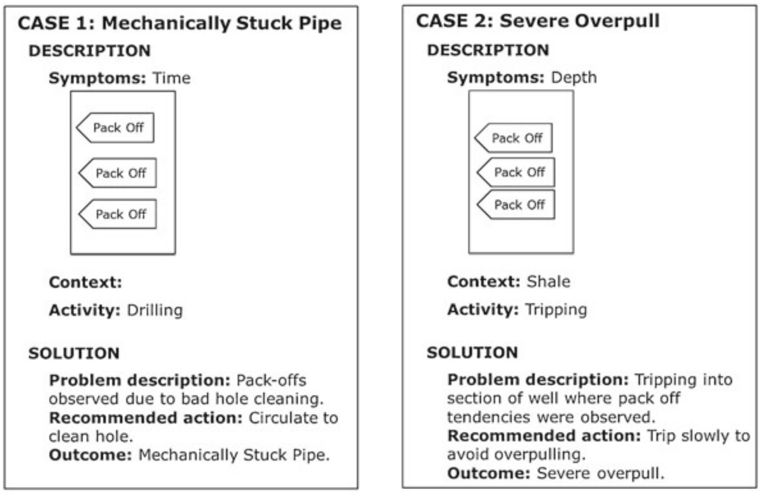
CASE 1: Mechanically Stuck Pipe
DESCRIPTION
Symptoms: Time
Pack Off
Pack Off
Pack Off
Context:
Activity: Drilling
SOLUTION
Problem description: Pack-offs observed due to bad hole cleaning.
Recommended action: Circulate to clean hole.
Outcome: Mechanically Stuck Pipe.
CASE 2: Severe Overpull
DESCRIPTION
Symptoms: Depth
Pack Off
Pack Off
Pack Off
Context: Shale
Activity: Tripping
SOLUTION
Problem description: Tripping into section of well where pack off tendencies were observed.
Recommended action: Trip slowly to avoid overpulling.
Outcome: Severe overpull.

Fig. 9 *Case 1* describes a past situation in which the pipe got mechanically stuck while drilling due to cuttings packing around the drill bit, while *Case 2* describes a past situation in which a severe overpull was experienced when the drill bit was tripping through a section of the well where pack offs were observed when drilling. An overpull is a situation where more force than expected is required to pull the pipe out of whole, which can occur for instance if the drill bit get stuck on a ledge

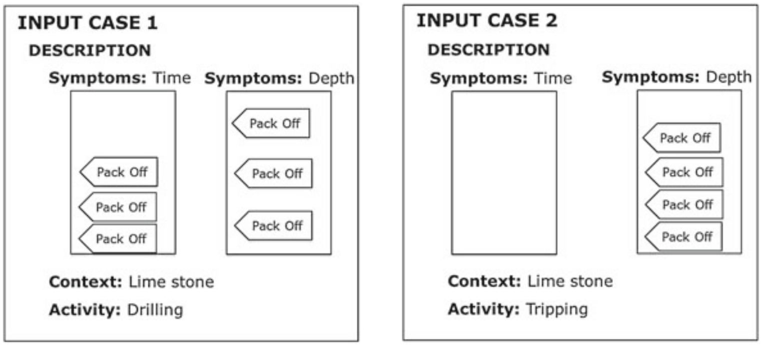

Fig. 10 *Input Case 1* describes a real-time situation in which pack offs are observed in lime stone while drilling, while *Input Case 2* describes another real-time situation in which the rig is tripping through a section of the well where pack offs were observed when drilling

4.2 Decision Support for Two Different Situations

The two input cases, which describe the two different situations that will be discussed in this section, are illustrated in Fig. 10. Each of the two situations will be analyzed by discussing how the two stored cases, which are retained in the case base of the CBR agent that was presented in 4.1, match in each situation. First, we will present an example of how a pack off symptom is recognized by the algorithm presented above.

Pack Off Recognition Fig. 11 shows some important parameter values around the time a pack off is recognized, so this is just a description of a generic pack off event. Time increases downwards on the y-axis while parameter values are plotted in a range on the x-axis. Pressure (SPP) can be seen increasing while the mud flow (MFI) is stable. This is right before the time a pack off is indicated by the *PackOff* agent. Hole depth (DMEA) is equal to the bit depth (DBTM), and thus the drill bit is at the bottom of the well. As can be seen in the graphs, the pressure decreases after the mud flow is decreased. The driller at the rig, which controls the mud flow, also recognizes the pack off and reduces the mud flow. Because of this reduction in the mud flow, the pressure decreases. The driller increases mud flow slowly and the drilling continues without any more pack off symptoms.

Situation 1: Pack Offs Experienced while Drilling *Input Case 1* describes a situation in which pack offs are experienced when drilling in lime stone. The pack off symptoms are close both in time and in depth in this situation, and there are three of them in the time sequence and three in the depth sequence, which is the same as for both of the stored cases illustrated in Fig. 9. The time sequence of *Case 1: Mechanically Stuck Pipe* is thus similar although not equal, as the distribution is somewhat different, and the similarity is computed to 85 %. The context and depth sequence are not specified in *Case 1*, so they are ignored, while the activity is drilling for both the input case and the stored case. Given that the total similarity is computed as the average of the features, the total similarity score is 92.5 % when comparing *Case 1: Mechanically Stuck Pipe* and *Input Case 1*. *Case 2: Severe Overpull* are dissimilar for both the context information and the activity, so both of these have a local similarity score of 0 %. The time sequence is not specified, so it is ignored. The depth sequence is quite similar, as they both contains three pack offs with quite similar distribution, and the system calculates

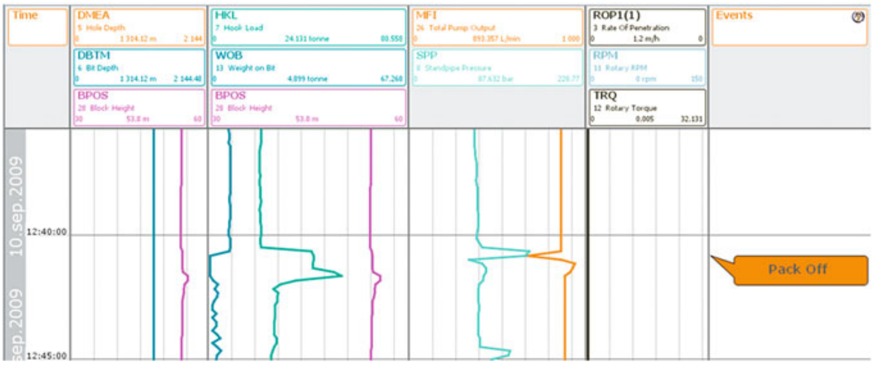

Fig. 11 A pack off event is recognized and shown in the time view when the pressure (SPP) increases while the mud flow (MFI) is stable

that there is a 90 % similarity. Even though the depth sequence is very similar, the total similarity score is just 30 %.

Therefore, in this situation the recommended action is to circulate to clean the hole in order to avoid getting mechanically stuck, as this is the recommendation from the most similar case, which is *Case 1: Mechanically Stuck Pipe*.

Situation 2: Tripping into a Section of the Well where Pack Offs Have Been Experienced Sitation 2 is specified by *Input Case 2*, which describe a situation where they are tripping into a section of the well where pack offs were experienced when the section was drilled. In this situation, no pack offs have been experienced over a certain time period, so the time sequence is empty. However, four pack offs were observed when drilling this section of lime stone, so the depth sequence contains four pack offs.

Case 1: Mechanically Stuck Pipe has three events in the time sequence, but as this is empty for the input case, the similarity score is computed to 0 %. Not context is specified in the stored case, so this is ignored, and the activity is dissimilar, which results in a 0 % similarity. The total similarity score for *Case 1: Mechanically Stuck Pipe* is therefore 0 %.

Case 2: Severe Overpull on the other hand, is found quite similar even though the context is dissimilar and has a similarity score of 0 %. The time sequence is not specified in the stored case, so it is ignored in the matching, but the activity is the same, and thus have a similarity of 100 %. The depth sequence in the input case have four pack offs and the stored case have three, but the distribution is similar, so the similarity is computed to 72 %. Hence, the total similarity between *Input Case 2* and *Case 2: Severe Overpull* is 57.3 %. This means that the recommended action in the situation represented by *Input Case 2* is to trip slowly to avoid a severe overpull.

4.3 Discussion

The example shows the strength of the architecture even with a small set of agents. It shows that complex recommendations can be given based on a small set of input parameters, as the *Activity* and the *Pack Off* agents perform multi-dimensional temporal abstraction on these parameters. Their output is used by the CBR agent, which provides the overall comprehension and predicts the outcome of the situation. This division between recognizing symptoms and diagnosing the situation has severeal advantages.

First, as many problematic situations in oil well drilling develop over a long time period and also are related to depth, a time series analysis only will provide poor results. Furthermore, given the many parameters that influence real-world situations, the number of examples required to train a machine learning system is enormous. Combining this with the scarcity of problematic situations reveals the impracticality of following this track. Drilling engineers are used to analyze time series data manually, and can easily verify or falsify single events when looking at the parameters relevant for a given symptom. Hence, the system is not thought of as a black-box system that just provides a recommendation without justifying it. The users can also see which symptoms are found relevant when comparing the current situation with a past case, which give them the possibility to decide themselves whether they agree or not. This increases the confidence of the users and they will more easily trust the advices than if no justification was given. Also, the modularity of the system makes it easier to expand to new problem types, as it only requires new agents to be deployed and new cases in the case base, which means that it can learn not only new situations, but also new problem types.

5 Results

The described architecture has been deployed in a commercial production environment since 2011. We give a short description of the deployment and present a case story from the production environment, which illustrate how the system can support decisions and save millions of dollars for oil well drilling companies.

5.1 Deployment

DrillEdge monitored the first live well during the summer of 2008 as part of the conclusion of the research project it was developed as part of. During 2011 around 15 pilot projects were finished, partly by analyzing historical data and monitoring live wells. In December 2011, Petroleum Development Oman (PDO) started monitoring live wells using DrillEdge, and in early January 2012 DrillEdge was running on more than 30 commercial wells concurrently. The current maximum number of operations monitored concurrently is 46, and it was reached in mid October 2012.

5.2 Case Story

Shell has run several tests both on historical and live data and deployed DrillEdge commercially in mid 2011. Shell experienced problems of twisting off the drill-pipe while drilling, and requested a solution that could predict twist off problems in advance. Twisting off the drill-string is a costly problem that often requires side tracking in order to avoid the parts of the drill-string that is left in the hole. Verdande Technology analyzed the data and found that long periods of maxing out the torque while drilling wore out the drill-string so that it finally twisted off. A symptom agent was developed to recognize when the torque was maxed out, and several cases were captured from historical data.

The solution was tested in a blind test using test data from a US land well while five cases was build from Middle East data. A total of 31 Maxed Out Torque events were fired, which resulted in three of the five Middle East cases appeared on the Case Radar before the drill-string twisted off. The first case appeared on the Case Radar two days before the twist off, potentially giving Shell long time to react to the problem. Cases were even captured from the US land blind test data and used to analyze the Middle East twist offs. Similar results were achieved. Other twist off tests were performed with varying results, but overall the tests were good enough for Shell to deploy DrillEdge commercially. In addition to the twist off tests, the stuck pipe solution was also put under pressure, but predicted stuck pipe six hours in advance. A detailed summary of the tests can be found in [22]. DrillEdge has been documented to accelerate learning in Shell [23].

6 Conclusion and Future Work

We have presented an architecture for multi-temporal abstraction that supports decision making in oil well drilling. The architecture scales well and is shown to predict problematic situations hours in advance. For this, Verdande Technology was awarded the Meritous Award for Engineering Excellence for its DrillEdge software platform by E&P Magazine in March 2011.

We will research how data mining techniques can be applied to detect symptoms of problematic drilling situations so that we do not need to rely on manually designed heuristic mathematical models. In order to do this, we need to be able to track event accuracy without human interference. Currently, we are implementing a system for quality ensuring event recognition automatically. Also, we will investigate how to combine a rule-based system with the case-based reasoning system in order to let domain experts control the case matching to a higher degree. Another task is to move in a more knowledge intensive direction and use the capabilities of the ontology as a part of the reasoning process. Furthermore, we would like to investigate ways to automatically identify where to capture cases and thus avoiding time consuming manual work. Finally, we are investigating whether this architecture and approach for data analysis and decision support can be expanded to other domains such as financial services and health care.

References

1. Aamodt, A.: Knowledge-intensive case-based reasoning in CREEK. In: Funk, P., González-Calero, P. (eds.) ECCBR, Lecture Notes in Computer Science, vol. 3155, pp. 1–15. Springer (2004)
2. Aamodt, A., Plaza, P.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Commun. **7**(1), 39–59 (1994)

3. Barbosa, R., Belo, O.: An agent task force for stock trading. In: Demazeau, Y., Pechoucek, M., Corchado, J.M., Pérez, J.B. (eds.) PAAMS, Advances in Intelligent and Soft Computing, vol. 88, pp. 287–297. Springer (2011)
4. Barbosa, R.P., Belo, O.: Autonomous forex trading agents. In: Perner, P. (ed.) ICDM, Lecture Notes in Computer Science, vol. 5077, pp. 389–403. Springer (2008)
5. Booth, J.: Real-time drilling operations centers: a history of functionality and organizational purpose—the second generation. SPE Drill. Compl. **26**(2), 295–302 (2011)
6. Catley, C., Stratti, H., McGregor, C.: Multi-dimensional temporal abstraction and data mining of medical time series data: trends and challenges. In: Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE, pp. 4322–4325 (2008)
7. Corkill, D.D.: Collaborating software: blackboard and multi-agent systems and the future. In: Proceedings of the International Lisp Conference. New York (2003)
8. Deeks, N.T.H.: WITSML changing the face of real-time. In: Intelligent Energy Conference and Exhibition (2008)
9. Endsley, M.R.: Toward a theory of situation awareness in dynamic systems. Hum. Factors J. Hum. Factors Ergon. Soc. **37**, 32–64(33) (1995)
10. Finin, T., Fritzson, R., McKay, D., McEntire, R.: KQML as an agent communication language. In: Proceedings of the Third International Conference on Information and Knowledge Management. CIKM '94, pp. 456–463, ACM, New York, NY, USA (1994), http://doi.acm.org/10.1145/191246.191322
11. Hughes, B.: Baker Hughes investor relations: overview and FAQ. http://investor.shareholder.com/bhi/rig_counts/rc_index.cfm (2012)
12. Lesser, V.R.: Reflections on the nature of multi-agent coordination and its implications for an agent architecture. Auton. Agent. Multi-Agent Syst. **1**, 89–111 (1998)
13. McGregor, C., Schiefer, J.: A web-service based framework for analyzing and measuring business performance. Inf. Syst. E-Bus. Manage. **2**(1), 89–110 (2004)
14. Montani, S., Bottrighi, A., Leonardi, G., Portinale, L.: A CBR-based, closed-loop architecture for temporal abstractions configuration. Comput. Intell. **25**(3), 235–249 (2009)
15. Nwana, H.S.: Software agents: an overview. Knowl. Eng. Rev. **11**, 1–40 (1996)
16. O'Brien, P.D., Nicol, R.C.: FIPA—towards a standard for software agents. BT Technol. J. **16**, 51–59 (1998)
17. Portinale, L., Montani, S., Bottrighi, A., Leonardi, G., Juárez, J.M.: A case-based architecture for temporal abstraction configuration and processing. In: ICTAI IEEE Computer Society, pp. 667–676 (2006)
18. Richter, M.: Similarity. In: Perner, P. (ed.) Case-Based Reasoning on Images and Signals, Studies in Computational Intelligence, vol. 73, pp. 25–90. Springer (2008)
19. Stacey, M., McGregor, C., Tracy, M.: An architecture for multi-dimensional temporal abstraction and its application to support neonatal intensive care. In: Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, pp. 3752–3756 (2007)
20. Stein, A., Musen, M.A., Shahar, Y.: Knowledge acquisition for temporal abstraction. In: Proceedings of AMIA Annual Fall Symposium. CIKM '94, pp. 204–208. ACM, New York, NY, USA (1996)
21. Strøm, S., Balov, M.K., Kjørholt, H., Gaasø, R., Vefring, E., Rommetveit, R.: The future drilling scenario. In: Offshore Technology Conference. CIKM '94, pp. 204–208. ACM, New York, NY, USA (2008)
22. van Oort, E., Brady, K.: Case-based reasoning system predicts twist-off in Louisiana well based on mideast analog. World Oil (2011)
23. van Oort, E., Griffith, J., Shneider, B.: How to accelerate drilling learning curves. In: SPE/IADC Drilling Conference and Exhibition (2011)

Springer

http://www.springer.com/978-3-642-36650-5

Combinations of Intelligent Methods and Applications
Proceedings of the 3rd International Workshop, CIMA 2012, Montpellier, France, August 2012
Hatzilygeroudis, I.; Palade, V. (Eds.)
2013, X, 156 p., Hardcover
ISBN: 978-3-642-36650-5