An Approach to Digitizing and Managing Well-Logging Parameter Graphs with Agent-Based Perspective

Chunsheng Li, Chengqi Zhang
Faculty of Information Technology
University of Technology, Sydney
PO Box 123, BROADWAY
NSW 2007, AUSTRALIA
{csli, chengqi}@it.uts.edu.au

Mei Wang, Qingfeng Song
Department of Computer Engineering
Daqing Petroleum Institute
Daqing, Heilongjiang 163318
CHINA
wm_may@cuninet.com

Abstract

The curves on well-logging parameter graph are very important to reservoir description. But how to keep the parameter graphs permanently and use the data implied in those curves efficiently are still open problems in petroleum industry. In this paper, we contribute an approach to digitizing well log curves and storing the digitized information into Oracle database or data file server with multi-agent perspective. We employed Gaia methodology and open agent architecture to analyze and design the system. According to the characteristics of the well-logging parameter graphs, we implement the SCTR (Scanning, Compressing, Tracing, and Rectifying) algorithms with four agents to digitize well log curves. Two data management agents are developed for operating database and data files in uniform agent communication language. The experimental results show that this approach is effective.

1. Introduction

A large amount of legacy information, which was produced from any specialized oilfield, faces to satisfy the current computation environment. Before computers were widely applied in petroleum industry, the information, such as well log data, might be drawn on parameter graphs in curve format [2]. In current computation environment, a novel system to deal with the processing and management of legacy well-logging parameter graphs is required in petroleum industry. The user’s requirements include mainly four aspects. Firstly, the well log curves drawn on parameter graph must be accurately converted to data (The maximum absolute value of the digitizing differences compared with original parameter graph is less than 0.5mm). The digitized well log data are stored into Oracle database. Secondly, the well-logging parameter graphs must be regenerated according to the well log data. Thirdly, because the original parameter graphs were the first hand documents, they must be compressed and stored in computers forever. If necessary in the future, the original well-logging parameter graphs can be restored from those compressed image files. Lastly, all information must be integrated in a uniform data operation platform. The data sources may include diversified data files and heterogeneous databases.

Because agents are being advocated as a next generation model for engineering complex, distributed systems and will succeed as a mainstream software engineering paradigm [1], we propose to implement well log information digitizing and managing system with agent-based technology. In this paper, we employed some ideas of Gaia methodology [8] and Open Agent Architecture (OAA) [4] to contribute an agent-based approach to achieving the user’s requirements. The framework of the approach consists of three levels, namely, role model, agent type, and agent instance. In role model level, a model with four roles is described. In agent type level, four agent types are designed. In agent instance level, seven agent instances are implemented. Four agents out of seven carry out the SCTR (Scanning, Compressing, Tracing, and Rectifying) algorithms. We employed Line Adjacency Graph (LAG) data structure in implementing the compressing agent and tracing agent. LAG is not only efficient to compress image files, but also very convenient to trace the curves based on compressed image files [6]. According to the characteristics of the well-logging parameter graphs, the rectifying agent with rectifying algorithm based on centimeter grid of coordinates is proposed for getting the exact well log data. We employed middleware agent to
operate database in uniform database-based agent communication language (DBACL) [3]. From agent viewpoint, the middleware agent is a wrapper of the database, which makes the database exhibit the agent characteristics in agent-based environment.

The remaining sections of this paper are organized as follows. In Section 2, the related work about this research is described. Section 3 presents the model with four roles for managing well log information. Section 4 discusses four agent types for designing the role model. Section 5 describes architectures of key agent instances for implementing the framework. Section 6 gives the result of case study. Section 7 concludes the paper.

2. Literature review

Much of the foundational work on agent technology has focused on methodologies to build agent-oriented systems, inter-agent communication protocols, and agent-based application systems [7]. Gaia, which is specifically tailored to the analysis and design of agent-based systems, is one of the methodologies [8]. Gaia is intended to allow us to go systematically from a statement of requirements to a design that is sufficiently detailed that it can be implemented directly. In applying Gaia, we move from abstract to increasingly concrete concepts. The main Gaian concepts can be divided into two categories: abstract and concrete. Abstract entities are those used during analysis to conceptualize the system, but which do not necessarily have any direct realization within the system. Concrete entities, in contrast, are used within the design process, and will typically have direct counterparts in the run-time system. But Gaia, as it presently stands, is not a general methodology for all kinds of multi-agent systems. It is not suitable for modeling open systems and controlling the behavior of self-interested agents. The OAA made it possible for software services to be provided through the cooperative efforts of distributed collections of autonomous agents [4]. Communication and cooperation between agents were brokered by one or more facilitators, which were responsible for matching requests, from users and agents, with descriptions of capabilities of other agents.

Many agent-based software systems evolved some notion of our framework have been developed. Li, et al. contributed a framework and developed an agent-based middleware with legacy software wrapping view to operate heterogeneous databases in DBACL [3]. Zhang, et al. proposed a framework to timely gather sufficient and correct petroleum information from databases, data files, or websites for constructing the system of petroleum information services [10]. All above work, to certain extent, promoted our research.

3. Role model of the system

In accordance with the user’s requirements described in Section 1, the system consists of four components, namely, interaction with user, curve digitizing, parameter graph regeneration, and well log data management. Each component makes use of significant computational resources and is of independent feature. The organization structure of the system is static because inter-agent relationships do not change at run-time. So it is suitable for applying Gaia to analyze the system [8].

3.1. Analysis with the Gaia methodology

The objective of the analysis stage is to develop an understanding of the system and its structure. In this stage, the system consists of roles, which role is defined by four attributes: responsibilities, permissions, activities, and protocols. Responsibilities determine functionality and, as such, are perhaps the key attribute associated with a role. Responsibilities are divided into two types: liveness properties and safety properties. Liveness properties describe those states of affairs that an agent must bring about, given certain environmental conditions. In contrast, safety properties are invariants. An acceptable state of affairs is maintained across all states of execution. The atomic components of a liveness expression are either activities or protocols. An activity is somewhat like a method in object-oriented terms, or a procedure in a PASCAL-like language. It corresponds to a unit of action that the agent may perform, which does not involve interaction with any other agent. Protocols, on the other hand, are activities that do require interaction with other agents. Permissions are the “rights” associated with a role. The permissions of a role thus identify the resources that are available to that role in order to realize its responsibilities. The activities of a role are computations associated with the role that may be carried out by the agent without interacting with other agents. Protocols define the way that it can interact with other roles. The roles model identifies the key roles in the system. A role will have associated with it certain
permissions, relating to the type and the amount of resources that can be exploited when carrying out the role [8].

In the well log information digitizing and managing system, one-to-one correspondence between components and roles is adopted. The role model comprises four roles, namely, interaction with user (IWU), curve digitizing (CD), parameter graph regeneration (PGR), and data management (DM).

IWU role involves scanning parameter graph into computer as an image file, inputting user’s requests, displaying the results (text or image mode) to user, and contacting with other roles. The protocols and activities are as follows:

\{InputRequest, ScanParameterGraph, DisplayImage, SaveDataByDM, GetDataByDM, ContactWithCD, ContactWithPGR, DisplayStates\}.

Responsibilities:
IWU = (A1)* | (A2)* | (A3)* | (A4)* | (A5)*
A1 = (InputRequest)* | (ScanParameterGraph)*.
A2 = (InputRequest)* | (GetDataByDM)* | (DisplayText)*
A3 = (InputRequest)* | (GetDataFromDM) | (DisplayImage)*
A4 = (InputRequest)* | (ContactWithCD) | (DisplayStates)*
A5 = (InputRequest)* | (ContactWithPGR) | (DisplayStates)*

where: ‘.’ means follow by; ‘|’ means ‘OR’; ‘x*’ means x occurs 0 or more times; ‘x+’ means x occurs 1 or more times; ‘x?’ means x occurs infinitely. We write activity names in a sans serif font, and use a similar font, underlined, for protocol names. The following descriptions about role submit to this comment.

The task of CD role is to complete mainly the SCTR functions. The relationships between the roles are shown in Figure 1. Indeed, such interplay is central to the way in which the system functions. The relationships between the roles are shown in Figure 1.

3.2. Structure of the role model

There are inevitably dependencies and relationships between the four roles in the system. Indeed, such interplay is central to the way in which the system functions. The relationships between the roles are shown in Figure 1.

Figure 1. Structure of the system

{ContactWithIWU, GetDataByDM, ProcessQueryData, ProcessRegeneratingData, ExtractOriginalGraph, SaveResultByDM}.

Responsibilities:
PGR = (C1)* | (C2)* | (C3)*
C1 = (ContactWithIWU)* | (GetDataByDM)* | (ProcessQueryData) | (SaveResultByDM)
C2 = (ContactWithIWU)* | (GetDataByDM)* | (ProcessRegeneratingData) | (SaveResultByDM)
C3 = (ContactWithIWU)* | (GetDataByDM)* | (ExtractOriginalGraph) | (SaveResultByDM)

DM role involves operating database and data files that are stored in file server. Although this role manages all operations for every database used by the system, each database must be of its own agent instance. An agent instance is like a middleware between the specialized database and other agents that intend to operate the database. The protocols and activities are as follows:

{ContactWithIWU, ContactWithCD, ContactWithPGR, OperateFileServer, OperateDatabase}.

Responsibilities:
DM = (D1)* | (D2)* | (D3)* | (D4)* | (D5)*
D1 = (ContactWithIWU)* | (OperateFileServer)
D2 = (ContactWithCD)* | (OperateFileServer)
D3 = (ContactWithCD)* | (OperateDatabase)
D4 = (ContactWithPGR)* | (OperateFileServer)
D5 = (ContactWithPGR)* | (OperateDatabase)

\{ContactWithIWU, GetDataByDM, RectifyData, SaveDataByDM, CompressImage, TraceCurve\}.

Responsibilities:
CD = (B1)* | (B2)* | (B3)*
B1 = (ContactWithIWU)* | (GetDataByDM) | (CompressImage) | (SaveDataByDM)*
B2 = (ContactWithIWU)* | (GetDataByDM) | (TraceCurve) | (SaveDataByDM)*
B3 = (ContactWithIWU)* | (GetDataByDM) | (RectifyData) | (SaveDataByDM)*

PGR role involves organizing query data from database by DM role, regenerating well-logging parameter graph according to the well log data in database, and restoring the original parameter graphs by extracting information from compressed image files. The protocols and activities are as follows:

{ContactWithIWU, GetDataByDM, ProcessQueryData, ProcessRegeneratingData, ExtractOriginalGraph, SaveResultByDM}.

Responsibilities:
PGR = (C1)* | (C2)* | (C3)*
C1 = (ContactWithIWU)* | (GetDataByDM)* | (ProcessQueryData) | (SaveResultByDM)
C2 = (ContactWithIWU)* | (GetDataByDM)* | (ProcessRegeneratingData) | (SaveResultByDM)
C3 = (ContactWithIWU)* | (GetDataByDM)* | (ExtractOriginalGraph) | (SaveResultByDM)

DM role involves operating database and data files that are stored in file server. Although this role manages all operations for every database used by the system, each database must be of its own agent instance. An agent instance is like a middleware between the specialized database and other agents that intend to operate the database. The protocols and activities are as follows:

{ContactWithIWU, ContactWithCD, ContactWithPGR, OperateFileServer, OperateDatabase}.

Responsibilities:
DM = (D1)* | (D2)* | (D3)* | (D4)* | (D5)*
D1 = (ContactWithIWU)* | (OperateFileServer)
D2 = (ContactWithCD)* | (OperateFileServer)
D3 = (ContactWithCD)* | (OperateDatabase)
D4 = (ContactWithPGR)* | (OperateFileServer)
D5 = (ContactWithPGR)* | (OperateDatabase)
The dependencies and relationships between the roles are presented in the interaction model (I Model). ‘I Model’ consists of a set of protocol definitions, one for each type of inter-role interaction. Here a protocol can be viewed as an institutionalized pattern of interaction. That is, a pattern of interaction that has been formally defined and abstracted away from any particular sequence of execution steps. Viewing interactions in this way means that attention is focused on the essential nature and purpose of the interaction, rather than on the precise ordering of particular message exchanges. A protocol definition consists of initiator (the role responsible for starting the interaction), responder (the role with which the initiator interacts), inputs, outputs, and processing. In IWU, SaveDataByDM, GetDataByDM, ContactWithCD, and ContactWithPGR interact with DM role, CD role, and PGR role, respectively, through ‘I Model’, but InputRequest and ScanParameterGraph are not relevant to ‘I Model’ because they interact directly with user. The protocols between users and IWU consist of the GUI interfaces and constraints users must follow. In CD role, ContactWithIWU, GetDataByDM, and SaveDataByDM interact with IWU role and DM role respectively through ‘I Model’. In PGR, ContactWithIWU, GetDataByDM, and SaveResultByDM interact with IWU role and DM role respectively through ‘I Model’. In DM roles, the protocols and activities with underlines interact with IWU, CD or PGR respectively through ‘I Model’.

4. Framework of the agent types

The Gaia design process includes three models, namely, agent model, service model, and acquaintance model. The agent model identifies the agent types that will make up the system, and the agent instances that will be instantiated from these types. The service model identifies the main services that are required to realize the agent role. Finally, the acquaintance model documents the lines of communication between the different agents [8]. In accordance with the analysis about the system in Section 3, four agent types are suitable for expressing the four roles. They are IWU agent type, CD agent type, PGR agent type, and DM agent type as shown in Figure 2.

In the framework, DM agent type wraps the data sources as an agent-based platform. From OAA point of view [4], DM agent type, file server, and databases comprise facilitator agent. The facilitator is a specialized server agent that is responsible for coordinating agent communications and cooperative problem solving. In our system, the facilitator is used to provide a global data manipulation for its client agents, which allows them to adopt a blackboard style of interaction [4]. The interactions between the agent types follow the rules described in acquaintance model. Acquaintance model simply defines the communication links that exist between agent types. It does not define what messages are sent or when messages are sent (It simply indicates that communication pathways exist). The pathways described in acquaintance model are in accordance with protocols of SaveDataByDM, GetDataByDM, ContactWithCD, ContactWithPGR, ContactWithIWU, and SaveResultByDM. In particular, the purpose of an acquaintance model is to identify any potential communication bottlenecks, which may cause problems at run-time. We employed the uniform database-based agent communication language for implementing the communication of acquaintance model [3].

5. SCTR and agent instances

Each agent type includes one or more agent instances (Agent instance is simply called agent) in accordance with the function described in Section 3. IWU agent type only includes one agent, which is called interface agent. The scanning process of the SCTR is included in the interface agent. In scanning stage, each parameter graph is scanned into computer in TIFF image format and occupies about 80 megabytes of storage space. CD agent type includes three agents, namely, image compressing, curve tracing, and data rectifying. PGR agent type only includes one agent, which is called graph-regenerating agent. DM agent type includes two agents, namely, data file management agent and middleware agent [3].

5.1. Image compressing agent

After a parameter graph is scanned and stored, the original image file must be compressed because not only it occupies too much storage space, but also the curve
The tracing efficiency is very low. The purposes of the compressing agent include two aspects. Firstly, the redundant message must be filtered out from the image file. Secondly, it is convenient to trace directly the curves based on the compressed image files. For solving above problems, we employed LAG data structure to compress the original image file. LAG is a kind of directed graph. Its nodes are all segments of continuous black pixels for all scanning lines of the image; the arcs of a node are the joint relations to other nodes in its adjacency scanning lines [9]. Assume two nodes $A(x_1, x_2)$ and $B(x_3, x_4)$ which locate in two adjacency lines, respectively. If $(x_2 \leq x_3)$ and $(x_1 \leq x_4)$ are true, $A$ and $B$ are joint. $A$ is $B$’s parent node and $B$ is $A$’s child node. The structure of a LAG node is shown in Figure 3.

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>Gray</th>
<th>Edge1</th>
<th>Edge2</th>
<th>Offset</th>
</tr>
</thead>
</table>

Figure 3. Structure of a LAG node

Let $X_1$ be the start value of the horizontal coordinates of a segment. Let $X_2$ be the end value of the horizontal coordinates of the same segment. Let Gray be the average gray of the segment. Let Edge1 be the number of joint nodes located in previous adjacency line. Let Edge2 be the number of joint nodes located in following adjacency line. Let Offset be the offset from current node to its first joint node located in following adjacency line in the LAG link list.

The algorithm adopted by the compressing agent includes two processes. Process 1, the centimeter grids are extracted from parameter graph image file for rectifying parameter graph distortions and recovering the parameter graph. Process 2, the nodes and their relations are extracted and organized with LAG. The algorithm of this process is described as following.

Input: TIFF dot matrix image data $I(m,n)$. A line $L(i)$ is loaded for each loop of the algorithm, where, $L(i)$ includes $n$ pixels and $0 \leq i \leq m$.

Output: LAG

Step 1: $i=0$ and load $L(i)$.
Step 2: Extract the $X_1$, $X_2$, and Gray of each segment in $L(i)$.
Step 3: Construct all LAG nodes in $L(i)$.
Step 4: While ($i < m-1$), do Step 5 to Step 9.
Step 5: $i=i+1$ and load $L(i)$.
Step 6: Extract the $X_1$, $X_2$, and Gray of each segment in $L(i)$.
Step 7: Construct all LAG nodes in $L(i)$.
Step 8: Fill in Edge2 and Offset (if Offset = 0) of related LAG nodes in $L(i-1)$.
Step 9: Fill in Edge1 of LAG nodes in $L(i)$.

After image file is compressed with LAG data structure, each well-logging parameter graph only occupies about 1.6 megabytes. The compression ratio is about 100:2.

5.2. Curve tracing agent

The goal of this agent is to link all nodes in a curve. The algorithm adopted by this agent is based on the LAG and always attempts to find the successor of current node in the next adjacency line. According to the value of Edge2 (0, 1, or greater than 1), there are three cases for finding the successor. Figure 4 is an example of two cross curves in LAG. The Edge2 of each segment of part A and D is 1. The Edge2 of the last segment of part C is 2. The Edge2 of the last segment of part B and E is 0.

Figure 4. Two cross curves in LAG

Curve segment (CS) data structure is proposed based on LAG for improving the speed and veracity when the curves are traced. A CS is defined as follow: if (Edge1 = 1) for any LAG node, it is the head of a CS; If (Edge2 = 1) for that head node or its son LAG node, it is the tail of the CS; All LAG nodes (Edge1 = 1 and Edge2 = 1) between the head and the tail (include the head and the tail) are members of the CS. For a specific CS, the number of joint Curve Segments (CSs) located in the previous adjacency line is Edge1 of its head LAG node; and the number of joint CSs located in the following adjacency line is Edge2 of its tail LAG node. The structure of a CS is shown in Figure 5.

<table>
<thead>
<tr>
<th>SHN</th>
<th>STN</th>
<th>Y</th>
<th>Depth</th>
<th>AW</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>A2</td>
<td>SS</td>
<td>Grid Segment</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5. Structure of a CS

Let SHN be the offset value of the head node of a CS in LAG link list. Let STN be the offset value of the tail node of a CS in LAG link list. Let $Y$ be the value of $y$ coordinates of the head of a CS. Let Depth be the depth of the CS along $y$ coordinates. Let $AW$ be the average width of the nodes in a CS. Let $A1$ mean the first attribute of a CS. The value domain of $A1$ is NOISE, DOT, and LINE. Let $A2$ mean the second attribute of a
CS. The value domain is BIG-DOT, SHORT-LINE, and NEITHER-BIG-SHORT-NOR-SHORT-LINE. The both above attributes indicate the forms of the curve. Let SS mean segment style of a CS. The value domain is PUBLIC-SEGMENT and PRIVATE-SEGMENT. Let Grid Segment indicate whether the CS is a centimeter grid.

The tracing algorithm consists of three processes. The first one is to generate all CSs based on LAG, and calculates the attributes of each CS. The second one is to link joint CSs according to their Edge2 values. The last one is to thin the linked curves. In linking CS process, the following rules are proposed: If Edge2 equals 1, the successor is the only joint CS located in the following adjacency line. Link the successor to the curve and change the successor as current processing CS until reaching the curve end. If Edge2 is greater than 1, the successor must be selected according to the characters (e.g. AW, attributes, SS, gray) of the joint CS located in the following adjacency line, or according to the extending trend of the linked curve. If it is still inexplicit, human aid is needed. If Edge2 is 0 and the depth of well does not reach the desired end, the original curve may be broken. In such case, define a zone that may include the possible successor in the following several lines and find a suitable CS as its successor according to the characters of the CSs and the extending trend of the linked curve (For dot line curve and broken line curve, the processing methods are similar with this rule). In thinning process, middle axis method is employed. The results are stored in a CURVEDATA data structure.

5.3. Data rectifying agent

The digitizing errors of the well-logging parameter graphs include mismatching two pages (when a graph is jointed with two or more pages), graph distortion (because of the environment factors), and rotating an angle (when the graph is scanned). The centimeter grids on the graph can implicate all above errors, that is, if an error occurs, the related centimeter grids will be correspondingly changed. The data rectifying agent will overcome all aforementioned problems. The distortion rectifying includes two steps. Step 1: Rectify all centimeter grids. Step 2: Rectify the curves grid by grid in accordance with the rectifying result of each centimeter grid.

5.4. Architectures of data management agents

The data file management agent consists of file manipulation (FM) Module, ACL Module, Control Module, and file server address library. FM Module deals with file manipulations. The file manipulations include creating, reading, updating, or deleting a file. The data file server and their addresses are listed in file server address library. ACL Module deals with the interactions with other agents.

The middleware agent consists of Socket Module, DBACL Checking Module, Task Executing Module, Database Operating Module, Output Result Generation Module, Knowledge For Database Operation Module, and Control Module. The architecture of the agent is shown in Figure 6.

The Socket Module deals with the communication with other agents. This part is the lower layer of DBACL and makes the DBACL formats satisfy Socket and vice versa. DBACL Checking Module deals with parsing of DBACL statement according to the rules and database information stored in the Knowledge for Database Operation Module through Control Module. If the DBACL statement is valid, it is transmitted to Task Executing Module, otherwise transmitted to Output Result Generation Module to generate an error message as response. Task Executing Module decompounds the DBACL statements as the SQL statements to operate database through Database Operation Module. On the other hand, this module organizes the results from database and transmits them to Output Result Generation Module. All function of this module abides by the knowledge stored in Knowledge for Database Operation Module. Database Operating Module is the interface to operate the database. Output Result Generation Module forms DBACL statements based on the rules stored in Knowledge for Database Operation Module. The inputs of the module are from DBACL Checking Module or Task Executing Module and the outputs are DBACL statements with the results accessed database or with responses. Knowledge for Database Operation Module...
manages and stores the DBACL rules and database operating information.

6. Case study

The framework and algorithms described in this paper have been applied in an industrial project----An agent-based curve-digitizing approach to well-logging data manipulation [10]. The parameter graph with well log curves is scanned as an image file (TIFF format) by scanner. The image file is preprocessed as the standard pattern file and then compressed to characteristic data files, which are saved into data file servers by data file management agent. The curves implied in those data files can be digitized to curve data. Transmit curve data to middleware agent by means of DBACL to store the data to Oracle database. A regenerating agent can access curve data or other information in Oracle database or data file servers by means of middleware agent or data file management agent and regenerate the parameter graph again. The agents worked well by processing test with thousands of parameter graphs. Figure 7 shows the part of result of well log curves. The proposed system is implemented using C and Socket techniques.

7. Conclusions

The system shows that the agent-based approach satisfies well-logging parameter graph digitization and management; furthermore, the architecture of agent-based framework can be used as a basis for multi-agent systems. The main contributions of this paper are as follows.

- A specialized framework, which satisfies both multi-agent and heterogeneous data source environment, is built.
- The SCTR curve-digitizing algorithms based on well-logging parameter graph are improved.
- An agent-based curve-digitizing system based on well log information management is developed.

8. References