Complex Learning System for Behavior Factor based Data Analysis

Guan Wei
Information Center, Xi'an University of Posts and Telecommunications, Xi'an; China
E-mail: guanw@xupt.edu.cn

***Abstract*—In this paper, we propose a complex learning system for data analysis that is based on behavior factor. This system combines features of both order based systems and order based data analysis framework. This system uses an appropriate behavior factor acquisition process, which is the weakest task in the current order and behavior factor based systems. As several research works have shown that data analysis is rather driven by domain behavior factor than by data, we have identified and analyzed the characters that distinguish behavior factor and orders from data for better determining the most components of the proposed system. This paper describes an approach that advocates the use of behavior factor order-based systems techniques for data analysis systems. In particular, we aim to make domain behavior factor explicit and separate from the system where will be used for data analysis. For representing domain behavior factor, we investigated a uniform and unified order representation form based on the creation of the environment component that stored information especially for the management of order and its quality.**

***Index Terms*—Data analysis, Behavior factor, Complex learning System, Components**

I. INTRODUCTION

Your goal is to simulate the usual appearance of papers in a Journal of the Academy Publisher. We are requesting that you follow these guidelines as closely as possible.

Data Quality (DQ) has always been an important issue and is even more the case today. The research works look at the role of Data Analysis (DA) tools in helping improve DQ and clarify the need to take an enterprise wide approach to DQ management, which is increasingly complex, open and dynamic [1, 2]. Although there is a steady stream of DA tools in practice and research, more recent studies have showed that the situation has not enhanced and companies do not invest adequately in DQ [3]. Therefore new approaches and architectures are required to help improve the DQ.

There is a wide variety of DA tools. Their functionality can be classified as follows: Declarative DA and Order-Based approaches for DA (OBDA). Whatever tools are chosen, a systematic, the order-based approach yields better results than an unstructured approach [4-6]. Although the Order-Based System (OBS) that encode behavior factor as orders and used to process complicated tasks have been firmly established for many years, they have not been well formally and adequately addressed for the DA tasks. Hence there is a need to an appropriate OBS for DA.

The current OBDA solutions often use Business Orders (BO), which are constraints on data contained in business policies into a formal form in order to allow computerization [7]. As BO is only a specific part of all Domain Behavior factor (DB), it is necessary to investigate, and how to exploit the all DB for DA. The main challenge in DB is behavior factor discovering, which is usually manually made by human experts (interviews or questionnaires) or collected from written sources (books or other reference material). Consequently, the first step in the development of any OBS for DA is to begin with a collection of the behavior factor from which the orders will be derived. Organizing the collected behavior factor so that translation to orders will be straightforward is also a challenging task for the behavior factor engineer [2, 8].

Several research works have shown the importance of DB in DA. However, there are few works on behavior factor management issues for DA.

Similar to data, behavior factor are also issued from different sources, which may be incomplete, inconsistent, heterogeneous and pervaded with uncertainty. Then, there is a need and a wide-ranging interest in managing the quality of behavior factor and orders in order to ensure high DA quality.

In this paper, we assume that a OBS can play yet another role; it can be an agent of change to improve the DQ. Our primary objective is to improve the functional capability of a DQ management by embedding it with behavior factor of the problem area. When the behavior factor acquisition is performed, the system states the behavior factor in the form of orders and uses these embedded orders to provide advice. As the currently OBS allow only the manipulation of production orders without precise under-standing of orders theoretical foundations and without considering order quality, our system provides an uniform order representation based on First Order Structure (FOS) theory of Logic that allows the representation of all the types of orders (derivation, production, integrity, transformation and reaction) and their quality characteristics. Hence the proposed OBS for DA contains two subsystems: the first is related to the orders management and the second is focused on the orders quality management. In this system, we have

doi:10.4304/jsw.8.4.1003-1010

incorporated two semiautomatic processes for the behavior factor acquisition and transformation.

Another feature of our proposal is that the control and the improvement of order (resp. behavior factor) quality are performed online, incrementally, during the design of the system. It permits an early clean of behavior factor and orders from anomalies and inconsistencies.

The proposed OBS for DA system lends itself to iterative process because in the first time it has not all the information needed to write the behavior factor and orders.

We assume that our proposal can be very advantageous in the meaning that the OBS can interject and provide necessary behavior factor at the appropriate points in the DA process.

II. Background and some Related Works

As our objective is to enhance the DQ by applying a Order based approach in DA, it is necessary then to represent some works related to Behavior factor Based System, Order Based System and Orders-Based Data analysis.

A. Behavior factor Based Systems

A BFS is defined as an interactive Software system that uses stored Behavior Factor of a specific problem to assist and to provide support for decision-making activities bound to the specific problem context. BFS have been developed and used for a variety of applications [9-11].

Behavior factor acquisition is the most important element in the development of BFS. Behavior factor could be obtained from domain experts, raw data, documents, personal behavior factor, business models and/or learning by experience.

The BFS is vague, and that it is generally hard to specify in advance a BFS completely because of the incremental nature of the behavior factor elicitation process.

Interviews are considered as the most popular and widely used form of expert behavior factor acquisition. One serious drawback of interviews is the fact that they are time consuming.

B Order Based Systems

OBS prove to constitute one of the most substantial technologies in the area of applied Artificial Intelligence (AI). They have used in diverse intelligent systems. Some interesting recent applications of OBS are the ones that are-related to business and the Semantic Web. Both current research works and applications go far beyond the traditional approach.

In the OBS, behavior factor is represented in the form of (If condition Then conclusion < action >) production orders. An order describes the action that should be taken if a order is violated. The empirical association between conditions and conclusions in the order base is their main characteristic. The general architecture of OBS includes domain independent components such as the order representation, the inference engine, the explanation system, an order engineering tool and a specific user interface. Most of this OBS are just shells providing an order interpreter and sometimes an editor.

Various order representation techniques are available. However, the orders for DA would require an appropriate order representation technique that allows the management of order and its quality, and the presentation of all type of order.

Let us notice that there are different categories of orders: integrity, production, derivation, reaction and transformations. Integrity orders consist of a constraint assertion. Derivation orders are used to derive conclusion whenever the conditions hold. Production orders produce actions if the conditions hold, while post-conditions must also hold after the execution of actions. A reaction order is a statement of programming logic that specifies the execution of one or more actions in case of occurrence of a triggering event and satisfaction of its conditions. Optionally, after executing the action, post-conditions may need to be satisfied. Orders can be also extracted from existing programs codes that are not expressed in any specific order dialect by using program slicing techniques.

The main problem encountered the implementation of OBS concerns the complete specification of non-redundant and consistent set of orders. This turns out to be a tedious task requiring significant effort of DB engineers. The approaches to verification and validation of OBS do not solve the problem entirely. The verification that is performed after the system designed is both costly and late. Moreover, after introducing the corrections of detected errors, the verification cycle must be repeated. The problem consists in the possibility of introducing new errors through correction of the old ones.

The distinctive feature of our system is the design of Order-based system for data analysis, which allows one to manage formally orders and their quality.

C Order based Data Analysis

Although DB has been identified as one of the main ingredients for successful DA, the issue of behavior factor representation to support DA strategies has not been well dealt with. Entire DQ process will be integrated complex learning system in order to achieve clean of data that is as automated, as correct and as quick as possible. Therefore only small number of data would be left for manual inspection and reprocessing.

The most current systems based on orders for DA use generated orders to check data in designated tables and mark erroneous records, or to do certain updates of invalid data. They will also store information about the orders violations in order to provide analysis of such data [8].

The incomplete and inconsistent information is a major challenge for OBDA. The quality of a order base which is an infinity problem that is difficult to solve is an essential issue for the OBDA processing. However, this can’t be done effectively using classical logic.

FOS theory is particularly interesting, as it can be a basis towards strong theoretical foundation for

developing an order form that permits us to manage order and its quality.

D Data Stream Classifying Method

Data-Stream classifying method expands the concept of ε-neighborhood and core-objects in Data Analysis with a fading function to maintain up-to-date information.

A ε-neighborhood is defined by Data Analysis as being a set of points that have a distance to another point less than the user-defined parameter ε. More specifically, given point p and dataset D, the ε-neighborhood of $p(N_\varepsilon(p))$ is equal to: $N_\varepsilon(p)=\{q\in D \mid dist(p,q)\le\varepsilon\}$, (1) where dist (p, q) is the Euclidean distance between point p and q.

A core-object is defined as a set of points within a ε-neighborhood that contain more points than the minimum point parameter. If p is part of a core-object, Data Analysis will expand the group around p.

The basic structure of the method is as follows:

1) Data Analysis takes the ε and minimum point parameters and then chooses a point p that has not been visited.

2) Data Analysis calculates N_ε (p). If the size of N_ε (p) is greater than minimum points, Data Analysis expands a group around p. Otherwise, the point is considered noise.

3) Data Analysis iterates to a new unvisited point and repeats the process.

Although Data Analysis was originally developed for a batch environment, it has provided an inspiration for stream classifying methods.

The fading function is defined as:

$$f(t)=2^{-\lambda t} \tag{1}$$

where $\lambda>0$ represents the decrease factor and t represents the time.

Data-Stream also modifies the core-object concept of Data Analysis, creating a core-group with three additional attributes: radius, center and weight. The radius must be less than or equal to ε, and the weight of a group must be greater than the user-defined parameter μ. The weight w, center c and radius r of a core-small-group are more formally defined at time t, for a set of close points, $p_1, p_2, \cdots, p_n$ with time-stamps $T_1, T_2, \cdots, T_n$ as:

$$w=\sum_{i=1}^{n} f(t-T_i) \tag{2}$$

$$c=\frac{\sum_{i=1}^{n} f(t-T_i)p_i}{w} \tag{3}$$

$$r=\frac{\sum_{i=1}^{n} f(t-T_i)dist(p_i,c)}{w} \tag{4}$$

Where dist (p_i, c) is the Euclidean distance between the point p_i and the center c.

Because Data-stream operates in a stream environment, the core-groups need to change dynamically as time passes. To facilitate this, a potential core-groups or p-small-group is introduced. P-small-groups are similar to core-small-groups, except they differ in that the center and radius values are based on the weighted sum and squared sum of the points ($\overline{CF^1}$ and $\overline{CF^2}$). Also, the weight must be greater than or equal to $\beta\mu$ where β defines the threshold between p-small-groups and outliers such that $0<\beta\le 1$. $\overline{CF^1}$ and $\overline{CF^2}$ are calculated using the formulas:

$$\overline{CF^1}=\sum_{i=1}^{n} f(t-T_i)p_i \tag{5}$$

$$\overline{CF^2}=\sum_{i=1}^{n} f(t-T_i)p_i^2 \tag{6}$$

This changes the center and radius values to be:

$$c=\frac{\overline{CF^{11}}}{w} \tag{7}$$

$$and \quad r=\sqrt{\frac{\overline{CF^2}}{w}-\left[\frac{\overline{CF^1}}{w}\right]^2} \tag{8}$$

Although the p-small-group permits the model to be updated dynamically, it generally will not provide a representative view of a workflow as new points appear. To handle this concept drift, Data-Stream also introduces the outlier-small-group (or o-small-group) and an outlier-buffer that temporarily stores o-small-groups and allows them to become p-small-groups. The operation of Data-Stream is as follows:

Initial Step: perform Data Analyzing on a set of initial points to generate starting p-small-groups.

Online Steps, when a new point p arrives in the stream:

1) The method attempts to me to merge p with the closest p-small-group. If the radius of the potential small-group is less than or equal to the value of ε, the point is merged.

2) If the point is not merged to a p-small-group, it tries to merge p with an existing o-small-group. If the radius is less than ε, it is merged with the o-small-group. Then if the o-small-group now has a weight large enough to become its own p-small-group, it is removed from the outlier-buffer and added to the model as a p-small-group.

3) If the point cannot be merged to an existing o-small-group, it creates a new o-small-group and gets placed in the outlier-buffer.

After the merging phase of the Data-Stream method, the lower weight limit is calculated for all o-small-groups in the outlier buffer. This is done using the formula:

$$\xi(t_c,t_0)=\frac{2^{-\lambda(t_c-t_0+T_p)}-1}{2^{-\lambda T_p}-1} \tag{9}$$

Where $\lambda>0$ represents the decrease factor, t_c and t_0 represent the current and starting time for the o-small-group, and T_p is the predetermined time-period.

4) If the weight of a particular group is less than the lower weight limit, the o-small-group can be removed from the outlier buffer.

E Discussion

The RDBC approaches proposed for both academic research and practical applications have certain persistent limitations related to the following aspects of order design:

No practical methodology for OBS is acceptable for OBDA systems because these methodologies are available only for order production and don't ensuring the quality of order.

Lack of formal relation of the order representation to logic.

Consequently the development of an appropriate OBS for DA is a crucial issue for the final success of OBDA where the order representation should be of satisfactory expressive power in order to express all of the required orders and be easy to handle and manage order and its quality. Order quality control consists in checking for some formal characters that an order base should exhibit. Let us notice that the most typical formal proprieties are: Consistency, completeness, determinism, redundancy and efficiency.

So in this paper, we target at dealing with above limitations by providing an appropriate OBS for DA in which the order form is based on FOS theory.

Our proposal started from the idea that the quality of DA results is determined by the quality of orders used for processing. The use of FOS provides a concise and elegant order representation for the management of order and its quality.

III. ORDER BASED DATA ANALYSIS SYSTEM

An important advantage of the proposed OBDA system is that its design is based on strong order-based DA meta-model. In addition the method is used as a basis for the development and specification of the uniform order representation.

A. Order based Data analysis Meta-model

The generic data analysis method in Figure 1 which is drawn with the UML notations illustrates the most important information which is worth tracking on OBS for DA. The information is captured through the life cycle of the order-based DA systems and its supporting data.

The role of the method is to provide the most components of the system and their relationships. Hence, it offers a good basis to design the OBDA system with the understanding of its functionality and to define a uniform order representation. The method is divided into two parts, one that presents the behavior factor/order based system and the other that contains the complex learning system and order quality management system. As depicted in Figure 1, the main types of orders considered in this meta-model are: reaction, production, integrity, derivation and transformation.

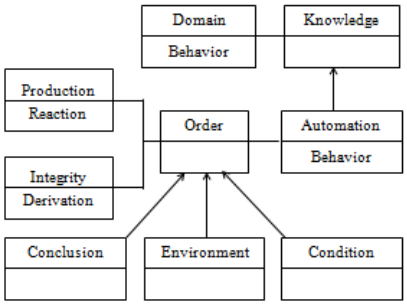

Figure 1. Order-based Data Analysis method.

The concept of Component aims at realizing a unified order form. As the condition and conclusion are the common components of each type of orders, the idea behind the use of Component concept is to represent the uncommon components (for example the component: post-condition is only used for a reaction order).

B Order Representation

The expressive power of our order representation constitutes a crucial feature with respect to the ability of behavior factor specification. Hence, it is of primary importance to properly find a uniform order form that deals with the limitations described above.

Although there are several order representation built on top of First Order Logic, the concept of Structure that gives a context for determining the value (true, false) of a given formulae has not been taken into account. In this work, we provide orders with the concept of Environment which is an adoption and an adaptation of the Structure in order to create a uniform order form. Therefore, the order form is defined as following:

IF Condition THEN Conclusion ON Environment The Environment which is built incrementally and iteratively has three main parts: Initial Environment (Initial_Env), Global Environment (Global_Env) and Order Environment (Order_Env). It contains three kinds of information: Universe (U), Quality Dimensions (QD) and Component (Cr). The Universe sets a desired target data where the order will be applied and the actions will be taken if order is violated. The Quality Dimension is a set of dimensions for the management of the order quality. The QMS selects suitable metrics from the metrics table for each dimension and so uses them to evaluate and to verify the quality of orders. The Component is a set of information (triggering event, post-condition and so on) that characterize each type of order. Thereafter the general form of order can be rewritten as follows:

IF A THEN C ON {U, QD, Cr}

where:

A is a conjunction of atomic conditions. This set is empty in the case of integrity order.

C is a set of atomic conclusions.

U = {DataSet (R), Action} where DataSet (R) is the set of data in which the order R is applied and the Action is the programs that will be taken when the order is violated.

QD is the list of quality dimensions that are used to evaluate the quality of a given order.

Cr = {Event, Post-Condition…}.

Table 1 show how the proposed order representation is wanted to be a general order form for writing all types of orders in comparison with related works that arrange orders types in a hierarchical structure.

As the goal of this work is to clean data, it is worth noted that our system defines the Action for each order differently to the currently OBS where Action is only defined for the reaction and production orders.

C Overview of the OBDA System Components

In this section, we provide an overview of our OBDA system. The proposed system after its installation enables users and behavior factor workers to create and to update behavior factor and order through a user interfaces. The order after its creation is affected to the corresponding temporary orders base, which are used to facilitate the extraction of order characteristics. These characteristics are exploited by the system to manage the orders and their quality.

Figure 2 shows the Order based Data analysis System. The system is a set of behavior factor sources, process, tasks, structures and subsystems. It is based on the RUI and RMS technologies. The proposed OBDA system consists of ten elements.

The main behavior factor sources used by the OBDA system are:

1) Domain Behavior factor: it is the set of behavior factor sources from which behavior factor can be obtained. In our work, the legacy system, programs and databases referred also to as domain behavior factor parts.

2) Behavior factor/Order bases: the Behavior factor Base (KB) is temporary area where the behavior factor transformations are applied. The Order Base (BS) is the physical area where orders are stored in order to will be used for DA. Once the behavior factor is transformed into a order, it will be stored in the RB and deleted from the KB.

TABLE I.
DIFFERENT ORDER FORMS

Type	Condition	Conclusion	Condition	Event
Integrity	0	0	0	0
Reaction	0+	1+	0+	1
Production	1+	0	0+	1
Derivation	0+	0+	0	0
Transform	1+	1	0	0

X=0.1 x exactly x^{+}= at least

The most typically tasks and processes of the system are:

1) Installation: as the design of OBDA system is independent of the databases to be cleaned; then, it becomes necessary to configure the environment required for performing the OBDA.

2) Behavior factor Acquisition: This process focuses on extracting behavior factor from expert behavior factor, books, documents, and so on by using manual form text or questionnaires.

3) Behavior factor Transformation: it provides a set of operations necessary to transform behavior factor onto orders.

4) Order Classification: the goal of this task is to affect each order to its type (Production, Reaction, Derivation, Integrity and Transformation). For each type, we have defined the corresponding order template, with which a given order will be compared.

5) Mapping: it provides a set of transformations orders that permit the expression of each order of temporary RB with respect to the uniform order syntax.

6) Order extraction: it uses the techniques of extraction of order automatically from computers (Databases, legacy systems, programs).

As the system is interactive, it provides three types of user interfaces. The first type is Behavior factor User Interfaces (BUI), which allows us to manage and interact with the information about behavior factor. The second type is Quality User Interface (QUI) which allows managing the quality dimensions and metrics. The third type is Order User Interface (RUI) which is similar to the BUI.

The OBDA system enables to manage the order by the Complex learning System and its quality by the Quality Management System (QMS). Each system which is considered as a subsystem of the OBDA system is suite of software applications that together make it possible to apply operations on order and its quality.

As time advances and the sources of DB from which order is extracted change, orders base must be synchronized with the underlying sources. Thus, after an initial built of our system, RB must be refreshed. For this end, our sys-tem allows the refreshment of RB for DA in real time.

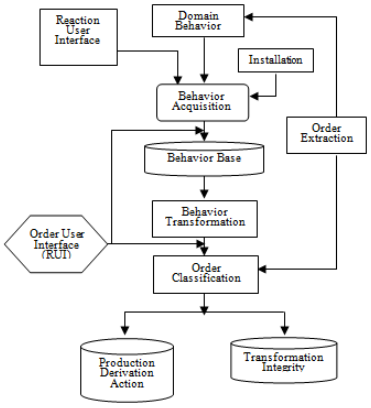

Figure 2. Order-based Data analysis System

IV. DESCRIPTION OF THE SYSTEM

As we have indicated above, the proposed system provides two subsystems for the management of orders and their quality. These subsystems which are fully integrated with the OBS for DA use the Environment component as basis for complex learning.

A Environment

This subsection gives the detail of the construction of the Environment of order. Let us notice that, contrary to the Global Environment and the Order one, the Initial Environment is available for any order.

1. Initial Environment

The Initial_Env is a set of common quality dimensions which are independent of particular target order base and applicable and available for each OBS. In particular, it is

the set of the most typical proprieties presented in discussion section. It is created during the installation task.

2. Global Environment

The Global_Env is also a set of quality dimensions specific to the order base and/or data set to clean. These dimensions can be performing in all orders or a subset (one) of orders. They are introduced to the system through the QUI by the user. They can be also introduced during the insertion or modification of orders through the RUI.

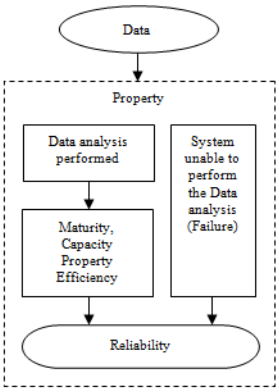

Figure 3. Context chart for Data Analysis property

3. Order Environment

The Order_Env is the set of information necessary to the building and execution of order. The information are introduced by the user through the RUI or extracted from the behavior factor base as depicted in Figure 2. The order is considered effective if the Universe (both dataset to clean and action are not empty) is defined.

B Complex Learning System

A Complex learning System is a software package that performs and controls the creation, modification, suppression and the archiving of orders. The orders are stored in relational tables. When the order is not yet applied on data, it can be deleted or archived. The user can be involved to manipulate the order through the RUI.

C Quality Management System

As the quality of DA results is determined by the quality of order, our system integrates a module for order quality management which can be performed automatically or by the user through the QUI. The QMS follows the continuous life cycle of Total Data Quality Management: Define, Measure, Analyze and Improve orders as essential activities to ensure high quality of data.

Differently to data where the DQ improvement can change the value of data, the QMS can only deactivate or archive order when it is evaluated poor. Therefore, the user defines the quality dimensions and their quality metrics which allow measuring the quality of order-through the QUI and after the QMS analyzes the quality of order in order to decide how to improve it. Let us notice that the use of Environment is to allow an incremental define of quality dimensions and their metrics.

Figure 4 shows possible outcomes of a Data Analysis.

Let us notice that the quality dimension can be measured by combining multiple metrics and also a metric can be used by multiple dimensions. So in our system each metric is implemented as atomic function (code).

V. OBDA FUNCTIONS

A Installation

The installation as we have described above is an important task because it allows to the user to define the information and requirements necessary to the OBDA. Among the most typically information, we cited:

Location of databases, Users authorized to use the order-based data analysis system.

List of sources that are parts of the domain behavior factor, A set of ontologically based data quality dimensions and metrics used in the literature and also are important to orders.

.Keywords of the domain behavior factor will be used to regrouping the behavior factor.

B Behavior Factor Acquisition

This process follows two steps: Collection and Storage. The Collection step aims at collecting the information extracted from sources of DB using questionnaires in natural language. The memorization of the information is made during the Storage step according to structured language sentences through the BUI in the KB. Note that the user can store directly the information through the BUI.

C Behavior Factor Transformations

Differently to the KDS which performs this task during behavior factor acquisition process, our system performs it separately (after) to the above process because it needs the intervention of behavior factor engineer. The main activities are:

Creation of behavior factor sets according to the keywords behavior factor.

Rewriting of behavior factor which does not respect the language sentences.

Elimination of duplicate behavior factor.

Decomposition of behavior factor in atomics behavior factor.

Elimination of duplicate atomic behavior factor.

Translation of behavior factor into orders.

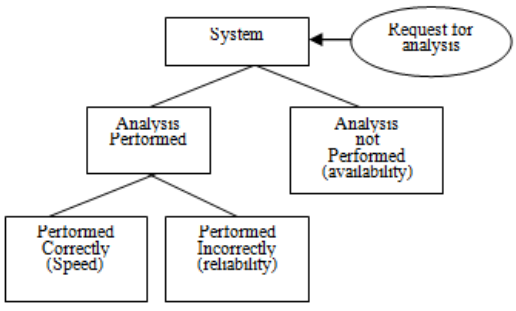

Figure 4. Possible Outcomes of a Data Analysis

D Order Extraction

As we have indicated above, the proposed phase allows the extraction of orders from programs, legacy systems and so on by using the data mining and programs slicing techniques. In generally, these techniques extract behavior factor as association orders with confidence and support metrics.

E Order Classification

At this phase, the majority of orders are not in the RB and don't respect the proposed order form. Then, there is a need to regroup them according to their types. To this end, each orders type has an appropriate order template with which a given order will be compared in order to determine its temporary order base (i.e. type). The order classification simplifies the order formalization and ensures a higher clarity and consistency of the order description. This last allows one to identify aspects that are necessary to compute certain quality dimensions of orders. For example the support and confidence metrics are used to compute the accuracy of a order.

F Orders Mapping

Orders mapping is a task that moves orders from temporary orders bases (integrity, transformation, derivation, production and reaction bases) to orders base (RB), where the orders will be written according to our general order representation. The system extracts Condition, Conclusion, Global Environment and Order Environment from orders before their transformations. The most typically orders transformations are:

1) Transformation 1: Order Condition:

It aims at transforming the Condition component of order in conjunction of atomic conditions.

Order Condition (R in Temporary RB) = $A_{i=0}^{i=n} C_i$

where C_i is an atomic Condition

This transformation is the same for all types of orders.

2) Transformation 2: Order Conclusion

It aims at transforming the Conclusion component of order in conjunction of atomic conclusions.

Order Conclusion (R in Temporary RB) = {C_i| C_i is atomic Conclusion}

This transformation is the same for all the types of orders.

3) Transformation 3: Order Universe

The Data Set (R) is extracted directly from the temporary order but if it is empty, then the system supposed that the order is applied in the all data. The physical emplacements of Action are also extracted and stored in the universe U. these emplacements can be indicate later by the user through the RUI.

4) Transformation 4: Order Components

The Event component of order, if it exists, should be extracted directly from the temporary order. The Post Condition component, if it exists, should be mapped by applied Order Condition (Transformation 1).

There are two goals underlying this algorithm. Firstly it computes the accuracy value of association orders from the support and confidence metrics. Secondly, it identifies the quality dimensions appropriate to a given order from its description and added them to the list of QD of the order. In the case where the QD is not defined, then the algorithm allows to the user to create this dimension and its metrics through a specific QUI. The algorithm also permits the manually selection of quality dimensions that are important from the user point of view.

VI. Case Study and some Experimental Results

In this section, we present some experimental results of the validation of our proposal in the health sector. We study the impact of embedded DB in DA for DQ performances. These experiments show that the proposed RMS and QMS subsystems deal well with some of the limitations of existing order based data analysis works.

The most typically components of the architecture of OBDA include:

Graphical User Interfaces for the dimensions quality, behavior factor and complex learning.

RMS and QMS libraries.

Programs corresponding to the proposed algorithms, process, tasks and transformations.

Association Orders mining tools.

These components are developed under the JAVA environment. They connect to the Order-bases, Behavior factor bases, Quality tables and Databases to be cleaned via JDBC Bridge. All the data and orders are in the relational schemata. SQL Query is used in our experimentation to perform the extraction and storage of Behavior factor/orders. The collection of behavior factor from KD is realized through a manual form designed especially for the collection of health information

The atomic conditions and conclusions of order in this experimentation are based in the use of attributive logic, which uses attributes for denoting some proprieties (A) of objects (o) and atomic values of attributes (d). It has this form:

A (o) op d

where op is operator.

The results of this case study provide several important observations:

1) Quality of orders: One of the main features of our proposal is the generation of orders and behavior factor with their dimensions and metrics quality that allow to the system to discover more interesting and higher quality orders.

2) Expected orders: This kind of order can be considered as rare, as they would be pruned by many objective interestingness measures (support and confidence). The use of Universe in order representation has allowed reducing the number of expected orders.

3) Uncertainty Order: contrary to the inference engine which does not reasoning with uncertain orders, our system reduces the uncertainty by specifying the domain and appropriate dimensions quality of order by the end user and the incrementally construction of the Global Environment of order.

4) Order and Quality Dimensions Groups Validation: this distinguished feature of this work is the ability for the end user to examine groups of orders or of quality dimensions for group/one orders and decide whether to

archive or deactivate a group as whole. For example, for a given order, we can deactivate its appropriate quality dimensions if it is necessary.

5) Autonomous: The QMS and RMS can be performed in online and off-line mode. During the initial building of OBDA, the offline mode is preferable. The clean of data can be also in real time i.e. when data is added to the database.

Although the design of the architecture allows one to use the notions of parallelism and pipeline, the time consuming is the major challenge of this proposition especially during the initial building of the OBDA system.

VII. Conclusions and Perspectives

This work confirms that conventional software engineering and behavior factor engineering are complementary and both essential for developing the increasingly larger systems of today. In this context, this paper describes an approach that advocates the use of behavior factor order-based systems techniques for data analysis systems. In particular, we aim to make domain behavior factor explicit and separate from the system where will be used for data analysis. For representing domain behavior factor, we investigated a uniform and unified order representation form based on the creation of the environment component that stored information especially for the management of order and its quality. Besides dealing with some limitations of currently OBS and BFS cited above, the system through the case study allows us to observe many advantages. The major challenge of our proposal is the time consuming. Then, it makes its implementation tedious and complex. This is related to the problem of automation of behavior factor acquisition that is not yet resolved.

Finally, once this system is designed, we intend to make it independent from specific database. Therefore, it would be interesting to perform this system with other possible databases. The QMS will also include a feed-back loop to enhance the order quality.

References

[1] C. Jones, “Applied Software Measurement: Assuring, Productivity and Quality”, 2nd Edition, McGraw-Hill, New York, 2009.

[2] W. Gibbs, “Software’s Chronic Crisis”, Scientific American, September, Vol. 167, No. 4, 2009, pp. 468-473.

[3] Standish group, extreme chaos, Standish group international. http: //www. standishgroup.com, 2010

[4] L. J. May, “Major Causes of Software Project Failures”, 2009. http: //stsc.hill.af.mil/ crosstalk/1998/jul/causes.asp

[5] A. Taylor, “IT Projects: Sink or Swim”, The Computer Bulletin, pp. 24-26. doi: 10.1093/combul/42.1.24, 2010

[6] J. Ropponen and K. Lyytinen, “Components of Software Development Risk: How to Address Them?” IEEE Transactions on Software Engineering, Vol. 26, No. 2, pp. 98-111. doi: 10.1109/32.841112, 2010

[7] D. R. Denison, Corporate Social and Management Effectiveness, John Wiley & Sons, New York, 2010.

[8] F. J. Heemstra and R. J. Kusters, “Dealing with Risk: A Practical Approach”, Journal of Information Technology, Vol. 11, pp. 333-346. doi: 10.1057, jit, 2011

[9] D. Radcliffe, “Project Leadership”, Software Magazine, Vol. 18, No. 5, pp. 38-44, 2008.

[10] N. H. Arshad, A. Mohamed and Z. Matnor, “Software Development Projects: Risk Reasons and Occurrences”, WSEAS Transactions on Computer Research, Vol. 2, No. 2, pp. 1991-8755, 2009.

[11] T. Addison and S. Vallabh, “Controlling Software Project Risks—An Empirical Study of Methods Used by Experienced Project Managers”, Proceedings of SAICSIT, pp. 128-140, 2008.

[12] A. Kakas and M. Denecker. Abduction in logic programming. In A. Kakas and F. Sadri, editors, Computational Logic: Logic Programming and Beyond. Part I, number 2407 in LNAI, pp. 402-436. Springer, 2010.

[13] .R. Moller, V. Haarslev, and B. Neumann. Concepts based information finding. In Proc. IT&KNOWS-98: International Conference on Information Technology and Knowledge Systems, 31. August- 4. September, Vienna, Budapest, pp. 49 (6), 2008.

[14] .R. Moller and B. Neumann. Ontology-based reasoning techniques for multimedia transformation and finding.In Semantic Multimedia and Ontologies: Theory and Applications. 2007. To appear.

[15] .B. Neumann and R. Moller. On Scene Transformation with Description Logics. In H. Christensen and H.-H.Nagel, editors, Cognitive Vision Systems: Sampling the Spectrum of Approaches, number 3948 in LNCS, pp. 247-278. Springer, 2009.

[16] .S. E. Peraldi, A. Kaya, S. Melzer, R. M oller, and M. Wessel. Multimedia Transformation as Abduction. In Proc. DL-2007: International Workshop on Description Logics, 2007.

Guan Wei: (1968.6-) Female, Master of science in educational technology, Senior engineer, Information Center of Xi'an University of Posts and Telecommunications; Xi'an; China, Degree was earned in 2004, The major field of study is computer network application.