An Approach to Advanced Data Synchronization in Complex Process Control Systems

D.Kuklenko, R. Gamzayev, V. Goloborodko, M.Tkachuk
National Technical University "Kharkiv Polytechnic Institute",
Frunze str., 21, Kharkiv, Ukraine
dmytrokuklenko@yandex.ru, gamzaev@mail.ru, vvgoloborodko@mail.ru,
tka@kpi.kharkov.ua

Abstract: Nowadays for any kind of distributed and multi-level, i.e. complex information systems it is vitally important to represent its data resources in a homogenized form. This paper presents our approach to data synchronization (DS) in complex Process Control Systems (PCS) designed with respect to our real project experience in the domain of PCS for gas-production industry in Ukraine. DS is considered as a stage of more sophisticated approach to data processing called Intelligent Data Engineering (IDE). The IDE framework and some of its modeling aspects are shortly described. The algorithm used at the DS stage for transferring process data between the levels of distributed PCS in XML-format is proposed and implemented. In order to estimate a system performance in a DS publisher node the metrical-based modeling is introduced in our approach.

1 Introduction

Modern trends in the domain of process control systems show that the usage of Web-based technologies in such systems is nowadays in common practice. These technologies are also used in the domain of Supervisory Control Access and Data Acquisition Systems (SCADA), the motivation and examples of such usage can be found, e.g., in [IG01]. We consider SCADA to be a part of PCS, and it collects and handles the technical data provided by Programmable Logical Controllers (PLCs). The other PCS functions consist in analyzing the data collected by SCADA system. When the system nodes that solve the SCADA and non-SCADA tasks are distributed, the data synchronizing problem for this distributed system should be solved. Also we should guarantee that the performance of SCADA-components functioning is reliable and the analytical subsystems receive the actual data.

Our research is closely connected with the development and re-engineering of Web-based PCS for the enterprises of gas and oil industry in East Ukraine we have been performing during last years. The results of our research are presented, particularly, in the papers [Tk01, Tk02, Ku01]. In this paper we present the framework for data processing in multi-level distributed PCS and consider the data synchronization and performance estimation tasks.

The paper is structured as follows: Section 2 introduces our common approach to Data processing in multi-level PCS called Intelligent Data Engineering (IDE), then in Section 3 we present our algorithm used at the Data Synchronization stage of IDE, and in Section 4 we describe our approach for performance estimation in SCADA system called metrical modeling.

2 Intelligent Data Engineering as common approach to data processing in distributed multi-level PCS

We propose the four-step framework of Intelligent Data Engineering to be used in PCS. It provides the solutions for the main tasks of data handling in information systems. The common scheme of IDE is shown on the Fig.1. Here we describe briefly the IDE main steps.

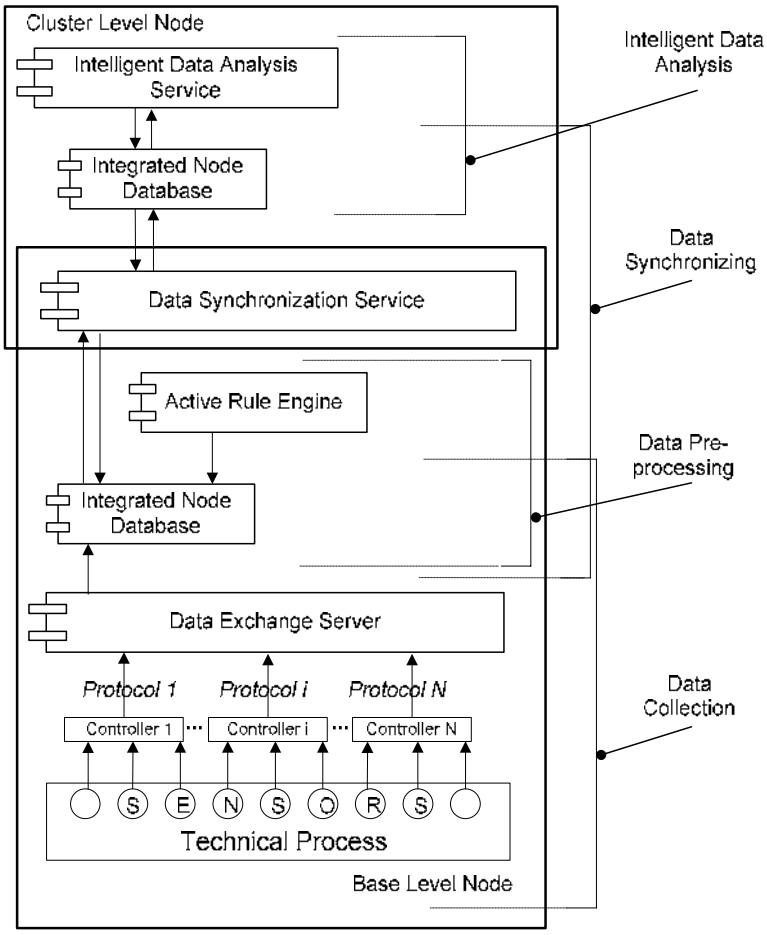

Fig. 1. Intelligent Data Engineering in Process Control Systems: common framework

1. Data collection. In PCS the data are acquired from Programmable Logical Controllers (PLC) with the high frequency. The data collection task consists in configuring the software component (Data Exchange Server, DES) which directly provides the interaction with PLC, with the actual data exchange protocol structure, and to store the data received

from DES in such form that allows us to easily associate them with the structure of technical process (TP).
The solution for this task is the set of data schemes which are combined into *Integrated Node Database* (INDB). INDB is the main software component in the architecture of a PCS which realizes IDE framework (see [Tk01]).
2. Data pre-processing. Here we can distinguish the following sub-tasks:
- To filter out the obviously mistaken data, which can be generated by the faults in censors, PLCs, and communication links;
- To average the parameters values, in order to reduce their size;
- To find out the specific situations in TP by analyzing the parameters values, and to store the primary data describing such specific situations as well as their origin and consequences (pre-history and post-history).

The solution for this task is the mechanism of Active Rules (see, e.g. [Pa01]), which consist of the active rule model and the software components (called Active Rule Engine, see [KU03]) providing the functionality of active rules mechanism.
3. Intelligent data analysis (IDA). The goals of this step is to find out the hidden knowledge in cumulative data and to use these knowledge for improving the quality of data processing in our PCS, i.e. for configuring the set of active rules on the level of data pre-processing. The knowledge is to be extracted in a form of association rules (see [SA01]) describing the regularities in parameters values. IDA process can be commonly described as follows. After a new volume of technical data is received, a set of candidate association rules is generated. Then, the domain expert approves some of these rules, and after that new active rules are generated on the basis of suitable association rules. Also, the functioning of existing active rules is analyzed, and appropriate changes are generated, e.g. the disabling of unused rules, changing their parameters, et ca.
4. Data synchronization. According to the typical architecture of multilevel PCS, the data pre-processing task is to be solved on the base level, while IDA is realized on cluster level, where the data from different base level nodes can be analyzed. So, the task of inter-level data synchronization should be solved.
We presented on the fig.1 the IDE framework for the case of two-tier distributed system. The base level node solves the standard SCADA tasks and also provides the additional functionality connected with the Data Pre-processing stage. The cluster level node is the node of decision making; it can integrate the data from several nodes of base level.

3 Data Synchronization: Relational-to-XML approach

The problems of data synchronization and integration in distributed systems are considered in our framework from such point of view, that the most popular and standard approaches to solving of this task (see, e.g. [HB011]) usually fail when they are used on the unreliable communication channels. In our application domain presented in [Tk01, Tk02, Ku01] we detected such a situation. So, the main goal of designing the special methods for data synchronization was to make it possible to synchronize the updated data only and to make it when the communication channels are available.
The goal of the proposed algorithm is to decompose relational scheme which stores the data to be synchronized into the set of sub-schemes which have to be directly trans-

formed to XML form. The synchronization algorithm receives on its input the list of relational tables (it is meant that they are grouped into relational scheme by relationships) and returns the set of relational sub-schemes.

We will use the following definitions in our algorithm. The relationship of a table N_1 with another table N_2 is called *import relationship* if N_1 contains the primary key of N_2 (or, with other words, when a relationship of the form many-to-one exists). In a similar manner, the relationship of the table N_1 with another table N_2 is called *export relationship* if N_2 contains the primary key of N_1 (when a relationship of the form one-to-many exists).

The information regarding each table is presented in the algorithm in the form of the following tuple: $X = \langle N, I, E, schema, tag, parent \rangle$. In this tuple, N is the name of the table; I,E – the sets of tables, the current table has import and export relationship with, correspondingly; *schema* – file number that will contain the XML data from the current table; *tag* – an XML-element number, that the current table corresponds with; *parent* – a parent XML-element number. Also we use an array M[I], which elements contain count of records in the tables presented in I. The proposed algorithm consists of the following steps:

1. Define S as a set of tables that have to be synchronized.

2. Find S_1 as a set of tables that have neither import nor export relationship. For these tables set the values schema=0, tag=0, parent=0.

3. Find S_2 as a set of tables that have two import relationships, but the number of their export relationships is less then two. Also, tables that are connected with table from a set S_2 with import relationship must have 3 export relationships in total. For these tables we should set the following values: *tag=2, parent=1*. For each table X_i belonging to S_2 we should set the value for *schema:* ($schema_i = i, i = \overline{1, \tilde{S}_2}$), where $\tilde{S}_2$ – the number of elements in S_2.

4. For each table $X_i \in S_2$ find I_i as a set of tables that are connected with import relationship. Set I_i contains 2 tables. For both these tables we set the value *schema=i* (corresponding to their parent), for the 1st table we set *tag=1*, *parent=0*; and for the 2nd table: *tag=3, parent=1*.

5. Remove elements $X_i \cup I_i$ $(\forall i \mid X_i \in S_2)$ from I_j and E_j for $(\forall j \mid X_j \in S / X_i \cup I_i)$. Remove $X_i \cup I_i$, $(\forall i \mid X_i \in S_2)$ from S and S_2.

6. If $\tilde{I}_j < 2$, $\forall j \mid X_j \in S_2$, then delete X_j from S_2. We set the values for X_j: *schema=tag=parent=0.*

7. Define S_3 as a set of tables that have the number of export relationships more than 1. If there are no such tables, go to the step 10. Else set for these tables the values: *schema$_s$=s* ($\forall s = \overline{\tilde{S}_2 + 1, \tilde{S}_2 + \tilde{S}_3}$), *tag=1 ,parent=0*; where $\tilde{S}_3$ –the number of elements in S_3.

8. For each table $X_i \in S_3$ define E_i $(\forall i \mid X_i \in S_3)$ as a set of tables that have export relationships. If for some non-equal *k,j* $E_j \cap E_k = X_l$ holds true (where

$X_l \neq \varnothing, X_l \in S$), and number of export relationships for the table j $\tilde{E}_j$ is less than $\tilde{E}_k$, then remove from E_k the elements belonging to X_l. For the tables belonging to $\mathrm{E_j}$ set the values: *schema=i*(corresponding to its parent), *parent=1*; and *tag=g* ($\forall g = \overline{2, \tilde{E}_i + 1}$). If $\tilde{E}_k < 2$, that means X_k has less then 2 export relationships with other tables, and doesn't belong to a set S_3, then delete X_k from S_3 and we set for it the values: *schema=tag=parent=0*.

9. For each table X_p from E_j check the number of export relationships, if $\tilde{E}_p = 1$ and $\tilde{I}_r = 1$, then for table X_r connected with X_p (for which we have received the values *schema=i, parent=1, and tag=m* on the previous iteration) set the following values: *schema=i, parent=m; tag=g* (where *g* increases on 1 from iteration to iteration, at the beginning it equals with the maximum tag number from all the elements of E_j added by one). Add X_r to the set *U*, where a set *U* is a set of tables that are connected with tables E_j on a first iteration. On the 2nd iteration we add to the *U* a set of tables that are connected with tables that had been received on a first iteration, and so on. Step 9 must be repeated for all the export relationship of received table. Delete from I_i and E_i $\forall i \mid X_i \in S / X_j \cup E_j \cup U$ the elements belonging to $X_j \cup E_j \cup U$ for $\forall j, X_j \in S_3$. Remove X_j from S_3.

10. Check all the other tables in S on their belonging to S_2 . If we have such tables then go to step 2, else go to step 12

11. Define S_4 as a set of tables that have exactly one export relationship. For such tables we should set the following values: *schema=j,* ($\forall j = \overline{\tilde{S}_2 + \tilde{S}_3 + 1, \tilde{S}_2 + \tilde{S}_3 + \tilde{S}_4}$), *tag=1, parent=0*; where $\tilde{S}_4$ – the number of elements in S_4 . A set of tables S_4 defines a set of root nodes for XML documents that has one child node.

12. For each table $X_i \in S_4$ find a table X_p, that has one export relationship. Set for it the values: *iter=1, schema=i, tag=iter+1, parent=iter*. Repeat the step 12 for tables such as X_p until there will be no tables with exactly one import relationship and on each iteration increase the value of *iter* by 1.

13. Add the rest of tables to S_1 and set for them the following values *schema=0, tag=0, parent=0*.

4 Performance analysis: metrical-based approach

When the Data Synchronization Service described in Section 3 had been designed, it was a very important point to guarantee the actuality of data DSS processes. We suggest the simulation of base level subsystems by using the *execution cost* evaluation. Hereinafter we call these modeling as metrics, and the model we receive during this process - as metrical models.

The evaluation of execution cost is based on the system decomposition principle. We split the system on some *functional units*, whose performing cost we can estimate by a

program-analyzer. The system must be decomposed until the received set of functional units represents some elementary operations (e.g., Select statement in Transact-SQL). Thereby we work with maximum level of detailing; it allows making the certain changes to code, basing on the analysis of execution cost for each functional unit in order to achieve the minimum expenses on its execution.
Here we present a small sample of using the metrical modeling. We analyzed the functioning of data processing algorithms implemented in INDB and Active Rule Engine. Due to the lack of space we do not present the intermediate diagrams and calculations. The final expression for execution cost EC (calculated in standard units) is shown in the Fig. 2.

$$EC = 0{,}29 + C * (0{,}03 + p_2 * (M * (0{,}04 + p_3 * 0{,}03) + 0{,}07) + P * *(0{,}04 + p_4 * 0{,}02)) + (0{,}09 + A * (0{,}06 + D * 0{,}01 + p_4 * (0{,}03 + Pn * 0{,}04)))$$

Fig 2. The received metrical model

The following variables were used in the model: number of controllers (*C*); numbers of messages (*M*) and parameters (*P*), which values are provided by controllers; number of active rules (*A);* their average difficulty (*D*); an average number of parameters in an active rule (*Pn*); probability of changes in the message sequence (p_1), average probability of message change (p_2), probability of a fact that current parameter is used in Action section of any Active Rule (p_3); and probability of an active rule triggering (p_4).
By the usage of a profiler we can find the execution time in milliseconds (ms) for application being tested. Than we calculate the coefficient of proportionality and make the transfer from standard units (s.u) to time units with respect to the given hardware-software configuration. For the SCADA system being tested we received: 1 s.u. = 4.618 ms. This result allows calculating the system response time when the values of its configuration parameters are known.

5 Conclusions

In this paper we presented the algorithm which provides the decomposition of a complex relational database scheme to a set of XML documents. This algorithm was designed for the usage in Intelligent Data Engineering framework for distributed multi-level PCS. The performance of the components which realize the Data Collection and Data Preprocessing stages of this framework is estimated with the help of the approach called metrical modelling.

References

[IG01] Information and Installation Guidelines for Advanced Control Systems for Isolated Power Networks. Technical report // The Symposium «Dissemination of the advanced control technologies and SCADA-systems», Ajaccio (France), 2000.
[HB01] Abdelsalam A. Helal, Bharat B. Bhargava „Replication techniques in distributed systems“, Kluwer Academic Publishers, 1996, p. 156.

[Ku01] D.V.Kuklenko, H.C. Mayr, et al. “Web-Based Information Systems For Technological Process Control: Architectural Framework And Software Solutions”, “Problems of programming”, Kyiv, #1-2, 2002 (special bulletin), pp.317-325

[Pa01] Norman Paton, Active Rules in Database Systems. New York, Springer, 1998.

[SA01] R. Srikant, R. Agrawal. Fast Algorithms for Mining Association Rules. Proceedings of the 20th International Conference on Very Large Databases, 1994

[Tk01] M. Tkachuk, H. Mayr, et al. Web-based Process Control Systems: Architectural patterns, Data Models, and Services, Proc. EurAsia-ICT 2002: Information and Communication Technology, Shiraz, Iran, 2002, pp.721-729.

[Tk02] N. Tkachuk, V. Shekhovtsov, et al. An Approach to Efficient Data Handling in Web-based Process Control Systems. Proceedings of the IASTED International Conference on Intelligent Systems and control, ACTA Press 2003, Editor M.H.Hamza. pp.242-247.