COMPUTER SCIENCE
ISSN1407-7493 DATORZINĀTNE 2007-30

TECHNOLOGIES OF COMPUTER CONTROL
DATORVADĪBAS TEHNOLOĢIJAS

MAIN PRINCIPLES OF A NEW CONCEPT OF DESIGNING DATA MANAGEMENT SYSTEMS

DATU VADĪBAS SISTĒMU MODERNO PROJEKTĒŠANAS LĪDZEKĻU KONCEPCIJAS PAMATPRINCIPI

Vadim Zuravlyov, *Riga Technical University, Pernavas 1-2, Riga, LV 1012, Latvia, doctoral student, zuravlov@gmail.com*

Radio Frequency Identification, XML, Physical Markup Language

1. Introduction

The huge quantity of hardware and software decisions for data management systems (such as Enterprise Resource Planning, Customer Relationship Management, smart house etc.) is created and maintained nowadays. There are new problems which can not be solved effectively enough in exists concept of designing of data management systems (such as database, Content Management System etc.).

At the moment there is a necessity for universal system that would allow using and manipulating any resource (from the person, up to a sheet of a paper). At the same time this system should not replace, but improve and fulfill already existing developments. It should allow any system going in production with the minimal changes in adjacent information systems. Therefore, to achieve universality, flexibility, not necessary to use other sources (such as database) was accepted the decision the information (data and business logic) save in resources.

The following examples of the appliance of this concept are: policy's execution control (internal company policy's, labours protection precisions etc.); working process analyzing, modeling and implementation in real time (manufacturing, work process improvement etc.).

The main principles (the basic rules that guides and influences thought) of the new concept (the basic unit of thought) of designing data management systems are described in this paper. The purposes of the given paper to describe one of possible schemes of work of the new concept on its basis represent language for it. The described scheme not obligatory should be the optimum decision, important define a new language (old language modification) and basic rules of a new concept. In the beginning of a paper author chooses technology (in our case it Radio Frequency Identification technology) and considers questions connected with technical solution. After that define how the information is stored (in our case it XML), describes the

structure of stored information (data and business logic). In a final part of article author represent new modification of XML in which it is used object-oriented design.

2. Technical solution

The key element of the new concept is a resource. A resource is any object of the alive or lifeless nature which is involved in working process of information system. All data of a concrete resource and its interrelationship with other objects (or types of objects) is necessary for storing in this resource. Therefore this technology on the basis of which will be created new concept should give this an opportunity.

2.1. Technology

Different technologies can be used to enable necessary functionality, but the author has chosen Radio Frequency Identification (RFID), as the most perspective direction nowadays. RFID (see more in [1]) is an automatic identification method, relying on storing and remotely retrieving data using devices called tags. Basic element of this technology is a tag, it is an object that can be attached to or incorporated into a product, animal, or person for the purpose of identification using radio waves. It is possible to store up to 1 Mb data in each tag. Such size of data allows uniting the list of attributes (data) with functionality (business logic) in resource.

Practical realization of the concept is based on describing of intercommunication interfaces on these tags with other tags, position of this tag in the overall tags network. Technologically it is built as circuit: Tags → RFID scanner → Computer, but nowadays there are already built the mobile devices that replace a circuit: RFID scanner → Computer, so the circuit is represented as: Tags → Mobile Device.

2.2. Data stored type

Further it will be necessary to define how the information is stored in a resource. The leader of information representation standards is XML. That is why the information is stored in resource in PML (Physical Markup Language) – the format modification of XML. PML is described in [2] as "a simple, general language for describing physical objects for use in monitoring and control of a physical environment - particularly through the Internet. Applications include inventory tracking, automatic transaction, supply chain management, machine control and object-to-object communication."

In the paper author represents a new modification of PML, that is been developed for our new concept. The author calls it RPML (Resource Physical Markup Language). Further in the document the author gives examples in language PML, but the places, where syntax coincides with the official version of language PML, will not be separately described (it is possible to find them on official page [2]).

3. The structure of stored information

Further it will be necessary to define how the information is stored in a resource. The leader

of information representation standards is XML. That is why the information is stored in resource in PML (Physical Markup Language) – the format modification of XML. PML is described in [2] as "a simple, general language for describing physical objects for use in monitoring and control of a physical environment - particularly through the Internet. Applications include inventory tracking, automatic transaction, supply chain management, machine control and object-to-object communication."

In the paper the author represents a new modification of PML, that is been developed for our new concept. The author calls it RPML (Resource Physical Markup Language). Further in the document the author gives examples in language PML, but the places, where syntax coincides with the official version of language PML, will not be separately described (it is possible to find them on official page [2]).

3.1. Data

Each resource has unique own features, but it is possible to emphasize some standard kinds of such features:

- Physical – any resource possesses physical properties, it is possible to fix from what material the resource is made, the length of the resource, etc.
- Entity – the resource can have its owner (it can be the person, the organization, the country, etc.), so here such parameters, as a name of the owner, the name of the organization, the address, etc., are registered.
- Location – the coordinates of a resource concerning the reading out device, it can be in 2D (X and Y), also in 3D (X, Y and Z) coordinates.
- Configuration – resource can consists of another resources (e.g. pallet shipment), here the unique numbers of other resources can be listed.
- History – parameters of any kind, that has timing component, when these parameters have been fixed. It is very important to remember, that memory of a resource is limited and consequently it is necessary to limit final number of possible records
- Variables – any program can be stored in a resource, and for its work variables are necessary. There are standard types of variables (such as Integer, String, Date etc.), and also nonstandard such as arrays (defined in PML).

So far the author has described the basic kinds of features, but there are also other, not listed kinds of features. All of this depends on defined tasks and on variants of their solution. It would also be desirable to note, that exist tags that communicate with the carrier. The typical example of such tag is sensors. For example, the thermal sensor control can write down a current condition of temperature in the tag. Also any mechanism (being carrier of tags) can write down on it the information. For example, the printer can write down on the tags the working condition of itself (such as printing, on, off, waiting, error etc.).

3.2. Business logic

The ideology is similar to usual programming languages, and for the best understanding of the paper, analogies to programming language C ++ are specified. In a resource the business logic is presented as the program, in the form of XML.

The executable block of the program, called "task", can be defined as procedure. The very first executable task in a resource is called "main". In the own task there is an opportunity to apply some kinds of operations:

1. Executable – actions which can execute resources. Typical actions: to include, to switch off, to start, etc.
2. Boolean – the Boolean operators, consisting of condition and task that will be executed (or on the contrary not to be executed) if condition is true.
3. Predefined – the built in operators who are predetermined by the system (for example, for definition of distance between two resources, operation "Locate" is used).

The syntax of the task:

```
<task label = "procedure" variable="value">
    ..
    <return type = "variable">value</return>
</task>
```

By the tag "task" it is defined the working block, where the name of the task (procedure) is required to be defined. Then the names of variables with their values are listed. The author wants to emphasize that it is necessary to define those variables which are transferred into "task". Inside of the procedure there are operations. If it is necessary to return any values, then the type of procedure (type) or the name (variable) or the returned value (value) are defined. If this task is used as a condition in verifying operation, then the last value in the last tag "return" is considered.

The syntax of the checking operation:

```
<if label = "name" condition="condition">
    <true>..</true>
    <false>..</false>
</if>
```

By means of tag "if" it is possible to define a condition. Conditions of comparison are set by an alphabetic combination: EQ (=), GT (>), LT (<), GE (=>), LE (<=), NE (<>). If the condition is true, then the working block, that is located in the tag "true", is to be executed. Otherwise, the block of the tag "false" must be executed. Verification operation has a name on which it is possible to identify verification operation (name). If don't exist tags "true" and "false", that will be considered tag "true".

4. Example

So far, the theoretical basis of the new concept of modeling systems of data management was briefly described in the paper. For better understanding the practical usage of the author's developed concept there is described a concrete problem, and solution of it in RPML.

Condition of the problem: it is necessary to organize regulation of temperature inside. Initial data are the conditioner (downturn of temperature), a heater (rise of temperature) and thermo sensor (capable to take temperature).

First of all the author needs to define data that can be used to solve the problem. In thermo sensor there is already been one parameter "Temperature", that changes on demand of data

from the environment. There are also 2 parameters: "Min" (admissible minimum) and "Max" (admissible maximum).

Example:

```
<mat label = "thermosensor">
    <extrude type = "Temperature"> 0 </extrude>
    <integer label = "Min"> 20 </integer>
    <integer label = "Max"> 21 </integer>
</mat>
```

The next step is definition of accessible executable operations. The conditioner and a heater have two executable operations "On" and "Off", that allow switching on and off these devices.

There will be a control block (or in other words – working algorithm) in thermo sensor:

1. If value of the parameter "Temperature" is greater then value of the parameter "Max", then switch off the heater and switch on the conditioner.
2. If value of the parameter "Temperature" is less then value of the parameter "Min", then switch off the conditioner and switch on the heater.

In RPML it looks as:

```
<task label = "main">
    <if label = "Hot" condition = "Temperature GT Max">
            <msr label = "Heater.Off"></msr>
            <msr label = "Conditioner.On"></msr>
    </if>
    <if label = "Cold" condition = "Temperature LT Min">
            <msr label = "Conditioner.Off"></msr>
            <msr label = "Heater.On"></msr>
    </if>
</task>
```

5. Object-Oriented Design

The approach of programming for RPML realization, which has been described so far in the paper, was focused on rules. I.e. actions are made on conditions of the certain rules. But usage of the object-oriented approach gives much more advantages.

As already has been mentioned in the paper, each resource contains RPML description (data and business logic). Interfaces of interaction between different types of resources are very similar. There can be small differences, but as usual they are minimal. Therefore it is important to create library of patterns of resources, or library of classes in OOP terminology. Each unique resource possesses its own features, its external interfaces can be connected with other unique resources, in OOP terminology - it is called an object. Each resource contains an object, and the description of the object class. Depending on concrete realization of the new concept, the library of patterns (library of classes) is used at creation of a network of resources. But the concept does not demand that classes are constantly accessible.

Well known (for example, see [3]), that for object-oriented style the conceptual base is an object model. Let's consider little elements of this model to look, as these elements can be applied in our case. Further the author will consider: abstraction, encapsulation, modularity and inheritance. Others elements of object-oriented style (such as typezation, parallelism, hierarchy, retentivity, etc.), will not be described in this paper. Theoretical basis of OOP is not

described in our paper, it can be found in other sources (for example, see [3, 4]). Further the description of basic elements of object model is given, and features of OOP application for our case are briefly described.

5.1. Abstraction

Abstraction it is simplifying complex reality by modeling classes appropriate to the problem, and working at the most appropriate level of inheritance for a given aspect of the problem. The basic element of our concept is a resource, but from the technical party tag. It means that the level of abstraction needs to be taken depending on the point of view on a task in view. For example we take "shipping pallet" resource. If us this resource interests as object which is necessary for moving on this resource one is placed tag. But if is considered the list of other resources, which is stored in the " shipping pallet " resource, then each entering resource must to have the tag.

5.2. Encapsulation

Encapsulation is described in [4] as “wrapping data and methods within classes in combination with implementation hiding”. In our case any class of resources is considered, how a black box - and on a concrete resource it is possible to influence through it interface (list of accessible attributes and methods), not knowing its internal realization.

For example take the “thermosensor” resource (see in character 4). If it is necessary to change temperature of a placing it is necessary to change attributes "Min" and "Max" of the "thermosensor" resource. On what it will work, what resources will provide this functionality there is no necessity to know to the one who gives such task.

5.3. Modularity

The information about each resource is stored in tag of this resource, and a modularity kind is caused by the concept. But there is an exception to the rules. For example, it is possible to present shipping pallet in which the information about each resource in the pallet is stored. Such modularity’s syntax is:

```
<module>
    <include>element_id</include>
</module>
```

The list of other resources, stored in the “shipping pallet” resource, is described in the "module" tag. Each entering resource is described in a tag "include", where its unique identifier (element_id).

5.4. Inheritance

In object-oriented programming, inheritance is a way to form new classes (instances of which in own concept are called resources) using classes that have already been defined. The new classes, known as derived classes, take over (or inherit) attribute and behavior of the pre-

existing classes, which are referred to as base classes (or ancestor classes). It is intended to help reuse of existing code with little or no modification.

For example, in character 4 of the paper the example of temperature's regulation is given. It is supposed, that "thermosensor" controls "heater" and "conditioner". It is possible to upgrade the "thermosensor" resource, up to "thermosensor_with_mail" where the electronic message is sent at changes of temperature.

The full description of a new class can look like this:

```
<mat label = "thermosensor_with_mail" class = "thermosensor">
    <task label = "main">
            <msr label = "thermosensor.main"></msr>
            <if label = "Hot" condition = "Temperature GT Max">
                    <msr label = "Mail">Temperature down</msr>
            </if>
            < if label = "Cold" condition = "Temperature LT Min">
                    <msr label = "Mail">Temperature up</msr>
            </if>
    </task>
    <mat label = "thermosensor ">
            ..
            <task label = "main">
                    ..
            </task>
    </mat>
</mat>
```

The new resource class "thermosensor_with_mail" is created on the basis of other resource class "thermosensor". The tag "mat" designates the description of a resource class, in parameter "label" the name of a new class of a resource is stored, and the name of a class of a resource on the basis of which the new class of a resource is created is stored in parameter "class". It is visible (from the example), that inside of resource class "thermosensor_with_mail", there is a resource class "thermosensor" (it is more in detail described in character 4 of this document). The task "main" is redefined, in which new functionality was added. If there will be a necessity to implement a new resource class on the basis of "thermosensor_with_mail" then it will place the description "thermosensor_with_mail" inside of itself.

Another resource can use functionality of the inherited class of other resource. In our case, there is an opportunity to define all resources "thermosensor" even if these resources were already improved up to "thermosensor_with_mail", and only inherit a resource class "thermosensor".

6. The practical usage of the new concept

Already represented theoretical material enables to understand on what principles and how the new concept of designing of data management systems works. It is necessary to define a practical component of the given concept.

First of all let us speak about the process of designing the new data management systems. Designing of system with in RPML written documents is very inconvenient, because it borrows a lot of time. The huge quantity of different case tools, that allow simplify visualization of designs, is created nowadays. The process can be presented by using diagrams in well-known case tool, for example: Unified Modeling Language (UML), GRADE-BM,

electric schemes, block schemes, etc. The ideal variant is to adapt already existing software for needs of the new concept.

If the author talks about modeling tools, then the new concept transfers them on a new development level. The opportunity of creating the scheme of the working process becomes possible in real time. It is also possible to analyze it and to make changes in a real time.

In theory there is an opportunity to export data in various modeling tolls. Data are always stored in the form of RPML documents – this is the standard of the new concept. But it is possible to change data in any other software. After changes it is necessary to return them in the same standard. For example, one user can use UML, another user GRADE-BM – so they have an opportunity to work simultaneously with the same project, using favorite's tools.

Conclusion

The new concept of designing of data management systems is described in the paper. It was necessary to develop such concept which allows tracing and controlling any resource in information system.

In paper described one of possible schemes of work of the new concept. In the beginning of a paper author chooses RFID (Radio Frequency Identification) technology (different technologies can be used to enable necessary functionality, but the author has chosen one of them) and considers questions connected with technical solution. Technical solution of the concept is based on describing of intercommunication interfaces on these tags with other tags, position of this tag in the overall tags network.

After that defined how the information is stored (in our case it XML) and described the structure of stored information (data and business logic). The main result is represent a basic rules of a new concept and new language for work with tags (the modification of PML (Physical Markup Language), called RPML (Resource Physical Markup Language) is given). After that author represents RPML in which it is used object-oriented design elements.

Only main principles of the given concept are described in the paper. Such serious questions as the optimal work scheme, safety of data, data exchange with other information systems, integration into other systems – will be considered in the next papers.

Acknowledgements

This work has been partly supported by the European Social Fund within the National Programme "Support for then carrying out doctoral study programm's and post-doctoral researches" project "Support for the development of doctoral studies at Riga Technical University".

References

1. Sandip Lahiri. "RFID sourcebook" // Upper Saddle River, N.J., IBM, London, 2005.
2. The Physical Markup Language / Internet. – http://web.mit.edu/mecheng/pml/
3. Booch G. *"Object-Oriented Design with Applications"* // The Benjamin/Cummings Publishing Company, Inc., Redwood City, California, 1991.
4. Bruce Eckel. "Thinking in Java. Third Edition." // President, MindView, Inc., U.S., 2003.

Žuravļovs V. Datu vadības sistēmu moderno projektēšanas līdzekļu koncepcijas pamatprincipi

Pašlaik Latvijā eksistē vairāki jauni tehnoloģiskie un telekomunikācijas izstrādājumi. Tiek realizētas un ekspluatētas vairāki aparātu un programmatūras risinājumi (ERP, CRM, „Gudrā māja" un citas). Šajā darbā tika aprakstīta koncepcija, kura ļauj uzlabot izstrādājumus. Iebūvēta „nervu sistēma", kas dot iespēju ieverot un vadīt jebkuru resursu (no cilvēka līdz papīra lapai).

Dotā darba mērķis ir aprakstīt dotas koncepcijas realizācijas piemēru. Uz piemēra pamatā aprakstīt tehniskus risinājumus informācijas glabāšanai, (XML dokuments) un struktūru (dati un bizness noteikumi). Kā arī prezentēt PML valodas modifikāciju darbam ar resursiem, izmantojot objektorientētu pieeju Parasti funkcionalitāti iebūvē programmatūrā, un pēc jebkurām neparastām izmaiņām nepieciešams atkal programmēt. Šajā darbā tiek aplūkota viena no iespējam, kā uzlabot šo situāciju – funkcionalitātes pārnese resursā, kas dot iespēju samainīt programmēšanu ar modelēšanu. Tipiskie piemēri koncepcijas izmantošanai ir nolikumu izpildes regulēšana (kompānijas iekšējie nolikumi, darba aizsardzības nolikumi utt.); modelēt, konstruēt un analizēt darba procesus reālā laikā (ražošana, darba procesa uzlabošana utt.).

Zuravlyov V. Main principles of a new concept of designing data management systems

There are a lot of modern IT and telecommunication products in Latvia. Various kinds of technical solutions (such as ERP, CRM, Smart house etc.) have been developed during last years. A new concept, which improves development of such kind of systems by developing "nervous system" controlling any resource (from human being to sheet of paper) is described in this paper.

The purpose of given article to describe a simple example of technical realization of the given concept. On the basis of this example to describe the technical decision, type (XML the document) and data stored structure (data and business logic) storages of the information. And also represent modification PML of language for work with resources using the object-oriented approach.

Usually functionality is built in to software and it is necessary to rebuild software after any crucial business changes. The way how to improve this situation is proposed in this paper. We can transfer business logic (functionality) into resource will more by which consequently will allow replacing programming with modeling. The following examples of the appliance of this concept are: policy's execution control (internal company policy's, labour protection precisions etc.); working process analyzing, modeling and implementation in real time (manufacturing, work process improvement etc.).

Журавлев В. Основные принципы новой концепции проектирования систем управления данными

В Латвии существует большое количество новых технических и телекоммуникационных разработок. Реализовано и эксплуатируется огромное количество аппаратурных и программных решений (ERP, CRM, Smart house и другие). В данной работе описывается концепция, которая позволяет улучшить такие разработки; встраивая „нервную систему", позволяющей наблюдать и управлять любым ресурсом (от человека до листа бумаги).

Цель данной статьи описать простой пример технической реализации данной концепции. На основе этого примера описать техническое решение, тип (XML документ) и структура (данные и бизнес правила) хранения информации. А также представить модификацию PML языка для работы с ресурсами, используя объектно-ориентированный подход.

Обычно функциональность встраивают в программное обеспечение, и после любых нетривиальных изменений необходимо заново программировать. В данной работе рассматривается одна из возможностей для улучшения этой ситуации – перенос функциональности в ресурс, что дает возможность заменить программирование моделированием. Типичные примеры использования этой концепции: регулирование выполнения правил (внутренние правила компании, правила охраны труда и другие); моделирование, конструирование и анализ рабочих процессов в реальном времени (производство, улучшение рабочего процесса и другие).