Proceedings of Informing Science & IT Education Conference (InSITE) 2014

Cite as: Zykov, S. V. (2014). Complex content management: From models to implementations Proce*edings of Informing Science & IT Education Conference (InSITE) 2014* (pp. 349-362). Retrieved from http://Proceedings.InformingScience.org/InSITE2014/InSITE14p349-362Zykov0434.pdf

Complex Content Management: From Models to Implementations

Sergey V. Zykov
National Research University Higher School of Economics
Moscow, Russia

szykov@hse.ru

Abstract

Managing the enterprise content is quite an issue due to its complexity. Moreover, the enterprise content size is growing exponentially. In order to efficiently manage the complex enterprise content, a systematic approach is suggested, which includes formal object-based models and software engineering tools. The paper focuses on the formal models for content management. These formal models are supported by problem-oriented languages and software engineering tools. The approach has been approved by a number of successful enterprise-scale implementations. The implementation areas include oil-and-gas industry, trading, banking, and nuclear power plant construction.

Keywords: enterprise content management, data modeling, data warehouse, legacy system, domain specific language.

Introduction and Related Works

Big data management is a critical issue for state-of-the art enterprises. However, not only size, which is exceeding petabytes in some cases, does matter. Heterogeneity, global distribution, and variable degree of structure (from relational databases to scanned images and video data) complicate efficient data management for complex business objects in the globally distributed environment. Generally, such collections of versatile data objects used in strategic corporate activities are often referred to as the ***enterprise content***. So, an innovative computing model is required for efficient management of enterprise content. Such a model should embrace different content forms, i.e., versatile data and metadata objects used in mission-critical software products and applications. The model (or a whole model set) should support the entire software development lifecycle: from conceptual modeling of the problem domain to enterprise system implementation and maintenance, including component-based expansion. The primary aim of the paper is the introduction into such formal model(s).

The problem of the big data management is even more complex due to heterogeneous nature of the data. The latter may vary from well-structured relational databases to non-normalized objects (such as trees and lists), and weak-structured multimedia data (such as scanned images and photos/videos). The approach presented is focused at more efficient heterogeneous enterprise and

uniform data representation and management procedures. It involves a set of mathematical models, methods, and the supporting CASE tools for object-based representation and management the enterprise content (Zykov, 2010b).

Brute force application of the so-called "industrial" enterprise software development methodologies (e.g., Rational Unified Process, Microsoft Solution Framework) to enterprise content management, without rigorous formal models, does not succeed. The results are usually either unreasonably narrow "single-vendor" solutions or inadequate expenses of time and cost. Also, a number of "pure model-based" approaches to content management exist, such as *Cyc* (Forbus, Birnbaum, Wagner, Baker, & Witbrock 2005), *SYNTHESIS* (Kalinichenko & Stupnikov, 2009) etc. and the others category and ontology-based (Guha & Lenat,1990; Güngördü & Masters, 2003; Lenat & Reed, 2002;, Witbrock et al., 2004). These "pure" approaches do not result in practically applicable (i.e., scalable, robust, ergonomic) implementations, since they are loosely integrated with the software engineering technologies. A number of international and federal research programs prove that the technological problems of heterogeneous enterprise data management are critical (Kanazawa & Fujiwara, 2009).

Another weak point of the content management approaches existing is their focus on problem domain uncertainties rather than partial instance inconsistencies. However, in practice, enterprise content management structure and processes are well-defined due to the enterprise-level standards and policies established and followed. The only exception available is long-term uncertainty of particular objects of the classes, which, in turn, are well-defined themselves. A typical example is a scanned invoice with unclear printed date. However, the type of the content can easily be restored by the undefined value position in the form, after its field-by-field comparison with similar invoices. Thus, the formal model for content management should be focused on well-defined class associations with undefined attributes rather than on the large fuzzy logic dictionaries. So, semantic networks are preferable to ontologies for content management modeling purposes.

The Approach Outline and Background

The approach suggested is based on rigorous formal models and it is supported by software engineering tools, which provide integration to standard enterprise-scale CASE (computer-aided software engineering) tools, commonly used with software development methodologies. The approach eliminates data duplication and contradiction of the enterprise content, thus increasing the robustness of the enterprise software systems. The idea is that the class associations allow duplication and contradiction detection and elimination at the metalevel of conceptual scheme of the data warehouse. The basis for such detection is semantic priority of certain classes and attributes. For instance, the enterprise content has multiple duplications of employee names in HR system, email service, payroll module etc. However, the semantic priority is HR, since the original of the personal data is stored there. So, the rest systems should contain a copy of the employee name form the HR system, or even better, handle its pointer to access the original content element.

The other approach elements include: a set of object models for enterprise content representation and management, engineering tools, which support semantic-oriented enterprise software development and smart content management (Zykov, 2010a, 2011), portal architecture, enterprise solution prototypes and full-scale implementations (Sushkov & Zykov, 2009; Zykov, 2010b).

The model set is based on creative synthesis of fundamental postulates of the following theories: finite sequences (Barendregt, 1984), variable domains (Scott, 1981), categories (Curry & Feys, 1958), semantic networks (Roussopulos, 1976), abstract machines and conceptualization (Wolfengagen, 1999). These foundations are used to declare and categorize content elements, to manage the enterprise content, and to model the software engineering environments behavior. Frames are used for graphical representation of semantic networks (Roussopulos, 1976).

The Business Case and Modeling Sequence

Let us assume a heterogeneous enterprise data warehouse, which contains data objects for the resource management software systems in HR (including employees and departments), documents (including contracts), and multimedia objects (including digital camera pictures). The HR management system is a legacy one; it resides on mainframes, and contains a number of non-normalized data items. For example, the employment history is a dynamically appended list, and it can not be normalized as a relational database structure). Also, the multimedia warehouse contains unstructured audio and video data stored as BLOBs.

Let us assume that the task is to find all employees, who have (at least one) contract in progress, and to publish the contract ID list, the employee names and the employee images in the enterprise portal. Such a task is not a trivial one, because of architectural and structural heterogeneity of data. To solve it, we need the heterogeneous data to be represented in the universal object form on the basis of the models mentioned above. Then we make an object-based query for the data, and represent it in a SQL-like form. However, this is not a common SQL query, since it addresses the heterogeneous enterprise data warehouse objects rather than relational database tuples. To map the model objects into the actual enterprise warehouse data, we will use a semantic network-based ConceptModeller tool (Zykov, 2010a).

Let us summarize the steps to acquire data for the object-based query.

1. Define schemes of the **classes**, where a class is a meta-structure over the heterogeneous enterprise data warehouse. The schemes should contain data and metadata format representation as ordered pairs of <*attribute, type*>. Derived classes are built using certain object-wise predicate-based selection criterion, which models selection of certain objects (i.e., class elements) out of the base class. The class hierarchy is modeled by partial order relationship, generally known as ISA.

2. Define schemes of the **roles** of the class users with respect to (i) user function lists (such as "author", "editor", "content manager", "editor-in-chief" etc.), and (ii) allowed operations over the class objects (such as "create", "delete", "update", "read", "publish"). The role schemes are matrices of <*role_name, operation*>.

3. Define **relationships** (possibly, in a hierarchy form) to model how are objects related to each other.

4. Create enterprise portal **templates** as ordered lists of the data object classes (possibly, nested), attributing the instantiated metadata elements of the classes, and their assigned values. The common form of an enterprise portal template is as follows:

<*…<class_name, <attribute_name, value>,…, <attribute_name, value>>…*>,

where certain attributes may contain a value of "⊥", that means "undefined".

The selection criterion for the elements to be included into the class of a template is done by a predicate-based assignment.

5. **Query** the heterogeneous enterprise data warehouse. The query is a "low-level", "embedded" function, which is modeled by a logical predicate that includes certain objects data classes. A predicate may contain quantifiers (such as "all", "some", "strictly N", "at least N", "not more than N"), logical operations (such as "AND", "OR", "NOT"), and conditional links (such as "prerequisite", "effect"). Direct SQL querying with such quantifiers is a possible option, however, this a non-trivial task due to bulky syntax and content complexity problems.

Compute the enterprise portal **page** by element-wise evaluation. Therewith, the user updates of the objects are managed in accordance with their roles, allowed operations (see step No.2), and

data object types. The selection criteria for the elements to be included into the enterprise portal pages are predicate-based assignments.

Modeling Classes

A class of complex ("multi-dimensional") enterprise content data objects is modeled by a n-arity dataobject relationship:

$K = \Im k:[T_1, T_2,..,T_n]. \forall(a_1:T_1) \forall(a_2:T_2) \ldots \forall(a_n:T_n)(k[a_1,\ldots,a_n] \leftrightarrow \Xi)=\{[a_1:T_1,\ldots,a_n:T_n]|\Xi\},$

where $\Im$ is the description operator (i.e., identifying quantifier meaning "the only").

Thus, the class of objects is a collection of ordered pairs of $(a_i:T_i)$, where ai is the *i*-th class attribute, T_i is the attribute type, and Ξ is the selection criterion in a predicate form:

$\Xi = 1$, *if* $(v_1:T_1,\ldots,v_n:T_n) \in K_n$, and $\Xi = 0$ otherwise.

In case of explicit enumeration of class objects, the selection criteria is a predicate of:

$\Xi = (v_{11}:T_1,\ldots,v_{1n}:T_n) \wedge (v_{12}:T_1,\ldots,v_{12}:T_n) \wedge \ldots \wedge (v_{1m}:T_1,\ldots,v_{nm}:T_n),$

and the class is a finite set of the elements of:

$K = (v_{ij}:T_i)$, $i=1,\ldots,n$; $j=1,\ldots,m$.

In more complex cases, Ξ criterion instantiates a more general type T_i for each element of the K class.

Thus,

$K \subseteq T_1 \times T_2 \times \ldots \times T_n,$

or

K *ISA* $T_1 \times T_2 \times \ldots \times T_n$

and Ξ criterion provides the required identification degree.

The business case in question yields to the following classes:

CONTRACT (Key: int, Name: char, Date: date, Side: char, Font: char);

PERSDATA (Key: int, Name: char, Dept: char, Photo:char, Font: char);

PHOTO (Key: int, Name: char, Height: int, Width: int);

or, in terms of the model,

$C (k, n, d, s, f)$ – contract, where:

k – key ID (a unique number);

n – contract name (a number);

d – contract termination date;

s – contract side;

f – font (to display the text in the enterprise portal);

$E (k, n, d, p, f)$ – employee, where:

k – key ID (a unique table number);

n – employee name (first name, middle initial, last name);

d – employee department;

p – picture (URI reference to a file with the digital employee photo);

f – font (to display the text in the enterprise portal);

$P\ (k, n, h, w)$ – photo image, where:

k – key ID (a unique number);

n – file name (including the path in the URI format);

h – image height (number of pixels);

w – image width (number of pixels).

The ConceptModeller toolkit is used for explicit selection of objects into the class through the visual interface. Implicit selection of objects into the class is done by a query in the same toolkit using a domain-specific language. The example of such a query is given below (see section "Assembling the Portal Page and Querying the Enterprise Data Warehouse").

Let us define function $V \equiv [[\cdot]]$ for the complete evaluation of the class (till the certain value in the web page of the enterprise portal):

$V = V(K) = [[K(a_1,...,a_n)]] = \lambda v_1,...,v_n\ .\ (v_1/a_1,...,\ v_n/a_n).$

According to the approach, evaluation is done in two stages: (i) from class to the object of the page template, and, (ii) from the page template object to the portal page object value.

The first stage is evaluation of the object of portal page template S:

$S = S(K) = [K(K(a_1,...,a_m,a_{m+1},...,a_n)] = \lambda s_1...s_m.(s_1/a_1,...,\ s_m/a_m,\ a_{m+1},...,a_n),$

where:

$0<m<n,$

$a_1,...,a_m$ are the metadata attributes for class $K(a_1,...,a_n)$.

Let us note that the function $S(K) \equiv [\cdot]$ does not necessarily evaluate all class metadata. Also, generally speaking, the types of a_i and of s_i are not necessarily equal. Generally, type checking procedure is required, which includes type inference using ISA hierarchy.

The second stage is portal page object evaluation

$V(S) \equiv [[\cdot]] : V(S) = [[K(a_1,...,a_m,a_{m+1},...,a_n)]] = [S(s_1,...,s_m,\ a_{m+1},...,a_n)] =$

$= \lambda v_{m+1}...v_n.(\ s_1,...,s_m,\ v_{m+1}/\ s_{m+1},...,s_n/v_n),$

where:

$0<m<n,$

$s_1,...,s_m$ are the pre-evaluated metadata attributes of class $K(a_1,...,a_n)$;

$s_{m+1},...,s_n$ are the currently evaluated data and metadata attributes of class $K_1\ (a_1,...,a_n)$ within the context of the object of the template S;

$v_{m+1},...,v_n$ are the values of $s_{m+1},...,s_n$ data and metadata attributes respectively.

Role Assignment and Object Selection

The collection of roles is represented by the ***rights matrix*** of:

$R = R(U, A)$, where:

R is a ***role predicate***, which controls the possible operation(s) on data warehouse object(s);

U is a ***role*** (type) of the enterprise user;

A is the type of operation on the data warehouse object(s).

Let us introduce an instance of the rights matrix $R = R(U,A)$:

$U = \{a_u, r_u, c_u, m_u, e_u\}$, where:

a_u – means "author" ,

r_u – means "editor",

c_u – means "updater",

m_u – means "content manager",

e_u – means "editor-in-chief".

$A = \{a_a, d_a, c_a, r_a, p_a\}$, where:

a_a – means "create" ("add"),

d_a – means "delete",

c_a – means "update" ("correct"),

r_a – means "read",

p_a – means "publish".

The template S instantiates the role predicate by the following rights matrix:

$\acute{R} = R|S = (\acute{r}_{i,j}),\ \forall i=1,...,m;\ \forall j=1,...,n$, where:

$\acute{r}_{i,j} = 1$, in case the user with U_i role has the right to perform the action A_j ,

$\acute{r}_{i,j} = 0$, otherwise.

According to the business case, the rights matrix for $\acute{R}$ template is as follows:

Table 1: Rights matrix

A / U	a_a	r_a	c_a	m_a	e_a
a_u	1	0	0	0	0
d_u	0	1	0	1	0
c_u	1	1	1	1	1
r_u	1	1	1	1	1
p_u	0	0	0	0	1

Selection Criteria for the Objects

Let us remind the selection criterion for class K with n attributes:

$K = \Im k:[T_1, T_2, .., T_n] . \forall(a_1:T_1)\ \forall(a_2:T_2) \ldots \forall(a_n:T_n)\ (k[a_1,\ldots,a_n] \leftrightarrow \Xi) = \{[a_1:T_1,\ldots,a_n:T_n] | \Xi\}.$

For illustrative purposes, let us limit the type system of class attributes by two types: integer number Z, and character string Σ. Let us define class P for the digital images:

$K(P) = \Im k:[k,n,h,w] . \forall(k:Z)\ \forall(n:\Sigma)\ \forall(h:Z)\ \forall(w:Z)(k \leftrightarrow \Xi(P)) = \{[k:Z,n:\Sigma,h:Z,w:Z] | \Xi(P)\}$;

$\Xi(P) = \Xi(k, n, h, w) = (\forall(k_1:Z)\ \forall(k_2:Z)\ P(k_1)=P(k_2) \Leftrightarrow k_1=k_2)\ \&\ (n \in Y)\ \&\ (h>0)\ \&\ (w>0)$,

where:

- the first condition means uniqueness of the key ID k;
- the second condition requires that the full file name n should belong to the domain $Y=$"*http://www...*", which contains only strings with internet locations in URI format (thus, naturally, $Y ISA \Sigma$);
- the third and the fourth conditions require positive values for digital image width and height.

In the latter criterion, we have omitted the natural constraints that the variables *k, n, h, w* should belong to the domains of their respective base types Z and Σ, and and that each attribute of class P should be defined under any valid instantiation (i.e., $k \neq \perp$, $n \neq \perp$ etc.).

In fact, the conditions for $K(P)$ and $\Xi(P)$ define class P for the digital images. Element-wise selection for the rest classes is done similarly (it will require checks whether certain objects belong to the "Date" type, and, possibly, a number of checks for subtypes "Payment name", "Contract side" etc.).

Building Class Relationships and Hierarchy

To build class associations, it is enough to build ***relationships*** (i.e. special purpose classes, which associate certain objects of the base classes) with the control predicates. Therewith, it is possible to consider just the case of describing associations for two variables only, which are key attributes of the base classes.

Also, we can define only the binary relationship R^2 for the two classes: K′ and K″ (the other relationships can be built on the basis of the binary relationship using induction):

$R^2 = R(K', K'') = \Im r:[k',k''] . \forall(k':T')\ \forall(k'':T'')\ (r[k',k''] \leftrightarrow \Psi) = \{[k':T',k'':T''] | \Psi\}$,

where:

- $k':T'$, $k'':T''$ are the key attributes of K'_m and K''_n classes, which have types of T' and T'' respectively;
- Ψ is the criterion of object belonging to relationship R^2.

Thus, the complete relationship has the type of

$R:[T'_1,\ldots,T'_m,T''_1,\ldots,T''_n]$, where $K':[T'_1,\ldots,T'_m]$ and $K'':[T''_1,\ldots,T''_n]$.

In case of key attribute coincidence (e.g., during an operation similar to relational database join) the attributes are not doubled in the resulting relationship. In general, the definition for a binary relationship is similar to that of class (the definition for associating its objects by the key attributes is given above).

A possible graphical interpretation of the relationship is a functional frame (Roussopulos, 1976),

with two characteristic sub-frames of the type “attribute – value”, linked by a predicate function with the name, which corresponds to the relationship name.

According to the business case, let us model the relationship “Process” between the classes “Employee” and “Contract” in the situation “Employee processes Contract”.

The relationship can be represented as follows:

$R(C,E) = \Im r:[k_C, k_E]. \forall(k_C:Z)\ \forall(k_E:Z)\ (r[k_C,k_E] \leftrightarrow \Omega) = \{[k_C:Z, k_E:Z] \mid \Omega\}$,

where:

k_C is the key attribute k of class C,

k_E – is the key attribute k of class E,

and the essential inclusion takes place:

$R(C,E)\ ISA\ (k_C:Z, n_C:Z, d_C:\Delta, s_C:\Sigma, f_C:\Sigma, k_E:Z, n_E:\Sigma, d_E:\Sigma, p_E: Y)$.

So, the criterion Ω of the object belonging to the relationship $R(C,E)$ is:

$\Omega = ((k_C, n_C, d_C, s_C, f_C, k_E, n_E, d_E, p_E) \neq \perp)$ or, in short: $\Omega = ((k_C, k_E) \neq \perp)$.

Therewith, similar to the previous definitions, the natural constraints for the attributes of the classes, which make the relationship, should be satisfied.
Concerning the type of the variable d_C, it can be inferred from a more general string (Σ) to a more specific “Date” (Δ), and the type of f_C – to a finite set of font names $\Theta = \{$“Arial”, “Courier”, “Times”, ...$\}$. The type of class Y corresponds to internet location of data object in the URI format.

Building the ISA Hierarchy for the Classes

ISA hierarchy for the classes is:

$K_1\ ISA\ K_2\ ISA\ \ldots\ K_n\ ISA\ K$,

where:
K_i is a class, $i=1,\ldots n$,

K is a type, i.e. a generic (base, system) class.

The ISA hierarchy for arbitrary classes, both built-in and user-defined, which are used for data and metadata modeling, is built in the similar way.

According to the business case, let us model a hierarchy for an organizational structure of an enterprise. A possible representation in terms of the approach concerned is as follows:

$K_0\ ISA\ K_1\ ISA\ K_2\ ISA\ K_3\ ISA\ K_4\ ISA\ K_5\ ISA\ K_6 \equiv K$, where:
K_0 is an “Employee”,
K_1 is a “Work group”,
K_2 is a “Section”,
K_3 is a “Department”,
K_4 is a “Sector”,
K_5 is a “Company”,
$K_6 \equiv K$ is an “Enterprise”.

Assembling the Portal Page and Querying the Enterprise Data Warehouse

Let us remind that the template S for each class K is:

$S(K) = [(K(a_1,...,a_m,a_{m+1},...,a_n)] = \lambda s_1...s_m . (s_1/a_1,..., s_m/a_m, a_{m+1},...,a_n)$.

In general, an enterprise portal page template is a list of $S = S(K_1,...,K_n)$ classes. The first template instantiation results in (partial) metadata evaluation.

According to the business case, the enterprise portal page template is:

$S = S(P,C,E)\ \mathfrak{I}s:[\ k_P,n_P,h_P,w_P,k_C,d_C,f_C,k_E,n_E,p_E,f_E]$.
$\forall(k_P{:}Z)\ \forall(n_P{:}\Sigma)\ \forall(h_P{:}Z)\ \forall(w_P{:}Z)\ \forall(k_C{:}Z)\ \forall(d_C{:}\Delta)\ \forall\ (f_C{:}\Sigma)\ \forall(k_E{:}Z)\ \forall(n_E{:}\Sigma)\ \forall(p_E{:}\Sigma)\ \forall(f_E{:}\Sigma)$
$(s[k_P,n_P,h_P,w_P,k_C,d_C,f_C,k_E,n_E,p_E,f_E] \leftrightarrow \Psi) = \{[k_P,n_P,h_P,w_P,k_C,d_C,f_C,k_E,n_E,p_E,f_E] \mid \Psi\} =$
$= \lambda s_{hP}\, s_{wP}\, s_{fC}\, s_{fE}.\ (k_P,n_P,s_{hP}/h_P,s_{wP}/w_P,k_C,d_C,s_{fC}/f_C,k_E,n_E,p_E,s_{fE}/f_E)$,
where:

$\Psi=(R(E,C)\neq\perp)\&R(E,P\neq\perp)\&(s_{hP}\ ISAZ)\&(s_{wP}\ ISA\ Z)\&(s_{fC}\ ISA\ \Sigma)\&(s_{fE}\ ISA\ \Sigma)\&(s_{hP}>0)\ \&(s_{wP}>0)$;
s_{hP} = *"150"* is the image height (in pixels);
s_{wP} = *"75"* is the image width (in pixels);
s_{fC} = *"Arial 12 Bold"* is the font name for general purpose text items in the page;
s_{fE} = *"Arial 12 Bold"* is the font name for general purpose text items in the page.

The above instance contains an abbreviated criterion of the object belonging to the template. The real-world criteria are often more complex; they may include other variables of *P, C, E* classes as well as the variables of the other problem domain classes. The templates for the image and the font usually assume such specific parameters as HTML tags for content decoration (such as FONT, IMG etc.) with a number of attributes. Instead of the HTML tags, the above instance contains other parameters; this has been done to conquer complexity of the illustration.

Querying the Data Warehouse

Let us remind the query to the data warehouse: Select all employees (Name, Reg. Number, Photo), who have at least one contract in progress.

The SQL query to the enterprise data warehouse is:

```
SELECT
      EMPLOYEE.EmpFio,
      EMPLOYEE.EmpTab#,
      CONTRACT.Contr#,
      PHOTO.PhotoEmpUrl
FROM
      EMPLOYEE,
      CONTRACT,
      PHOTO
WHERE
(CONTRACT.ContrFinDate < Date("Today"))
&(CONTRACT.ContrRespEmpTab#=EMPLOYEE.EmpTab#)
& (EMPLOYEE.EmpPhoto# = PHOTO.Photo#)
```

Let us evaluate the enterprise portal page template:
$V = V(S(P,E,C)) = \mathfrak{I}v:[k_P,n_P,k_C,d_C,k_E,n_E,p_E]$.
$\forall(k_P{:}Z)\ \forall(n_P{:}\Sigma)\ \forall(k_C{:}Z)\ \forall(d_C{:}\Delta)\ \forall(f_C{:}\Sigma)\ \forall(k_E{:}Z)\ \forall(n_E{:}\Sigma)\ \forall(p_E{:}\Sigma)\ \ (s[k_P,n_P,k_C,d_C,k_E,n_E,p_E] \leftrightarrow \Phi) =$

$\{[k_P,n_P,k_C,d_C,k_E,n_E,p_E] \mid \Phi\} = \lambda v_{kP},v_{nP},v_{kC},v_{dC},v_{kE},v_{nE},v_{pE}.$
$(v_{kP}/k_P,v_{nP}/n_P,s_{hP},s_{wP},v_{kC}/k_C,v_{dC}/d_C,s_{fC},v_{kE}/k_E,v_{nE}/n_E,\ v_{pE}/p_E,s_{fE})$,
where:
$\Phi = (R(E,C) \neq \perp)\ \&\ (R(E,P) \neq \perp)\ \&\ (v_{kP}\ ISA\ Z)\ \&\ (v_{nP}\ ISA\ \Sigma)\ \&\ (v_{kC}\ ISA\ Z)\ \&\ (v_{dC}\ ISA\ \Sigma)\ \&$
$(v_{kE}\ ISA\ \Sigma)\ \&\ (v_{nE}\ ISA\ \Sigma)\ \&\ (v_{pE}\ ISA\ \Sigma)\ \&$
$(v_{dC} <$ "20/10/2006") $\&\ (v_{nP} = v_{pE})\ \&\ (v_{rC} = v_{kE})$
is the criterion of the page belonging to the template,
and:
v_{kP} is the key ID of the photo image;
v_{nP} is the employee photo image (the URL of the file);
v_{kC} is the key ID of the contract;
v_{dC} is the date of the contract termination;
v_{kE} is the key ID of the employee;
v_{nE} is the name of the employee;
v_{pE} is the employee photo image (the URL of the file).

In the business case, the data warehouse query predicate Φ requires only its arguments to be defined and well-typed (i.e., matching the enterprise portal page template elements, such as the contracts, which have not been processed by the employees). The extra constraints, in case they are required, are modeled by quantifiers of *[N]*, *[>N]*, *[<N]* etc., the essence of which is explained below (see section Role Assignment and Object Selection).

Special Cases of Content Publishing and Legacy Systems

Relation-based object interaction with computing environment for content publishing

The relation-based object interaction is implemented by relationships of styling and publishing, which correspond to building template S, which is based on classes $K_1,\ldots,K_n$, and on subsequent instantiation of the enterprise portal page. The process is modeled by a triplet of roles<*a, o, d*>, where

- *a* = "agent" is a subject performing the action,
- *o* = "object" is an entity, upon which the action is performed,
- *d* = "destination" is an addressee (receiver) of the action.

The above relationships can be reduced to the binary relationships, which have been discussed above. Therewith, no more details are given here to save space. The above frame-based diagrams may be mapped into respective UML collaboration diagrams.

The relationship schemes for the styling and publishing operations are:
$C = C\ (R(r_{11},\ldots r_{kl}),\ K(a_1,\ldots,a_m),\ S(s_1,\ldots,s_n))$;
$P = P\ (R(r_{11},\ldots r_{kl}),\ S(s_1,\ldots,s_n),\ V(v_1,\ldots,v_q))$,
where:
C is styling (the operation of getting a template by the object class);
P is publishing (the operation of evaluation of the template to obtain the enterprise portal page content);
$R = (r_{11},\ldots r_{kl})$ – user rights matrix;
$K = (a_1,\ldots,a_m)$ – class;
$S = (s_1,\ldots,s_n)$ – template;
$V = (v_1,\ldots,v_q)$ – portal page.

Generally, the user rights matrices for styling and publishing are different. The styling template description, in general, may contain several classes; the same holds true for publishing template description.

Situational Modeling for Legacy Systems

Let us consider the query: "Every employee is busy in less than *n* projects", an alternative version of which is: "Every employee is employed in less than *n* companies":

$\forall s\ \forall p_i, i=1,\ldots,n.(R(s,p_i) \in \{0,1\}) \& (\forall s\ \forall p_i,\ i>n.(R\ s,p_i)=0))$

where:
s is an employee;
p_i is a project (company);
R – "busy" ("employed").

The formal model for the constraint: "In any company there is employed only one employee assigned personal ID n":

$\forall s_1\ \forall s_2\ \forall c\ \forall t.(R(s_1,c) \& R(s_2,c) \& N(s_1,t) \& N(s_2,t)) \Rightarrow s_1=s_2,$

where:
$s_1,\ s_2$ are employees;
c is a company;
t is a personal ID;
R is to be employed;
N is to be assigned.

The above constraints lack the natural requirements for the modeled objects belonging to problem domain classes and basic data types; this is done for the sake of intuitive transparency.

Implementation Examples

Portal-Based Implementation: ITERA Oil-And-Gas Group

The approach has been approved by a number of internet and intranet portals in ITERA International Group of Companies. The enterprise warehouse supports integrated storage of content, including data and metadata business objects. During the design stage, problem domain model specifications are transformed by the ConceptModeller software development tool to UML diagrams, then into target IS and enterprise content warehouse storage schemes. The portal design scheme is based on a set of the data models that integrate object-based enterprise content management. A toolkit has been implemented including ConceptModeller visual problem oriented software engineering tool and the content management system. The full-scale enterprise portal has been customized for content management and implemented in a 10,000 staff enterprise. Portal architecture has been designed, implemented and customized according to technical specifications outlined by the author and tested for several years in a heterogeneous enterprise environment. Advanced frame-based personalization and content access level differentiation substantially reduce risks of the enterprise data damage or loss. Due to the approach, costs of enterprise content management, maintenance and integrity support have been essentially reduced, while portal maintenance, customization and performance tuning have been simplified.

Domain-Driven Messaging System for a Distributed Trading Company

A trading corporation used to commercially operate a proprietary Microsoft .NET-based message delivery system for information exchange between the headquarters and over a hundred of dis-

tributed local shops. The original system was client-server based. The old version of the client included a local database and a Windows-based messaging service, while the server side consisted of a Web service and central database. Thus, all the shops client-side software had to be reconfigured and verified manually in case of any update. The project challenges were: complicated client-side code refactoring; difficult error localization/reduction; inadequate documentation; and decentralized configuration monitoring/management for remote shops. The approach used frame-based object representation and Domain Specific Languages (DSL) (Cook, Jones, Kent, & Wills, 2008; Evans, 2003) development for business objects management. The model set helped to conquer problem domain complexity, to structure the data, and to treat data representation and manipulation uniformly. The DSL scope included rules/parameters of message transfer, and new types of messages. Each retail shops had its own configuration instance, which made the client-side message processing/transfer structure. The approach included DSL-based semantic network object model (Fowler, 1997; Hohpe & Woolf, 2003). The DSL modeled system dynamics in terms of object metadata messages and states (i.e., configurations and manipulation rules). To define operations with the messages (such as transformation and routing), a set of high-level templates was developed. Other templates were created for system reconfiguration, server interaction, etc. Channels were used for message management. In the frame-like graph of map messaging, templates were represented as nodes, while channels were the arcs between certain templates. Based on DSL classes, messaging maps were built and used for future parsing to generate system configurations. At this stage, DSL syntax and semantics were built. DSL-based refactoring resulted in an enterprise trade management system with transparent configuration management (and a dramatically simplified change management), rigorous object-based model, and efficient messaging. The approach made the proprietary system an open, scalable, maintainable, and easily customizable solution. Currently, all the shops client-side software can be reconfigured and verified automatically in case of any update, based on the DSL.

The Air Transportation Planning System for Russian Central Transportation Agency

An air traffic planning system architecture had been developed. The problem was to add remote access to the planning data. The previous solution was based on a legacy TAXXI-Baikonur technology, which involved component-based visualized assembling of the server application, legacy (VCL-type) library components, and a "thin" XML browser at the client side. According to the State Program of Planning System Updates, the Main Air Traffic Management Centre was going to create the new remote access solution. The internet-based architecture was going to be implemented in Java technology and to operate on the Apache web server platform. The solution was supposed to query Oracle-based data centre, process the query output and retrieve the results of the air traffic planned capacities to an intuitive and user-friendly interface. The practical application of the approach is the global enterprise-scale integrated system, which is providing a uniform and equal information access to all of the international air traffic participants. The similar globalization processes are underway in Europe and the U.S.A. The frame-based approach is going to unify the issues of the architecture updates and application migration in Russia. The approach is going to simplify the integration challenges of the global air traffic management software solution.

6D-Modeling for Nuclear Power Plants

Another challenging aspect of the approach implementation is related to high-level template-based software re-engineering for nuclear power plants (NPP). To provide worldwide competitive level on the NPP production, it is necessary to meet the quality standards throughout the lifecycle, high security under long-term operation, terms-and-costs reduction for new generation facilities development. Each stage of the NPP lifecycle is mapped into a set of fame-based business pro-

cesses, where people and enterprise systems interact. Heterogeneous nature of the data objects, and millions of units, make NPP a high complexity information object. The aggregated NPP model is often referred to as a 6D model, which includes 3D-geometry, time and resources for operating the plant. Since mechanisms for information searching, scaling, filtering and linking, should provide complete and non-contradictory results, the information models should have well-defined semantics. The frame-based approach assists in rigorous multi-level abstractions required for such a complex problem domain modeling. While a single information model can be derived out of a single system, the 6D model should combine information models of a number of systems. The methodology for 6D model suggests portal-based system integration, which can be based on a single platform capable of entire lifecycle support. Information model development assumes monitoring system state changes and their influence to the other parts of the system. This helps to immediately react on critical issues in NPP construction, which can be used for decision making. A wrong decision would be made otherwise under incomplete or incorrect information. The approach assists in rigorous constraint-based data consistency tracking.

Conclusion

The formal model-based approach to enterprise content management has been developed. Practical application of the approach resulted in implementation of heterogeneous enterprise-level software systems, which integrate state-of-the-art and legacy components, well-structured and weak-structured content. The advantages of the approach include uniform modeling of the contemporary and the legacy systems content, including the non-relational content, such as hierarchical, network-based, and non-normalized databases. This is due to the object-based nature of the approach to content modeling and management. Support of the model by software engineering tools boosts content management productivity and considerably reduces the enterprise system reaction time. Another benefit of the approach is straightforward declaration and management for weak-structured and multimedia-based content objects, such as digital images and video clips.

The implementation of the approach is based on the architecture of web portals, which manage the heterogeneous enterprise content. The approach resulted in successful implementations for ITERA Group with around 10,000 employees in nearly 150 companies of over 20 countries. The systematic approach to enterprise content management provides integration with a wide range of state-of-the-art software engineering tools and enterprise standards. Other implementations include: air transportation planning system, messaging system for a trading enterprise, nuclear power plant and banking solutions. Since each of the implementations is a domain-specific one, the system cloning process is not straightforward. Instead, it requires certain analytical and software re-engineering efforts. However, in case of content object heterogeneity the approach results in substantial term-and-cost reduction of at least 30%.

References

Barendregt, H. (1984). *The lambda calculus* (rev. ed.), Studies in Logic, 103. Amsterdam: North Holland.

Cook, S., Jones, G., Kent, S., & Wills, A. C. (2008). Domain-specific development with visual studio DSL tools. Pearson Education, Inc.

Curry, H., & Feys, R. (1958). *Combinatory logic* (Vol.1). Amsterdam: North Holland.

Evans, E. (2003). *Domain-driven design: Tackling complexity in the heart of software*. Addison Wesley.

Forbus, K., Birnbaum, L., Wagner, E., Baker, J., & Witbrock, M. (2005). Combining analogy, intelligent information retrieval, and knowledge integration for analysis: A preliminary report. *Proceedings of the 2005 International Conference on Intelligence Analysis*.

Fowler, M. (1997). *Analysis patterns: Reusable object models*. Addison Wesley.

Guha, R., & Lenat, D. (1990). *Building large knowledge-based systems: Representation and inference in the Cyc project*. Addison-Wesley.

Gungurdu, Z., & Masters, J. (2003). Structured knowledge source integration: A progress report. In *IKIMS* 2003, Cambridge, MA, USA.

Hohpe, G., & Woolf, B. (2003). *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison Wesley.

Kalinichenko, L., & Stupnikov, S. (2009). Heterogeneous information model unification as a pre-requisite to resource schema mapping. In *ITAIS 2009*, Springer, 373-380.

Kanazawa, S., & Fujiwara, M. (2009). R&D trends for future networks in the USA, the EU, and Japan. *NTT Technical Review, 7*(5), 1-6.

Lenat, D., & Reed, S. (2002). Mapping ontologies into Cyc. In *AAAI CWOSW 2002*, Edmonton, Canada.

Roussopulos, N. (1976). *A semantic network model of databases.* Toronto Univ.

Scott, D. (1981). *Lectures on a mathematical theory of computations*. Oxford Computing Laboratory Technical Monograph. PRG-19.

Sushkov, N.; & Zykov, S. (2009). Message system refactoring using DSL. In *CEE-SECR\'9209*, Moscow, Russia, 153-158.

Witbrock, M., Panton, K., Reed, S. L., Schneider, D., Aldag, B., Reimers, M., & Bertolo, S. (2004, November). Automated OWL annotation assisted by a large knowledge base. In *Workshop Notes of the 2004 Workshop on Knowledge Markup and Semantic Annotation at the 3rd International Semantic Web Conference ISWC2004* (pp. 71-80).

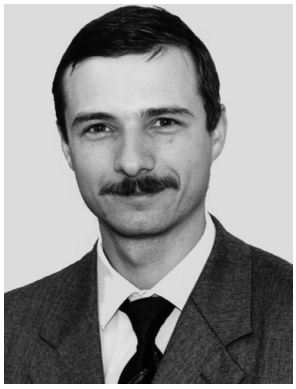
Wolfengagen, V. (1999). Event driven objects. In *CSIT'99*, Moscow, Russia, 88-96.

Zykov, S. (2010). Concept modeller: A frame-based toolkit for modeling complex software applications. In *IMCIC 2010*, Orlando, FL, USA, 468-473.

Zykov, S. (2011). Pattern-based development of enterprise systems: From conceptual framework to series of implementations. In *ICEIS 2011*, Beijing, China, SciTePress, 4, 475-478.

Zykov, S. (2010). Pattern development technology for heterogeneous enterprise software systems. *Journal of Communication and Computer, 7* 4), David Publishing Co., 56-61.

Biography



Dr. Sergey V. Zykov was born in Moscow in 1971, M.Sc. in Computer Science from Moscow Engineering Physics University (MEPhI, Russia, 1994). He holds a PhD in Computer Science from MEPhI (2000, formal models and methods of ERP integration). He has a 20-year experience in IT, including 2 years as Vice-CIO of ITERA Group. He has a Certified Web Professional certificate in web design and e-commerce. He has a 20-year experience in teaching computer science and software engineering, and a mentor certificate from Carnegie Mellon University. He holds a PGCert in Higher Education from the London School of Economics.

Currently, he is an Associate Professor of Higher School of Economics, Moscow Aviation Institute, Moscow Engineering Physics University, Moscow Institute of Physics Technology, and Innopolis University. He authored 10 books and over 100 papers on computer science and software engineering. Primary research fields: enterprise software development, enterprise application integration, data modeling, web portals.