International Journal

INFORMATION TECHNOLOGIES & KNOWLEDGE

2009 Volume 3 Number 1

International Journal
INFORMATION TECHNOLOGIES & KNOWLEDGE
Volume 3 / 2009, Number 1

IJ ITK is official publisher of the scientific papers of the members of the ITHEA International Scientific Society

IJ ITK rules for preparing the manuscripts are compulsory.
The **rules for the papers** for IJ ITK as well as the **subscription fees** are given on ***www.ithea.org***
The camera-ready copy of the paper should be received by e-mail: info@foibg.com.
Responsibility for papers published in IJ ITK belongs to authors.

General Sponsor of IJ ITK is the **Consortium FOI Bulgaria** (www.foibg.com).

International Journal "INFORMATION TECHNOLOGIES & KNOWLEDGE" Vol.3, Number 1, 2009

Edited by the **Institute of Information Theories and Applications FOI ITHEA®**, Bulgaria,
in collaboration with the **V.M.Glushkov Institute of Cybernetics of NAS, Ukraine**,
and the **Institute of Mathematics and Informatics, BAS, Bulgaria**.

Publisher: **ITHEA®**
Sofia, 1000, P.O.B. 775, Bulgaria. www.ithea.org, e-mail: info@foibg.com
Printed in Bulgaria

ISSN 1313-0455 (printed) **ISSN 1313-048X (online)** **ISSN 1313-0501 (CD/DVD)**

METAMODELLING AND MULTILEVEL METADATA AS A BASIS OF TECHNOLOGY THAT IS INTENDED FOR DEVELOPMENT OF ADAPTABLE INFORMATION SYSTEMS

Lyudmila Lyadova

***Abstract**: Methods of creation and development of distributed information systems dynamically adapted to varying requirements of users and service conditions are considered. Described means are based on use of multilevel models. These models represent various parties of system functioning and system domain with various levels of abstraction and from different points of view. System functioning is based on the models interpretation. This approach is realized in the CASE-technology METAS intended for maintenance of all life cycle of dynamically adaptable systems. The basic levels of the system description are the logical level (the system objects description with data domain terms), the physical level (the description of data presentation in a database) and the presentation level (the description of user interface). All models can be changed during system functioning. New models can be developed on the basis of these models. Now the model of reporting, the Web-model and the business-processes model are created. Adaptation possibilities are based on various means of model development such as language toolkit, tools of data re-structuring, means of generation and adjustment of user interface and so on. Software tools of external data import, means of replication of data and models, tools of program components integration and document management subsystem are included in CASE-system too. This technology is based on DSM and DSL: domain specific languages are used as tools for development of domain models. Objects attributes, operations and business rules are described via DSL. The language toolkit is used for DSL creation. It is the base for the multilevel metamodelling. In such a way development of DSL for the specific domain is the first step of information system creation.*

***Keywords**: Adaptable information systems, CASE-technology, metadata, metamodelling, domain-specific modeling, DSM, domain specific languages, DSL.*

***ACM Classification Keywords**: D. Software: D.2 Software Engineering: D.2.2 Design Tools and Techniques – Computer-aided software engineering (CASE); D.2.11 Software Architectures – Domain-specific architectures; D.2.13 Reusable Software – Domain engineering, Reuse models.*

Introduction

The *adaptability* (ability of systems to adapt to changes of environment and functioning conditions) is one of the most important requirements shown to information systems (IS) of different functions. This concept is sufficiently wide: adaptability includes such interrelated nonfunctional requirements as ability to development, flexibility, expansibility, interoperability, etc. [4, 15, 20]. Such wide nature of adaptability makes this concept interesting to research. This property of information systems is also critical in practice of information system development. Adaptability defines efficiency of investments to implementation and introduction of information systems, their operation and maintenance. It guarantees "survivability" of information systems. *Adaptability* is the characteristic defining ability of system to development according to needs of their users and business.

We distinguish among concepts of adaptive and adaptable systems:

- *adaptable systems* are the "easily changeable" systems including means, which would provide their adjustment for new requirements and conditions dynamically (during operation), would facilitate their support;
- *adaptive systems* are systems which change the behaviour automatically according to the changes occurring in their environment ("context"), they are adjusted to changes of environment without application of any means of "manual" adjustment.

Adaptation of information systems is a process of their adjustment for varying functioning conditions and requirements of users and business processes both during creation of new systems, and at support existing. It is possible to consider this iterative process as the major part of information system life cycle.

It is clear that adaptation process needs adequate tools. There are some *general approaches* to development of adaptable information systems. We consider here appropriate technologies of information system development allowing adaptation possibilities during all information system life cycle.

A *software development environment* consists of a platform of tools and infrastructure, a program under development, people that are involved, and a processes those guide and direct all activities. Typically, however, software development environment is used in the sense *integrated development environment* (IDE) which only covers a certain part of the whole process and of the people involved, for example, the programming activity and the programmers. Participation of experts (non-programmer, specialists in Information system domain) in information system development and maintenance is very important for efficiency of life cycle processes. But it is possible only with use of appropriate tools. As software projects become increasingly complex and development processes extend beyond simply the write-compile-execute style of working. The complex interactions of different people in various roles should be managed and their work should be integrated seamlessly into the overall development.

Generally accepted principles of modern technologies of information system development are based on modeling [1, 2, 5, 7, 14]: analysts create models of system domain and its environment and then these models are realized as data and programs by database designers and programmers. All requirements of business and needs of users can be included to these models and must be taken into consideration by development.

Metamodeling and Technologies of Information Systems Development

Model-driven engineering (MDE) is a software development methodology. It focuses on creating *models* (abstractions more close to some particular domain concepts rather than computing (or algorithmic) concepts). This methodology is targeted at increase productivity by maximizing compatibility between systems, simplifying the process of design, and promoting communication between specialists working on the system development. A modeling paradigm for MDE is considered effective if its models make sense from the point of view of the user and can serve as a basis for implementing systems. The models are developed through extensive communication among product managers, designers, and members of the development team.

In MDE the model is built *before* or *parallel* to the Information system development process. Recently the model is created mostly manually, but there are also attempts to create the model automatically: for instance, model can be built as result of documents analysis and text mining (analysis of technical documentation or instructions, job descriptions and etc.). The model can be constructed from the completed system. The data structure and program code are the models of system too. Therefore one important way to create new models is by model transformation. For example, new model can be taken out of the source code; new model can be extracted from data scheme. Model transformations are executed with using languages like DSL (Domain Specific Language).

The first tools to support MDE were the *Computer-Aided Software Engineering* (CASE) *tools* entered the market in 80s. The CASE-tools and various means of augmented Integrated Development Environments have delivered great value to developers and managers now.

The best known MDE initiative is the *Model-Driven Architecture* (*MDA*), which is a registered trademark of Object Management Group (OMG). There are some other projects of development of MDE-oriented tools.

The main idea of *domain-driven design* (*DDD*) is to concentrate attention at the heart of the application, focusing on the complexity that is intrinsic to the business domain itself. The core domain (unique to the business) is distinguished from the supporting sub-domains. Domain-driven design consists of a set of patterns for building enterprise applications from the domain model out. Applying them together will allow building systems that genuinely meet the needs of the business.

The adaptation means can be based on use of the *dynamic programming languages* (DPL), supporting possibility of redefinition of programs and data structure, modernization of programs via extension of their functional components and etc.

Domain-Specific Languages (DSL) are developed for modeling of various subject domains and sub-domains. The idea of using a domain-specific language has gained a lot of interest in the last decade. The use of domain-specific languages has been shown to raise the level of abstraction, increase productivity and ease maintenance and evolution in software development. Using DSL allow to involve non-programmers in information system development at the different stages of Information system implementation process.

Successful development of a modern enterprise system requires the convergence of multiple views and experience: business analysts, domain experts, database experts and developers with different kinds of expertise, interaction designers – all take part in the process of development such a system. The different products with various functionalities must be integrated to create a running system.

Domain-specific multimodeling (DSMM) is a development paradigm where each view is made explicit as a separate domain-specific language (DSL). Every participant of the system development process has a particular language tailored to solve problems specific to its view on the information system. Development of and tooling for a single DSL is well-studied, but the tools of interplay of different languages in a single system are created now. Multiple DSL are required when moving from simple examples to real enterprise applications. Therefore methods and tools support is needed if multiple DSL development is to succeed [8, 12].

The maximum flexibility at information system development is obtained, if system work is under use of models operating its behaviour which can be changed in the system functioning. Models can be organised at various levels of abstraction and platform independence that provides peak efficiency of development process and possibility of models transformation.

Development of the adaptable systems assumes using of the appropriate tools supporting the demands to system properties. Thus, it is possible to consider that property of adaptability is necessary not only for developed systems, but also for means applied for their creation.

The approach to the development of the adaptable information systems based on creation and interpretation of multilevel models and CASE-system based on this approach are presented it this paper.

The model is a “substitute” object of “original” object. The model is in certain conformity with the original system and provides representation about its some properties. The system model is the abstract description in some formal language of system characteristics important from the point of view of the purpose of modeling. The model describes system behaviour too. It is impossible to be limited to creation only one model at system development process. If system is complex and the number of its characteristics is large then describing system by one model will lead to its extreme complexity. The best approach to the development of any not trivial system is to use set of

several models which can be almost independent from each other and will allow to make accents on the different parties of system at the decision of various problems of maintenance of its life cycle.

The models can be generally divided into following kinds:

- *static models*, describing structural properties of systems;
- *dynamic models*, representing behavioural properties of systems;
- *functional models*, describing functional properties of systems.

The static model describes system components, their structure, attributes, interrelations between them and operations which they can carry out. The operations of static model are events of dynamic model. They are functions of functional model. The dynamic model describes sequence of performance of operations in the process of system functioning. The functional model describes the transformations which are carried out by system. It opens the maintenance of operations of static model and events of the dynamic.

According to the degree of model abstraction it is possible to divide models into group of

- *conceptual models* representing a higher-level sight of a problem in terms of system domain;
- *models of the specification* defining "appearance" and external behaviour of system;
- *models of realisation* which reflect the internal structure of system, a concrete way of realisation of observable behaviour of system.

Various metamodel definitions exist. We recognise that the models, created by developers of information systems, should be described in any *formal language – modeling language*. We'll consider that the *metamodel is a model of modeling language* applied for formalisation of the system description. The *linguistic metamodel* is a metamodel described with domain-independent language of modeling. The *ontological metamodel* is a metamodel which describes system in domain-dependent modeling language (domain specific language).

The four-level hierarchy of models is a classical variant of metamodeling at information system development (Fig. 1). In this hierarchy each higher level defines language for the description of subordinate level.

Fig. 1. Classical four-level hierarchy of information system models

There are the data describing a state of a subject domain at the level M0, i.e. the *domain state model* of system is described at the level M0.

The level M1 is an *ontological metamodel* for level M0. It contains *domain specific model*. This model represents structure of data, rules and operations, describing the state of system domain. It is represented with *metadata*.

The level M2 defines a *linguistic metamodel* for the levels M1 and M0: at the level M2 there is a *model of modeling language*, used by analysts and developers for the system development. CASE-tools operate with this model too. The forth (the uppermost) level M3 defines *language used for description of metamodels* of level M2.

The number of levels can be changed at development of concrete systems with different tools.

Information systems and development tools can be divided into *some classes* according to the number of levels in created hierarchy of models and to the way of their use at the information system development and roles of

models in technologies applicable for the creation and maintenance of information system.

In *traditional information system* (Fig. 2) there is data describing a state of information system domain and conditions of system functioning. Data is contained "*in*" information system (if file system or in database).

This data corresponds to the model of information system domain which can be described in any language, including natural language or graphical notations.

The model of information system is developed by analysts. Then developers (database designers, programmers) realise this model by means of the chosen tools of information system development, programming systems and database management systems.

In case of model change it is necessary to copy and recompile source codes of applications of information system and/or to restructure information system database. More often at the process of information system development and at the information system maintenance developers "leave" from initial model which is not updated at modification of the data structures and algorithms of their processing. "Mismatch" of models leads to various problems.

In *traditional CASE-technologies* (Fig. 3) the domain model is defined formally. It is contained "*in*" CASE-system. This model is described *in terms of a metamodel* which can be defined in any language and realised by means of CASE-system.

The model change conducts to necessity of change of the code generated by CASE-system. As well as in a case of traditional information system, the metamodel is developed by analysts and then it is realised by developers. The change of metamodel involves copying and recompilation of CASE-tools. However such changes occur extremely infrequently.

Modern CASE-systems include tools for creation and editing of models, and they also allows to generate the database (DB) and the main part of information system applications code. The system developed with CASE-tools usually realises all necessary structures of data defined by the system model, provides access to the data in databases and standard user interface for work with them. The program components, realising behavioural and functional aspects of concrete information system, are detailed and refined more often manually.

In case of change of model the CASE-system allows to generate anew code of applications. A program code added by programmers "manually" remains at observance of certain rules of its writing. After "regeneration" of system applications manual completion of a code usually is required.

An advantage of this approach is that time of the working out initial stages is essentially reduced. Besides, conformity between system and model is supported by CASE-tools.

The *information systems operated the metadata* (Fig. 4) provide more powerful possibilities for dynamic adaptation.

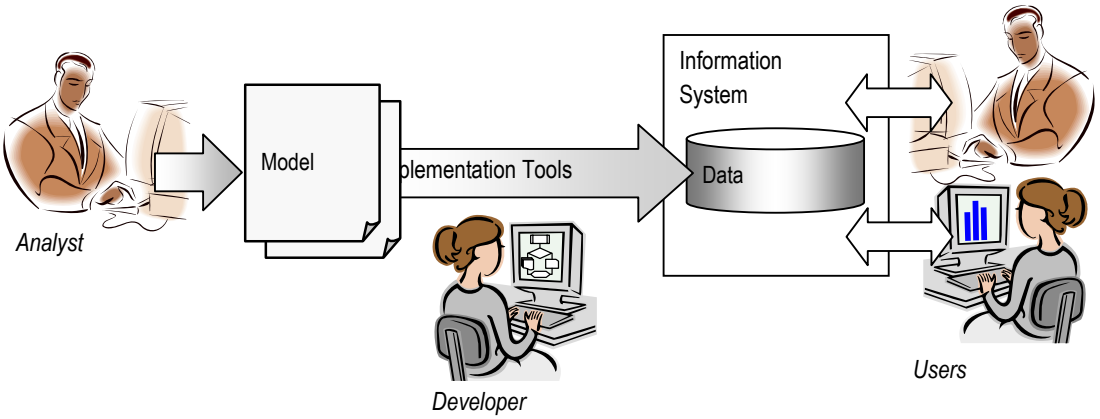

Fig. 2. «Traditional» information system

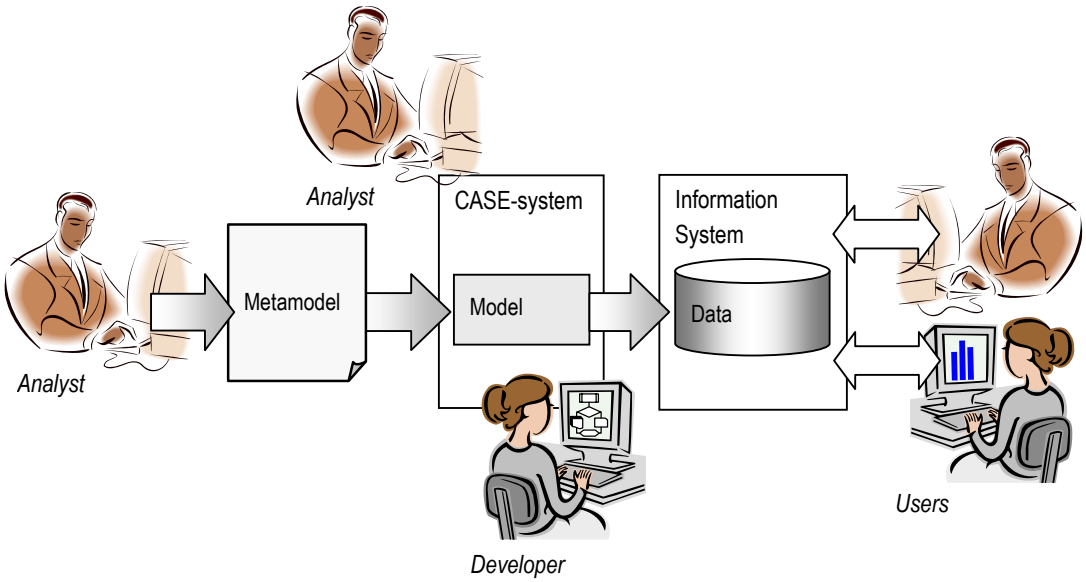

Fig. 3. «Traditional» CASE-system and information system

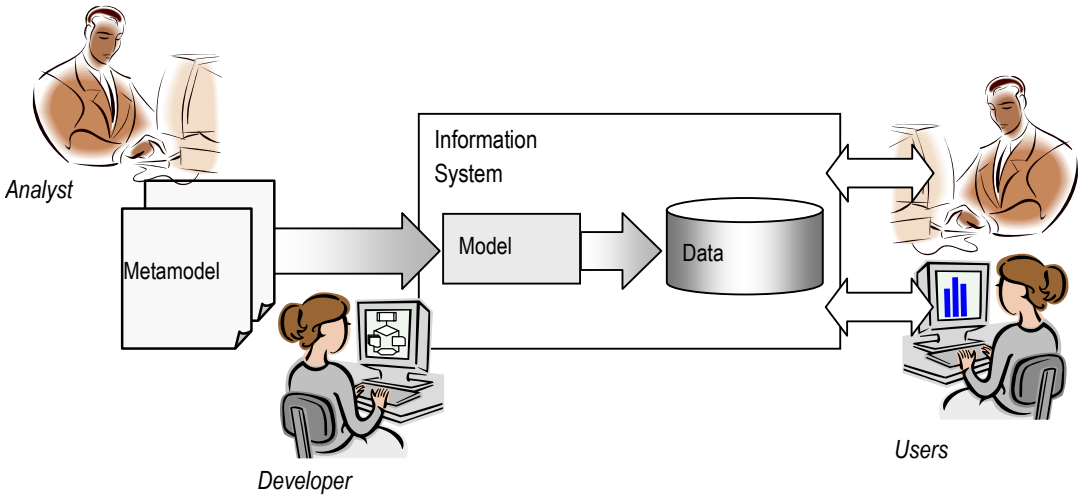

Fig. 4. Information system controlled by metadata

In this case also three levels of the models are used. However the model of a system domain, created by analysts and developers and represented with metadata, is "*in*" information system during its operation and it is used by run-time components of system. Thus, the software of information system acts like a interpreter of model, and model represents a "controlling system", setting rules of information system functioning.

A negative aspect of such approach is that performance of system decreases in its operation. Besides, if there are not possibilities of including to information system of external program components, expanding standard functionality of system, then it is impossible to realise the functions specific for the concrete system, reflecting the particular business processes rules and domain conditions. Therefore the metamodel should be as much as possible powerful and expressive.

The advantage of this technology is that in case of changes of system model it is not necessary to recode and recompile applications of system. It is not required to change source code of programs – the information system simply starts to work according to new model. But "old" data and "new" data can be "miscoordinated" with restructuring. Data loss is possible as result of changes of models.

Technology DSM with code generation provides modeling in terms of information system domain. Models are defined with languages using concepts and relations of information system domain. In case of *domain multimodeling* the specific language of modeling is applied to the decision of each problem in system domain. The domain specific languages are created and used exclusively for each system and problem.

There are two levels of metamodels in this technology (Fig. 5). The meta-metamodel is realised by analysts and developers with Meta-CASE-tools. Metamodel is described on the base of meta-metamodel with Meta-CASE-system. The metamodel defines domain specific modeling language. It is the base for CASE-tools generation. The model of system domain is described with this CASE-system. Then information system (data and applications) is created with CASE-tools on the base of this model. The Meta-CASE- and CASE-tools can be integrated in uniform CASE-system. This CASE-system forms a whole platform of development tools – DSM-platform.

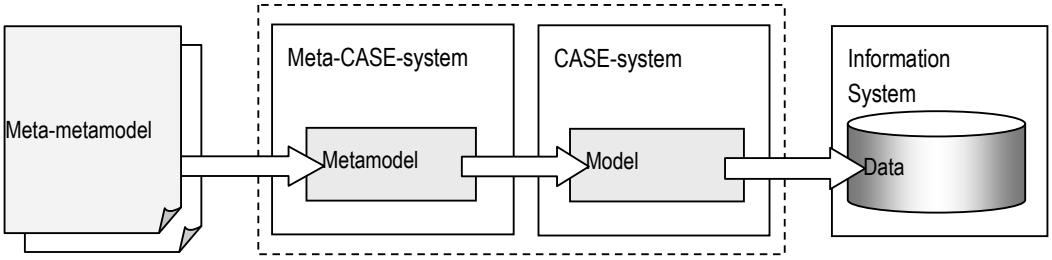

Fig. 5. DSM-technology with code generation

Development of domain-dependent languages (Domain Specific Language, DSL) allows essentially simplifying process of creation of domain models. Experts in the system domain (in applied problems) can take active part in models development process. Other advantages and lacks are related to code generation. They coincide with corresponding characteristics of traditional CASE-technology.

Technology DSM with interpretation of the metadata (Fig. 6) provides the maximum possibilities of dynamic adaptation of information systems. This variant is a combination of two previous. The metamodel, model and data are "*in*" information system developed with CASE-tools based on this technology. In this variant CASE-tools allow to develop models and to interpret them at system operation (special language toolkits and run-time components are included to system for this purpose). In order to use this approach in practice of complex information system

development it is necessary that the meta-metamodel was as much as possible expressive and powerful. Interpretation of two levels of metamodels leads at once to conspicuous loss of performance. However at sufficient expressiveness of a meta-metamodel extremely flexible system can be developed.

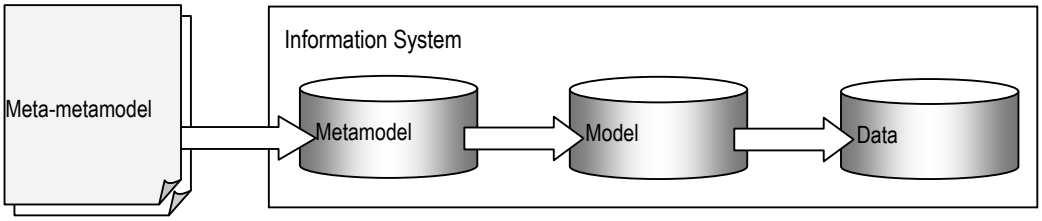

Fig. 6. DSM-technology with metadata interpretation

Such approach is realized in CASE-technology METAS presented in this paper [20]. An experience of some projects demonstrates an applicability of this technology.

The Technology of Development of Dynamically Adaptable Information Systems, Operated under Metadata

The maximum of information systems flexibility can be achieved, if both at system engineering and at its operation the metadata are applied. This metadata describe objects of a system domain, their properties and behaviour, conditions of its work and the characteristics of business processes. The developed information system operates under metadata in interpretation mode. The metadata describe the model of domain state, data and user interface.

CASE-technology METAS (METAdata System) is used for development of dynamically customized information systems operated under the metadata [20]. The base of the adaptability tools is the multilevel model of information system (Fig. 7).

The key aspect of this technology is use of the interrelated metadata describing information system and its environment from the various points of view and at the different levels of abstraction.

In most cases information system development tools work in mode of generation of the system applications in any programming language according to the set of specifications, describing information system domain and environment. The basic difference of METAS CASE-technology from other is in mode of functioning: metadata is used at operating time by program kernel of system in interpretation mode. METAS program kernel (Metadata system kernel, MDK) carries out system functions (data presentation and their processing) according to rules, defined by system metadata, interprets models.

System metadata represents the formalized description of information system. Metadata is stored in special base of the metadata (*MDB*, *Metadata Base*). This metadata, used for system customizing at development and functioning, describe the following aspects of information system: system domain objects and their behaviour, business operations and business processes, primary documents and reports, visual user interface and system security model. The metadata represent *base level models* (*MDK models*), each of which describes a certain part, aspect of information system (some models can describe the same parts of system, but from the various points of view). The metadata in METAS is divided into some interrelated models (Fig. 7). Program components of system work with the metadata of corresponding level (or with the several interconnected levels).

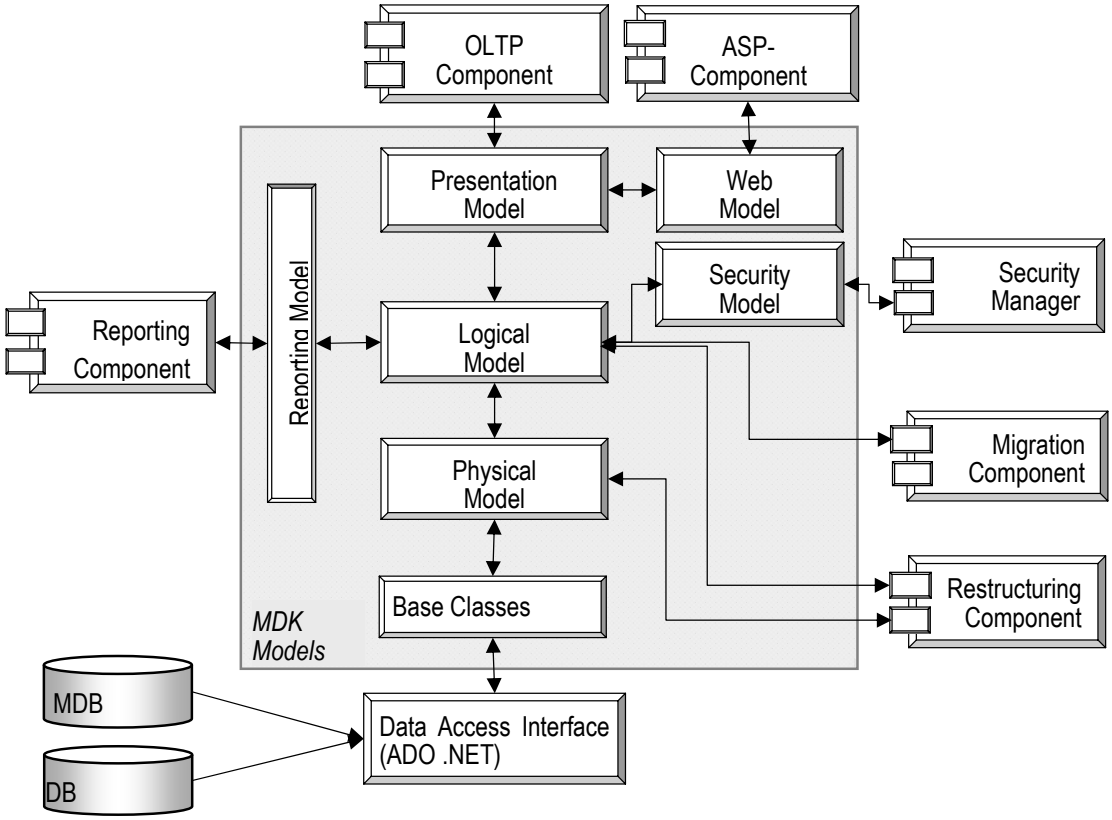

Fig. 7. Metadata models and METAS components

Metadata include three base layers representing following basic MDK models (Fig. 7):

- *Physical model* – the metadata, describing representation of information system objects in a database (DB) of system (for instance, DB scheme (description of tables and interrelations between them, etc.) is stored in this layer). At process of system functioning they form a basis of logical model. This model is automatically generated according to the description of system created in logical level [6, 18].
- *Logical model* – the metadata, describing entities of a system domain, their behaviour (through business operations) and also the general applied operations of information system. This model is based on notations of language UML and allows working in applied domain terms for users of information system. The model of this level is created by analysts and developers. It is the main layer of metadata [6, 10, 18].
- *Presentation model* – the metadata describing the visual user interface at work with objects of information system [10]. This model is automatically generated on the base of logical model. According to this model screen forms are generated. The main form of applications includes tree of objects for easy-to-use navigation and table presentation of basic information on objects of information system. A separate form is generated for all entity of information system. It is used for presentation of detail information on corresponding object and relational objects. Developers and users of information system can customize this model (i.e. user interface) adapt it to their informational needs and preferences.

The set of the characteristics represented in models can be dynamically modified and extended. The set of the models can be extended too via addition of new models describing new parties of information system or new properties of objects or existing, but from the new points of view.

For building algorithms of functioning METAS-system it is necessary to formalize the description with using metadata for all models. Let us consider the formal definition of metadata systems for physical and logical levels in terms of *graph models* [6, 10, 18].

The *nodes* of the *logical model graph* are corresponded to *entities of data domain*. On the physical level all entities are presented with tables of DB (one entity is described by *set of tables*: any entity has one *main table* and *some dictionaries* in addition).

All entities are a *set of attributes*. There are some types of attributes, presented in logical model, listed below:

- *Own attributes*. Own attribute is any one from the *main table* of entity.
- *Dictionary attributes*. Any entity's dictionary has corresponding attribute. When we work with it we may choose necessary values from the list or add a new value in dictionary.
- *External attributes*. If the entity relates to another as "M:1", then it must contain additional attribute with information about this relation.
- *Key attribute*. In fact it is an ordinary own attribute. It contains a link to key field of entity's main table, so it unambiguously determines values of all the rest attributes. This term is introduced only for handy work.

Sometimes it is necessary to represent the main information about real object of some entity in the short form. For example let two entities are related as "1:M". In this case entity on side "M" has an attribute representing another entity (*presentational attribute*). It is not necessary to represent in this attribute all information about object of domain, but only the main, which is enough for its identification. Such main information for entity is determined by the special *presentational expression*. It looks like SQL expression and may contain any attributes of corresponding entity. Any operation and function of SQL language are allowed.

Let us introduce the concept of *relation between entities*. It is analogy with table's relation, but on logical level, in terms of application domain.

There are two basic relation types:

- "1:M". This relation corresponds to physical relation "1:M" between entities' main tables.
- "M:M". This relation corresponds to physical relation "M:M", which is realized using intermediate table in relational database management system.

For any entity relation we may concretely define the count of entity on both sides. They have minimal and maximal number of entity, which may participate in relation. For example the following subtype for "1:M" are possible: "1 : 2..3", «0..1 : 0..1», «0..1 : 1» and other.

The *relations* between entities are presented as *directed arcs* of the graph of logical model.

The main task of logical level is to provide users possibility to work in term of application domain, without thinking about physical information storage in DB. In other words, there is no necessity on logical level to take care about distribution of fields between physical tables. Now user may work in terms of entity, simply calling its name in combination with the name of attribute. Entity builds SQL query itself and either executes it itself (operations of inserting, deleting and updating information in database) or returns ready SQL expression.

The graph presentation of logical model is automatically mapped to graphs of physical and presentational levels.

In addition developers and users can extend logical model with XML.

All new models are based on these models. Creation of new model requires development of programs mapping new model to models of basic levels or interpreting this model.

Now the following models extending basic MDK models are included in system:

- *Reporting model* – the metadata describing queries of users, templates of reports formed at execution of

business operations and business processes. This model is used for the analysis of the data and visualizing of analysis results [11].

- *Business-processes model* – the metadata describing business operations and business processes supported by the system.
- *Web-model* provides access to resources of information system for remote users through the Web-interface. This model is the base for development in Web-portal of system.

The *security model* allows to realize monitoring of privileges of users at work with system resources, controlling access rights [3, 9]. The protection subsystem works with own DB. This DB contains information on users and their privileges used for authorized access of users to system data and functions.

The CASE-tools allows to describe objects of information system domain and business processes, to build queries and reports in terms of domain, to customise standard user interface and also to export and import models and the data dynamically. The component of data migration allows to import data from external structural sources with schemes described by users. Users can export all results of data processing to various formats (text, MS Word and Excel documents etc.). *Means of information retrieval* and *documents analysis* allow searching documents and getting information from them. An adaptation of information system is realized without reprogramming of system components and without participation of programmers. The standard business logic can be extended via definition of new types and new operations that are specific for concrete domain.

Language Toolkits and METAS DSM-platform

As it is seemed above use of DSL allows to simplify information system development and to involve experts to process of creation and modification of domain models. Special language toolkits ("meta-CASE-tools") are necessary for development of these languages (meta-metamodels). Meta-CASE- and CASE-tools are integrated for development of systems based on DSMM in order to create multiple interrelational models of information system domain. These models are to be conformed. One approach to models conformation is to use uniform metalanguage for models development. The language used for creation of other languages is named *metalanguage*. Process of models development can be iterative: created languages can be used by developers as metalanguages for creation of other languages and so on. All languages will be conformed on the base of common metalanguage. Such approach is realized in METAS DSM-platform (Fig. 8) that is presented in this article [21, 22, 24, 25].

Now uniform universal DSL does not exist. Different CASE-tools are based on such languages of visual modeling as SDL and MSC – for modeling of telecommunication systems, IDEF, DFD, EPC, BPEL and BPML – for modeling of business processes, etc. Recently for the modeling language standard applies UML.

Key aspect of the DSM-approach is use of DSM-platform including language toolkits, translators or interpreters, etc. The modern DSM-platforms support optimum variants of visual modeling of information system domain and mode of system functioning. The decision on choice of concrete language completely lies on the developers of information system choosing means of system development.

Nowadays some systems for development of DSL exist. They include graphical editors of visual languages and allow possibility of definition of own graphic notations for specialists (experts in different domains and applied problems). Such means are MetaEdit+ [16], MS DSL Tools, Eclipse GMF, State Machine Designer, REAL-IT [19], UFO-toolkit [23] and so on. Technologies Microsoft DSL Tools, GMF, MPS are strongly connected with corresponding DSM-platforms – MS Visual Studio, Eclipse, IntelliJ-IDEA accordingly, therefore the languages created by means of these toolkits cannot be used in other platforms and external applications. These technologies do not allow possibility of creation both visual, and text DSL. Besides, any of technologies listed

above (except MetaEdit+) do not allow developing dynamically adjusted languages.

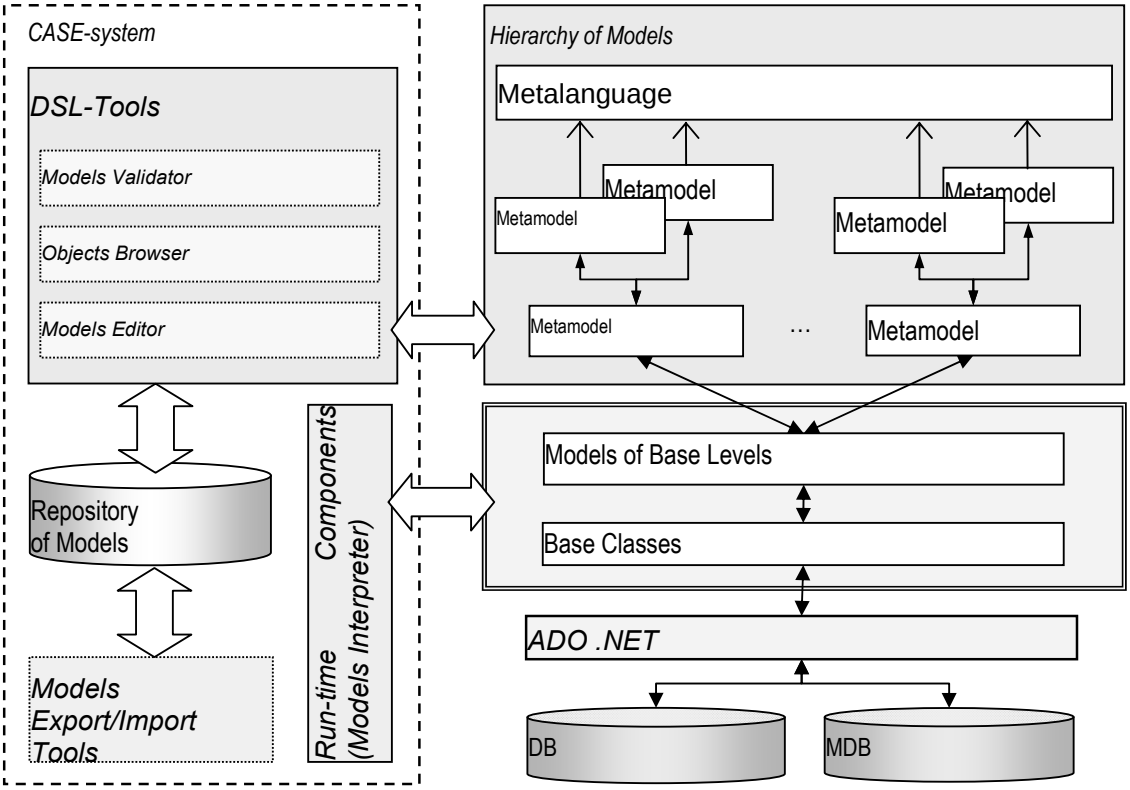

Fig. 8. Simplified structure of information system based on the DSM-platform with (meta)models interpretation

Elimination of noted lacks is the purpose of working out of a metalanguage and DSL-toolkits MetaLanguage – METAS DSM-platform (Fig. 8).

An approach to creation of dynamically changeable languages is based on storage of the language description in the base of metadata (repository) of information system accessible at its functioning.

New possibilities of toolkits and information systems:

- dynamic adjustment of languages for varying conditions of system functioning and requirements of business processes and needs of users;
- work in terms of the information system domain that are habitual for the users;
- reuse created languages and models in similar projects;
- integration of several domain specific languages into one system.

For the language development it is necessary to create its model – a metalanguage. On the base of metalanguage developers can create new languages. They can make changes in existing languages adapting DSL for their needs. If the description of the metalanguage is presented in the form of the metadata a possibility of changes in this description is real too. Besides, process of creation of metamodels becomes iterative and this process can be infinite.

It is necessary to integrate into a developed metalanguage both text, and graphic representation: graphic representation is convenient for using at the description of domain models, and text – at DB creation (for definition of restrictions, for description of templates of reports, etc.).

By analogy to traditional formal languages visual languages can be defined as a set of graphic symbols and formalized rules for development of visual models. Graph grammar represents the traditional approach to the formal description of syntax of visual languages. The most suitable model of internal representation of developed languages at system functioning is multigraph (despite simplicity of its topology). It is possible to describe all operations at development of metamodels in terms of oriented labeled multigraphs.

METAS Language toolkits include graphic editor, objects browser, repository and validator of models. Process of DSL development begins with definition of its metamodel: the basic constructions of language, the relation between them and the restrictions are described. As a result of metamodel creation developer receives new visual modeling language. Developers and users can create new models with this language. It is necessary to check created models with validator before use for system development. Models are stored in the system repository. Browser allows finding suitable models.

The approach described above is realised in the form of library of classes. The primary goal of library consists in maintenance of work with repository systems.

Architecture of Information Systems Developed with METAS Technology

CASE-technology METAS allows to create distributed information systems with client-server architecture of applications (Fig. 9). Distributed information system includes independently functioning subsystems. Subsystems can interact in different modes according to functioning conditions and user's needs. Subsystems are integrated with Simple BizTalk Server – integration component of METAS platform. The integration components support interaction of subsystems applications with some protocols.

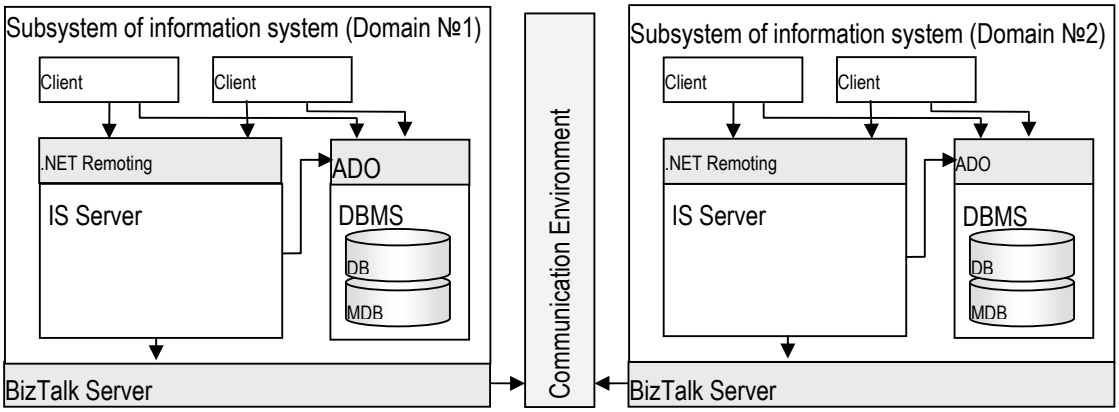

Fig. 9. Domain architecture of information system created with METAS DSM-platform

Simple BizTalk Server allows to replicate system metadata and data. The data replication tools realize two scheme of replication: data and transactions can be replicated.

Documents and templates can be transferred too.

Users of system can set the timetable and choose protocol for the applications interaction.

The components of interaction are the base of means of subsystems conformation and databases synchronization.

Conclusion

The basic advantages of the METAS technology are

- flexibility of systems and possibility of dynamic adaptation to changes of operating conditions, requirements of users with the minimum expenses;
- possibility of integration with external systems via different tools;
- there is no necessity of special training of users: system allows possibility of work in habitual environment in familiar terms of a domain;
- low requirements to a hardware-software platform.

The project of an integrate DSM-platform which provides all stages of information system development since working out of subject-oriented languages of modeling is realized now. Updating of metamodels can be made at any stage of creation DSL and system lifecycle. Thus after modification of a metamodel the system will automatically make all necessary changes to the models created by means of this metamodel. The developed tools form a basis for automatic system adaptation using ontologies in the capacity of domain model.

Acknowledgments

This work was supported by the Russian Foundation for Basic Research (Project 10-01-00794) and by the Russian Humanitarian Scientific Fund (Project 09-02-00373в/И).

Bibliography

[1] Almeida J.P., Pires L.F., van Sinderen M. Costs and Benefits of Multiple Levels of Models in MDA Development // Proceedings of the Second European Workshop on Model Driven Architecture (MDA) with an emphasis on Methodologies and Transformations “Computer Science at Kent”. September 2004. Canterbury, UK. P. 12-21.

[2] Atkinson C., Kühne Th. The Essence of Multilevel Metamodeling // Proceedings of UML 2001 – The Unified Modeling Language. Modeling Languages, Concepts, and Tools: 4th International Conference, V. 2185 of LNCS. Toronto, Canada, October 2001. Springer. P. 19-33.

[3] Chichagova M., Lyadova L. The Application of Graph Model for Automation of the User Interface Construction // International Journal "Information Theories & Applications" Vol.14, Number 2, 2007. P. 189-192.

[4] Chung L., Subramanian N. Adaptable system/Software architectures // Journal of Systems Architecture: the EUROMICRO. Special issue: Adaptable system/Software architectures. Vol. 50, Issue 7 (July 2004). P. 365-366.

[5] Cook St. Domain-Specific Modeling and Model Driven Architecture // MDA Journal, January 2004. P. 2-10. [PDF]. [www.bptrends.com/publicationfiles/01-04%20COL%20Dom%20Spec%20Modeling%20Frankel-Cook.pdf].

[6] Eremina M. About Methods of Mathematical Modelling in the Development of Information Systems // International Journal "Information Technologies and Knowledge". Vol.1, Number 2, 2007. P. 189-195.

[7] Favre J.-M., Nguyen T. Towards a Megamodel to Model Software Evolution Through Transformations // Preproceedings of the Workshop on Software Evolution through Transformations: Model-based vs. Implementation-level Solutions “SETra 2004”. A satellite event of ICGT 2004. October 2004, Rome, Italy. P. 56-70.

[8] Hessellund A. Domain-Specific Multimodeling. Ph.D. Dissertation (preprint), Supervisors: Sestoft P. and Osterbye K. [PDF]. [www.itu.dk/people/hessellund/work/hessellund-thesis-preprint.pdf].

[9] Kourilov D., Lyadova L. Complex Protection System of Metadata-based Distributed Information Systems // International Journal "Information Technologies and Knowledge" Vol.2, Number 2, 2008. P. 116-122.

[10]Kudelko E. The Application of Graph Model for Automation of the User Interface Construction // International Journal "Information Theories & Applications" Vol.14, Number 4, 2007. P. 366-373.

[11]Lanin V. Architecture and Implementation of Reporting Means in Adaptive Dynamically Extended Information Systems // International Journal "Information Technologies and Knowledge" Vol.2, Number 3, 2008. P. 273-277.

[12]Lochmann H., Hessellund A. An Integrated View on Modeling with Multiple Domain-Specific Languages Proceedings of Software Engineering (SE 2009) February 17-19, 2009 Innsbruck, Austria Editor(s): R. Breu. [http://www.docin.com/p-50777268.html].

[13]Mazanek S. Visual Languages. MetaEdit+ : [http://visual-languages.blogspot.com/2007/11/metaedit.html].

[14]Savidis A. Dynamic software assembly for automatic deployment-oriented adaptation // Preproceedings of the Workshop on Software Evolution through Transformations: Model-based vs. Implementation-level Solutions “SETra 2004”. A satellite event of ICGT 2004. October 2004, Rome, Italy. P. 191-198.

[15]Subramanian N., Chung L. Software Architecture Adaptability: An NFR Approach // Proc. Int. Workshop on Principles of Software Evolution (IWPSE`01) / Vienna, Austria. ACM Press, (September 2001). P. 52-61. [PDF] [www.utdallas.edu/~chung/ftp/IWPSE.pdf].

[16]Tolvanen J.-P., Rossi M. MetaEdit+: defining and using domain-specific modeling languages and code generators: [Электронный документ] [http://portal.acm.org/citation.cfm?id=949365].

[17]Tsybin A., Lyadova L. Software Testing and Documenting Automation // International Journal "Information Technologies and Knowledge" Vol.2, Number 3, 2008. P. 267-272.

[18]Борисова Д.А., Лядова Л.Н. Иерархическая модель данных как основа реализации информационной системы, управляемой метаданными // Математика программных систем: Межвуз. сб. научн. тр. / Перм. ун-т. Пермь, 2006. С. 4-13.

[19]Иванов А.Н. Технологическое решение REAL-IT: создание информационных систем на основе визуального моделирования // Сб. «Системное программирование» / Под ред. проф. А.Н. Терехова и Д.Ю. Булычева. – СПб: Изд. СПбГУ, 2004. С. 89-100.

[20]Лядова Л.Н. Технология создания динамически адаптируемых информационных систем // Труды междунар. науч.-тех. конф. «Интеллектуальные системы» (AIS’07). Т. 2. – М.: Физматлит, 2007. С. 350-357.

[21]Лядова Л.Н., Сухов А.О. Метаязык построения визуальных языков моделирования // Технологии Microsoft в теории и практике программирования / Материалы конф. Нижегородский ун-т. Нижний Новгород, 2009. С. 267-273.

[22]Лядова Л.Н., Сухов А.О. Языковой инструментарий системы MetaLanguage // Математика программных систем: Межвуз. сб. научн. тр. / Перм. ун-т. Пермь, 2008. С. 40-51.

[23]Маторин В.С., Маторин С.И., Попов А.С. CASE-инструментарий UFO-toolkit. Автоматизация построения УФО-моделей // Проблемы программирования. 2004. № 2-3. С. 144-149.

[24]Сухов А.О. Предметно-ориентированный язык в адаптируемых информационных системах // Технологии Microsoft в теории и практике программирования / Материалы конф. Новосибирск, 2008. С. 25-26.

[25]Сухов А.О. Среда разработки визуальных предметно-ориентированных языков моделирования // Математика программных систем: Межвуз. сб. научн. тр. / Перм. ун-т. Пермь, 2008. С. 84-94.

Author's Information

Lyudmila Lyadova *– Perm Branch of the State University – Higher School of Economics, Associate professor of the Department of Information Technologies in Business; 38, Studencheskaia St., Perm, Russia; e-mail: LNLyadova@mail.ru.*