Lomonosov Moscow State University
Faculty of Computational Mathematics and Cybernetics

Master course

DISTRIBUTED OBJECT TECHNOLOGIES PART 2: DESIGN OF SEMANTICALLY INTEROPERABLE INFORMATION SYSTEMS

Summary of the course

By

Professor Leonid A. Kalinichenko

September, 2009

Moscow

1 Motivation

This part of the DOT course [1] is intended to show how the initial, most creative stage of an information technology development might look like. Main objective of the specific technology used as an example is a move from technical interoperability technique (e.g., CORBA) to semantic interoperability.

We remind that emphasis in object technologies has moved to development of common models and architectures which efficiently provide for a capability of joint usage of (i) heterogeneous, (ii) pre-existing, (iii) distributed software and information sources for application problem-solving. Such collaboration of heterogeneous components (called *'interoperability'*) allows for composition of systems from pre-existing heterogeneous and distributed components. This development paradigm of interoperable open systems technology is unprecedented in scale.

At the same time Information Systems (IS) or Enterprise Information Systems (EIS) in particular should become the infrastructures provided for collecting information and services from across the entire enterprise thus giving all its workers complete and transparent access to information. Diverse forms of enterprise inter- and intra-organizational models were developed (such as, virtual corporation, extended enterprise). A basic issue for modeling of such distributed business activities, their collaboration is that there are many types of elements to be modeled in an enterprise, and many perspectives and contexts in which those definitions would be viewed. Often such elements can be implemented as legacy applications. Advanced enterprise modeling approaches share the fundamental strategy of integrating at the model level - taking fragments of information within the enterprise and placing them in a larger context. What model is to be taken, how a proper context is to be formed and implemented and how semantic interoperability in such context can be organized are the basic issues to study in this course.

It is important to note that there exist two principally different approaches to the problem of integrated representation of multiple information resources for an EIS: 1) moving from resources to problems (an integrated schema of multiple resources is created independently of specific applications) and 2) moving from an application to resources (a description of an application subject domain (in terms of concepts, data structures, functions, processes) is created, into which resources relevant to the application are mapped). The first approach driven by information resources is not scalable with respect to the number of resources, does not make semantic integration of resources in a context of specific application possible, does not lead to justifiable identification of relevant to EIS resources, does not provide for enhancing of EIS stability w.r.t. evolution of the relevant to EIS resources. These deficiencies are inherent to the Global as View (GAV) approach. GAV might be used as a basic technique for the information resources driven approach. The second approach (application-driven) assumes creation of an application model [2] that supports an interaction between an application and resources on the basis of the application domain definition Such technology has obvious advantages over the approach driven by specific information resources.

The middleware approach (e.g., CORBA) provides no application model but individual components that can implement the application when assembled and interconnected by a software bus. The problem of components is that they do not have *clean semantic specifications* to rely on for their reuse.

The gap between the existing Object Analysis and Design (OAD) methods applying mostly top-down technique and the demand of the middleware architectures and methodologies for the development based on a *composition of interoperating components* remains to be large.

A number of various computational, data and knowledge models (respective languages) based on an object paradigm is continuously increasing. These models are used for development of software and data services, information systems and their subsystems that technically can easily become components of the middleware. Such heterogeneity and lack of well-defined semantics of the respective models creates a big

[1] The summary of the DOT course (Part 2) provides an overview of the main sections of the course. Thus the structure and content of the course are defined. To simplify mastering of the course, the summary contains hyper references to the documents in electronic form recommended for studying of the respective course sections. Among these documents are the course materials and lecture notes, research papers and books. The referenced documents are in the form of PDF,PPT and PS files.

[2] Actually another concept is required here - the *mediator* instead of the application model. The mediation approach is a comprehensive technique that requires separate course to study it. This is why we avoid using the *mediator* term in frame of the DOT course. In case when this term or the term "specification of requirements" will be met in the materials referenced by this summary or by the list of the recommended literature, please, interpret a mediator specification or a specification of requirements just as an application model specification.

obstacle for their interoperability.

But probably the largest obstacle for the interoperability of components consists in *the application semantics of components technically interrelated through the middleware (e.g., CORBA)*. Reconciliation of their application concept bases (an obvious prerequisite for their interoperation) constitutes a problem.

This part of the Distributed Object Technology (DOT) course is focused on the approaches investigated to fill in the gaps mentioned. We focus on the issues of semantics of the specifications we get on different phases of the information system development.

- Semantics are the key issues to resolve. We strictly distinguish between application semantics and object model semantics. Object models of IS as well as object specifications of pre-existing components should be semantically clear to infer their most important for reuse relationships - whether a given specification can be correctly substituted by another one. Such reasoning becomes possible only in frame of *formal modeling languages* applying an idea of abstract interpretation. An introduction into this area of software engineering will be provided.

- Another issue is how to compose specifications of reusable components or their parts to form an artifact that can substitute a part of the application model of IS. *Strict definition of subtyping* as well as a *type algebra* will be introduced.

- An advanced language for specification of IS and pre-existing components independently of the actual models and languages used for their development will be also introduced as a *canonical object model*.

- *Ontological modeling* is discussed for expressing of the application domain semantics of the IS and of the pre-existing components. Various approaches are introduced.

- Applying such fundamentals, and having in mind object-based interoperability-oriented middlewares (such as CORBA) finally we go into the world of *component-based information system design*.

General ideas related to the problem of the Semantic Interoperability Reasoning (SIR) can be found in the reports SIR framework, Ideas for SIR and Analysis and Design

2 Formal models and methods

2.1 An introduction into design based on formal models

The software engineering community has devised many techniques, tools, and approaches aimed at improving software reliability and dependability. These have had varying degrees of success, some with better results in particular domains than others, or in particular classes of applications. A popular approach is known as *formal methods*, whereby a specification notation with formal semantics, along with a deductive apparatus for reasoning, is used to specify, design, analyze, and ultimately implement software systems.

A key issue is the need for those applying formal methods to be able to abstract and to model systems at an appropriate level of representation, that is, to develop solid design principles and apply them to software development. This is particularly true when proving properties of more complex systems involving significant concurrency and interoperation among components. When we wish to prove component properties and relationships, it is often the only way to prove them at a more abstract level, exploiting the idea of *abstract interpretation.*

Considering formal methods the course is focused mostly on those of them that provide for proof of a possibility to correctly substitute a specification of an application with the specification of a component.

A strategy for compositional development applying formal methods is briefly considered in the slides of the file A Strategy for formal methods

More on that can be found in the Strategies paper.

2.2 Models and languages

A brief introduction into formal methods is provided. It is given in a form of several examples of interrelated specifications of abstract data types (ADT) applying well known Z and Object-Z notations.

This introduction is given in the first section of the document ADT basics

If required, the description of Z notation can be found in Using Z as well as a collection of references to the publications on Z is available in Z referenes.

Object-Z is an extension of the formal specification language Z to facilitate specification in an object-oriented style. It is a conservative extension in the sense that the existing syntax and semantics of Z are retained in Object-Z. An introduction to the Object-Z is provided in Object-Z

The main formal model and language required for the DOT course is the Abstract Machine Notation (AMN) chosen due to the fact that for the purpose of design with reuse we need a formal basis to reason that pre-existing component or its part can serve as an implementation of a fragment of an application model specification. For the B technology (based on the AMN) specific tools exist providing for a proof of specification refinement ()[3]. For understanding the relationship between Z and AMN please read about Jean-Raymond Abrial.

An introduction into AMN is provided in the second section of the course material AMN. Another text, introducing B AMN with more examples is given in B Method. References to the publications on B can be found in B-related references.

In a form of lecture notes the information on formal methods is provided in Formal methods

3 Object types

3.1 Subtyping: strict definition

Subtyping (or subclassing in the object-oriented programming languages) is a well known relation. Regretfully very rare it is possible to meet a student who would know a strict definition of this relation semantics. According to this definition a fact that type specification A and type specification B are in a subtype relationship should be proved.

Good explanation of such definition is provided in the paper by Barbara Liskov A Behavioral Notion of Subtyping

In a brief form this definition is given in the first section of the course material ADT basics. In a form of lecture notes this information is provided also in Formal methods.

It is important to understand how definitions of subtyping and refinement are interrelated.

3.2 Type algebra (calculus)

Type specifications and their reducts are chosen as the basic units of specification manipulation. For their manipulation, the algebra of type specifications is introduced. The algebra includes operations of *reduct, meet, join* and *product* of type specifications. These operations are used to produce the specifications of the respective compositions of their operands.

The *reduct* of a type is chosen as the minimal unit for reuse, manipulation and transformation of type specifications. Reducts of the component type specifications can be used as minimal fragments potentially reusable for the respected reducts of the analysis model types. The identification of the fact of type reducts reusability is the basic concern of the design. For that the type specifications should be *conformant*: that is, a common reduct should exist for them. A *common reduct* for types T_1, T_2 is such reduct R_{T_1} of T_1 that there exists a reduct R_{T_2} of T_2 such that R_{T_2} is a *refinement* of R_{T_1}. For reusability decision the *most common reduct* of two types (the application model type and the component type) is the main target.

The composition operations (*meet, join*) of the algebra of type specifications are based on the notion of the most common reduct. It is important that though in the algebra we assume manipulation of the specifications of the semi-formal canonical object model, the mapping of such specifications into the formal notation is

[3]It is said that component (or a type) A refines component (or type) B if A can be used instead of B so that the user of B will not notice such substitution.

defined. This makes verifiable the results of the algebraic operations. Such approach makes possible to keep a balance between non-tractability of taking a common reduct for two types and a possibility to verify each step in the process of development (if required).

Definition of type algebra with examples showing AMN usage is provided in Calculus

How type algebra operations are used for determining the resulting type of a query (formulae) is defined in the Type Inferencing paper

4 Canonical model

For the object modeling the course shows how a "canonical" object model may look like. This model (the SYNTHESIS language) is used for uniform representation of various object and data models (including semistructured data models) in one paradigm. The canonical model serves as the common language, "Esperanto", for adequate uniform expression of semantics of various information models surrounding us. In particular, this model incorporates various widely used constructs in the current object models. The model is based on important notions of *algebra of object type specifications* used to compose new types or infer them during the object algebra formulae evaluation.

To give the canonical model exact meaning, a mapping of this object model into the B AMN notation has been constructed. This mapping provides precise meaning for the language. Thus, we get the semi-formal object model and its formal counterpart that we can use together as a common paradigm for:

1. uniform representation of various object models;
2. uniform specification of pre-existing components;
3. different models used on the phase of analysis of the information systems.

Such canonical model provides capabilities of consistency check of specifications on different phases of the information system development. But what is more important is that the concept of refinement of the specifications relying on the pre-existing components becomes inherent in the model. This property can be fruitfully used on various phases of the process of the IS design. In particular, the specifications of components of existing software or legacy system descriptions can be extracted and transformed into a collection of homogeneous and equivalent specifications for further reuse at the design phase.

An introduction into the SYNTHESIS language is defined in the course material Canonical object model for semantic interoperation reasoning.

Description of the SYNTHESIS language is provided in a book on the SYNTHESIS language.

5 Ontological modeling

5.1 General considerations

Application semantics of components we consider separately in frame of the ontological approach. Ontological definitions provide a conceptual framework for talking about an application domain and an implementation framework for problem solving. Ontology is treated as a well organized collection of *concept definitions*. Concepts are composite descriptions of individuals or types defining their attributes, concept interrelationships and concept "micro-theories" including rules (constraints) and functions. Semi-formal and formal (model-based) specifications for concepts are provided. We base the ontological model on the canonical object model.

5.2 Models (languages) for definitions of ontologies

Various ontological models have been developed so far to provide for ontological modeling in various applications of IT.

Here three ontological models will be presented.

The first one is based on verbal definitions of the application domain concepts expressed in a natural language. The ideas and examples of such not formal way of concept definitions are considered in sections 1,3 and 4 of the course material on Ontological modeling and in Ontomodeling presentation.

The second one, ONTOLINGUA model (developed at the Stanford University), is based on the full first order logic (FOL). An introduction to ONTOLINGUA is provided in sections 2.1 and 2.2 of the course material on ONTOLINGUA.

The third one, Web Ontology Language (OWL) developed by W3C for the Semantic Web is based on the description logic being a tractable subset of FOL. Description of OWL can be found elsewhere, e.g., in the W3C documents OWL Overview, OWL Reference, OWL Semantics, in Wikipedia Web Ontology Language, etc. Role of OWL and ontologies in Semantic Web is discussed in Ontologies and Semantic Web

5.3 Methodology for ontology integration, canonical ontological model

To work with ontologies of different resources a unified representation of ontologies is required. It is provided by a canonical ontological model developed so that the ontological model (language) of any resource could serve as its refinement. After analysis of various ontological models the kernel of canonical ontological model has been identified as a subset of the SYNTHESIS language mentioned above. This subset includes:

- facilities for definition of concepts as abstract data types (ADT);
- type invariants expressing concept constraints;
- generic ontological metaclasses, instances of which are ADTs defining concepts;
- verbal definitions of concepts applying metaframes annotating types expressing concepts.

An introduction into the ONTOLINGUA model and the ideas of mapping of the ONTOLINGUA into the SYNTHESIS can be found in the second section of the course material Ontological modeling as well as in Ontomodeling presentation.

The OWL DL ontological model has been mapped into the ontological canonical model and the respective definition of the canonical model kernel extension has been defined. Details are presented in Reversible ontological model mapping paper.

For each of the component specification suspected to be relevant for the application the reconciliation of its ontological context with the application domain ontological context should be made. Technically it means construction of the extended ontological specifications establishing the component/application names and concepts relationship. The basic idea of the step is to find for the component names proper synonym or hypernym associations in the application domain ontological specifications. The relationships between concepts are established on the basis of subtyping association that can be justified using the refinement technique.

An approach for the resource and application contexts reconciliation is presented as follows.

Issues of ontological identification of relevant specifications for semantic context integration of heterogeneous semistructured sources are discussed in the Intercontext correlation paper. Here metainformation model is defined which includes uniform features for ontology, thesaurus and classifier modeling. Special technique for integration and mapping of different ontologies in this model is defined. The method for identification of specification element correlations in different contexts is considered.

An approach for ontologies reconciliation in terms of type refinement is presented in the Ontology reconciliationpaper.

An approach for establishing semantic linking of an application model and of the resource specifications is considered in Object specifications linking.

6 Component based compositional design

6.1 General considerations

The component-based information system design is organized around the object middleware concept providing for interoperability (like CORBA). We base the development process on a conventional object analysis

and development methods. The requirement planning and analysis phases of the conventional process are augmented with ontological specifications and complete specifications of type constraints and operations in the canonical model. The design phase is completely reconsidered: this is the design with reuse of the pre-existing components uniformly specified in the canonical model.

Application model specification for EIS includes definition of terminology and concepts of the subject domain of the application. They are expressed by the respective dictionaries (thesauri) and ontological definitions in the canonical model. Application definitions include also specifications of object classes corresponding to the application subject domain, specification of instance types of these classes and of their methods defining behavior of the objects, specification of processes characteristic for the application. On the early stages of design the specifications mentioned can be expressed by means of various specification languages (e.g., by UML). However, it is assumed that finally such specifications (including ontological ones) are mapped into the canonical model specifications having formal semantics. It is worth of remark that application-driven specifications are formulated independently of specific pre-existing information resources. The result of this specification activity fulfilled by a community interested in a specific EIS application constitutes the application model specification created as the result of reaching a consensus in such community. The application model specification activity is called the application model consolidation phase.

General approach for the application problems statement and solving consists in problem formulation in terms of the application model specification and transformation of this formulation into a set of tasks (queries) to the real information resources registered at the application. Such transformation in the database theory is known as the view based query rewriting.

To discover component types (classes) and their fragments relevant for the concretization of an analysis model we undertake an associative search of the component constituent names based on the ontological correlation with the proper application names. Then we take into account interconcept associations interpreted as subtyping relationships. As the result we get sets of specifications of probably relevant component types and classes. Further we should select among probably relevant component types (classes) those that really may be used for the concretization of the application domain type (class). For reuse a model of composite object integrating data and behaviour from various resources is applied. The data and/or behaviour residing at each resource is regarding as a fragment of a composite object. Reducts (projections of type specifications) are considered as patterns of reuse and composition. In design, specifications of the concretization types (classes) are constructed as the mediating definitions above the reusable reducts of the component types involved. Correctness of the results of design can be verified using formal facilities of the canonical model. The integrated presentation in the context of the single, object paradigm is unique in filling the gap between the existing object analysis and design methods, the technical interoperation architectures and methodologies for specification refinement and reuse of pre-existing components. The development method of semantically interoperable information systems serves as a glue between these techniques.

6.2 Resource Identification and Registration

Identificatio (discovery) and registration of resources to be re-used for the IS implementation is a process that includes decomposition of an application model specifications into consistent fragments, search among specifications of relevant resources of such types (or their reducts) that could be used as candidates for refining by them of the application model specification types, construction of expressions defining resource classes as a composition of the application model classes. For such manipulation a specification composition calculus (type algebra) is used. A principle of type specification decomposition into a set of specification reducts serving as the basic units of reuse and composition has been declared. An operation of identification of the most common reduct of resource type and application model type specifications has been introduced. Type lattice and type algebra have also been defined. Important point in this scheme consists in a provision of the type refinement proof applying logical model of the application and resource type specifications in AMN. The compositional calculus emphasizes complete type specifications and expressiveness sacrificing tractability in complex cases. Such decision is motivated by orientation of the modeling facilities on type refinement and composition. The benefits we get include rigorous way of identification of common fragments of resource and application model type specifications leading to justifiable mapping of resource type into application model type.

How a concretization of a type of the application model by a component type can be built applying formal proof in the AMN is defined in the Component-based design paper.

Similarly a problem of concretization with the emphasis on semantic reconciliation of resource types to satisfy the application model is introduced in Concretization construction paper.

A methodology for applying of the canonical model and the AMN for the automation of refinement proof in the process of component reuse is considered in the Automation of Verification paper.

A process of registration of heterogeneous information resources in an application model is based on GLAV that combines two approaches - Local As View (LAV) and Global As View (GAV). According to LAV the specifications of resources being registered are considered as materialized views over virtual classes of an application model. GAV views provide for reconciliation of various conflicts between resource and application model specifications and provide rules for transformation of a query results from resource into the application model representation. Such registration technique provides for stability of IS application specification during any modifications of specific resources and of their actual presence (removing, addition new ones, etc.) as well as for scalability of applications w.r.t. the number of resources registered in them.

Identification of resources relevant to an application (that precedes the registration) is based on three models: metadata model, characterizing resource capabilities represented in external registries (basic ideas for such registries are provided in CORBA naming and trading services), canonical ontological model, providing for definition of application domain concepts, and canonical model providing for definition of structure and behavior of application and resource objects. Reasoning in canonical models is based on the semantics of the canonical model and facilities for proof of refinement. Reasoning in the metadata model is a heuristic one based on nonfunctional requirements to the resources needed in application (similarly to CORBA trading). For the design, the application model and resources specifications are given uniformly in canonical model, though in process of design a transformation into such model from another specification language (e.g., from UML) might be required.

Complicated problem of registration consists in reconciliation of contexts of an EIS application and specific resources. Ontological definitions annotate elements of application model specification and of resource specifications given in a form of types, classes, processes. A similarity of concepts is established in two steps: first, by means of verbal ontologies, and then by establishing refinement relationship of concepts as abstract data types. Thus the conformity of the concepts is verified. Depending on complexity of specifications such verification can be provided automatically or interactively. In the simple cases the verification can be reduced to justifying of concept subsumption in description logics applying practically such systems as FaCT or Pellet.

The techniques listed are used as a basis for identification and registration of information resources in the application model. In this process ontological specifications are used for identification of application model classes semantically relevant to a resource class. The maximal subset of the application model class specification semantically relevant to a resource class is identified as the most common fragment (reduct) of specifications of respective instance types given for these classes. Concretizing types reconciling the conflicts (of values, structures, behaviors) are defined so that an instance type of the application specification class would be refined by an instance type of the resource class. The main registration result is a GLAV expression defining how a resource class is determined as a composition of the application model classes. In process of resources evolution a specification of the application model remains stable, only such expressions need to be modified. An instrument for supporting techniques of registration of resources in an application specification should include:

- facilities identifying relevant information resources by metadata;
- facilities for reaching consensus of ontological contexts of information resources being registered and of the application model;
- facilities for automation of the heterogeneous information resources registration in the application model based on the GLAV approach;
- metainformation repository storing specifications of application models, information resources and the results of registration.

An approach for heterogeneous resources registration at the application model is described in the Resouce registration, in the Compositional approach, in Compositional approach in Russian papers or in the dissertation on component-based design.

6.3 Compositional design

The information system design is a process of systematic transformation of specifications. In the component-based development a transformation from the application model to the design model consists in constructing of a composition of relevant fragments of pre-existing components to be included into a design specification that should finally serve as a concretization of the application model. The component-based development in the interoperable environment imposes a strict demand on semantic interoperability of components as the basis for such compositions. To cope with such strict demands, we should provide complete specifications in a notation suitable for their manageable and justifiable transformations.

Type specifications and their reducts are chosen as the basic units of specification manipulation. For their manipulation, the algebra of type specifications is introduced. The algebra includes operations of *reduct, meet, join, product* of type specifications. These operations are used to produce the specifications of the respective compositions of their operands. The reduct of a type is chosen as the minimal unit for reuse, manipulation and transformation of type specifications.

Reducts of the component type specifications can be used as minimal fragments potentially reusable for the respective reducts of the analysis model types. The identification of the fact of type reducts reusability is the basic concern of the design. For that the type specifications should be *conformant*: that is, a common reduct should exist for them. A *common reduct* for types T_1, T_2 is such reduct R_{T_1} of T_1 that there exists a reduct R_{T_2} of T_2 such that R_{T_2} is a *refinement* of R_{T_1}. For reusability decision the *most common reduct* of two types (the analysis model type and the component type) is the main target.

The composition operations (*meet, join*) of the algebra of type specifications are based on the notion of the most common reduct. It is important that though in the algebra a manipulation of the specifications expressed in the semi-formal canonical object model is assumed, the mapping of such specifications into the formal notation is defined. This makes the results of the algebraic operations verifiable. Such approach makes possible to keep a balance between non-tractability of taking a common reduct for two types and a possibility to verify each step in the process of development (if required).

A semi-formal canonical object model for all phases of the design process is applied. The semantics and notation of this model is defined. Object-oriented nature of the model makes possible to establish a mapping between the canonical model and existing notations in object technology (e.g., UML). This property is important for incorporation of the graphical notation (UML-based) into the development process.

After establishing interconcept correlation and ontological concept integration between different application domains, the process of semantically interoperable information systems design starts. The heuristic procedure for the most common reduct construction for a pair of ontologically relevant type specifications is defined. The process of design is based on this procedure driven by ontologically relevant pairs of attribute types. The common reducts discovered by the procedure are used in type algebra expressions and object calculus formulae to define new types that should be constructed in the design phase. The process is illustrated by an example showing various steps of the design including verification of the results in B AMN.

A framework for evolving from technical to semantic interoperability design is proposed in Complementary architecture paper.

A process of semantically interoperable information systems design is described in the course material Design, Design presentation, in the paper devoted to Component-based development. as well as in the Compositional development report or in dissertation on component-based IS construction.

7 Web Services compositions

How the approach for semantically interoperable information systems development with reuse applied in frame of the object middleware (CORBA) could be extended to the Web service framework is considered in the Extension paper as well as in the Composition of Web services paper.