Canonical model development techniques aimed at semantic interoperability in the heterogeneous world of information modeling

L.A.Kalinichenko

Institute of Informatics Problems
Russian Academy of Science
leonidk@synth.ipi.ac.ru

Abstract. This paper addresses the context of enterprise modeling aiming at a methodology showing how a unified model intended for the information integration and interoperability in heterogeneous world can be systematically synthesized. We can observe an explosion of diverse information modeling facilities and respective platforms around the world. Various attempts to reach technical interoperability among the platforms are well known. Diverse middleware frameworks have been developed emphasizing different basic artifacts (objects, messaging, services, etc.). Taking into consideration information modeling semantics in such frameworks is rarely attempted. The approach considered here is based on the following idea. To provide semantic information integration and interoperability in a heterogeneous environment including different modeling facilities an *extensible* canonical model is required. A core of the canonical model is fixed. Then for each specific information model M_i, taken from the environment, an extension of the canonical model core is built so that this core together with such extension is *refined* by M_i. Such refinement-based model transformation should be *provable*. The canonical model for the environment is synthesized as a *union of extensions* created for all the models M_i in the environment.
Techniques for such approach are considered separately for *structured* data models with behaviors, for *object* models and for *process* models. The paper has a form of a short overview of basic results related to these techniques and obtained in IPI RAS in different periods during more than 20 years. Experience reached, lessons learnt and perspectives of the approach for the Interop project (enterprise modeling) are discussed.

1 Introduction

Enterprise modelling provides the means to structure and decompose the enterprise system into less complex parts and to describe functionality and behaviour of any part of the system. Enterprise models support various requirements of enterprise inter- and intra-organizational engineering and integration. Diverse forms of enterprise modeling were developed (such as, *virtual corporation, extended enterprise*). A common feature for these forms is that they imply crossing-boundary business activities. Virtual enterprises should unite forces and jointly

behave as one producer towards the customers. Modeling of such *distributed business activities*, their collaboration constitutes a problem. A basic issue 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. Enterprise integration combines partial definitions and their uses for multiple purposes in such a way that the whole system can be seen coherently. All 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 uniform context.

Current situation in enterprise modelling is characterised by a large number of modelling languages and tools leading to heterogeneous and unintegrated definitions of the enterprise and by absence of communication among tools. Common representation of enterprise definitions is required. One of the possible solutions is to develop a UEML (Unified Enterprise Modelling Language) [23] as a means to mediate between different Enterprise Modelling tools, as an inter-lingua among such tools.

Technically, interoperability of business processes and enterprise models can provide for heterogeneous models collaboration. Interoperability requires the timely and meaningful exchange of information among the models. It requires reliable transfer of information, a clear agreement on its syntax, and shared well-defined understanding of its semantics. The benefits of information sharing between collaborators can only be exploited if interoperability on the data, behavior and process models can be assured. Information from many independent sources and with different semantics should be associated, organized, and merged.

Various middleware frameworks facilitate technical interoperability of enterprise models. Data-level interoperability deals with moving data between multiple data-stores, message-level interoperability assumes message exchange between the integrated information systems or services, process-level interoperability is responsible for handling message flows, implementing rules and defining the overall process execution. At the same time taking into account the information modeling semantics in such frameworks is rarely attempted.

This paper is focused on inter-model interoperability based on the semantics of heterogeneous modeling facilities. For uniform representation of various modeling facilities in one paradigm a canonical model is needed that should provide for semantically equivalent representations at the canonical level of definitions expressed heterogeneously. To reason that a definition in one language can be substituted by a definition in another one, a formal specification framework and a commutative model mapping method are provided. According to this method, to provide semantic information integration and interoperability in a heterogeneous environment including different modeling facilities an *extensible* canonical model is required. A core of the canonical model is fixed. Then for each specific information model M_i taken from the environment an extension of the canonical model core is built so that this core together with such extension are *refined* by M_i. Such refinement-based model transformation should be *provable*. The canon-

ical model for the environment is synthesized as a *union of extensions* created for all the models M_i in the environment.

The paper defines techniques experienced for such approach separately for *structured* data models with behaviors, for *object* models and for *process* models. The paper has a form of a short overview of basic results related to these techniques and obtained in IPI RAS in different periods during more than 20 years. The cited bibliography reflects this period. Analysis of related works can be found in [10, 11, 13, 22].

Though heterogeneous model mapping and canonical model construction techniques are considered in specific contexts (data base integration and mediation, object type specification re-use and compositions, compositional development of processes), they can be easily generalized for many other applications. Enterprise modeling and model driven architecture of OMG (MDA) [15] are two examples of contexts showing the range of possible applications of the approach. The relations between the UEML development and data base integration issues have already been discussed in [19].

The paper is structured as follows. Section 2 provides an overview of the results on extensible canonical model development and commutative data model techniques experienced in the beginning of 80ies for structured data models. Section 3 extends approaches of Section 2 with the refinement technique that has been applied to object models in 90ies. Section 4 overviews the recent attempts to apply the extensible canonical model development techniques to the process models. Experienced reached and perspectives of their use in various environments are summarized in conclusion.

2 Structured data models mapping and canonization

2.1 Principles of canonical model construction

An approach for rigorous definition of data models and for handling them as formal objects in the process of construction of data model (DM) mappings [10, 11] is considered in context of heterogeneous database integration. In each database management system a data model is completely defined by data description language (DDL) and data manipulation language (DML) semantics. In transition from a specific (source) DM to the canonical (target) one it is required to preserve information and operations. To reach that a source DM should be equivalently represented in the canonical one in the process of DM mapping. A notion of *data model equivalence* is introduced in the following manner. Database states in a source and a target DM are equivalent if they are mapped into the same state in the *abstract data metamodel*. It is assumed that equivalent database states represent one and the same collection of facts. Database schemas are equivalent if they produce sets of database states of equal power related by bijective dependency in such a way that the states being in one-to-one correspondence are equivalent. Two data models are *equivalent* if each database schema in one model can be put into a one-to-one correspondence with the equivalent schema

in the other model (and vice versa), while providing completeness of the DML operator set in each data model.

The following are the basic principles of a canonical model construction:

The *data model axiomatic extension principle*. Canonical data model should be extensible while new data models are involved into considerations. Such extension is implemented axiomatically. An extension of target DM is formed by adding to its DDL of a system of axioms determining (in terms of the target model) logical data dependencies of the source data model. The result of the extension should be equivalent to the source data model.

The *data model commutative mapping principle*. In the process of mapping a source DM into the canonical one it is necessary to preserve information and operators. This requirement is satisfied if DM mapping is commutative.

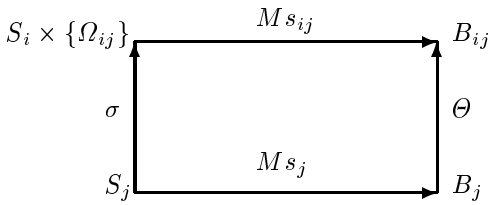
The set of all schemas expressible in DDL of M_i is denoted by S_i, set of data manipulation statements of M_i DML is denoted by O_i. A *space of admissible states* expressible in M_i is denoted as B_i.

$Ms_i : S_i \to B_i$ is a semantic function of M_i DDL

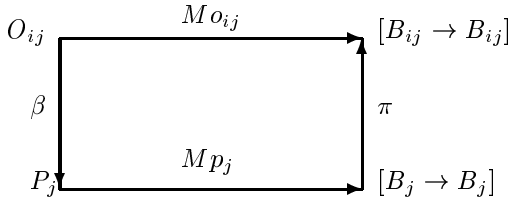
$Mo_i : O_i \to [B_i \to B_i]$ is a semantic function of M_i DML.

Mapping $f =< \sigma, \theta, \beta >$ of data model M_j into an extension M_{ij} of data model M_i is commutative if the following conditions hold:

— schema mapping diagram is commutative:



— DML operators mapping diagram is commutative:



— θ mapping is bijective.

Here Ω_{ij} denotes a set of axiom schemas expressing the data dependencies of M_j in terms of M_i; P_j denotes sequences of M_j DML operators (procedures).

The *unifying canonical data model synthesis principle*. Canonical data model synthesis is a process of construction of canonical data model core extensions equivalent to various data models included into the environment and a process of

merging of such extensions in the canonical data model. Following this process, the unifying canonical data model is formed in which various source data models have homogeneous equivalent representations.

Axiomatic extension of the target model means that its DML operator semantics should be adequately modified to preserve the axioms of the extension.

2.2 Denotational semantics as the data metamodel

Main reason for formal definition of the data models is to obtain their compact and precise descriptions, making possible manipulation by different data models as by mathematical objects. The metalanguage used for the formal definition of the data models is called the data metamodel (DMM). DMM should be general (i. e. independent of particular data model concepts), allowing precise expression of semantic properties of different data models, of their similarity or difference on the basis of one and the same language. Formal DM definitions provide for strict discipline for design of data model mappings, according to which construction of the mapping and proof of its correctness should be done simultaneously.

Denotational semantics [20] has been experienced as a formal DMM for structural data model mapping [11]. The cornerstone of the denotational semantics consists in the introduction of the class of "data types" — domains of functions — as partially-ordered sets and of a class of functions (generally recursive) for creation of a natural and precisely defined computational model. Relation $\sqsubseteq$ ("less defined or equal") is a partial order on domain D such that for all d belonging to D the relationships $\perp \sqsubseteq d$ and $d \sqsubseteq d$ holds. Data type (domain) in the data metamodel is a set partially ordered by relation $\sqsubseteq$.

Abstract data metamodel consists of:

- a set of elementary domains $D_1, D_2, ...$ corresponding to primary sets of objects;
- operations allowing construction of complex domains (data types) from simpler ones and facilities for data type formal definition;
- set of data types defined on the basis of elementary domains by means of data type constructing operations;
- a set of primitive functions and predicates, defined on the data types;
- a set of functional forms, used for new function definition and facilities for formal definition of functions;
- a set of function definitions;
- a set of rules of equivalent function transformations.

Any data type (domain) T is constructed recursively from data types (domains) $D_i (1 \leq i \leq n)$ using domain constructing operations $OP = +| \times | \rightarrow$ (domain of sum, product and functional domain respectively) according to the following rules:
$T = D_i$ (any domain is a data type), $T = T^\star$ (list domain), $T = T^n$, $T = T\ OP\ T$. In domain constructing expressions parentheses may be used: $T = (T\ OP\ T)$. For readability the functional domain is inserted into square brackets: $T = [T \rightarrow T]$.

A functional form (conditional form, functional substitution, composition) in a data metamodel is an expression denoting function.

Equivalent function transformation is used to demonstrate the commutativity of the operator mapping diagrams in DM mappings. Examples of basic transformations [4] are shown below. Here *f,g,h* denote arbitrary functions, *p,q* denote any predicates.

$$\bar{\perp} \circ f \equiv f \circ \bar{\perp} \equiv \bar{\perp} \tag{1}$$

$$(p \longrightarrow f, g) \circ h \equiv p \circ h \longrightarrow f \circ h, g \circ h \tag{2}$$

$$h \circ (p \longrightarrow f, g) \equiv p \longrightarrow h \circ f, h \circ g \tag{3}$$

$$p \longrightarrow (p \longrightarrow f, g), h \equiv p \longrightarrow f, h \tag{4}$$

Complete specification of DMM can be found in [10, 11].

2.3 Structure of formal definition of a data model

The definition of DDL of data model M_i consists of the following:

1. *Abstract DDL syntax* as a domain of all schemas expressible in the data model;
2. *DDL semantic domains* as data types representable in the data model;
3. *Schemas of DDL semantic functions*;
4. *Functions of DDL construct interpretations in semantic domains.*

The definition of DML semantics of M_i consists of the following partitions:

1. *Abstract DML syntax* as a domain of all DML operators expressible in M_i ;
2. *DML semantic domains* containing DDL semantic domains and data types characterizing state of a DML program;
3. *Schemas of DML semantic functions*;
4. *DML statement interpretation functions* interpreting changes of database enforced by DML statements execution.

2.4 Process of commutative data model mapping construction

The generic process of commutative structural data model mapping construction is shown on Fig. 1. This process solves two problems: 1) construction of canonical model core extensions equivalent to source data models; 2) development and verification of canonical DML interpretation by the source DML. This process was used intensively in the beginning of 80ies. Details of the process application and respective examples can be found in [10].

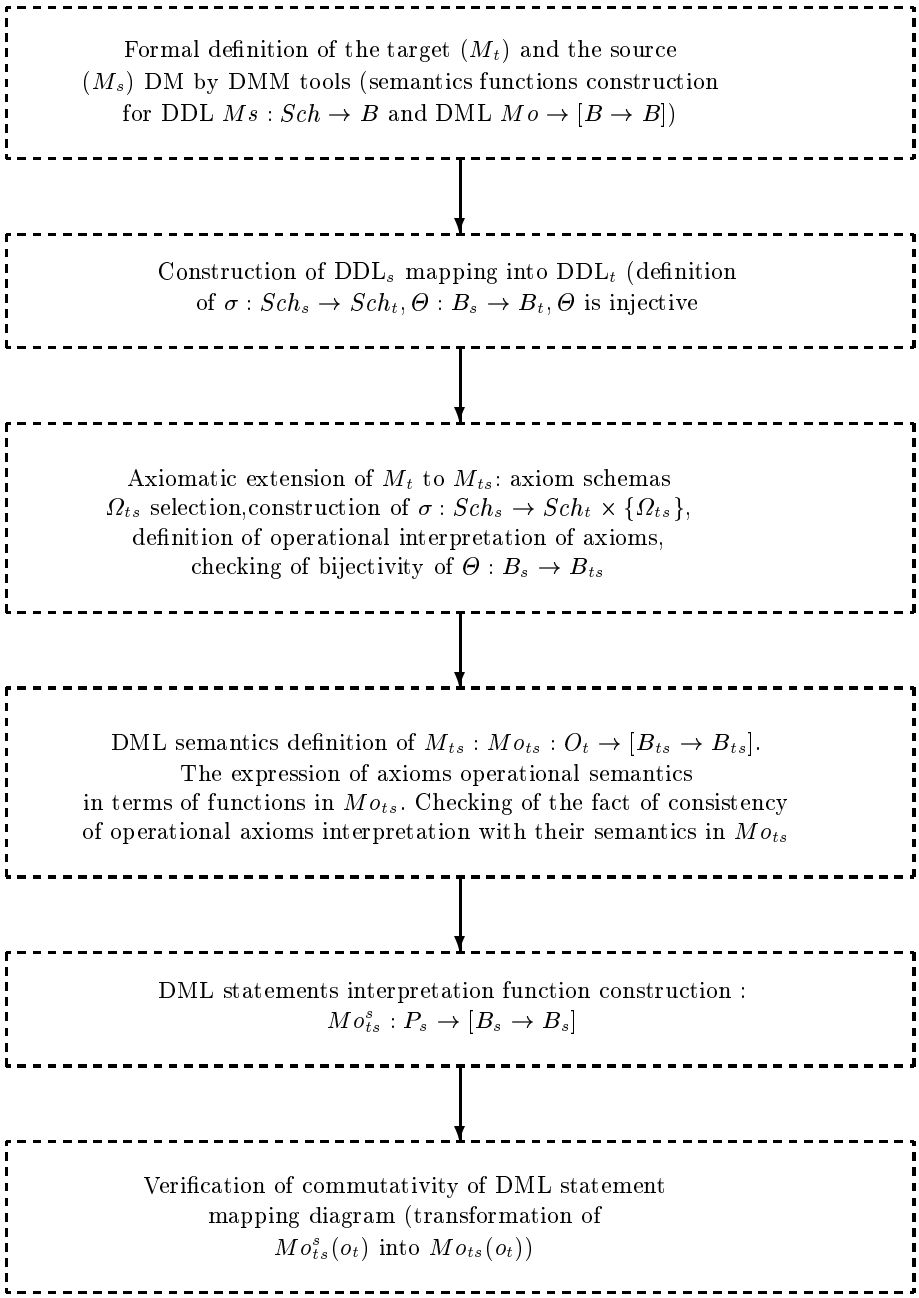

Formal definition of the target (M_t) and the source (M_s) DM by DMM tools (semantics functions construction for DDL $Ms : Sch \to B$ and DML $Mo \to [B \to B]$)
Construction of DDL$_s$ mapping into DDL$_t$ (definition of $\sigma : Sch_s \to Sch_t, \Theta : B_s \to B_t, \Theta$ is injective
Axiomatic extension of M_t to M_{ts}: axiom schemas Ω_{ts} selection,construction of $\sigma : Sch_s \to Sch_t \times \{\Omega_{ts}\}$, definition of operational interpretation of axioms, checking of bijectivity of $\Theta : B_s \to B_{ts}$
DML semantics definition of $M_{ts} : Mo_{ts} : O_t \to [B_{ts} \to B_{ts}]$. The expression of axioms operational semantics in terms of functions in Mo_{ts}. Checking of the fact of consistency of operational axioms interpretation with their semantics in Mo_{ts}
DML statements interpretation function construction : $Mo^s_{ts} : P_s \to [B_s \to B_s]$
Verification of commutativity of DML statement mapping diagram (transformation of $Mo^s_{ts}(o_t)$ into $Mo_{ts}(o_t)$)

Fig.1.

2.5 Canonical model for the structural source data models

On the basis of systematic application of the commutative data model mapping method and of the method of canonical model synthesis the following results were obtained [10, 11].

Axiomatic extensions of relational DM equivalent to various source structural DM (network, hierarchical, binary, descriptor) were constructed. An example of axiomatic extension of relational DM equivalent to CODASYL network data model [6] is given in fig. 2. Compound axioms reflect CODASYL sets semantics with MANDATORY and OPTIONAL membership as total and partial functional dependencies.

Simple axioms (for relation R_i; A_i, A_j, A_k- collections of attributes of R_i)

1. Axiom of uniqueness
   UNIQUE A_i
2. Axiom of constancy
   CONSTANT A_i
3. Axiom of definiteness
   OBLIGATORY A_i
4. Axiom of conditional uniqueness
   UNIQUE NONNULL A_i
5. Axiom of order
   R_i[RESTRICTED BY A_i] IS ORDERED[<order>]
   [BY < direction > A_j{, < direction > A_k }]

Compound axioms (for relations R_i, R_j)

6. Axiom of total functional dependency(f.d.)
   $R_j(A_j) \rightarrow R_i(A_i)$
7. Axiom of partial functional dependency(p.f.d.)
   $R_j(A_i) \Longrightarrow R_i(A_i)$
8. Axiom of partial strong functional dependency
   $R_i(A_i) = S \Longrightarrow R_i(A_i)$
9. Axiom of partial functional dependency with initial connection
   $R_j(A_j) = L \Longrightarrow R_i(A_i)$

Fig.2

A set of canonical model facilities obtained as a result of canonical data model synthesis is shown in fig. 3. Application of the structural model mapping technique was demonstrated for the case when a combination of relational and semi-structured models was used as a canonical model core. For the source models, twelve of widely used diverse structural data models of the beginning of 80ies were used (fig. 3). Among them are network data models (including CODASYL

DM), hierarchical data models (including IMS), binary relational data models, semi-structured data models (BASIS). The required data model mappings were constructed and the canonical data model has been synthesized.

An important property of the process of synthesis consists in relatively fast saturation of the canonical data model when consideration of new source data model brings no new axioms on the target DM level. The resulting data model with respect to the known (at the beginning of 80ies) DM is a saturated one. This circumstance allows to consider the resulting model to be a canonical one.

It is important to note that for the denotational semantics as DMM no tools providing for justification of commutativity of DDL and DML mapping diagrams existed. Proof of the commutativity of DDL mapping diagrams was based on the method of structural induction. Proof of commutativity of operator mapping diagrams was based on the rules of equivalent function transformation of the metamodel. Due to absence of any specific tools, construction of a model mapping according to this approach was a hard work.

Generally, for the relatively small number of the source models in the environment, the approach based on manual construction of data model mappings and justification of their commutativity has been practically applicable, though labor consuming.

The results obtained for the structural data model mapping keep their methodological value today. Similar approaches can be applied for other purposes (e.g., common representation of enterprise models, model mapping in MDA).

More advanced technique providing for automatic proof of correctness of model mappings is considered in the next section.

3 Object model commutative mapping technique

The period of 90s can be characterized by active development and use of object models and middleware frameworks intended for the interoperable information systems development (e.g., [18]) , when alongside with new object models (being inherently extensible) new formal languages and program development methods (based on refinement calculus and step-wise refinement technique) appeared. Taking advantage of that, the model mapping and canonical model development methods defined in the previous section have been modified in the following way. As a formal data metamodel instead of denotational semantics the Abstract Machine Notation (AMN) has been applied. AMN provided for manipulation with model-theoretic specifications in the first-order logic and proof of specification refinement [2, 3]. Refinement technique allowed to expand basic definitions of relationships between data types, schemas, data models so that instead of equivalence of the respective specifications it would be possible to reason of their refinement [3]. Specific tools (B-technology [2]) provided for the proof of model mapping diagrams commutativity in a semi-automatic way: proof obligations required for justification of model refinement are generated by B automatically, their proof may require human assistance.

Canonical model facilities	Data models											
	1	2	3	4	5	6	7	8	9	10	11	12
Canonical data model core												
Normalized relations	*	*	*	*	*	*	*	*	*	*	*	*
Hierarchical relations		*				*	*	*		*		*
Positional aggregates							*	*				
Core extension												
Axiom of uniqueness		*	*	*	*				*		*	*
Axiom of constancy		*	*	*	*				*			*
Axiom of definiteness		*	*	*	*				*		*	*
Axiom of conditional uniqueness		*										
Axiom of conditional constancy		*										
Axiom-function											*	
Axiom-partial function											*	
Axiom of order		*	*	*		*	*		*	*		
Axiom-predicate											*	
Axiom of total f.d.		*	*	*					*	*	*	*
Axiom of partial f.d.		*									*	*
Axiom of strong p.f.d.		*										
Axiom of p.f.d.with initial connection		*										
Axiom of total f.d. with backward connection			*									
Axiom of p.f.d. with backward connection											*	
Axiom of stable total f.d.(s.t.f.d.)			*									
Axiom of s.t.f.d. with backward connection			*									
Axiom of of duplex dependency					*	*						

Data models denotation

1.Codd relational DM (1970)
2.CODASYL network DM
3.IMS Hierarchical DM
4.IDS Network DM
5.DM of DBMS PALMA
6.ADABAS DM
7.Descriptor DM of DBMS BASIS
8.DM of DBMS POISK
9.TOTAL network DM
10.Hierarchical DM of DBMS INES
11.Binary relational DM
12.Codd relational DM (1979)

Fig. 3

3.1 AMN as an object metamodel

AMN as a model-theoretic notation makes possible to integrate the specification of the state space and behavior (specified by operations on the states). Specification of the machine state consists in providing of state variables together with invariants - constraints that always should be satisfied. Operations are defined using an extension of Dijkstra's formalism of guarded commands.

The state specification notation is based on a set theory and a typed first-order language with the built-in types and type (sort) constructors. The collection of complex sort constructors includes: cartesian product ($\times$), powerset ($\wp$), set comprehension ($\{x|x \in s \wedge P\}$), relational sort constructors ($s \leftrightarrow t$), functional sort constructors ($s \longrightarrow t$.) Here s,t denote sets, P denotes a predicate. Predicates in the notation are defined using the first-order language. The set of well-formed formulae for that can be defined using the logical connectors and quantifiers. The interpretation of a state of an abstract machine is given by the machine variables assigning to each variable an element in a certain domain.

The operations of the abstract machines are based on the generalized substitutions. Every generalized substitution S defines a predicate transformer binding with some postcondition R its *weakest* precondition $[S]R$ that guarantees the invariance of R after an operation execution. If it is so, one says that S *establishes* R. "Weakest" precondition means that the "initial state" predicate associated with some given "final state" predicate should allow as many states as possible.

A machine N is said to *refine* a machine M if a user can use N instead of M without noticing it. Applying algorithmic and data refinement the refinement machine can be constructed [2]. In AMN it is possible to prove formally that a machine is a refinement of another one using specialized tool. Modeling type and subtype in object environment by an abstract machine and its refinement, we can formally establish the subtyping conditions.

More details on the B AMN can be found in [3].

3.2 Redefinition of data model mapping principles

A notion of *data model equivalence* is redefined as a notion of *data model refinement* introduced in the following manner. Two database states - one in a source model and another one in a target model - are in a refinement order if they are mapped into two states in the *abstract data metamodel* so that an image of a source model state becomes a refinement of an image of a target model state. A type t_s bijectively data refines a type t_t iff the types produce sets of database states of equal power related by bijective dependency in such a way that the states being in one-to-one correspondence are in a refinement order. A schema Ss refines a schema St iff for each type t_s of Ss there is a type t_t in St (St does not contain other types) such that t_s is a refinement of t_t. A data model Ms *refines* a data model Mt iff for each admissible schema Ss of Ms there exists an admissible schema St of Mt such that Ss is a refinement of St.

Data model axiomatic extension principle is redefined as a data type axiomatic extension so that one-to-one mapping θ in the commutative data type

state mapping diagram (that replaces schema mapping diagram) becomes data refinement instead of data equivalence. Instead of DML operator mapping diagram, a commutative diagram of data type behavior is used in which π becomes an algorithmic refinement.

Generally for data model mapping we should construct 1)M_j into an extension of M_i mapping; 2) AMN semantics of M_j ; 3) AMN semantics of the extended M_i. After that we can apply B technology to prove a) the state-based properties of the mapping (commutativity of the data type state diagrams); b) the behavioral properties of the mapping for all type models defined for a particular internal data model. This leads to a proof that M_j is a refinement of the extension of M_i.

Note that data model mapping based on AMN and on a refinement technique is applicable to structured data models as well as to the object data models.

An example of mapping of the ODMG'93 [17] relationship type into the SYNTHESIS association metatype [12] applying refinement technique can be found in [13]. The SYNTHESIS association metatype is a loose, unconstrained type that should be properly extended so that it could be refined by the ODL relationship type (a concrete, built-in type). Data refinement of the data type state diagram has been reached by introducing for the association metatype of the adequate axioms (axiom of partial functional dependency and axiom of order). Mapping of operations has been provided by specification of the relationship operations create, delete, add_one_to_one, remove_one_to_one and traverse in the association metatype. Abstract interpretation of the extended SYNTHESIS association metatype in AMN and a machine corresponding to the ODMG relationship type as a refinement of the association metatype machine were defined and the refinement association has been proved.

3.3 Generic features of the commutative data model mapping approach

The data model axiomatic extension principle, the data model commutative mapping principle, the unifying canonical data model synthesis principle based on the notion of *data model refinement* are exploited to create the necessary ground for the design of semantically interoperable heterogeneous data modeling environment. According to the approach for commutative data model mapping construction, the basic steps of the mapping design should:

- construct the mapping of a source data model type specifications into type specifications of an extension of the canonical data model (including state and behavior mapping);
- provide an interpretation of source data model types in abstract machine notation;
- provide an interpretation in abstract machine notation of the types resulted in mapping of the source data model types into an extension of the canonical data model types;

– justify the state-based and behavioral properties of the type mappings proving that a source data type is a refinement of its mapping to the canonical data model.

It is important to note that the process of design of the canonical data model core axiomatic extensions (state mapping diagram construction) and the process of design of behavior of types in the extended target data model refined by means of the source one (behavior mapping diagram construction) can be separated. Thus, it is allowed to separate and treat independently the process of the canonical unifying data model synthesis from the process of the definition of the types behavior.

The spectrum of data models included into a heterogeneous environment may include object-oriented as well as structured data models. A unifying canonical data model may be synthesized integrating data models of various DBMSs on the basis of the data model refinement conception. This approach can be easily extended to any area where heterogeneous information models are extensively applied.

4 Towards extensible canonical process model

Extensible canonical model development and commutative model mapping techniques were considered so far for the structured and object-oriented models. Very important class of models in the context of enterprise modeling constitutes a class of process models. Enterprises are modeled through their activities defined as concurrent processes. Virtual enterprise specifications are based on the integrated processes of actual enterprises involved into the integrated activities. Processes are implemented as workflows. Workflow technology continues to be developed for its traditional applications of enterprise process modeling and coordination, as well as for component frameworks and inter-workflow interaction. A large number of workflow products (workflow management systems (WFMS)) are commercially available. They apply a large variety of languages and concepts based on different paradigms. Workflow specification is a complex construct highly integrated (correlated) with specifications of other types. Designing a workflow, we should consider semantics of workflow objects and objects involved in them in an integrated way. Here we consider workflows in the process perspective.

In the previous sections we studied how to preserve information and operations while mapping a source model into the target one. Mapping process models, concurrent behavior is to be preserved.

Processes are related to the class of interactive computations [24, 25]. The concept of interactive computations is characterized by non-determinism (the choice of the exhibited behavior is influenced by the computation environment), concurrency (viewed as compositions of communicating concurrent processes), communication between components and the outside world, constraints imposed by an environment on computations.

Classical "effective computability" (in frame of which we were in the previous sections) is purely algorithmic, while interaction paradigm deals with concurrent and distributed computations coordinated by protocols. Well known nondeterministic concurrent systems are based on CCS of R. Milner [16], process algebras [5] and other formalisms which were designed to study communication and interaction in concurrent processes. At the same time, none of existing model cannot serve as an extensible core of the process canonical model due to the lack of the unifying model of concurrency. In the rest of this section we shall report on some steps made for understanding how such model could be created.

In [14] we attempted to show how to apply the compositional approach of object type specification design to the workflow design with reuse. For that we rely on a script-based process specification framework [12]. The script model combined features of the canonical object model with the coloured Petri nets [9]. Notions of types, functions and predicates defined in object calculus are basic constituents of the canonical model for the process definition. It was shown how such process model can be applied to the homogeneous definition of industrially supported workflows. FlowMark and Staffware [8, 21] were used as examples. The script refinement concept attempted in [14] merges conventional algorithmic specification refinement technique considered previously with detecting of process bisimulation equivalence. For the latter, admissible patterns of activities of a script were modeled as process algebra expressions [5] implied by a script model. Basic process algebra with iteration was used for experiments. Complete axiomatization of process algebras [5] provides for equivalent transformation of process algebra expressions and their partial ordering making possible to reason that a concurrent behavior given by a pre-existing workflow specification can be considered as a refinement of the required process behavior.

This analysis showed that coloured Petri net due to its power might be a good choice for the canonical process model, but it is too complicated for the refinement reasoning. Only very simple process patterns of Petri nets can be modeled as process algebra expressions.

Another approach that would provide for rigorous process model mapping is based on a combination of the process specification facilities with the facilities formalizing the refinement concept and providing for proof of specification refinement. Abstract Machine Notation (AMN) and B-technology [3, 2] provide such capabilities. AMN has been considered as a formal model for the object language [12] intended as the canonical model core for the integrated information systems development. In this context creating a combination of a process model with the refinement capabilities of B-technology was considered as a basis for commutative process model mapping. An expression in AMN of the comprehensive CSP [7] capabilities (sequential processes, parallel composition on the arbitrary level, hiding, nondeterministic choice including general choice, assignment operator, main timing primitives of Timed CSP (delay, timeout and time prefixing)) has been defined. Algorithms mapping such process specifications into B-machines have been developed [22]. Such way of process interpretation allowed to construct refinements of process specifications and to prove correctness of refinement us-

ing B-technology. This is a necessary prerequisite for commutative process model mappings as well as for compositional virtual enterprise processes development involving workflow processes as pre-existing components (a refinement of the process specification of requirements by a composition of the pre-existing workflows can be formally justified). The extended AMN notation allows to model reactive concurrent systems operating with complex data types. It is important to note that to justify correctness of process model mapping into the AMN model, the third model, Label Transition System (LTS) has been used. Process specification and the result of its mapping into AMN have been mapped into LTS. Then an equivalence of the resulting specifications in LTS has been demonstrated.

These investigations showed possible approach to the process model mapping development preserving process behavior. Simultaneously, a variety of workflow process modeling facilities currently understood has been systematically defined in [1] as workflow patterns. We can imagine a course of the extensible canonical process model formation as a systematic definition of the canonical model core extensions corresponding to the groups of such workflow patterns so that the canonical model core together with each extension could be refined by a respective group of patterns. Serious investigations are still required to develop a suitable canonical process modeling framework that could cope with strong demand of [1] and combine extensibility with rigorous refinement facilities.

5 Conclusion

Current situation in enterprise modelling is characterised by a large number of enterprise modelling languages and tools leading to heterogeneous and unintegrated definitions of the enterprise and by absence of communication among tools. Various middleware frameworks facilitate technical interoperability of enterprise models. At the same time, taking into account the information modeling semantics in such frameworks is rarely attempted. This paper is focused on intermodel interoperability based on the semantics of heterogeneous modeling facilities. For uniform representation of various modeling facilities in one paradigm a canonical model is needed that should provide for semantically equivalent representations at the canonical level of definitions expressed heterogeneously. To reason that a definition in one language can be substituted by a definition in another one a formal specification framework and a commutative model mapping method are provided. The paper gives a short overview of techniques experienced for such approach separately for *structured*data models with behaviors, for *object* models and for *process* models. Though heterogeneous model mapping and canonical model construction techniques are considered in specific contexts (data base integration, service type re-use and compositions, compositional development of processes), they can be easily generalized for the enterprise modeling environment. Refining model mappings and extensible canonical model construction techniques are necessary pre-requisite for re-use (substitutability), integration and compositionality of information and services in the interoperation and mediation infrastructures.

References

1. W.M.P. van der Aalst, et al Workflow Patterns, Distributed and Parallel Databases, 14(3):5-51, 2003.
2. Abrial J.-R. B-Technology. Technical overview. BP International Ltd., 1992, 73 p.
3. J. -R. Abrial. The B-Book. Cambridge University Press, 1996.
4. Backus J. Can programming be liberated from the von Neumann style ? A function style and its algebra of programs. - CACM, 1978, v.21, N8.
5. J. Bergstra, I. Bethke, A. Ponse, Process Algebra with Iteration and Nesting, Computer Journal, V. 37, N 4, 1994.
6. CODASYL Data Description Language Committee Journal of Development. January 1978.
7. C. A. R. Hoare. Communicating Sequential Processes. Prentice-Hall, 1985.
8. IBM FlowMark, Modeling Workflow, IBM Corporation. September 1994
9. K. Jensen. Coloured Petri Nets: a High Level Language for System Design and Analysis", High level Petri Nets. Theory and Application. Springer Verlag, 1991.
10. Kalinichenko L.A. Methods and Tools for Integration of Heterogeneous Databases.- Moscow, Science Publ., 1983, 423p. (in Russian).
11. Kalinichenko L.A. Methods and tools for equivalent data model mapping construction. Proc. of the EDBT'90 Conference, 1990, Springer Verlag,p.92-119.
12. Kalinichenko L.A. SYNTHESIS: the language for desription, design and programming of the heterogeneous interoperable information resource environment.- Institute of Informatics Problems, Russian Academy of Sciences, Moscow, 1995.
13. Kalinichenko L.A. Method for Data Models Integration in the Common Paradigm. Proceedings of the First East-European Conference, ADBIS'97, St.Petersburg, September 1997.
14. Kalinichenko L.A. Workflow Reuse and Semantic Interoperation Issues. In Advances in workflow management systems and interoperability, A.Dogac, L.Kalinichenko, M.T. Ozsu, A.Sheth (Eds.). NATO Advanced Study Institute, Istanbul, August 1997.
15. MDA Guide Version 1.0.1, OMG, document number: omg/2003-06-01, June 2003.
16. Robin Milner. Communication and Concurrency. Prentice Hall, 1989.
17. The Object Database Standard: ODMG-93. Ed. by R.G.G. Cattell, Morgan Kaufmann Publ., 1994, p. 169.
18. Object Management Group, "The Common Object Request Broker: Architecture and Specification", OMG Document Number 91.12.1, December 1991.
19. Petit M. Methodological clues for the design of a standard enterprise modelling language. EI3-IC workshop on Common Representation of Enterprise Models, 2002.
20. Tennent R.D. The denotational semantics of programming languages.- CACM, 1976, v. 19, N8, p.437-453.
21. Staffware for Windows Graphical Workflow Definer, Staffware plc., 1995.
22. Stupnikov S.A., Kalinichenko L.A., Jin Song DONG Applying CSP-like Workflow Process Specifications for their Refinement in AMN by Pre-existing Workflows. In Proceedings of the Sixth East-European Conference on Advances in Databases and Information Systems ADBIS'2002, September 8-11, 2002, Bratislava, Slovakia.
23. Vernadat F. B. UEML - towards a Unified Enterprise Modelling Language. Proceedings of MOSIM'01, Troyes, France, 2001-04-25/27.
24. Peter Wegner. Interaction as a Basis for Empirical Computer Science. ACM Computing Surveys, 27(1), March 1995.
25. Peter Wegner. Why interaction is more powerful than algorithms. CACM, 40(5), May 1997.