УДК 004.89

Burov Y
National university "Lvivska polytechnica",
Department "Information systems and networks"

A FORMAL MODEL OF KNOWLEDGE REPRESENTATION IN ONTOLOGICAL TASKS MODELLING SYSTEM

In this paper we propose a formalization of ontology-based task execution modelling system. It is built using approach of algebraic systems theory. We show that proposed algebraic system based on multiple domains can be used for ontological models representation of tasks and knowledge elucidation, storage and processing

Keywords – knowledge base, mathematical model, ontology, ontological model

Introduction

Design and implementation of intelligent systems based on the formalization and reuse of knowledge is a promising area of practical application of artificial intelligence in software systems. At the core of such systems is a formalized representation of knowledge about the subject area, for example in the form of ontology. Ontology definition made by Gruber [1] as a specification of a conceptualization leaves open the selection of formal system and language for building such a specification. Not completely resolved are the problems of analisys of the ontological knowledge representation method itself, the definition of its inherent limitations and advantages. In order to resolve those problems, formal models for various aspects of ontological modeling should be built and analysed. So the study of ontological modelling method using mathematical models of ontologies and ontological modelling is important research area.

Analysis of recent research and publications

The use of formal methods for the study of systems that use information and knowledge has a long history. Most researchers use formal methods, based on the algebraic approach, set theory and first-order predicate logic [2, 3, 4].

Codd [2] developed relational algebra based on first-order predicate logic and algebra of sets which was used to build the theoretical foundations of relational databases, including database query language SQL.

In [3] a general algebraic approach to the representation of relational database is developed, including algebraic database model, is analyzed a symmetry and equivalence relations in databases using multisort algebras. The author creates a model of the relational database in the form of algebraic structure using multisort algebras. Author performs an algebraic modeling of operations of databases union and decomposition.

In [4] an algebraic approach is used to modelling of concepts and concepts databases. Abstract data types are used to represent concepts and algebraic operations over data types are defined.

In [5] is proposed an algebraic approach to the construction of models and model transformations are defined as algebraic operations. The authors investigated the possibility of automating the complex inference process using algebraic models. As a result, an executable meta-language was created with purpose to identify, build, evaluate and transform models of complex systems.

In [6] was proposed an approach to software systems construction using interpreted ontological models, the architecture and functioning of simulation system based on ontological models was described. However, the mathematical formalization of the proposed approach, including knowledge representation using ontologies and tasks models, relationships between them and the issues of practical implementation of modeling were not researched.

Purpose of the article

At present, the main research in ontology modeling is focused on declarative ontologies - domain ontologies, and general ontologies [7]. Task ontologies and ontological models constructed on the basis of them are studied insufficiently.

Historically, the tasks ontologies have been developed as a result of scientific analysis of tasks (task analysis). Methods of task analysis are used to define and formalize all factors that affect or are used in the process of solving the problem by an expert. Such methods are widely used for designing interfaces of computer programs, in expert systems, decision support systems [8].

Task analysis is focussed on the analysis and specification of the components of common tasks and problems, determination of its structure and constraints. This allows the expert to better understand the problem, identify possible errors and omissions. Expert can simulate the process of problem solving and task execution and is able to evaluate the results of simulation in order to gain and pass the knowledge to other experts.

The area of task analysis has experienced a significant change with the advent of ontologies. It was proposed to use task ontologies to formalize the concepts and relations for the any given task[9]. Unlike other types of ontologies, such as general or domain ontology, task ontology

- is created for some class of tasks;
- the concept of task goal is important and its formalization mandatory
- the concept of action is intoduced[10] in the context of task execution;
- task ontology modelling environment provides execution (or simulation) of actions;

Tasks ontologies research area is closely related to conceptual modeling, because in the process of building of task ontology expert actually creates a formalized conceptual model for task[11]. An important aspect of both conceptual and ontological modeling is the interaction with domain expert who creates and validates an ontology.

In the process of tasks ontology research were implemented simulation environments allowing users to create and execute ontological models for specific classes of tasks. The most advanced of these environments is CLEPE (Conceptual level programming environment) [7]. However, available research is focussed on studying tasks ontologies for different tasks separately. Also, the problem of task ontologies mathematical modelling and formal representation is not resolved.
The goal of this paper is to build a formal model of knowledge representation in intelligent systems based on tasks ontologies.

Main part. Mathematical model for knowledge representation

The use of ontological models for software systems construction requires the development of mathematical groundwork and the corresponding formal models, which will be used for the formal representation and validation of methods used in intellectual modelling system. To construct a formal model approach we applied an algebra of systems [5], which defines the algebraic system by a combination of several algebraic domains.

Let our application area have n sets of objects:

$$A_1, A_2, \ldots A_n$$

Objects from a single set are classified as instances of particular concept. Those sets are carrier sets for n multisort algebras. Particular exemplars belonging to those sets we will designate as $a_1, \ldots a_n$.

Concepts (entities) domain E. Based on each set A_i we will define an abstract data type E_i

$$E_i = (Name, \Sigma, Ex)$$

Where *Name* – type name, Σ - the signature of multisort algebra, *Ex*- definitional constraints of the type. The signature contains only the set of elements and does not contain operations and relations.

Let's designate particular types as E_i (any type, its name is yet not specified) or E_{name} where *name* – type name, in case if in the context of presentation it is important to refer to specific type. Variables,

which are taking values of specific type we will write as X_{name}, or simply *Name*. Types E_i form a set of algebraic domains *E*, corresponding to concepts of application area.

$$E = \{E_1, E_2, ..., E_n, \}$$

Attributes domain At. Let's define an algebraic attributes domain At on lists of attributes values in form of tuple (key, value). The element of tuple *key* specifies attribute's id, and value – its value. According to [5] for this domain are defined *merge*, *substitute*, *delete*, *interp* operations.

In practice helpful are functions defined on arguments from different domains, such as function of attribute's value selection:

$$F_{selval}(At, key) \to value$$

Boolean domain Cs. This domain includes expressions which are evaluating to Boolean values set {true, false}. Those expressions have operands belonging to different algebraic domains. Let's interpret the elements of Boolean domain as constraints *Cs*. In Boolean domain are defined boolean operations {*and*, *or*, *negation*}, also interpretation operation *interp*, which is used for simplifying and calculating boolean expressions evaluation. An instance of boolean domain is a specific initialized constraint.

Entities with attributes domain T. Let's specify this domain on a set of tuples $< E_i, At_j, Cs_j^* >$. For each i exists only one j, which is a part of this domain element:

$$\forall i \exists^1 j :< E_i, At_j >$$

Each constraint $Cs_j \in Cs_j^*$ is an expression with operands from domain At_j. The operations specified on this domain's elements are operations of meging and splitting {*merge*, *split*}. The operation of entities merging is defined as creation of new entitity having combined attributes and constraints from all parent entities. The operation of entity splitting is reverse to merging. An instance of entity with attributes domain is a fact.

Relations domain Rl. The carrier set of this domain is represented by a set of tuples:

$$\{(T_{1,i} \times T_{2,i} \times ... \times T_{k,i}, At_i^r, Cs_i^*)\}$$

Each structure is a tuple, having cartesian product of algebraic types from domain T, and type from attributes domain At (which defines relation's attributes), and a set of constraints Cs.

Each constraint $Cs_i \in Cs_i^*$ is an boolean expression with operands from domains

$At_{1,i}, At_{2,i}, ..., At_{k,i}, At_i^r$.

In relation domain are defined operations of merging, splitting, substituting relations {*merge*, *split*, *substitute*}. The operations if merging and splitting are interpreted similarly to same name operations from domain *E*. The substitute operation is understood as a reification of relation.

Ontology is a tuple including domains of entities with attributes, relations and Boolean domain constraints describing constraints for entities and relations.

$$On =< T, Rl, Cs^* >$$

Each relation $R \in Rl$ is defined on a set of roles $\{P_1, P_2, ..., P_n\}$:

$$R(P_1, P_2, ..., P_n)$$

In a general case for each role P_k is specified an initialization function $F_{in}^{\ k}$, which defines a subset of entities allowed to substitute a role:

$$F_{in}^{\ k} : P_k \to T_{in}^{\ k} \subseteq T$$

In a simplest case, when $\forall k : | T_{in}^{\ k} | = 1$ for each role there is only one entity type which can initialize this role. In this case in relation we can substitute roles with corresponding entities from ontology: $R(E_1, E_2, ..., E_n)$

The entities from ontology form an hierarchical structure (taxonomy) using inheritance (*is-a*) relation R_{isa}

Binary relation R_{isa} is defined on an ordered pair of roles *Child-Ancestor*:

$$R_{isa}(P_{ch}, P_{pr})$$

Inheritance relation is transitive, so if $R_{isa}(E_1, E_2)$ and $R_{isa}(E_2, E_3)$ then $R_{isa}(E_1, E_3)$ holds.

Let's create a function F_{pr} which for every entity E_j defines an ordered list of its ancestors $(E_1, E_2, ..., E_k)$, so $R_{isa}(E_i, E_{i+1})$ and $R_{isa}(E_j, E_1)$ hold. Also we will define a function $F_{pr}{}^1(E_j)$ which returns an immediate ancestor of E_j entity or empty set $\varnothing$.

An important kind of relation is *Whole-Part* relation (*has-parts*): $R_{has}(P_{wh}, P_{pt})$, where P_{wh} - is the role of *Whole*, a P_{pt} - is a role of *Parts*. A subtype of this relation R'_{has} defines specific entities for *Whole* and sets of allowed entities – for *Parts*:

$$R'_{has}(E'_{wh}, \{E^1_{pt}, E^2_{pt}, ..., E^n_{pt}\}')$$

For the sake of simplicity we will use such notation for such relation representation:

$$E_{wh} = (E^1_{pt}, E^2_{pt}, ..., E^n_{pt})$$

On the other hand relations and models can be also considered as separate entities from ontology. In algebraic type system T those entities are represented by data types which store metadata about those models and relations.

An important part of relation definition in ontology are consistency constraints applied to entities in relation and relation itself. In our system those constraints are represented by set of boolean expressions $Cs_{Rl,i}$ for each relation type Rl_i, which should evaluate to true:

$$\forall cs^i{}_{Rl,i} \in Cs_{Rl,i} : ev(cs^i{}_{Rl}) = true$$

where *ev()* – is an evaluation function.

In some cases constraints are also applied to attributes of entities. Let's consider those constraints as consistency constraints for unary relations.

Let's defne function *TypeParents()* which for every type T_i will return an ordered list of it's supertypes and is a transitive closure of inheritance relation.

$$TypeParents(T_i) = (T^1, T^2 ..., T^n)$$

where T^{i+1} is a supertype for T^i

We will also define function *TypeName()*, which for every data type *T* returns its unique identifier (name, description) of this type. For example for data type T_{MD}, corresponding to model:

$$TypeName(T_{MD}) = "Model"$$

For instances t of every type T we will define function which return its type:

$$Type(t) = T$$

Let's name the multiset of instances of T as $Population(T)$.

$$Population(T) = \{t \mid Type(t) = T\}$$

Let's denote the multiset of instances of type T_i *as* $\hat{t}_i$:

$$\hat{t}_i \mid \forall t_i \in \hat{t}_i : Type(t_i) = T_i$$
$$\hat{t}_i \subseteq Population(T_i)$$

An abstract datatype which correspond to multiset of instances $\hat{t}_i$ we will denote as $\hat{T}_i$

In general case every object o can be identified as belonging to multiple types.

Let's define a function TypeId(), returning a set of types which can be associated with a given object.

$$TypeId(o) = \{T_o^1, T_o^2, ..., T_o^m\}$$

In cases when a semantic interpretation of a given type is important we will show type name as an index. For example, let's denote a model datatype as T_{MD}, a particular model instance as - t_{MD}, and multiset of model instansies as - $\hat{t}_{MD}$.

We will specify a task ontology On_{TS} which is a part of general ontology On and contains objects used for this particular task execution

$$On_{TS} \subseteq On$$

Ontological model *Md* will be defined as three elements tuple:

$$Md = < On_{TS}, Ac_{TS}^*, Cs_{TS}^* >$$

Where On_{TS} – is a task ontology, Ac_{TS}^* - set of actions, Cs_{TS}^* - a set of additional constraints.

Action *Ac* is an entity with attributes (algebraic domain T), which is interpreted as a command to some externale service or command to execute another model. Parameters are specified for this command. Those parameters arte obtained from attribute's values of elements included in On_{TS} (or they are defined as constants). We will introduce an initialization function *InstPar* specified as a set of mappings between action attributes and attribute's values of elements included in ontological model:

$$InstPar : (Ac_{l,i}, pkey_{l,i}) \rightarrow SelValue(At_{l,j}, key_{l,k})$$

$$Ac_{l,i} \in Md_l, At_{l,j} \in On_{TS}^l \in Md_l$$

In the table below are shown algebraic domains used in mathematical model

Table. Algebraic domains of model

Domain	Carrier set	Operations
Entities (n domains) – E	$\{a_i \mid Type(a_i) = E_i\}$ facts if specified types	
Attributes – At	$\{(key, value)^*\}$	{merge, substitute, delete, interp}
Boolean domain Cs	{true, false}	{and, or, negate, interp}
Entities with attributes –T	$\{(E_i, At_j, Cs_j^*)\}$	{merge, split}
Relations	$\{(T_{1,i} \times T_{2,i} \times ... \times T_{k,i}, At_i^r, Cs_i^*)\}$	{merge, split}
Ontology	(T, Rl, Cs^*)	{merge, split}
Ontological models	$\{(On_{TS}, Ac_{TS}^*, Cs_{TS}^*)\}$	

The structure and working of the modelling system

The modelling system, which implements the proposed approach consists of the following components: (Fig.) Information base contains facts about objects and events of the outside world required for executing tasks by the system. All facts are semantically interpreted, that is presented as objects of certain types defined in ontology . Facts in the information base are ordered by time, allowing to track the state of an information in arbitrary time moment or analyse effects of some change. The information base, ontology and model repository together form a knowledge base system.

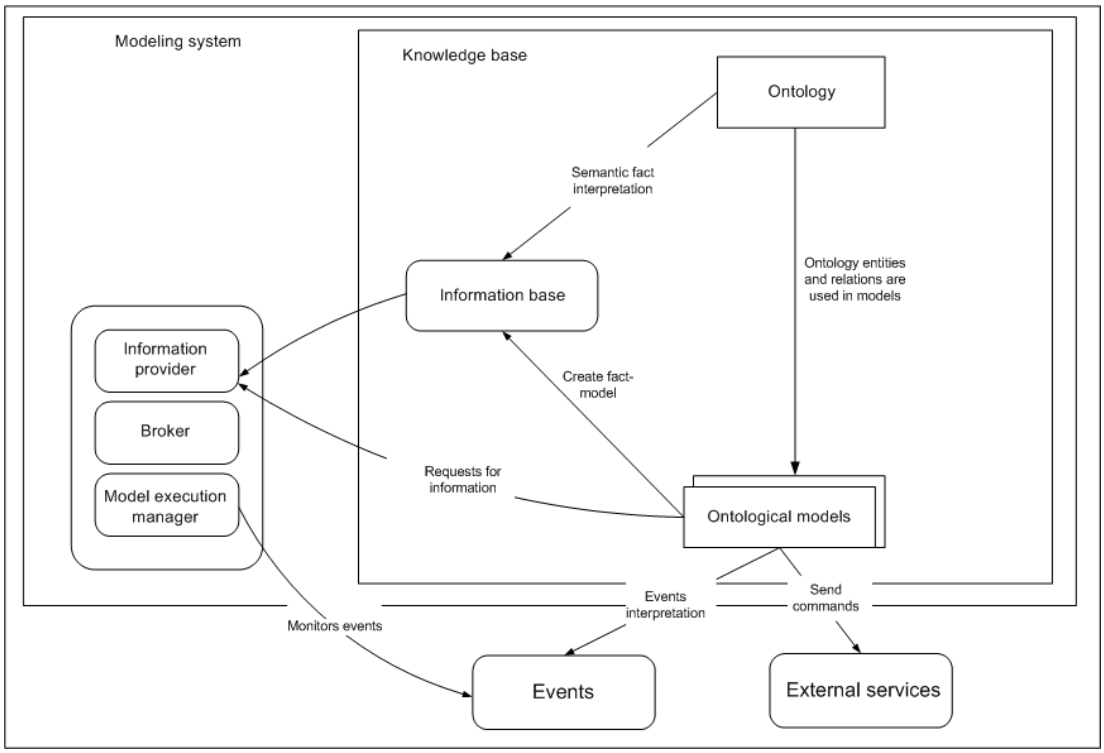

Fig. The structure of modeling system

Ontology contains a domain model presented as a taxonomy of classes. This creates the possibility of unambiguous interpretation of all objects from information base, identify common attributes and properties for them. In addition, the ontology includes relations, rules and constraints that are common for a given domain.

When describing complex ontology rules and constraints references to models can be used including dynamic constraints, i.e. the value of which depends on the state of information base or the outside world.

The system reacts to a defined range of events in the external world and handles this by creating new or modifying existing facts. In order to process those events, a suitable model shoul be developed. The important components of the modelling system are services which ensure the execution of models, their interactions, obtaining the required information from external sources.

In this way the *Model Execution Manager* starts, and stops models, monitors the usage of resources. The *Broker* finds relevant models fot task at hand, initializes them, preparing for execution. *Information provider* on model's request looks for needed data in external sources and semantically interprets search results.

Models perform operations according to their logic, and send requests to extermal services. For example, those services can be operational system services, enterprise information system services built according to SOA guidelines [12] or arbitrary web-services.

Processimg knowledge in modelling system

Ontological models and their relations form a network of type T_{NMD}, which is defined as a set of model multisets of different model types $\hat{T}_{MD}$ and their relations $\hat{T}_{RMD}$.

$$T_{NMD} = \{\hat{T}_{MD}, \hat{T}_{RMD}\}$$

Differently from ontology entities, models don't form a strict hierarchy, but form a dynamic network where relations and model themselves can change reflecting the process of learning, changes in external world or even in process of particular task execution.

Each model can be in active or passive state. Accordingly, the set of fact- models can be split in two subsets of active and passive models:

$$Population(T_{MD}) = \hat{t}^{ac}_{MD} \cup \hat{t}^{ps}_{MD}$$
$$\hat{t}^{ac}_{MD} \cap \hat{t}^{ps}_{MD} = \varnothing$$

where $\hat{t}^{ac}_{MD}, \hat{t}^{ps}_{MD}$ are multisets of active or passive fact-models.

Active model is a model initialized by information from specific context. Models goes active on command from other models or when specified events occur. Active models are used for executing current tasks in system or interpreting knowledge. If active fact-model is no longer needed (result obtained, goal reached) then fact model goes into passive state and is archived.

Model relation (interaction) T_{RMD} - is a data type which reflects an activation relation. It is used in making decision about model activation.

Let's consider the model activation process in more detail. Model activator initiates activation process when in order to execute its main task it needs to execute helper-tasks represented by other models. For example, if input data obtained by model from context are not fully defined, it activates other models to fill data gaps. This may require additional searches in databases or web, or even asking human expert.

For each type of model relation T^i_{RMD} exists a class of tasks T^j_{PR} which should be executed in process of model interaction. Additionally for class of tasks T^j_{PR} exist a set of models $\hat{T}^j_{MD}$ which can be used for executing tasks of this class.

During model interaction are executed sequentially the tasks of relevant models definition, optimal choice of a model-candidate from the set of relevant models and model-candidate initialization.

The relevance function is a mapping of current context t_{CON} and the set of alternatives $\hat{t}_{MD}$ into Boolean set {true, false}

$$F_{rel} : (t_{CON}, \hat{t}_{MD}) \to (true, false)$$

Relevance function allows to form a set of relevant models:

$$\hat{t}^{re}_{MD} \subseteq \hat{t}_{MD}$$

i.e. models t^{re}_{MD} for which holds

$$F_{rel}(t_{CON}, t^{re}_{MD}) = true$$

If the set of relevant models is empty, modeling system informs model-activator that required task cannot be executed.

In process of optimal choice task execution in a set of relevant models is selected one model t^{op}_{MD}. The usage of this model maximizes a selection function F_{ch} taking in consideration selection criterea $\hat{t}_{CR}$ and context t_{CON}.

$$F_{ch}(t^{op}_{MD}, \hat{t}_{CR}, t_{CON}) \to \max$$

Initialization function F_{in} maps current context t_{CON} to a set of selected model's attrubutes values – t_{VSL}.

$$F_{in} : t_{CON} \to t_{VSL}$$

In summary let's define T_{RMD} as a set:

$$T_{RMD} = \{T_{PR}, \hat{T}_{MD}, F_{rel}, F_{ch}, F_{in}\}$$

Model type T_{MD} includes schema type T_{SCM} and realization type T_{IMD}:

$$T_{MD} = \{T_{SCM}, T_{IMD}\}$$

The schema of model specifies its structure, components, defines rules and constraints of model usage and also a list of possible interactions and operations. Schema is a part of model externally visible. It is used for interactions with model.

The shema of model has slots $\hat{T}_{SLM}$, their relations $\hat{T}_{RSM}$, rules $\hat{T}_{RUM}$, constraints $\hat{T}_{CSM}$, and operations $\hat{T}_{OPM}$:

$$T_{SCM} = \{\hat{T}_{RO}, \hat{T}_{RRO}, \hat{T}_{RUM}, \hat{T}_{CSM}, \hat{T}_{OPM}\}$$

Slot in model is an attribute - role. For each slot is defined function F_{RG} which specifies a set of types $\hat{T}_{CL}^{RG}$, which instances are allowed to initialize this slot:

$$F_{RG} : T_{SLM} \to \hat{T}_{CL}^{RG}$$

For each slot is defined rules and constraints within slot's scope – $\hat{T}_{RUSM}$, $\hat{T}_{CSSM}$, operations on slot's values $\hat{T}_{OPSM}$:

$$T_{SLM} = \{\hat{T}_{CL}^{RG}, \hat{T}_{RUSM}, \hat{T}_{CSSM}, \hat{T}_{OPSM}\}$$

The relation of slots T_{RESM} is specified bu a set of slots it links $\hat{T}_{SLM}^{resm}$, and a set of ontology entities used for semantic relation interpretation – $\hat{T}_{CL}^{resm}$, the set of models used for operations with relation – $\hat{T}_{MD}^{resm}$.

$$T_{RESM} = \{\hat{T}_{SLM}^{resm}, \hat{T}_{CL}^{resm}, \hat{T}_{MD}^{resm}\}$$

For each instance of relation, linked slots belong to model's slots:

$$\hat{t}_{SLM}^{resm} \subseteq \hat{t}_{SLM}$$

Model relation corresponds to one of relation types specified in ontology On:

$$TypeEn(T_{RESM}) = E_{RESM} \in \overline{E}$$

Model, describing relation, belongs to a set of models in system:

$$t_{MD}^{resm} \in Population(T_{MD})$$

Let t'_{BFC} - to be a state of information base. We will define a type for a goal T_{GL} for which every its instance is a specification of a set of information base states where goal is achieved:

$$t_{GL} = \hat{t}_{BFC}^{GL}$$

For goal definition it is handy to define goal function which allows to check whether in particular state t'_{BFC} goal Gl is achieved:

$$F_{gl}(t'_{BFC}) = \begin{cases} true \mid t'_{BFC} \in t_{GL} \\ false \mid t'_{BFC} \notin t_{GL} \end{cases}$$

Goal function can be specified, for example, as a ordered list or assertions $\hat{t}_{ASR}$ about information base facts which can be verified.

$$F_{gl}(t'_{BFC}) = \hat{t}_{ASR}(t'_{BFC})$$

where $t_{ASR}(t'_{BFC})$ – is an assertion about values of atrributes of facts and their relations inn state t'_{BFC}.

Each assertion $t_{ASR}(t'_{BFC})$ is a function defined on Boolean set {true,false}:

$$Range(t_{ASR}(t'_{BFC})) = \{true, false\}$$

Where function *Range(f)* specifies the area of function definition f.
So,

$$F_{gl}(t'_{BFC}) = true \Leftrightarrow \forall t_{ASR}(t'_{BFC}) \in \hat{t}_{ASR}(t'_{BFC}) : t_{ASR}(t'_{BFC}) = true$$

Executable ontological models are intended for resolving specific tasks defined by their goals. For simplification of model search it is advisable to organize information about models as an ontology of goals $OnGl \subset On$, having their own entities and relations. In this ontology we will define model categiries according task classes resolved. So, for example we can define classification models, algorithmic nodels, access control models etc. The information about fact-models is used by *Model Execution Manager*. The *Broker* is using goal ontology to find model suitable for current task execution.

Conclusion

Research and development of mathematical models of intellectual systems using ontologies and task models allows to understand theoretical aspects of those systems, elucidate their constraints, provide validation and verification for systems of ontological models.

1. Gruber, T.R. A translation approach to portable ontology specifications. / Gruber, T.R. // Knowledge Acquisition. - 1993. - vol 5. - P.199-220.. 2. Codd E.F. Relational Completeness of Data Base Sublanguages/Codd E/F//Database systems.- 1972, vol 54, #2.-P.65-98. 3.Плоткин Б.И. Универсальная алгебра, валгебраическая логика и базы данных /Плоткин Б.И.. – М.:Наука, 1991.-446с. 4. Бениаминов Е.М. Алгебраические методы в теории без данных и представлений знаний./ Бениаминов Е.М..- М.: Научный мир, 2003г.- С.184. 5. Koo B Algebra of systems: a metalanguage for model synthesis and evaluation./ Koo B, Simmons W. //IEEE Transactions on systems, man and cybernetics, vol 39, N 3, may 2009 6. Буров Є.В. Концептуальне моделювання інтелектуальних програмних систем: монографія/ Є.В. Буров.- Львів: Вид-во Львівської політехніки, 2012.- 432 с. - ISBN 978-617-607-189-1 7. Mitsuri Ikeda. Task ontology: Ontology for building conceptual problem solving models/ Mitsuri Ikeda, Kazuhisa Seta, Osamu Kakusho, Riichiro Mizoguchi. // Workshop note of application of ontologies and problem-solving meyhods, ECAI98, 1998 8. Johnson, Peter. Task-Related Knowledge Structures : Analysis , Modelling and Application / Johnson Peter, Johnson Hilary, Waddington Ray, Shouls, Alan// Knowledge Creation Diffusion Utilization, 1988.- P. 35-62 9. Van Welie Martin. An Ontology for Task World Models/ Van Welie, Martijn, Van Der Veer, Gerrit C Eliëns // Methods, v. 98, 1998.- P. 57-70. 10. Raubal Martin. Ontology-Based Task Simulation/ Raubal M., Kuhn W. // Spatial Cognition and Computation, v.4, 2004 .- P. 15-37 11. Seta Kazuhisa. Building ontologies for conceptual model management/ Seta Kazuhisa, Koyama Kazuya, Hayashi Yusuke, Ikeda Mitsuru// WSEAS Transactions on Information Science and Applications, v.3, 2006 .- P. 546-553. 12.Erl, T. SOA Principles of Service Design. / Erl, T. - Prentice Hall, 2007