ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

Rainer Manthey and Viacheslav Wolfengagen (Eds)

Advances in Databases and Information Systems 1997

Proceedings of the First East-European Symposium on Advances in Databases and Information Systems, (ADBIS'97), St Petersburg, 2-5 September 1997

Appropriateness of the Activity-centred Graphs in Information System Modelling

Alexandra Galatescu

Published in collaboration with the
British Computer Society

©Copyright in this paper belongs to the author(s)

ISBN: 3-540-76227-2

Appropriateness of the Activity-centred Graphs in Information System Modelling

Alexandra Galatescu

ICI- Research Institute for Informatics

8-10 Averescu Avenue, 71316 Bucharest 1, ROMANIA

Fax: 40 1 312 85 39, E-mail: agal@std.ici.ro

Abstract

We emphasize some abilities of a representation formalism, called activity-centred graphs (ACGs), in complex information system modelling. Derived from the conceptual graph formalism, ACGs offer linguistic, structured and logically consistent means for concept and process representation. They respond to two main conclusions: 1) natural language is the ideal metamodel, able to master the heterogeneity of the existing data/knowledge/process models and 2) a structured and logic-based language is needed to master the complexity of the today's information systems. ACG linguistic features are integrated with techniques like declarative and procedural structuring, modularity and encapsulation. As complex inferences and knowledge retrieval technique, we define the thematic views over ACGs and exemplify them for an event detection case. We also motivate the ACGs 1) for their unifying role relative to the data, functional and control perspectives of the information systems, 2) for their contribution in supervising the concept and process interoperability, 3) for their impact in code and repository reusability.

1 Introduction

Without an exhaustive presentation of the activity-centred graphs (ACGs), we try to point out here their abilities in information system modelling, based on our present experience in multidatabase modelling (inside of a project on developing a multidatabase CASE environment).

The activity-centred graphs have derived in our research from the classical conceptual graph (CG) formalism (see [10, 11, 7, 1-3]), that tempted us with its flexible, linguistic representation means, with its logical foundation and with its elaborate rules to graph formation and coreferencing. Our modifications (with consequences mainly in structuring the modelling/execution processes and in code and repository reusability) consist of 1) the transformation of the role-like relationships, implemented by the conceptual graphs, into operator-like relationships among concepts and, then, 2) the combination of the ACGs with the classical CGs in, what we called, *ACGs with qualified concepts*.

The activity-centred graph formalism is our representation solution to the multidatabase modelling, integration and execution inside complex, distributed and evolving information systems. Although these graphs have been first created for this particular purpose, we advocate their universal appropriateness, from information system modelling and execution to natural language processing.

Among the *characteristics and requirements of a today's information system*, some of the most difficult we need to cope with are:

- the heterogeneity of the data/knowledge models, representations and languages distributed over the heterogeneous hardware and software platforms;
- the perspective of future conceptual extensions with new types of concepts, structures, relationships, operators etc, in order to represent and process more and more sophisticated aspects of the real world;

- the unification of the three perspectives (data, functional and control perspective) of an information system (see [8]). *Data perspective* aims at modelling the essential information that one needs to represent in a system. *Functional perspective* aims at modelling the functions and data flow between functions. *Control perspective* aims at modelling the dynamic, time-dependent behaviour of a system.
- an increasing need for code and repository reusability inside and among information systems;
- a simple user interface, close to human reasoning and understanding.

In Section 2 we present the basic definitions and notations of the activity-centred graph formalism. In Section 3, we motivate our proposal, mainly following the above enumerated requirements of the information systems.

2 Controlled Diagrams of Activity-centred Graphs

2.1 Basic Definition of an Activity-centred Graph

We define an *activity-centred graph* as a star graph (see Fig. 1(a)) composed of:

- two kinds of *nodes*:
 - an *operation*, that abstracts a static (declarative) or dynamic (run-time) activity, and
 - more *concepts*, that define or describe the operation. A concept may be *primitive* (represented by a value) or *structured* (defined by another activity-centred graph). As concepts we may represent: objects, properties, abstract knowledge or types, events, states, points in time, expressions, structures etc.
- *conceptual connectives* (concept-operation links) meaning the *roles* of the concepts *relative to the operation*. Each conceptual connective represents an additional meaning of a concept and, also, a label for the graphic conceptual link (arrow or line) between the concept and the operation. As acronyms and meanings of these roles, we propose: AGNT (agent of the activity, i.e the person or the object that produces the activity), PTNT (patient, i.e the object the activity operates upon), RSLT (result of the activity), INST (instrument to achieve the activity), RCPT (recipient, i.e the person or the object receiving the result of the activity), LOC (location of the activity or of an involved concept), CHRC (characteristic of the activity or of an involved concept), SRC (source of an activity or its initial state), DEST (destination of an activity or its final state), PART (part of another concept meaning a whole), TIME (point in time when the activity begins), DUR (duration of the activity), EVNT (event that stimulates the activity execution, represented as a concept), DSCR (activity description, represented as a concept), STAT (the state of the activity), CAUS (activity cause, represented as a concept), GOAL (activity goal, represented as a concept). The list may be enlarged, depending on the specific domain which they are used for.
 These roles have linguistic synonyms such as: 'by' for AGNT, 'of', 'upon' for PTNT, 'into' for RSLT, 'with' or 'through' for INST, 'at' for TIME, 'when' for EVNT, 'as', 'of' for CHRC, 'in', 'on', 'over' for LOC, 'from' for SRC, 'to' for DEST, 'of', 'out of' for PART, 'like' for DSCR, 'for' for RCPT, 'for' for DUR, 'in', 'as' for STAT, 'because', 'because of' for CAUS, 'to', 'in order to' for GOAL.
- *declarative constraints*, which are domain specific constraints:
 - *applied to the operation*, consisting in the 'modal' verb (e.g. 'must' or 'may', that specify the compulsory or optional activation of the operation during the process execution);
 - *applied to the concepts*, consisting of:
 - * *concept type*, that represents an abstraction for the individual concept (e.g. PERSON may be the type of 'John Smith'). The 'concept' is an instance (which for the classical CGs is called 'referent' or 'individual') of a concept type.

Figure 1 Graphic (a), Linguistic (b), Linear (c) Notation and λ -definition (d) of an Activity-centred Graph

- * *concept* (universal or existential) *quantification* by the symbols: $\forall$ for 'any', $\exists$ for 'must exist' and $\exists?$ for 'may exist', meaning the compulsory or optional instantiation of the concept types they precede.
- * *plural type of the concept*. Each concept may be either *singular* (an instance) or *plural* (composed of more instances). A plural concept may be either *collective* (whose elements represent all together the respective concept) or *distributive* (whose elements may separately represent the concept and participate in the graph). We denote by C{ } the collective plural and by D{ } the distributive one. If necessary, we indicate the *plural cardinality* by C@n{ } or D@n{ }. Between parentheses, concrete values of a primitive plural concept may be inserted, separated by commas.
- * *concept subtyping*, that endows each concept type with lower-level meanings in the modelled domain, generally not disjunctive and composing a lattice for each concept type. In our view, the concept subtyping may be achieved in one of the following three ways: 1) by *concept restriction* to particular values, 2) by *concept specialization* to specific roles in the modelled domain, 3) by *concept renaming* with synonyms in other languages or ontologies.

The auxiliary verb 'do' in Fig. 1 ((a), (b)) is just a formal indication on the activity-oriented meaning of the representation. It helps for the linguistic interpretation of the graph (Fig. 1 (b)). Each operation has a linear notation (see Fig.1 (c)) and a λ -definition as in Fig. 1 (d).

We denote the concepts by [concept_type: individual], the operation by (OPERATION) and the conceptual connectives by <<role>>. The graphic conceptual links (by arrows or lines) specify the *orientation of the connected concepts in the operation signature*. The single-headed arrows specify the input or output concepts and the double-headed arrows specify the input-output concepts. The local concepts are suggested by a simple line.

2.2 Concept Qualifying. Attributive Concepts and Graphs

Using pure ACGs, we are forced to qualify concepts by means of operations, overloading the representation. Recently, we have overcome this defect, by separating the *active concepts* (direct participants in operations)

from the *attributive concepts* (that only qualify the active concepts) and by correlating them using role-like relationships (which we called *attributive roles*), as in the classical CGs. So that, an active concept may be qualified by an attributive one using an *attributive graph* like (see also Fig. 2):

$$[attributive_concept] \xrightarrow{ROLE} [active_concept]$$

The attributive roles may be selected from a system-predefined set of roles or may be user-defined roles.

For procedural purposes (in the case we need to control the concept qualification), this kind of link may be simulated by means of an ACG that describes a generic (SETTING) operation:

$$[active_concept] \xrightarrow{<<PTNT>>}(SETTING) \\ \quad \quad \quad \begin{array}{c} <<ROLE>> \\ \downarrow \\ [attributive_concept] \end{array}$$

The attributive role *ROLE* in the attributive graph becomes here a concept-operation link, i.e the role of the concept in the *SETTING* operation.

In our opinion, interleaving these two kinds of graphs (ACGs and the attributive CGs) and managing them by, at least, contraction, expansion, join and coreference (see below) offer a complete, simple and universal representation solution to the today's complex applications.

2.3 Defining Structured Concepts by Activity-centred Graphs

For defining structured concepts, in our view, the general definition of an activity-centred graph has the following specific features: 1) the defined concept has always the role of recipient (denoted by $<<RCPT>>$) relative to the *definition operation*, whatever is that operation; 2) the other concepts are *definition (attribute) concepts*, having *definition roles* relative to the definition operation and, implicitly, relative to the defined structured concept; 3) each definition operation has the same signature (concept types and definition roles) for all the structured concepts instantiating that type. The λ -definition of a structured concept (a particular case of the expression in Fig.1(d)) is:

$$STRUCTURED_CONCEPT = (\lambda x)(x_2, \dots, x_p)(\forall x)(STRUCTURED_CONCEPT(x) \wedge (RCPT(x)) \supset \\ (\exists x_2) \dots (\exists x_p)(CONCEPT_2(x_2) \wedge role_2(x_2) \wedge \dots \wedge (CONCEPT_p(x_p) \vee NULL) \wedge (role_p(x_p) \vee NULL)) \\ DEFINITION_OPERATION(x, x_2, \dots, x_p))$$

where the bound variable x addresses an instance of the structured concept and *NULL* indicates the optional existence (instantiation) of a concept.

In [4], we used activity-centred graphs to define complex, multiple, group and list concepts, as well as subtypes by restriction, specialization and renaming, all needed for the multidatabase modelling and integration. Those definition operations are comparable with the descriptive operators introduced by various conceptual models or knowledge representation languages because of their utility. What we consider important in our representation is the unitary and flexible way to describe and process various definition operations, by unifying the definition operators with the invariant roles of the involved concepts in each definition, relative to the defined structured concept. It impacts on: 1) the natural definition of new types of structured concepts, by implementing new definition operators and 2) the processing of each structured concept only depending on the operation that defines it and on the invariant roles of the concepts in that operation, hence independently of their specific type and meaning.

2.4 Contraction, Expansion, Join and Coreference of the Activity-centred Graphs. Declarative Encapsulation

Relying on the λ -definition of its main operation and for encapsulation purposes, any ACG may be contracted and represented either 1) by the type of a defined structured concept, in the case of a definition operation or 2) by the name of the operation, in the case of a domain specific operation. Expansion is the reverse operation

Figure 2 Join and Coreference Links (dashed lines) Representing the Situation:
 "The diligent boy John goes to school to learn algebra. School is near his home"

relative to the contraction of a graph. We may expand either 1) a structured concept into its graphic definition or 2) an operation into its graphic signature (activity-centred graph).

When two graphs have a common concept or, even, a common subgraph (e.g. in the case of a common structured concept), we may operate a (maximal) join between the two graphs, by representing the common component just once and simplifying the intended diagram. We may join the graph around a domain specific operation with a graph around a definition operation or we may join the graphs representing two domain specific operations.

Two concepts that refer to the same individual are *coreferent*. If they belong to different graphs, those graphs are connected by a *coreference link* between the coreferent concepts.

If necessary, for the representation expressiveness and interpretation, the two graphs (the main graph and the coreferent or joint one) may be represented (and also stored) together. The coreferent or joint graph will be nested in the main graph and the operation that defines it will be a *collateral* operation in the main graph.

In the example in Fig. 2, the graphs (I), (II) and (II), (III) are coreferent because the concept [BOY] is coreferent with the concept [PERSON]. The graphs (I) and (II) are joint on the concept [HOME]. The graphs (II) and (III) are joint on the concept [SCHOOL]. Without the contextual correlation between them, the graphs (I), (II), (III) may be considered independent of each other. The operation (LEARN) may be considered the purpose of the operation (GO). The operation (HAVE) may be represented and stored as collateral operation for the operation (GO). We underline the difference between the qualifying role CHRC of the attributive concept [DILIGENT] relative to the active concept [BOY] and the role $\llcorner CHRC \llcorner$ of the concept [DISTANCE] relative to the operation (GO).

2.5 Controlled Diagrams of Activity-centred Graphs. Procedural Encapsulation

In order to interactively control any process described by ACGs, we introduced a procedural semantics over ACGs, based on two kinds of statements:

- *executable statements*, represented by contracted operations describing a process. They compose hierarchies or hierarchical diagrams of operations, as in Fig. 3.
- *control statements* represented by interoperation connectives (IOCs), as procedural constraints upon ACGs describing the executable operations.

The interoperation connectives suggest the sequential, iterative and conditional control of the operations together with their stepwise refinement, their logical or semantic correlation and their dynamic character. They impose the *constraining rules* on the process execution and may be classified as follows:

- *operation semantic grouping*, represented by the connective GROUP;
- *operation refinement*, inferred from connectives that specify:
 - *semantic specialization* of a parent operation, by SPEC,
 - *compulsory or optional execution* of the child operations, by MUST or MAY;
 - *alternative* (but not necessarily disjunctive) *execution* of the child operations, by CASE.

Figure 3 Hierarchy (a) and Hierarchical Diagram (b) of Contracted Operations Controlled by Interoperation Connectives (IOC). $IOC \in \{GROUP, CASE, SPEC, MUST, MAY, THEN, BFOR, AFTR, AND, OR, XOR, NOT, DO, RSLT, IF - THEN - ELSE, WHILE - DO, MUSTREPEAT, MAYREPEAT, EVNT, GOAL, DSCR, TIME\}$

- *execution timing*, by THEN, BFOR, AFTR between two operations;
- *(exclusive) disjunctive execution* of two operations, by OR (XOR);
- *execution conjunction* of two operations, by AND;
- *execution negation* of two operations, by NOT;
- *imperative execution* of the operation that follows DO or RSLT connective;
- *conditional execution*, by IF-THEN-ELSE, depending on a predicate condition composed of operations and logical operators;
- *conditional repetition*, by WHILE-DO, depending on a predicate condition on operations;
- *unconditional, compulsory or optional, repetition*, by MUST REPEAT or MAY REPEAT;
- *purpose subjunctive execution* of an operation, by GOAL. GOAL is preceded by an operation whose execution is motivated by the execution of another operation (which we call *subjunctive operation of purpose*) that follows GOAL.
- *operation description*, by DSCR, that introduces the description of an operation as one or more correlated operations;
- *execution stimulation*, by EVNT, that introduces an *event operation* stimulating another operation;
- *execution starting point in time*, by TIME, that introduces the starting point in time of an operation as a *time condition* (a predicate condition of operations, whose true value is automatically associated to a certain point in time).

The logic introduced by each interoperation connective has been naturally translated, in our project, into corresponding rules in predicate calculus and automated accordingly (see [5,6]).

For a gradual understanding of a process, the diagrams of operations are structured in procedural modules, whose encapsulation is achieved by procedural contraction, expansion and nesting (see also [5,6]).

2.6 Thematic Views over Activity-centred Graphs

Besides direct interrogations upon concepts and operations described by ACGs, we also use thematic interrogations by means of *thematic views* upon ACGs. They help us to solve complex inference upon ACGs and to refine the inference goals according to semantic criteria.

2.6.1 Basic Definition of the Thematic View over Activity-centred Graphs

We call *subject*, a domain specific entity (e.g. in multidatabase modelling, a schema, class, attribute, method etc). We call *theme*, the range of interesting knowledge on one or more subjects, at a very moment during a process execution (e.g. in the multidatabase modelling process we need themes like: structure, semantics, behaviour, topology etc). We call *thematic view* about a specified theme and on some specified subjects, the overall acquired knowledge complying with that theme and describing those individual subjects.

Let be 'theme' a specified theme, 'subj' an individual of a specified subject type, $\{G_i\}_{i=1:n}$ the ACGs comprising the knowledge that satisfies 'theme' and $\{O_i\}_{i=1:n}$ the operations they describe. They may be definition or domain specific ACGs.

On each G_i , we first apply an *intensional projection* operation (Π), by extracting from G_i and correlating just those concept types (CTs) (or their subtypes) that match with 'theme' for that subject type. Implicitly, one of these types (say CT_i) is the type that abstracts 'subj'. So that, we define 'theme' on G_i by a predicate expression of all theme-compliant conceptual tuples extracted from the graph G_i :

$$theme(G_i)[CT_i] = \Pi_{theme} G_i = pred_expr\{O_i(CT_i, CT_{i_1}, \dots, CT_{i_k})\}$$

When G_i is a definition graph, instead of O_i , the formula will contain the name of the structured concept defined by G_i .

On the extension of each conceptual tuple in the predicate expression ' $theme(G_i)[CT_i]$ ', we then apply the *extensional selection* operation (σ), by extracting the corresponding value tuples that instantiate 'theme' for the given subject 'subj'. We then get:

$$theme(G_i)[CT_i : subj] = \sigma_{subj} (\Pi_{theme} G_i) = pred_expr\{O_i(subj, v_{i_1}, \dots, v_{i_k})\}$$

In the case of more subjects, the selection operates on all subjects at once:

$$theme(G_i)([CT_{j_1} : subj_1], \dots, [CT_{j_p} : subj_p]) = \sigma_{subj_1, \dots, subj_p} (\Pi_{theme} G_i) = pred_expr\{O_i(subj_1, \dots, subj_p, v_{i_1}, \dots, v_{i_k})\}$$

The thematic view on a given subject is, in the general case, a predicate expression, whose operands (expressions of value tuples as those extracted before) are logically connected by logical operators.

$$view(theme, subj) = pred_expr\{theme(G_i)[CT_i : subj]\}_{i=1:n}$$

Because each concept involved in a thematic view has an invariant role relative to a certain operation, we can extract each theme independently of a specific domain by a formula like:

$$theme(G_i)[role_i] = \Pi_{theme} G_i = pred_expr\{O_i(role_i, role_{i_1}, \dots, role_{i_k})\}$$

Thematic views have multiple applications. In the next section, we exemplify their usage for an event detection case.

2.6.2 Example of Event Detection using a Thematic View

Starting with the simple example in Fig. 2, we are trying to supervise the event that stimulates John to go to school, that is to answer the question "When does John go to school?".

Suppose that John goes to school in one of the following three situations:

- when an *event operation* stimulates him: Mother urges John to go to school.
- when an *event concept* has a certain state/value: When homework is finished, John goes to school.
- when a *time condition* becomes true: No later than twelve hours, John goes to school.

These situations will be represented by extending the graph that describes the operation (GO) in Fig. 2 and by representing the new graph that describes the event operation (URGE) (in the linear notation):

(GO)

```
<<AGNT>> ∀[BOY: 'John']
<<SRC>> ∃[HOME]
<<DEST>> ∃[SCHOOL]
<<EVNT>> ∃?[HOMEWORK_STATE: 'finished'] Event concept
<<TIME>> ∃?[MAXIMUM_TIME: 12H] Time condition
```

(URGE) *Event Operation*

```
<<AGNT>> ∃[MOTHER: 'Mary']
<<PTNT>> ∃[CHILD: 'John']
```

In order to supervise the events that stimulate John to go to school, we need to create a thematic view about the *theme* *GO_TO_SCHOOL* on the *subject* John. We synthetise the construction of this thematic view, according to the general definition presented in the previous section:

Theme: *GO_TO_SCHOOL*

Subject: John

Graphs selected for theme: GO and URGE

Intensional Projection:

$$GO_TO_SCHOOL(GO) [BOY] = GO(BOY, HOMEWORK_STATE) \vee GO(BOY, MAXIMUM_TIME)$$

$$GO_TO_SCHOOL(URGE) [CHILD] = URGE(CHILD, MOTHER)$$

Extensional Selection:

$$GO_TO_SCHOOL(GO) [BOY: 'John'] = GO([BOY: 'John'], [HOMEWORK_STATE: 'finished']) \vee \\ GO([BOY: 'John'], [MAXIMUM_TIME: '12H'])$$

$$GO_TO_SCHOOL(URGE) [CHILD: 'John'] = URGE([CHILD: 'John'], [MOTHER: 'Mary'])$$

Thematic View: WHEN John goes to school

$$WHEN(GO_TO_SCHOOL, John) = GO_TO_SCHOOL(GO) [BOY: 'John'] \vee GO_TO_SCHOOL(URGE) [CHILD: 'John'] = \\ GO([BOY: 'John'], [HOMEWORK_STATE: 'finished']) \vee \\ GO([BOY: 'John'], [MAXIMUM_TIME: '12H']) \vee \\ URGE([CHILD: 'John'], [MOTHER: 'Mary'])$$

Supervising the target event means to supervise the moment when the predicate expression representing the thematic view becomes true (i.e. when at least one of the selected graphs have been instantiated with the indicated values of the indicated concepts).

3 Motivating Activity-centred Graphs

3.1 Unifying Role of the Operation

In Fig. 4, we synthetize our view on the unifying role of the 'operation' (as an abstraction of the real world activity) relative to the three perspectives (data, functional and control perspective, see also [6]). The main advantage of the activity-centred graphs is that we use a single representation for the three perspectives. Other solutions (the most important we consider OMT [8] and KADS [9]) use a distinct representation for each perspective.

In short, for the *data perspective*, by means of general or domain specific declarative operations, we may define any type of primitive or structured concept, the concept behaviour (operations upon concepts) and the concept interoperability (dependencies among concepts). For the *functional perspective*, the process is

Figure 4 Concept, Process and Knowledge Flow Representation Converging into Operation Representation and Control

decomposed into domain specific dynamic operations. These operations are correlated by knowledge flows (input/output concepts) and control flows (continuation conceptual conditions) during the process execution (see [6]). For the *control perspective*, the operation is the uninterruptible unit of execution of the process. Each process is a sequence of dynamic operations, controlled by events, by a procedural strategy (interoperation links) and/or by a (human or automated) decision strategy. The operation may determine the transition of the state (extension) of certain concepts, its own state (active, stopped, waiting etc) or the state of the whole process. Finally, the conditions, whose true values may also determine the state transition are defined as list concepts obtained by a concatenation operation.

3.2 Universality, Linguistics, Flexibility and Extensibility of the Activity-centred Graphs

Universality of the activity-centred graphs (stylized sentences correlated by join and coreference operations and by interoperation connectives) derives from the universality and metamodeling feature of the natural language, the ideal model (metamodel), that helps us to express any information and situation in the real world.

Linguistics in the activity-centred graph representation appears from the representation of the nouns (as concepts), verbs (as operations), modal verbs (preceding the operations, see Fig. 1), pronouns (as concepts coreferent to the related nouns, e.g. 'his' in Fig. 2), adjectives (as attributive concepts introduced by concept qualifying roles), adverbs (as concepts introduced by the concept-operation link $\langle\langle DSCR \rangle\rangle$ or $\langle\langle CHRC \rangle\rangle$ or by the interoperation connectives DSCR and CHRC), prepositions and conjunctions (as concept-operation links), the noun plural (represented by the concept (collective or distributive) plural).

ACG linguistics and semantics is enhanced with 1) concept subtyping, that introduces the *concept polysemy* [10], 2) *concept homonymy* [10] (alternative disjunctive meanings of a concept) by means of, what we call, multiple concept [4], 3) *anaphoric sentences* (by means of coreferent graphs), 4) *active and passive voices* (by reversing the operation and the concept roles).

Flexibility and extensibility of the activity-centred graphs are the consequences of the ACG ability 1) to define any kind of first or higher-order concept and operator (concrete or abstract one), 2) to describe the concepts by two kinds of roles (relative to the domain and process and, also, relative only to the operation) and 3) to be correlated and controlled by control statements, whose logic (directly expressed in predicate calculus) assures their correctness. Also, both the set of conceptual connectives (roles) and the set of interoperation connectives we propose can be enlarged or shortened, depending on the representation needs.

In conclusion, 1) ACG linguistics helps us to master the heterogeneity of the data/ knowledge models we need to integrate in a complex information system and 2) ACG flexibility and extensibility help us to naturally integrate future representation requirements.

3.3 Code and Repository Reusability using Activity-centred Graphs

Both code and repository reusability are the consequence of the *operation invariant signature*, represented by a named tuple of invariant roles of the concepts that define or describe that operation:

$OPERATION(role_1, \dots, role_n)$

Code Reusability. Any inference on an activity-centred graph and, implicitly, on the operation it describes (including the inference like concept and operation storing and retrieving), will be independent of the domain specific concept types. It will rely only on the roles of those concept types relative to the operation. For instance, processing (MOVE) operation may only depend on the following roles: $\langle\langle PTNT \rangle\rangle$ denoting the moved entity (person or object), $\langle\langle SRC \rangle\rangle$ denoting the source place, $\langle\langle DEST \rangle\rangle$ denoting the destination place and $\langle\langle AGNT \rangle\rangle$ meaning the person or object that operates the respective move.

Repository Reusability. Similarly to the active and passive voices that change the subject and the complement roles in natural language, the domain specific operations may have reverse counterparts, acting on the same concept types, but changing their roles. Storing both operations (the direct and the reverse one) in the same internal structure, but interpreting and processing them differently, is another consequence of the roles we introduced in the definition of the activity-centred graphs.

3.4 A Representation Close to Human Reasoning and Understanding

We motivate this ability of the ACGs by:

- their *graphic and linguistic representation*, with a one-to-one correspondence to verbs, nouns, adverbs, prepositions, conjunctions, pronouns etc (see above).
- their *inferential capabilities* and *logical consistency* derived from:

Logical support of the activity-centred graphs consisting of:

- *natural mapping of the node and link representation to first or higher order logic.* Any diagram of activity-centred graphs will be represented by:
 - * monadic and first order predicates (representing primitive concepts and concept-operation roles),
 - * dyadic (first or higher order) predicates (representing the attributive roles),
 - * λ -definitions of n-ary (first or higher order) predicates (representing operations and concepts with structured definition).

NOTE When the concepts are predicates instead of values, the respective connection (attributive role or the operation) is a higher order predicate.

 - * combinations of conjunctions, disjunctions and negations of n-ary (first or higher order) predicates, representing the logic of the interoperation connectives.
- *λ -definition of the operations and of the concepts with structured definition.* This is the logical support for their encapsulation and connectivity (the bound variables address the operation parameters, see Fig.1(d)).

Logical constraints we imposed upon the operations and concepts, consisting of:

- *operation modality*, by a simplified deontic logic reduced to the operation obligation (by 'must' and 'may' verbs preceding the operation type, see Fig. 1(a)). It may be extended with verbs expressing the possibility, intention, wish, belief etc, relative to the operation execution.
- *concept (universal and existential) quantification* by: $\forall$ for 'any', 'all', $\exists$ for 'must exist' and $\exists?$ for 'may exist', with important benefits in specifying both the inference direction (premise and conclusion) inside each graph (see the λ -definition of the operation) and the obligation of the concept instantiation.

3.5 Structured Modelling/ Execution using Activity-centred Graphs

An important characteristic of the ACG formalism is its ability to extrapolate the software engineering principles (abstraction, structuring, modularity, connectivity, encapsulation) to the concept and process modelling and execution. These principles help, in our opinion, any conceptual/ execution model to approach the ideal structured modelling. Without the structuring techniques we propose (declarative and procedural structuring, modularity, connectivity, encapsulation) the representation of the activity-centred graphs would be overloaded and difficult to be managed.

The implementation of these principles also gets conceptual modelling closer to an advanced conceptual programming and decreases the gap between the modelling and the programming steps.

3.6 Procedural Guiding and Control of the Process Execution

The control statements (interoperation connectives) and their mapping rules to predicate calculus help us 1) to guide (predict) a process execution by a forward inference along the predefined procedural modules composing that process and 2) to control (by a backward inference) the correctness of each operation execution. We already implemented the procedural guiding and control for a multidatabase modelling process, based on a theme-oriented and event-driven reasoning upon the declarative and procedural modules of ACGs. The final goal of this reasoning is the knowledge acquisition on all compulsory themes (e.g. structure, semantics, dependencies, inheritance, mapping, implementation etc) for each modelled subject (schema, class, attribute etc)

3.7 A Natural Framework for Supervising the Concept and Process Interoperability

We distinguish the concept and process interoperability as the clue in the complex information system modelling and execution. Activity-centred graphs facilitate it because:

Concept Interoperability appears 1) when a structured concept is defined by means of other concepts, correlated by a definition operation, 2) when some source concepts influence the destination concepts, correlated by a dependency operation, 3) when the execution of a functional operation interconnects more concepts, previously defined for the respective process. After its declarative supervising during the modelling step, the concept interoperability becomes operational during the process execution.

Process Interoperability is the consequence of: 1) interprocess events, when an event operation in a process stimulates an operation executed in another process, 2) interprocess conceptual dependency, when the source and destination concepts in a dependency operation are defined in distinct processes, 3) interprocess conceptual transfers, when two operations in distinct processes transfer concepts (as parameters) to each other, 4) interprocess conceptual and logical sharing, when two processes share concepts/ graphs/ rules/procedures in a common knowledge base. Interprocess transfer and sharing are possible due to the code and repository reusability using activity-centred graphs (see above).

3.8 Benefits in Data/Knowledge Acquisition, Storing, Interrogation and Interpretation

Acquisition and Interrogation. Activity-centred graphs can be used as stylized acquisition and interrogation sentences, whose logic and semantics are (as we justified above) universal, flexible, extensible and close to the human reasoning and understanding. Their declarative and procedural structuring contributes to their gradual knowledge and understanding.

Storing. Natural mapping of the activity-centred graphs into frames makes natural their storing into the widely spread relational databases. We use ORACLE7 Server to store ACGs in the knowledge base of the multidatabase CASE environment and ORACLE Developer 2000 to acquire and interrogate this base. The distinction we made between the input, output, input-output and local concepts participating in operations allows the ACG integral or partial storing (e.g. storing local concepts only).

Interpretation. Besides the interpretation of (inference upon) a single graph, as in natural language, we use a complex thematic and multicriterial interrogation and interpretation upon the graphs, involving one or more target subjects (see above the definition of the thematic views over ACGs). We use a theme-oriented reasoning for checking the modelling correctness and, also, for comparing the distributed schemas. Both are based on predefined themes on predefined subjects.

4 Conclusions and Future Work

The activity-centred graphs sketched in this paper are our response to two main conclusions: 1) natural language is the ideal model (or metamodel) able to express any information and situation in the real world and 2) a structured and logic-based language is needed to master the complexity of the today's information systems. Structuring helps us for a gradual understanding of the representation and logic helps us to prove the representation correctness and consistency.

Along the paper, we also emphasized the benefits of the invariant roles of the concepts relative to the operations in the code and repository independence of a specific domain.

Linguistics of the ACGs may be further improved. Also, some more aspects in information system execution should be studied using ACGs, starting from the execution model we proposed in [6] (regarding the dynamic operations, transactional goals etc).

Further, an ACG general editor and an ACG translator to relational and O-O languages (including the representation and translation of the concept behaviour) are two important objectives of our future work.

References

- [1] Catach L., Fargues J. Deduction et operation pour le modele des graphes conceptuels. IBM Paris Scientific Center, No. F087, 1985
- [2] Chein M., Mugnier M. Conceptual Graphs: fundamental notions. Rev. d'intellig. artif., Vol. 6-no 4/1992
- [3] Fargues J., Landau Marie-Claude, Dugourd A., Catach L. Conceptual graphs for Semantics and Knowledge processing. IBM Journal of Research and Development, vol. 30, nr.1, Jan.1986
- [4] Galatescu A: An Activity-based Ontology using Conceptual Graphs. Proc. Intl. Symp. KRUSE (Knowledge Retrieval, Use and Storage for Efficiency), Santa Cruz, USA, Aug. 1995
- [5] Galatescu A. An Approach to Representing Procedural Semantics over Modelling-dedicated Conceptual Graphs. Proc. 10th Congress on Cybernetics and Systems, Bucharest, Romania, Aug. 1996
- [6] Galatescu A. Toward a Unifying Representation of Concepts, Processes and Knowledge Flows using Activity-centred Graphs. Proc. 7th Workshop on Knowledge Engineering: Methods & Languages (KEML97), Milton Keynes, UK, Jan. 1997
- [7] IRDS Conceptual Schema, Part 1: Conceptual Schema for IRDS, Part 2: Modeling Language Analysis. ANSI Report X3H4/93-196
- [8] Rumbaugh J., Blaha M., Premerlam W., Eddy F., Lorsen W. Object-Oriented Modeling and Design. Prentice-Hall, 1991
- [9] Schreiber G., Wielinga B. Breuker J. A Principled Approach to Knowledge-based System Development. Academic Press London, 1993
- [10] Sowa J.F. Lexical Structures and Conceptual Structures. Semantics in the Lexicon. Edited by J.Pustejovsky, Kluwer Academic Publishers

- [11] Sowa J.F. Toward the expressive power of the natural language. Principles of Semantic Networks. Explorations in the Knowledge Representation. Morgan Kaufmann Publ., 1991