Applying Conceptual Graph Theory to the User-Driven Specification of Network Information Systems[1]

Aldo de Moor

Infolab, Tilburg University, P.O.Box 90153, 5000 LE Tilburg, The Netherlands,
e-mail: ademoor@kub.nl

Abstract. Users need to be strongly involved in the specification process of network information systems. Characteristics of user-driven specification are described, and process composition is proposed as a feasible approach. The knowledge representation framework used in the RENISYS specification method is introduced, using conceptual graph theory as its underlying formalism. The role of ontological and normative knowledge is explained. The presented theory is used to show how legitimate process definitions can be generated by the users. The facilitation of user-driven process composition is discussed.

1 Introduction

Research networks are goal-oriented networks of professionals that focus on supporting certain stages of the research process, such as the planning and conduct of research activities and the dissemination and implementation of the results.

One characteristic of such networks is that their activities are often highly complex as well as innovative in nature, implying that their work processes need to be continuously remodelled, and their supporting network information system redesigned. A second trait is that research networks, which are often Internet-based, make use of an ever increasing set of standard information tools, rather than custom-designed programs, to implement their information systems. These tools enable a fixed set of information and communication processes, which in turn allow participants to carry out their network activities. Another important feature is that for reasons such as the relative lack of hierarchical organization of the participants, and the participants themselves being the task-experts, user-driven network information system development is essential [1]. This means that the participants are responsible for determining the exact roles that a suite of tools plays in the enabling of their work processes.

In this paper, we aim to first delve in more detail into some key aspects of network information system development. More particularly we focus on how users should be involved in the system specification process through process composition. We introduce the knowledge representation framework used in a specification method for research network information systems currently under development (RENISYS), using conceptual graphs as the underlying formalism. The importance of ontologies as the conceptual foundation of specification knowledge is highlighted, and some operations to change ontological type definitions are presented. Based on the ontologies, sets of norms can be defined, which regulate user-behaviour both on the operational and the compositional level. By means of an example adapted from a concrete research network case we illustrate how legitimate process definitions can be

---

[1] This paper has been presented at the Fifth International Conference on Conceptual Structures - Fulfilling Peirce's Dream, University of Washington, Seattle, August 4-8, 1997. The conference proceedings have been published in the Springer-Verlag Lecture Notes in Artificial Intelligence series, No.1257

generated. Subsequently, we briefly discuss how the previously developed theory can be applied to facilitate more active user-driven process composition.

2 Involving the Users in the Information System Development Process

2.1 What Is a Network Information System?

The network information system (NIS) can be described from an analysis as well as a (high-level) design perspective. From the analysis point of view, the NIS is seen as the set of meaningfully combined and configured information and communication processes which are necessary to support and coordinate the activities of the network participants in their various roles [1]. In the analysis view we thus focus on how meaningful process requirement definitions can be made. However, we do not take into account the roles that the available information tools play in the implementation of the required process functionality. These roles are determined during design, in which the set of required information and communication processes is mapped to the available enabling information tools. When looking at the information system in this way, we consider each tool to afford a number of generic information and communication processes, which can be used in a particular network to enable specific work processes.

2.2 Characteristics of User-Driven System Specification

User-driven NIS development is based on a paradigm very different from the one underlying traditional information systems development approaches, such as ISAC or SDM (Fig.1). Traditional methods are typically based on the waterfall paradigm. These methods are used in large scale projects, in which a group of external experts (represented in Fig.1 by the small pencils) analyzes the organization according to a pre-defined series of steps. Users play a rather passive role, which is often limited to being interviewed by the analysts. At a certain moment in time, the latter produce a blueprint of the information system, which is approved of by the users, and then implemented. However, this implementation is static and the functionality provided often rapidly becomes obsolete, due to quickly changing user requirements. When the mismatch between required and implemented functionality becomes too large, the whole process has to start all over again. An excellent analysis of the many problems encountered in this kind of large project-based software development is given by Brooks in his famous book [2].

Information system development for research networks takes place very differently. First of all, most of it is done by the users on their own. Network participants themselves discuss what activities they should carry out, and determine which information tools to use to accomplish their goals. Furthermore, as has been recognized for quite some time already, such user-driven system development should be based on an evolutionary approach [3]. In contrast with the waterfall-based approaches, the required and implemented functionality gradually evolves, rather than expands with leaps and bounds. Characteristically, a few researchers meet and decide to

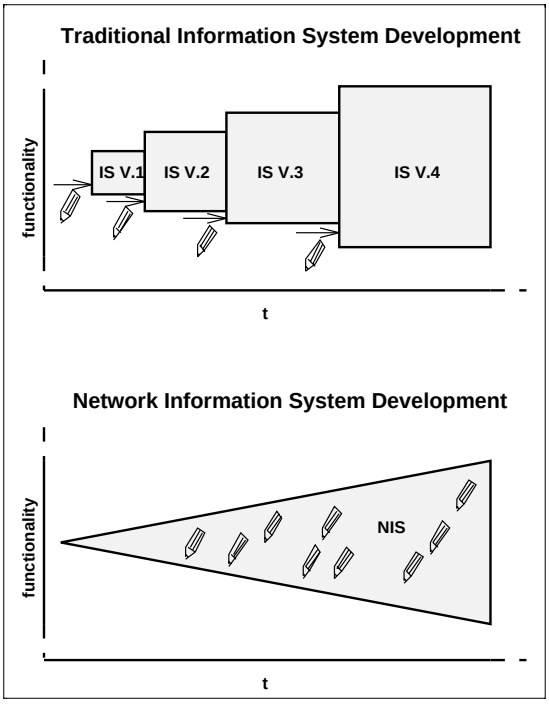

Fig. 1. Network Information System Development: Evolution Instead of Revolution

form a small network, supported by, for example, a simple mailing list in order to freely exchange various kinds of information. Such a network can expand rapidly, however, both in organizational and task complexity, requiring ever more sophisticated information technological support. Another characteristic is that there is not a single, bird's eye view on the universe of discourse, which the external analysts of the waterfall methods (should) have. In user-driven development, there are rather multiple ants' eyes views, each user in general only being knowledgeable about, and interested in, the small part of the workflow she is involved in.

2.3 Process Composition

When classifying a NIS according to its purpose, it can be considered an ad hoc workflow management system. A workflow management system allows for the design, execution and management of work processes [4]. Ad hoc means that such a system supports creative knowledge activities. These activities are notoriously difficult to model; the workflow support provided can therefore at best provide some sort of control to ensure that tasks, responsibilities, etc. are delivered [5]. However, current workflow management modelling methods (input-process-output as well as speech act-based methods), and user-oriented development approaches (e.g. prototyping, radically tailorable tools), are not sufficiently capable of supporting such user-driven system specification [6,7].

A more promising approach seems to be the one taken by Fitzpatrick and Welsh to facilitate so-called process composition. A core concept they introduce is that of process space. This is `a semantically rich and relatively well defined space, both physically and conceptually, which constrains and bounds the very possibilities of work' [8]. Through process composition, participants can determine the (maximally) required information system functionality by describing their work processes, the total

information system process space being equal to the sum of the individually composed processes.

Although some initial attempts have been made to formalise process composition support (especially by [8]), there is still a lot of research needed before it can be effectively used for network information system development.

The research agenda should include at least the following issues:
(1) How to adequately represent specification knowledge?
(2) How to ensure that only legitimate process definitions are made?
(3) How to foster more active user involvement in the specification process?
(4) How to match legitimate process definitions with the functionality enabled by the available suite of information tools?

2.4 The RENISYS Specification Method

The RENISYS method (**RE**search **N**etwork **I**nformation **SY**stem **S**pecification method) is currently being developed to provide a concrete specification approach and tool for network participants to compose their own network information systems. The scope of the method and methodological design criteria were discussed in [1,6]. A model of the user-driven development process was described in [7]. In the model, we subdivide this process into three main subprocesses: the specification, implementation, and use of the network information system. Furthermore, we strictly separate specification and implementation, while on the other hand we strongly connect the specification and use process. For the method to be successful, it will at least need to incorporate answers to the points raised in the above-mentioned research agenda. For lack of space, in this paper, we will only focus on providing an initial answer to the first three issues. It seems natural to address these questions first, since process requirements must be defined before they can be assigned to (enabling) tools in the system design, which the final issue is about.

2.5 Applying Conceptual Graph Theory

A user needs to be involved in a specification discourse which makes use of a restricted form of natural language. The language must be rich enough to allow the efficient expression of complex specifications. At the same time it must be sufficiently formal and constrained to allow meaningful specification inferences to be made for the method to adequately facilitate process composition. Conceptual graph theory is more than a syntactic variant of first-order logic because it can enforce conceptual definitions of concepts and relations in terms of natural language-related primitives [9]. The formal structures and operations of the theory have been shown to be useful for representing and processing terminological knowledge in concrete applications, see e.g. [10]. This natural language-focus is an important reason for choosing conceptual graph theory as the knowledge formalism of choice in RENISYS. We will illustrate its potential by giving some preliminary solutions to the research questions posed. In future research we hope to expand the application of CG theory in the method; the purpose of this paper is only to generate some ideas and discussion about the relevance of the theory to this particular kind of application.

3 Knowledge Representation in RENISYS

In RENISYS, the network information system is modelled from three different perspectives, resulting in different domains. The domains are combined in a *reference*

framework. The framework is used to generate and store specification knowledge, of which two important kinds are ontological and normative knowledge.

3.1 The Reference Framework

The reference framework consists of three, interconnected domains: the problem domain, the human network, and the information system. In the problem domain, the universe of discourse is interpreted from a task perspective (what are the goals and activities?), in the human network it is described from the organizational point of view (what are the participant interaction processes allowed by the organizational positions?). Together, these domains form the usage context, which describes the determinants of the (generic) information and communication processes. These processes, having been assigned to the set of available information tools, constitute the information system. Directed mappings connect the entities from the different domains (for more details on the framework, we refer the interested reader to [7,11].

In RENISYS, we distinguish three types of knowledge. Ontologies contain descriptions of the terminology used in the different domains. Norms describe the desired behaviour of the various actors in the network. State knowledge can be used to describe the actual or potential behaviour of actors. In RENISYS, it plays an especially important role as a trigger of specification processes, when users use state knowledge to describe the actual situation they are in, and the workflow problems they experience. As the focus of this paper is on the specification process itself, rather than on how exactly it is triggered, and since the correct expression of state knowledge demands an ontological and normative knowledge basis, in the next sections we will study the format and use of ontological and normative knowledge only. In our treatment of these knowledge categories we have been influenced by the semiotic theory of Stamper [12], who takes a strong subjectivist instead of the more regular objectivist stance on the reality to be modelled. An extensive discussion on the pressing need for, and feasibility of such subjectivist information system development methods can be found in [13].

3.2 Ontologies

An ontology is an explicit specification of a conceptualization, which itself is an abstract, simplified view of the world that needs to be represented for some purpose [14]. In our case, this purpose is network information system specification. An ontology can practically be used to organize the storage of information and access to knowledge [15], which for us concerns specification knowledge. Thus, an ontology forms only part of a knowledge base, as it contains a vocabulary useful for describing a domain rather than knowledge about the state of the domain itself [14]. In this way, it is an important instrument in supporting the correct reuse and extension of already generated, complex knowledge structures.

The basis for the ontologies is the concept type hierarchy. Each concept used in any ontological, norm, or state definition must be in this type hierarchy. However, not all concepts that are included in the type hierarchy need to be defined by a differentia as well. For relation types we use a, slightly modified, fixed subset of the case relations used by Sowa [16]. For ontological purposes, type definitions as described in CG theory are useful. We use type definitions instead of schemas because we want ontological definitions only to contain *necessarily*, not just *possibly* existing knowledge. In this way, type definitions provide a clear canonical core for norm and state knowledge, and are useful in the enforcement of selectional constraints on what are considered to be meaningful specifications.

Sources of Type Definitions

In RENISYS, two main sources of type definitions exist: a stable set of theory-grounded definitions, and an evolving set of user-specified definitions.

Theories relevant to the analysis and design of research network information systems provide axiomatic primitives. Some of these primitives are useful for the natural expression of requirements by participants, some are relevant to the design of the network information system, a third category is used to describe the reference framework that provides the connection between these two sets of concepts. In our approach we draw from several such theories, notably language action theory (as introduced in [17] and concrete methodological approaches for organizational communication theory (e.g. the Dynamic Essential Modelling of Organizations method [18]. However, these provide only an initial set of specification entities, which needs to be adapted by the users to match their specific work processes. Our basic ontological goal therefore is not to come up with *the* definition of *the* determinants of the information system in a (research network) usage context. Rather, the basic, theory-grounded set of primitives forms a customizable conceptual foundation that users can tailor to their unique, evolving requirements.

The second, more interesting source of ontological definitions is the network participant herself. Users must be provided with the mechanisms to legitimately customize ontological constructs. Developing facilities which assist users in efficiently modelling their own worlds may furthermore help to considerably increase the limited intellectual capacity available for ontology construction. The pioneering, small groups of experts who currently try to do this cannot handle the sheer volume of the tasks involved [9]; user-defined ontologies can be used directly, or at least form an important input, in larger scale construction efforts.

Type definition operations form the basic tool set to implement such ontological construction mechanisms.

Type Definition Operations

We distinguish three kinds of type definition operations which prescribe how users can carry out such ontology customizations. These operations are: the *creation, modification, and termination* of type definitions.

• Type Creation

In the creation of a type definition, both the type label and its differentia (if needed) are generated. The type must be a subtype of an existing type, the differentia a specialization of the differentia of the supertype. This operation can be implemented using the standard form of type definition as described in [16].

Type creation can serve two uses: the specialization of a concept, or the generalization of a set of concepts. If a concept is to be specialized, no further action is required. If a set of concepts is generalized, all links to the original parent of each concept must be reassigned to the newly created supertype.

Example: The Creation of a Group_Report_Editor Type

Pre:
type EDITOR(x) **is**
[ACTOR:*x] <- (agnt) <- [CONTROL] -> (obj) -> [EDIT].

Post:
type EDITOR(x) **is**
[ACTOR:*x] <- (agnt) <- [CONTROL] -> (obj) -> [EDIT].

type GROUP_REPORT_EDITOR(x) **is**
[EDITOR:*x] <- (agnt) <- [EXECUTE] -> (obj) -> [EDIT] -> (rslt) -> [GROUP_REPORT].

The example given is one of type creation for concept specialization purposes. Let us say a user intends to create a group report editor type. A group report editor is an editor who can perform the actual editing of a group report. An editor type has already been defined previously. In graph terms the operation could be represented as above (an execution is a subtype of a control process; items that have changed after the operation have been underlined).

- **Type Modification**

During the modification of a type, the type label is kept, only the differentia is changed. All (type, norm, and state) definitions including the modified type remain the same, although the role of this concept in these definitions changes due to the type modification.

Example: The Modification of a Group_Report_Editor Type

Pre:
type GROUP_REPORT_EDITOR(x) **is**
[EDITOR:*x] <- (agnt) <- [EXECUTE] -> (obj) -> [EDIT] -> (rslt) -> [GROUP_REPORT].

type PUBLISH_GROUP_REPORT(x) **is**
[TRANSFORMATION:*x] -
(obj) <- [EXECUTE] -> (agnt) -> [GROUP_REPORT_EDITOR]
(obj) <- [EVALUATE] -> (agnt) -> [CLIENT]
(matr) -> [GROUP_REPORT]
(rslt) -> [PUBLISHED_GROUP_REPORT].

Post:
type GROUP_REPORT_EDITOR(x) **is**
[EDITOR:*x] <- (agnt) <- [EVALUATE] -> (obj) -> [EDIT] -> (rslt) -> [GROUP_REPORT].

type PUBLISH_GROUP_REPORT(x) **is**
[TRANSFORMATION:*x] -
(obj) <- [EXECUTE] -> (agnt) -> [GROUP_REPORT_EDITOR]
(obj) <- [EVALUATE] -> (agnt) -> [CLIENT]
(matr) -> [GROUP_REPORT]
(rslt) -> [PUBLISHED_GROUP_REPORT].

In this example, the meaning of the concept type 'group report editor' is changed. For example, the actor who has the modification authority decides that no longer is the editor somebody who does the actual execution of the editing process, but the editor becomes the one responsible for the final assessment of the produced report. The actual execution can be distributed among a number of reviewers instead. This is a real-life situation often experienced in growing research networks. As the example shows, the differentia of 'group report editor' is changed, but the definition of 'publish group report', in which a group report editor plays the role of executor, remains identical. Changed type definitions thus have global implications, while at the same time responsibilities for these definitional changes are clearly divided among the network participants with the proper definitional authorities.

- **Type Termination**

When a type is terminated, both the type label and the type differentia are removed. In this case, all occurrences of the old type in any ontological, norm, or state definition

must be replaced with for example the supertype or one of its subtypes, although other termination scenarios are conceivable as well.

Example: The Termination of a Group_Report_Editor Type

Pre:
type GROUP_REPORT_EDITOR(x) **is**
[EDITOR:*x] <- (agnt) <- [EVALUATE] -> (obj) -> [EDIT] -> (rslt) -> [GROUP_REPORT].

type PUBLISH_GROUP_REPORT(x) **is**
[TRANSFORMATION:*x] -
(obj) <- [EXECUTE] -> (agnt) -> [GROUP_REPORT_EDITOR]
(obj) <- [EVALUATE] -> (agnt) -> [CLIENT]
(matr) -> [GROUP_REPORT]
(rslt) -> [PUBLISHED_GROUP_REPORT].

Post:
type GROUP_REPORT_EDITOR(x): *removed*

type PUBLISH_GROUP_REPORT(x) **is**
[TRANSFORMATION:*x] -
(obj) <- [EXECUTE] -> (agnt) -> [EDITOR]
(obj) <- [EVALUATE] -> (agnt) -> [CLIENT]
(matr) -> [GROUP_REPORT]
(rslt) -> [PUBLISHED_GROUP_REPORT].

In the example, the ontological definition of 'group report editor' is removed, for example because of a reorganization in the network. RENISYS must now find all definitions in which the concept to be removed occurs. Note that, like in the other type definition operations, the replacement of concepts in definitions with other concepts is not trivial. The owners of these concept definitions may need to be consulted, or at least notified that other actors plan to change the way in which 'their concept' is being used. Finding out what kind of protocols best suit user-driven specification is an important objective of our research. We have recently started modeling such protocols along the line of speech-act based discourse protocols, as described in [19].

3.3 Norms

Each network participant plays several actor roles. An actor is an interpreting entity capable of playing process control roles. The actor states his requirements in terms of the operational actions he is involved in. An action is a combination of a control process (initiation, execution, evaluation) and a transformation (a process in which a domain object, such as a paper, is generated). To produce specifications, actors can also make compositions. These are combinations of control and definitional processes, such as the creation of a type definition.

The dual control/controlled process approach allows users on the one hand to define workflows and specification processes in terms of concrete deliverables, on the other hand to clearly define the responsibilities for these processes.

Responsibilities have to do with the rules that the network participants agree on, which specify their desired (non)behaviour, and have a normative character. However, to think of norms only as responsibilities produces specifications that are too coarse to properly model specification changes. Norms are therefore subdivided according to the role they play in restricting or affording behaviour. This results in the following categories: privileges, responsibilities, and prohibitions. Privileges comprise those actions and compositions that an actor is permitted to carry out. If an actor is obliged

to carry out an action when appropriately triggered (e.g. by taking part in a workflow), the actor has a responsibility. Thus, responsibilities consist of mandatory actions and compositions. All responsibilities should be privileges as well, if not, respecification of norms is necessary. Prohibitions are actions and compositions that an actor is not permitted to carry out.

In order to define the required functionality of an information system, it is important to know what actions specific actors actually are - or are not - allowed to carry out. This knowledge we define in action norms, which of course must be expressed in terms of the concepts that have been constructed using the type definition operations described in the previous section. Besides action norms, in user-driven system specification we also have a need for compositional norms. These norms indicate which actors in the network can make what kind of knowledge definitions about the network.

In CG terms, the basic representation of action norms is:

[ACTOR] <- (agnt) <- [CONTROL] -> (obj) -> [TRANSFORMATION].

An action norm thus shows which control rights an actor has over which (operational) transformations. Compositional norms are similarly represented as:

[ACTOR] <- (agnt) <- [CONTROL] -> (obj) - [DEFINE] -> (rslt) -> [DEFINITION].

A definitional process is here either a creation, modification, or termination of a type, norm, or state definition.

4 Generating Legitimate Process Definitions

An important issue in process composition is how to determine what are legitimate knowledge definitions. Who must be involved in the evolution of norms and types? This is often very unclear in the non-hierarchical kind of professional networks which are our object of interest, as decision-making authority is role, instead of position-bound.

A legitimate definitional change is one of which (1) the canonicity, and (2) the authorization have been checked, thus combining meaningfulness with validity. This helps to ensure as much as possible a model of the network and its information system that represents the interests of, and is acceptable to all network participants.

4.1 Case: a Research Group Writing a Group Report

We will give a concrete illustration of the previously developed theory by producing a legitimate process definition. The example concerns the definition of a group report editor type, and is based on a real case. The B.C. Forests and Forestry Project Group (BCFOR) is an Internet-mediated group aiming to do 'public research' on deforestation in British Columbia, Canada[2]. Like most other such groups, it initially merely allowed unstructured discussion through a mailing list.

However, after a while, the group decided to create a structured group report on a specific discussion topic. More explicit workflows now needed to be defined, in order to allow for both an adequate division of tasks and more tailored information technological support. This specification process turned out to be very hard without a proper conceptual framework to manage the system evolution [7]. We expect an

[2]BCFOR is part of the Global Research Network on Sustainable Development. More information is available at: http://infolabwww.kub.nl:2080/grnsd/proj/gp-bcfor/

approach as described in this paper to allow participants in a network to better manage the complexities of network information system evolution.

4.2 The Problem: Creating a Group Report Editor Type

In this section, we will illustrate how legitimate (canonical and authorized) process definitions can be generated. In our current approach, the various knowledge categories are represented in simple graphs (only using complex referents to store definition graphs), as we expect this to considerably reduce the complexity of knowledge processing operations. In the implementation of the tool, the different knowledge categories could for example be distinguished by storing them in separate ontological and norm knowledge bases. In the example, the boxed comments indicate the status of the graphs (proposed type definition, accepted norm, etc.).

Example

At t=0, an editor(<actor), edit (<transformation), and paper (<object) type are distinguished in the type hierarchy, all in the problem domain. The editor and edit types also have a definition; the paper type (still) goes undefined:

Category: Type Definition, Modality: Accepted

type EDITOR(x) **is**
[ACTOR:*x] <- (agnt) <- [CONTROL] -> (obj) -> [EDIT].

type EDIT(x) **is**
[TRANSFORMATION:*x] -> (rslt) -> [PAPER].

When the group decides to write a group report, it is agreed that a special kind of editor, the group report editor, is required. The group does have the authority to create a subtype of editor, as the following compositional norm was already previously defined:

Category: Permitted Composition, Modality: Accepted

[GROUP] <- (agnt) <- [CONTROL] -> (obj) -> [CREATE] -> (rslt) -> [TYPE:[EDITOR]].

After some group discussion, based on the existing definition of editor, it is decided that a necessary condition for an editor to be a group report editor is that the edit process she is responsible for should result in the group report as an output. Thus, the proposed group report editor definition becomes:

Category: Type Definition, Modality: Proposed

type GROUP_REPORT_EDITOR(x) **is**
[EDITOR:*x] <- (agnt) <- [CONTROL] -> (obj) -> [EDIT] -> (rslt) -> [GROUP_REPORT].

However, although the group has the authority to create this definition, the definition is not legitimate, because it is not canonical. The new concept type 'group report', which forms part of the definition, has not even been included in the type hierarchy. The method checks if there is an actor who has the default authority for doing this inclusion by (minimally) expanding the edit concept node in the editor definition (using the existing edit type definition), which results in the following graph:

Category: Type Definition, Modality: Derived (by method)

type GROUP_REPORT_EDITOR(x) **is**
[EDITOR:*x] <- (agnt) <- [CONTROL] -> (obj) -> [EDIT] -> (rslt) -> [PAPER].

As (in RENISYS) a transformation can only result in a single type of output object, the method knows that a group report *must* be a subtype of paper for the proposed definition to be canonical. Subsequently, the method needs to identify the proper authority for a paper type creation operation. It does this by projecting the following query graph on its set of existing compositional norm graphs:

Category: Query, Modality: Derived (by method)
[ACTOR] <- (agnt) <- [CONTROL] -> (obj) -> [CREATE] -> (rslt) -> [TYPE:[PAPER]].

Let us assume that this projection results in just one permitted composition, namely:

Category: Permitted Composition, Modality: Accepted
[EDITOR] <- (agnt) <- [CONTROL] -> (obj) -> [CREATE] -> (rslt) -> [TYPE:[PAPER]].

The method can now propose the users who play the editor role to create the requested group editor type. If the editor does not agree, a negotiation discourse between editor and group should be initiated and supported by RENISYS.

5 Facilitating User-driven Process Composition

Knowing how to create legitimate knowledge definitions is a necessary, but not yet a sufficient condition for successful user-driven system specification. The dynamics of the specification process deserve special care, due to the fact that users have only a very limited interest in participating in this process. One possible solution to this problem is to present users with customized views on the total process space, views that make them better comprehend their privileges, responsibilities, and prohibitions in the network, and that increase the incentives for users to initiate and participate in more useful specification discourse. Therefore, we refine the concept of process space into action and process composition spaces, and facilitate discourse initiation by focusing on breakdowns.

5.1 Action and Process Composition Space

All permitted actions of an actor together form his action space. An example of a possible action space of an actor 'author' at a certain moment in time is the conceptual graph shown here:

[AUTHOR] -> (attr) -
  [PERMITTED_ACTION: [EXECUTE] -> (obj) - [WRITE] -> (rslt) -> [CONTRIBUTION]]
  [PERMITTED_ACTION: [INITIATE] -> (obj) -> [EDIT]].

The graph says that an author can execute writing a contribution, as well as initiate an edit process. Depending on the exact mapping to the information and communication processes enabled by the available information tools, this would result in certain functionality specifications. As said before, we do not explain in this paper how this information system design should be done.

In order to *adapt* his set of possible actions, called making compositions in our terminology, an actor must have the authority to produce knowledge definitions, or otherwise negotiate with an actor who does have this authority. The total set of compositions that an actor is allowed to make we define as his (process) composition space. The composition space of an actor generally, but not necessarily, comprises

some compositions required to update definitions that shape his own action space, plus other kinds of compositions he has been authorized to carry out. An illustration of a possible composition space for actor 'author' is given here:

```
[AUTHOR] -> (attr) -
   [MANDATORY_COMPOSITION: [CONTROL] -> (obj) - [CREATE] -
        (rslt) -> [TYPE: [CONTRIBUTION]]]
   [PERMITTED_COMPOSITION: [INITIATE] -> (obj) -> [MODIFY] -
        (rslt) -> [PERMITTED_ACTION: [EDITOR] <- (agnt) <-  [CONTROL] -
            (obj) -> [EDIT] -> (matr) -> [CONTRIBUTION]]].
```

The graph should be interpreted as meaning that an author must decide on creating subtypes of the concept type 'contribution' (for example, 'abstract' or 'paragraph'), when requested. Furthermore, an author may initiate the modification process of the norm which says that an editor is permitted to control the editing of contributions.

5.2 Recognizing Breakdowns

A fundamental idea regarding the facilitation of process composition is that the method should not trigger users to define requirements according to the rigid analysis and design steps that are prescribed by waterfall methods. Instead, it should support them in resolving concrete functionality problems. These problems are experienced when they play the roles as defined in their action spaces.

According to Winograd and Flores, 'breakdown' plays a key role in the adequate design of artifacts. "A breakdown is not (necessarily) a negative situation to be avoided, but a situation of non-obviousness, in which the recognition that something is missing leads to unconcealing some aspect of the network of tools that we are engaged in using" [17, p.165]. Thus, the proper identification and handling of breakdowns, as soon as they occur, is crucial for users to become interested and actively involved in the (re)specification of their network information system.

The specification method, besides helping the user in becoming aware of his breakdowns, must also help formulate the breakdowns in network terminology, involve relevant other actors in the specification discourse, and support the process of making the actual changes in knowledge definitions.

How exactly this should be done is in our current research focus. The human computer interface now being constructed allows the method tool to have a pseudo-natural language dialogue with users who either explore their own breakdowns, or are involved in a specification discourse triggered by other users resolving their particular breakdowns. Users are prompted to participate by tool-generated e-mail, and once they log on to the RENISYS (web) server will be presented immediately with the appropriate dialogue screens.

It is important to realize that users are not forced to participate in specification discourse. They are allowed to delegate their authority. If they do not respond at all, while not having delegated their responsibilities, network-agreed upon decision rules should determine how the legitimacy of knowledge definitions should be established.

6 Conclusions

In this paper, we have attempted to indicate how conceptual graph theory can help to enable the complex process of user-driven specification of network information systems. Its intuitive semantics resulting from its roots in natural language, combined with its formal power, make the theory a 'logical' candidate as the underlying knowledge representation and reasoning formalism to be used in the RENISYS specification method. We do not claim that our current use of conceptual

graph theory to represent the various specification constructs is the most appropriate way; rather, it is an initial attempt meant to generate discussion on better representations, and the algorithms needed to produce them.

Since the early days of conceptual graph theory, a lot of progress has been made in extending and refining its terminology and operations. However, only little attention has so far been paid to finding practical applications of the generic theoretical constructs [20]. We hope that the RENISYS method can serve as such an application, the research on the method both benefiting from the wealth of existing conceptual graph resources and contributing to their further development.

RENISYS is going to be implemented as a client/server system, the clients being standard web browsers. This will allow for maximum participation by the average user, who is the main source of specification knowledge. Our original intention was to develop the complete server in TCL, a powerful scripting language. However, it would be worthwhile to see to what extent one of the 'standard CG workbenches' [21] could be used as the heart of the server.

7 Acknowledgment

The author wishes to express his gratitude to Hans Weigand for his helpful suggestions for improvement of the original manuscript.

References

1. De Moor, A. Toward a More Structured Use of Information Technology in the Research Community. *The American Sociologist*, 27(1), 1996, pp.91-101.
2. Brooks, F. *The Mythical Man-Month: Essays on Software Engineering*. Addison Wesley, anniversary edition, 1995.
3. Knight, K., editor. *Participation in Systems Development*. Applied Information Technology Reports, Unicom, 1989.
4. Abbott, K., Sarin, S. Experiences with Workflow Management: Issues for the Next Generation. In Furuta, R., Neuwirth, C., editors, *Proceedings of the ACM Conference on Computer Supported Cooperative Work, Chapel Hill, October 22-26, 1994*. ACM, pp.113-120.
5. Khoshafian, S., Buckiewicz, M. *Introduction to Groupware, Workflow, and Workgroup Computing*. John Wiley & Sons, 1995.
6. De Moor, A. Coordinating the Specification Process of Information Systems for Research Networks: Methodological Design Principles. In Fidler, C., editor, *14th International Association of Management Conference, Toronto, August 2-6, 1996, Information Systems Proceedings*, pp.95-103.
7. De Moor, A., Van der Rijst, N. Fostering Active User Involvement in the Specification of Network Information Systems. In *Dutch Interdisciplinary Research Conference on Information Science, December 13, 1996*, Delft University of Technology, pp.105-118.
8. Fitzpatrick, G., Welsh, J. Process Support: Inflexible Imposition or Chaotic Composition? *Interacting with Computers*, 7(2), 1995, pp.167-180.
9. Lehmann, F. CCAT: The Current Status of the Conceptual Catalogue (Ontology) Group, With Proposals. In Ellis, G., Levinson, R., editors, *Proceedings of the Third International Workshop on PEIRCE: A Conceptual Graphs Workbench, University of Maryland, August 19, 1994*, Lecture Notes in Artificial Intelligence, Vol. 835, Springer-Verlag, pp.18-28.
10. Angelova, G., Bontcheva, K. DB-MAT: Knowledge Acquisition, Processing, and NL Generation Using Conceptual Graphs. In Eklund, P., Ellis, G., Mann, G., editors, *Proceedings of the 4th International Conference on Conceptual Structures: Knowledge Representation as Interlingua, Sydney, August 19-23, 1996*, Lecture Notes in Artificial Intelligence, Vol.1115, Springer Verlag, pp.131-134.

11. Van der Rijst, N., De Moor, A. The Development of Reference Models for the RENISYS Specification Method. In Nunamaker Jr., J.F., Sprague Jr., R.H., editors, *Proceedings of the 29th Hawaii International Conference on System Sciences, January 3-6, 1996*, pp.455-464.
12. Stamper, R. A Semiotic Theory of Information and Information Systems / Applied Semiotics. In *Invited Papers for the ICL / University of Newcastle Seminar on "Information", September 6-10, 1993.*
13. Hirschheim, R., Klein, H., Lyytinen, K. *Information Systems Development and Data Modeling - Conceptual and Philosophical Foundations*. Cambridge University Press, 1996.
14. Gruber, T. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Technical Report KSL 93-04, Knowledge Systems Laboratory, Stanford University, 1993.
15. Mizoguchi, R. Knowledge Acquisition and Ontology. In *KB&KS '93, Tokyo*, 1993, pp.121-128.
16. Sowa, J. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.
17. Winograd, T., Flores, F. *Understanding Computers and Cognition - A New Foundation for Design*. Ablex Publishing Corporation, 1986.
18. Dietz, J. Modelling Business Processes for the Purpose of Redesign. In *Proceedings of the IFIP TC8 Open Conference on Business Process Redesign*, North-Holland, 1994, pp.249-258.
19. Chang, M., Woo, C. A Speech-Act Based Negotiation Protocol: Design, Implementation and Test Use. *ACM Transactions on Information Systems*, 12(4), 1994, pp.360-382.
20. Lukose, D., Mineau, G., Mugnier, M.-L., Möller, J.W., Martin, P., Kremer, R., Zarri, G. Conceptual Structures for Knowledge Engineering and Knowledge Modelling. In Ellis, G., Levinson, R., Rich, W., Sowa, J., editors, *Proceedings of the Third International Conference on Conceptual Structures - Conceptual Structures: Applications, Implementation and Theory, Santa Cruz, August 14-18, 1995*, Lecture Notes in Artificial Intelligence, Vol. 954, Springer-Verlag, pp.126-137.
21. Mann, G. What Conceptual Graph Workbenches Need for Natural Language Processing. In Ellis, G., Levinson, R., Rich, W., Sowa, J., editors, *Proceedings of the Third International Conference on Conceptual Structures - Conceptual Structures: Applications, Implementation and Theory, Santa Cruz, August 14-18, 1995*, Lecture Notes in Artificial Intelligence, Vol. 954, Springer-Verlag, pp.70-78.