An Ontological Framework for User-Driven System Specification*

Aldo de Moor, Hans Weigand
Infolab, Tilburg University
P.O.Box 90153, 5000 LE Tilburg, The Netherlands
tel.+31-13-4663020, e-mail:ademoor/weigand@kub.nl

Abstract

User-driven specification of network information systems is required in many virtual professional communities, such as research networks. This paper introduces a method and an accompanying tool being developed to support the legitimate user-driven specification of network information systems. The need for method ontologies, organized in an ontological framework, is motivated. A key part of this framework, the core process ontology, is described. The application of the framework is illustrated with some case study examples. It is shown how ontological and normative knowledge can be combined to model the user-driven specification process. Its formal semantics are described using dynamic deontic logic.

1. Introduction

Collaborative work is increasingly being done in a distributed fashion. Collaborating professionals from within and across organizations are more and more expected to set their own goals and ways of working. As such collaboration often entails having people from many different locations working together intensively on complex tasks, distributed computer technologies, such as Internet-based groupware, have the potential to considerably increase productivity. This paper focuses on supporting the creation of information systems for such *virtual professional communities*, which are defined as communities (or networks) of professionals whose collaboration on activities that are required to realize shared goals is mostly or completely computer-mediated.

1.1. Virtual Professional Communities

The concept of virtual professional communities deserves further explanation. One definition of the term 'community' that well captures its essence is given by [1], who sees it as 'a group of people *bound* together by certain mutual concerns, interests, activities, and institutions.' Thus, key to distinguishing a community from an arbitrary set of communicating persons is that certain bonds exist, resulting in a synergy in thinking and acting.

A community is a professional community if it is goal-oriented, and its members adhere to certain norms of rational discourse. A community is a virtual community if most participant interactions are mediated by computer technology. Like their counterparts in 'real communities', members of a virtual community have shared mental models and social norms [2].

To successfully collaborate, virtual professional communities have a growing need for integrated *network information systems*, which can be composed out of standard Internet-based information tools. Such tools include mailers, mailing lists and a great variety of Web-applications. Still, from an analytical instead of such a design perspective, a network information system can be regarded as a set of meaningfully configured and combined information and communication processes necessary to support and coordinate the activities of the network participants in their various roles [3]. This implies that the *usage context* in which the information system operates, needs to be very clearly defined.

1.2. User-Driven System Specification

Active participation of members of a virtual professional community in the specification process is essential in order to successfully develop the desired community partnerships and the required supporting information systems [4]. Such user-driven development is based on a paradigm very different from traditional, waterfall-like development methods [5]. First, the network information system is generally implemented as a set of commonly available information tools. This entails a socio-technical approach to system design, as the specified requirements and the roles that the available tools play must co-evolve [6]. Second, network participants themselves take the initiative to system specification. Third, the specification and implementation of the, initially small, system are gradually expanded during system use. Fourth, there is not a single view on the universe of discourse, which an external team of analysts normally develops in traditional

[0]Proceedings of the Thirty-Second Hawaii International Conference on System Sciences (HICSS-32), Maui, Hawaii, January 5-8, 1999.

information system development. Instead, each user generally is only knowledgeable about and interested in respecifying the workflows he is actually involved in, although at the same time the need for global specification consistency remains.

To foster user participation, a norm-based specification approach is essential. A distinguishing characteristic of many virtual professional communities is that they are egalitarian in nature, because participation is voluntary. Such a community can only thrive if all of its members see the particular interests they represent sufficiently reflected in the communal goals, activities, organizational structures, and supporting information system. Contrary to a hierarchical organization, claims to power cannot be used to control communal specification processes. Instead, permitted, obliged, and forbidden specification behaviour is guided by norms. Unlike explicit rules, norms are often left implicit. Furthermore, norms can occur in very complex, interrelated clusters [7]. Based on these norm groupings, *legitimate specifications* which are (1) semantically meaningful and (2) acceptable to the network participants concerned need to be produced by participants in a form of rational discourse, i.e. discussion which is open and allows background assumptions on usage context factors to be examined [5].

Several specification approaches have been developed, which aim to increase the acceptability of specifications, such as the REMAP method [8]. However, these approaches focus on optimizing a fair *deliberation* process with given participants, but do not indicate how these participants are to be selected in the first place. In the RENISYS (**RE**search **N**etwork **I**nformation **SY**stem **S**pecification) project, a specification method, implemented by a Web-based tool, is being developed that does support such a legitimate user-driven specification process. An initial prototype of the tool has been implemented (Fig. 1, http://infolab.kub.nl/renisys.phtml), which allows for relatively simple specification changes to be formulated and negotiated.

We first explain the approach to user-driven specification adopted in RENISYS.

2. The RENISYS Method

One way to increase the precision of user participation is to let users themselves trigger and define specification processes. Users in general are not continuously involved, but produce specifications intermittently [9]. Empirical evidence also shows that users often initially only need to have an essential understanding of their (business) processes and information tools in order to achieve concrete results [10]. These natural tendencies can be supported by focusing the support of the specification discourse between users on the resolution of *breakdowns*. The support provided needs to aid the user in fostering awareness of a breakdown, and the formulation and negotiation of specification changes that can resolve the breakdown.

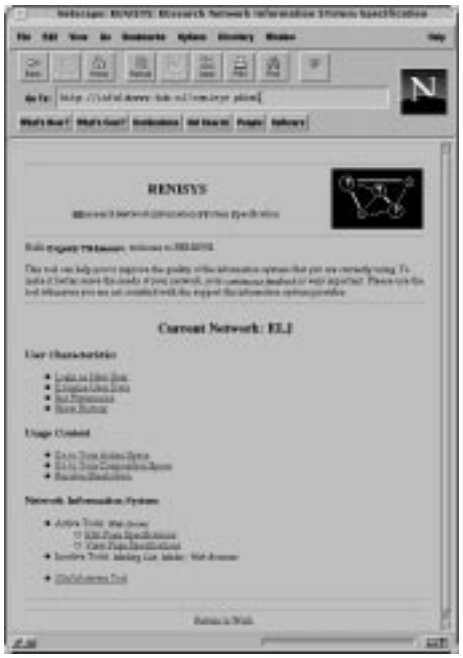

Figure 1: Home page of the RENISYS tool

2.1. Breakdowns

According to Winograd and Flores, the 'breakdown' has a central role in human understanding, and, more particulary, in design. "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 (p.165)" [11]. Breakdowns play two fundamental roles in the design process. On the hand hand, they are the things that the designer will try to avoid, by anticipating on what can go wrong. But when they occur, they also trigger a reconsideration of the functionality of the system. It is only when a user fails in accomplishing a certain goal, that he feels the need for an instrument, and can express what the instrument should do. RENISYS aims at supporting user participation in the specification process precisely at the point a breakdown occurs (no sooner, no later). We hypothesize that such a breakdown-focused user-driven specification approach is especially useful for information systems that support creative knowledge processes, which need a lot of subtle fine-tuning before they are acceptable to a user community. This claim we will try to substantiate in future empirical research by applying RENISYS in several case studies.

2.2. Specification Knowledge Representation

As an underlying knowledge representation formalism, RENISYS makes use of conceptual graphs [12]. To organize the various kinds of complex specification knowledge,

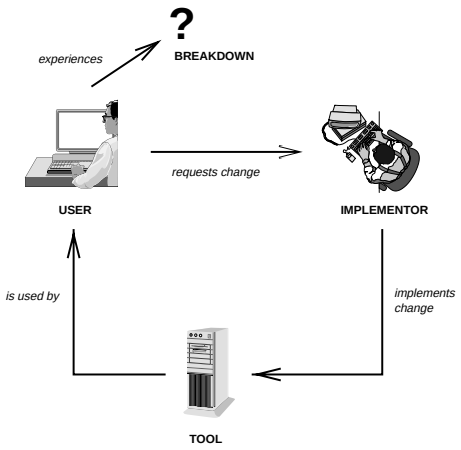

Figure 2: Individualistic User-Driven Specification

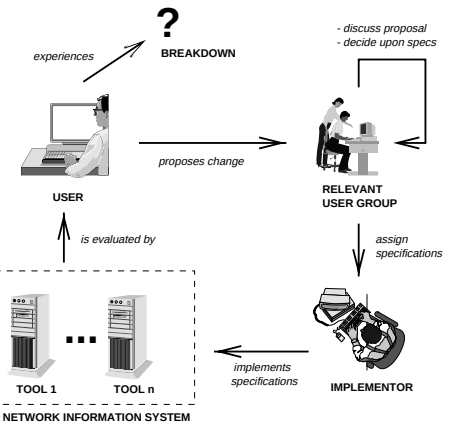

Figure 3: Legitimate user-driven specification

three different categories of specification knowledge are distinguished [5]. Ontological *type* definitions contain descriptions of the terminology used in the community. *Norms* describe the accepted behaviour of the various actors in the network. *State* specifications describe the actual or potential state-of-affairs of network entities.

2.3. A Model of Legitimate User-Driven Specification

In individualistic user-driven approaches, user-involvement in system development generally takes place in relative isolation (Fig. 2). For instance, a user facing a breakdown contacts the system administrator, and asks him to implement a desired change in a Web server. Subsequently, the updated tool is used and the process is possibly repeated.

However, this relatively simple and straightforward specification and implementation loop does not work in a virtual professional community, as any change in tool implementation may profoundly affect group collaboration. As not only communal work, but also specification behaviour is guided by norms, any specification change to be implemented must be legitimate, that is, meaningful and acceptable (as described in Sect. 1.2.)

The process goes as follows (Fig. 3). Again, a user faces a breakdown during work. However, instead of directly requesting an implementor to realize a change, the user formulates his desired specification in terms meaningful to the network. Then, this change is proposed to a *relevant* group of users (see Sect. 5. for more on how to determine what are relevant actors). This group discusses and decides upon the proposed change, while possibly modifying the proposal and initiating new specification processes. The approved specifications are then assigned to a particular implementor. Upon implementation, the user is notified and asked to evaluate the updated functionality, and, if necessary, start a new loop.

Before focusing our attention on the structure and use of these ontologies, we first introduce the case with which we illustrate the theory developed in this paper.

2.4. Case: The Electronic Journal on Comparative Law

IWI, a Dutch organization stimulating new ways of distributing scientific information, funded a project to create an Electronic Journal on Comparative Law (EJCL, http://law.kub.nl/ejcl/). The project group included participants from various academic law institutes, university libraries, and computer centres. The goal was to have all publishing activities, ranging from paper submission to editing, peer review, and publication done completely electronically, making use of the Web. The project started in spring 1997 and ended in summmer 1998.

The initially basic set of requirements defined by the users (e.g. editors or authors) gradually evolved in scope and complexity. Furthermore, the initial basic set of simple information tools over time included more advanced groupware applications. For instance, at first only a relatively simple Web site and mailing list were being used, whereas later also the much more sophisticated BSCW tool [13], supporting advanced group work, was integrated into the network information system. Because of these clear user-driven specification aspects, the EJCL-project was an interesting case to test and refine the theory underlying the RENISYS method.

3. Ontology-Based Support of the Specification Process

The user-driven specification approach applied by RENISYS is briefly described. Next, some advantages of using ontologies are listed. Then, the reference framework used to organize specification knowledge from different domains is introduced. Finally, the ontological framework used to model entities to be stored in the reference framework is presented.

3.1. User-Driven Specification in RENISYS

Key to a successful application of RENISYS is its smooth integration with the network information system it helps to specify. Two important features facilitate this interaction. First, each tool that is part of the information system must include an option for the users to call RENISYS, passing it on a parameter containing the exact location from where the call is being made. For instance, in the case of a Web server, a link can be included in each page, via which the RENISYS server may be contacted. Second, RENISYS includes an authorization mechanism, which identifies the calling user and checks which actor roles he is allowed to play. Through these roles, RENISYS can simply determine which specification authorizations a particular user has.

Furthermore, once the actor roles of a particular user have been determined, the knowledge from (1) these actor roles, and (2) the mappings from domain specifications to the calling page can be combined. RENISYS can then generate a very specific context of entities possibly related to the breakdown the user experiences. This greatly helps the user to - with minimum effort - produce a precise specification proposal to present to the relevant group of users. To determine who are the relevant users and what they should negotiate about, a conceptual graph-based mechanism applying context lattices that can handle such specification knowledge context evolution has been proposed [14]. A language/action theory-based way for such groups of users to discuss and decide upon specifications is described in [15]. However, what has not been described in detail yet, is the structure and evolution of the entities to be specified. To this purpose, ontologies can play an important role.

3.2. Ontologies in RENISYS

In [5], a formal conceptual graph-based approach for specification knowledge, including ontologies, was introduced. The approach allows for the simple customization of ontologies by users, when its graph-based queries are mapped to semi-natural language statements. Three kinds of ontology evolution operations were distinguished, allowing for the creation, modification, and termination of ontological definitions. In this paper, the focus is not on these (essential) mechanisms of ontological change, but rather on the development of a set of ontological primitives that are the starting point for the required ontology evolution. These primitives form the basis for the (user-customized) specifications that define a particular network information system.

The reasons for our use of an ontology-based approach are similar to some proclaimed advantages of domain ontologies (cf.[16]):

1. Domain ontologies may be used in knowledge acquisition where teams have to work together and an ontology serves as the agreed upon, *common understanding* of the terms in the domain [17]. This argument is of particular relevance for networks where groups of users and developers with widely varying backgrounds are involved.
2. A second important function of ontologies is that such a common understanding often entails certain *commitments* for participants playing network roles. These bind users to the implications of the definitions agreed upon. For instance, if an edit process definition says that a set of review guidelines is to be an input into the process, all users implicitly accept that these guidelines need to be available and adhered to. Such commitments, if made explicit, enable more controlled development and maintenance of a knowledge-base system [18].
3. The most often cited role is in enabling *reuse* of knowledge for building new applications. In our user-driven network system environment, reuse not only can be fruitful at the very beginning, but each time a new specification needs to be defined (i.e. when a breakdown has occurred).

In our view, the ontologies form the language necessary to describe the community-wide defined and accepted properties of network actors, objects, and processes, thus providing a description of required functionality. Normative knowledge, on the other hand, provides a way to precisely denote the acceptable information system usage and specification behaviour. As functionality of an information system generally changes less than the usage aspects, the explicit distinction between ontological and normative knowledge allows for more stable system specification and implementation.

We already gave a formal definition of norms in [5], and showed what role they play in user-driven system specification. In this paper, we introduce our ontological framework.

3.3. The Reference Framework

All specification knowledge in RENISYS is organized in a reference framework, consisting of three interrelated domains. Five ontologies provide the concepts needed to describe entities from these domains as well as their linkages.

The Domains

RENISYS specifications belong to, or link entities from, three different domains. A domain is a system of network entities that can be observed by analyzing the universe of discourse from a particular perspective. The *problem domain* is the UoD seen from the task perspective, the *human network* (domain) is the UoD observed from the organizational perspective, the *information system* (domain) is the UoD seen from the functionality perspective. The domain interrelationships are shown in the reference framework (Fig. 4)

The *problem domain* and the *human network* form the two top domains. The distinction between problem domain

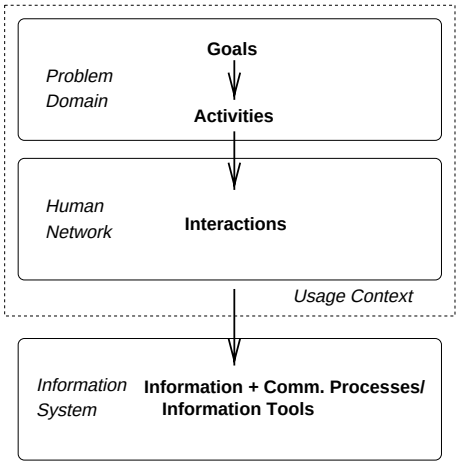

Figure 4: The RENISYS reference framework

and human network is based on Habermas' distinction between work and interaction [19]. In his theory of communicative action these concepts designate respectively a domain of purposive-rational action and a subsuming institutional context.

Together the problem domain and the human network are called the *usage context*, which models the determinants of change in the network information system. In the usage context, the requirements are formulated that determine the high-level design specifications that are generated at the information system level. To understand how the information system evolves, these system determinants need to be described.

Finally, in the *information system*, the actual network information system is represented.

3.4. The Ontological Framework

For each of the domains, a *domain ontology* has been developed, which contains a set of predefined ontological domain axioms or primitives. These primitives can be applied to users either to modify existing definitions or to define new terminology. A somewhat similar approach was used by [8], who extracted primitives from the literature and modified these according to empirical findings. In RENISYS, however, these theory-grounded primitives are meant to be accepted and modified by (authorized) users. To increase the efficiency of this customization process, reference models predefining concepts of particular domains can be useful [20].

To link entities from the various domains, the *framework ontology* contains a set of mappings. These mappings can be used to represent *functionality determinants*, which are entities in the usage context that are connected to entities from the information system domain. For example, the edit process (represented in the problem domain) is linked to a set of communication processes in the information system domain, such as those that enable the distribution of documents to the mailers of users playing the editor roles.

Space does not permit to describe the domain and framework ontologies in detail. We therefore focus on the fundamental ontology of which the domain ontologies are specializations: the core process ontology.

4. The Core Process Ontology

The heart of the ontological framework is the core process ontology, consisting of elementary network process concepts. The concepts in this ontology are derived from language action theories (for an overview of current work see [21]), which are useful for modelling complex goal-oriented communication processes. The core process ontology contains the common vocabulary for three domain ontologies. Such a core process ontology is necessary, because various domain ontologies provide different perspectives on the same reality.

The most generic concepts are *entities*. An entity is an identifyable phenomenon in the universe of discourse. Entities include actors, objects, and processes. Different users can have different views on these entities, but each entity definition has global scope.

4.1. Actors

An *actor* is a context interpreting entity capable of playing transformation control roles. The total set of roles that a specific actor has in turn is played by one or more *subjects*: sentient physical entities in the universe of discourse. A subject can play the roles of more than one actor.

4.2. Objects

An *object* is defined as an entity that can be acted upon or produced in a transformation.

4.3. Processes

Central to RENISYS is its process model (Fig. 5). A *process* is an abstract entity mediating some kind of change in other entities. This change can, for example, be one of space, time, or quality. However, we constrain ourselves to processes that are relevant for the user-driven specification of network information systems. Three main kinds of processes are distinguished: control processes, delegation processes, and transformations. Transformations are further subdivided into workflow, specification, implementation, and use processes. Delegation, implementation, and use processes are not described here, as they are not essential for the modelling of specification processes.

Control Processes

In a *control process*, the status of a controlled process, called a transformation, is changed. A *controller* is an actor playing transformation control roles. Three such roles are distinguished: *initiator*, *executor*, and *evaluator*. The corresponding control processes are: initiations, executions and

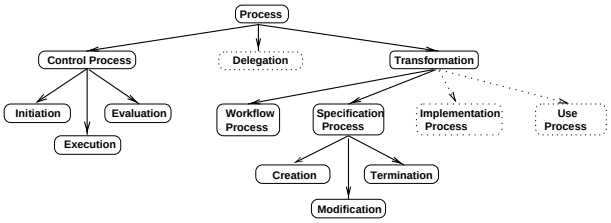

Figure 5: The RENISYS process model

evaluations. In an *initiation*, a transformation is started. In an *actor-initiation* this happens on the own initiative of the initiator; in an *event-initiation* a transformation is required to start because of the successful completion of another transformation, resulting in an explicit inter-transformation dependency. It is also possible that both an actor-initiation and an event-initiation apply. In that case the initiator can decide whether to start the transformation he controls on the condition that the event has taken place. In an *execution*, the intended result of a transformation is realized. In an *evaluation*, an assessment of the change realized in the execution is made. If this result is accepted by the evaluator, the evaluation completing the transformation may result in an event that triggers the initiation of other transformations.

Transformations

We call a process affected by control processes a *transformation*. In a transformation, a single type of *output object* is produced from a set of *input objects* under the control of a set of actors playing the control process roles. This restriction of allowing only a single output object type is necessary to simplify transformation change, as it is now clear which (sub)transformation is affected. This would not be the case if two or more output object types are generated in a single transformation.

Several kinds of transformations are distinguished, based on the kinds of objects they produce. Two categories of transformations treated in this paper are workflow and specification processes.

In *workflow processes*, the production of domain objects is described: a set of input domain objects is transformed into an output domain object.

In *specification processes*, definitions are handled. A *definition* is a proposition in which specification knowledge is asserted. Definitions consist of a defined term and a defining set of terms, which include entities from the universe of discourse. Definitions are of either types, norms, or states. A *type* is class of entities sharing certain properties. A *norm* is the accepted (permitted, mandatory, and forbidden) behaviour of an actor. A *state* is the actual or potential state-of-affairs in the universe of discourse. In specification processes, actors from the universe of discourse (operational level), in which they are involved in workflow processes, play the roles of *specifiers*, which can be further subdivided into *creators*, *modificators*, and *terminators*. In order to do so, three kinds of specification processes are possible: creations, modifications, and terminations. A *creation* is a specification process in which a new definition is created. In a *modification*, the defining set of terms of an existing definition is changed. A *termination* is a specification process in which an existing definition is removed, and references to the terminated definition are reviewed. Examples of such specification processes for types were given in [5].

Actions and Compositions

Control processes combined with transformations form new entities. For each transformation at least one initiation, execution, and evaluation control process must be given, in order for the transformation to be sufficiently defined.

A control process plus its controlled workflow process is called an *action*, in combination with a specification process it is called a *composition*.

Every transformation can only be realized by an ensemble of initiations, executions, and evaluations. Each such action or composition in turn is assigned to actors by means of norms, which, under certain conditions, either permit, require, or forbid them to do these control processes. Norms will be further discussed in the next section.

5. Applying the Ontological Framework

We have used the ontological framework in the EJCL-case to model the specification evolution occurring in the setup of an electronic journal. This was done by constructing a set of process diagrams, based on the concepts as defined in the framework. An example of such a diagram is given in Fig. 6. Users playing various key actor roles in the project were asked to fill in the diagrams as much as they could. Using the framework, it was then determined which actors should evaluate, and, if necessary, modify or complement the specifications. In this way, in a very short period of time specifications were generated which were found to be both accurate and complete by the participants and could easily be mapped to the available set of enabling tools. The manual approach used in this particular case is now being incorporated in an automated version in the RENISYS tool.

A particular advantage of our approach has proven to be that the framework allows for the simple expression of high-level specifications in which details need to be filled only when the users themselves deem this necessary. In other words, the specification follows the users in their thinking rather then forcing them into some (to them) foggy methodological direction. A related workflow modelling approach based on the same line of thought is described in [22].

An actual case example of an ontological specification was the definition of the main workflow of producing a journal needed to accomplish the network goal: publishing an on-line series of electronic journal issues.

At the start of the project, the users identified the main workflow to consist of seven sub-workflows, represented in

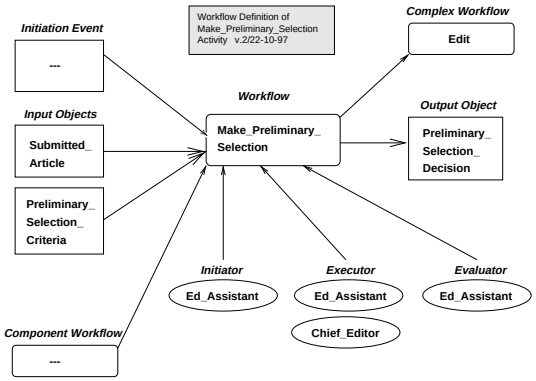

Figure 6: A simple workflow process diagram

the following ontological specification (in conceptual graph notation):

```
[TYPE: [PRODUCE_JOURNAL:*x] -> (def) -
  [ACTIVITY:?x] -> (part) -
    [COMPILE_ISSUE]
    [EDIT]
    [MAINTAIN_SITE]
    [REVIEW]
    [START_UP]
    [SUBMIT]
    [USE]].
```

The ontological knowledge becomes useful when applied in norms that are used to guide the actual specification process. Two main categories of norms are distinguished. *Action norms* describe acceptable behaviour at the usage level (i.e. an editor is permitted to carry out editing). *Composition norms*, on the other hand, prescribe acceptable behaviour at the specification level (i.e. an editor is required to evaluate the modification of any review process type definition). A key entity in both types of norms is the *actor*.

Ontologically, any domain actor is defined as a specialization of some, more generic type of actor. In this case, an editor is considered to be an actor in the problem domain. From a normative point of view, however, an actor is a collection of acceptable actions or compositions.

Actors are defined as context roles both independent of certain subjects (subjects can play an actor role), and independent of the control role (such as 'initiator' or 'executor') that these actors themselves play in transformations. The definition of actors is rather sober, as we have seen above. However, they become alive by the norms that are defined on them. This is illustrated by a brief example adapted from the case.

A network participant using the 'Edit Journal' *web page* finds that this page does not really capture the needs of the network, as only the chief editor is currently allowed to access it. The participant thinks that the assistant editor should also be able to do so. Through interaction with the RENISYS tool, he finds out that currently only the following *action norm* applies to this particular page:

```
[REQ_ACT: [CHIEF_EDITOR] <- (agnt) -
   [CONTROL] -> (obj) -> [EDIT]].
```

This norm says that the chief editor must start, carry out, and assess the the outcome of the editing process, as it is one of his responsibilities. Note that the user is not directly confronted with this abstract representation. Instead, the tool uses it to generate a pseudo-natural language dialogue with the user. In order to also require the assistant editor to at least evaluate the results of the edit process, the additional norm proposed by the participant is the following:

```
[REQ_ACT: [ASS_EDITOR] <- (agnt) <- [EVAL] -
  (obj) -> [EDIT]].
```

However, although this change is proposed by the participant having the breakdown, it is not yet clear if it is also acceptable to the community as a whole. To determine which users are to be involved in the change process of the action norm, the tool needs to project the following query on the existing base of *composition norms* (see [14]):

```
[REQ_COMP: [ACTOR:?] <- (agnt) <- [CONTROL] -
  (obj) -> [CREATE] -> (rslt) -> [REQ_ACT:
  [ASS_EDITOR] <- (agnt) -> [EVAL] -
  (obj) -> [EDIT]]].
```

In a further simple projection, the set of returned actors can then be queried to determine the particular subjects playing those roles. These users are subsequently invited to participate, for instance by sending them an e-mail containing the login code for this particular specification process.

In sum, our approach allows for evolving specification knowledge to be concisely represented, as well as to precisely identify which users should be involved in particular specification processes. Regarding the implementation of these specifications: the specifications at the information system domain level are ultimately represented in terms of those attributes of information tools that can easily modified, such as the public or private status of a mailing list. In this way, a close match between specification and implementation processes can be guaranteed, thus ensuring that the evolution of the technical information system rapidly follows the changes in requirement definitions.

6. Formal Semantics of the Specification Process

The specification process illustrated in the previous section needs to be formally represented in a logical language. This is necessary in order to automate the calculation of which particular users can legitimately be involved in a particular stage of a specification process.

In [23], a formal language called L_{ill} for an integrated semantics for information and communication systems was described. In this modal logic it is possible to specify object types and roles (static part), actions and constraints (dynamic part) as well as communicative actions (illocutionary part). The constraints can be objective (what is possi-

ble or necessary) and deontic (what is permitted or obligatory). The illocutionary part adds the special action category of "speech acts". L_{ill} contains also a special (declarative) speech act for granting, as well as for retracting, authorizations. The language L_{ill} can be used for describing the semantics of the specification process in which the ontological framework is used. Using the language we formally describe how users are authorized to be involved in the subsequent specification process stages. The details of the language are discussed in [24], here we will only introduce the ideas most relevant to the discussion.

In the RENISYS ontology, a composition was defined as a combination of a control process and a specification process. The ontological definition of a *particular* specification process S_Δ (e.g. 'create *some* knowledge definition') includes the knowledge definition being changed (e.g. of the 'edit process type'). To formally represent this, we introduce a set C of control processes, a set S of specification processes, and a set D of knowledge definitions, and define the set $Comp$ of compositions in L_{ill} as the Cartesian product of $C \times S \times D$.

In RENISYS, each particular specification process S_Δ consists of three subsequent compositions $Comp_i$, which indicate the initiation, execution, and evaluation process that make up S_Δ. To carry out S_Δ, a *conversation for specification* needs to be successfully completed, of which the result is an accepted knowledge definition D_Δ. In this conversation, three conversational roles are distinguished: the initiator (I), executor (X), and evaluator (E). A conversational role can be fulfilled by more than one actor. Composition norms determine which actors (and thus which subjects) play the various conversational roles in a particular specification process, by, for example, stating that an editor is permitted to initiate the 'create edit type definition' specification process:

$$Perm\_Comp(Editor, I, Create\_Type\_Edit)$$

In this paper, we do not work out how this mapping from composition norms to conversational roles takes place (see [14] for a conceptual-graph based approach to accomplish this).

The following stages are distinguished in a successful conversation for specification:

- I initiates a specification process by *proposing* to start S_Δ.
- X executes S_Δ by *discussing* a D_Δ.
- E evaluates S_Δ by *deciding* on a D_Δ.

We start using L_{ill} by defining a finite set $Subj$ of constants to represent the subjects. We also have a finite set A of actors. Φ is a first order logic formula and α some speech act $\in \{PROPOSE, DISCUSS, DECIDE\}$. The intuitive meaning of $[\alpha]\Phi$ is that after the execution of α, Φ necessarily holds.

In L_{ill}, different validity claims for speech acts are distinguished. In this context, we use only one, to *authorize* a conversational role to perform a speech act in a specification process, which is represented by means of the $auth$ predicate. A conversational role in RENISYS is authorized to perform a speech act depending on the (initiation, execution, or evaluation) stage of a particular conversation for specification. To say that some conversational role x is authorized to perform speech act α on specification process S_Δ resulting in (intermediate) knowledge definition D_Δ, we propose the following axiom (assuming universal quantifications):

$$P(DO(x, \alpha, S_\Delta, D_\Delta)) \Leftrightarrow auth(x, \alpha, S_\Delta, D_\Delta)$$

which expresses formally that some role is authorized to perform a speech act, if and only if the role is allowed to do it. The predicate $DO(x, \alpha)$ is used to denote the performance of a particular speech act by a particular role. Instead of $[\alpha]\phi$, we write $[DO(x, \alpha)]\phi$, to represent that x is the role carrying out the speech act α. The predicate P is formally defined in terms of the deontic modalities as expressed in deontic logic. These modalities can be represented with the following abbreviations (cf. [24]):

$$\begin{array}{lll} O(\alpha) & == & [\overline{\alpha}]Violation \text{ (}"\alpha \text{ is obligatory"}\text{)} \\ F(\alpha) & == & [\alpha]Violation \text{ (}"\alpha \text{ is forbidden"}\text{)} \\ P(\alpha) & == & \neg[\alpha]Violation \text{ (}"\alpha \text{ is permitted"}\text{)} \end{array}$$

This can be interpreted as that α is obliged if not doing α leads to a violation state, α is forbidden if doing α leads to a violation state and α is permitted if it is not forbidden to do α. Note that the P predicate in the abovementioned axiom is of a higher order than the $Perm$ of permission (norms). In other words, having some $Perm\_Comp$ apply to an actor is a necessary, but not always a sufficient condition to be authorized to perform speech act α. For example, an actor must be permitted by some composition norm to execute a specification process. However, he is only actually authorized to do so after this process has first been initiated as well.

We now need to define exactly when actors playing conversational roles are authorized to perform the various speech acts constituting a conversation for specification. Knowing these authorizations is essential in order to give particular users access to the proper specification tool functionality at the right moment in time. As mentioned before, we assume that we already know which actors $\{i, x, e\}$ in the network are allowed to play the conversational roles $\{I, X, E\}$, as determined by the composition norms that apply to S_Δ.

First, any initiator i of a S_Δ (as determined by the relevant composition norms) is always authorized to propose that S_Δ starts.

$$auth(i, PROPOSE, S_\Delta, \emptyset)$$

The result of this speech act is a proposed specification process according to the following rule:

$$[PROPOSE(i, S_\Delta)\,]prop(S_\Delta)$$

Discussion by the executor x resulting in (tentative) knowledge definition D_Δ can start after a proposal has been made:

$$P(DISCUSS(x, S_\Delta, D_\Delta)) \Leftrightarrow prop(S_\Delta)$$

Once D_Δ has been finalized by the executor, the discussion is closed:

$$[DISCUSS(x, S_\Delta, D_\Delta](disc(S_\Delta, D_\Delta) \wedge \neg prop(S_\Delta))$$

Note that the specification process is no longer in 'proposed' state and that $prop(S_\Delta)$ therefore needs to be removed. Once this has been done, evaluator e is authorized to make a decision:

$$P(DECIDE(e, S_\Delta, D_\Delta, Decision)) \Leftrightarrow disc(S_\Delta, D_\Delta)$$

The actual decision can be either to accept (+) or reject ($\Leftrightarrow$) the discussed D_Δ. Since we assume that the specification process was successful, we represent the acceptance of D_Δ in the following way:

$$[DECIDE(e, S_\Delta, D_\Delta, +)](accept(D_\Delta) \wedge \neg disc(S_\Delta, D_\Delta))$$

which says that a positive (+) decision results in an accepted definition, while its discussed status is retracted. With a negative decision, the proposal is retracted as well, but of course, no accepted status is given.

The specification process was only described in a very rudimentary way, as the main objective was to illustrate how the ontological concepts can be used in the authorization of users involved in the dynamic specification process. Some possible directions of extension of the formal framework are outlined next. In current research we are addressing these and related issues.

It was assumed that all actors playing conversational roles are assigned by means of *permitted* composition norms. However, if required and forbidden compositions are also known (as in any realistic case), other authorizations will be additionally needed.

Another simplification was that compositions were considered to be atomic, in the sense that no generalization/specialization hierarchy of definitions is supported. However, in real-world specification the latter is extremely useful. In [14] we propose a context-lattice based conceptual graph approach, which we are currently developing further.

Furthermore, in RENISYS, two types of specification process (and workflow) initiations are distinguished: the 'actor initiation' and 'event initiation' (as described in Sect. 4.3.). The initiation process described here referred to actor initiations only. To also represent event initiations, the discussion authorization needs to be extended (note that the retraction of $event_{init}$ also needs to be addressed in later speech acts):

$$P(DISCUSS(x, S_\Delta, D_\Delta)) \Leftrightarrow prop(S_\Delta) \vee event_{init}(S_\Delta)$$

Finally, the distinction that is made in L_{ill} between permitted and possible actions is worth noting. In RENISYS, this distinction is particularly relevant, since *possible* is useful in the information system domain ("enabling information tools"), whereas *permitted* is the modality that prevails in the usage context. Formally dealing with these issues is important to describe the matching process between required and (tool-)enabled functionality. The relationships between these two domains can be of various kinds, for instance:

- "everything that is permitted is possible" is a completeness requirement on the implementation functionality. This is a very basic requirement.
- "everything that is possible is permitted" means that only permissible behaviour is implemented, a kind of correctnes requirement (cf. a speed control system that makes it impossible for drivers to violate the speed limit). Note that this requirement is seldom met completely, which means that users can do things wrong, leading to a violation that has to be repaired or sanctioned later. So although many 'permit' constraints at usage context level are translated into functionality of the implementation, and hence lose their deontic status, also in the information system domain there is a need to express norms.

7. Related Work and Conclusion

An ontological framework was described as part of the RENISYS specification method for research network information systems. The framework can be used to structure and support the discussions that users have as part of the specification process. An important feature of our approach is that it allows users to consider functionality (re)specification of the network information system in terms of the particular context in which they use it, thus strongly connecting cooperation for work with cooperation for specification.

A number of tools have been developed, many of them Web-based like the earlier mentioned BSCW tool [13] that allow users to relatively easily modify often complex functionality characteristics. However, structured support for the specification processes is often not given. This makes it hard to ensure the legitimacy of the specifications, i.e. that they are both meaningful and acceptable to a professional community of users. To increase the strong link between usage and specification, some form of descriptive terminology is needed. End-user modifiable tools such as 'radically tailorable tools' [25] do provide a, basic, language for integrating specification and implementation change. However, this tailorability is typically single-user and single-tool focused. On the other hand, socio-technical design methods such as Mumford's ETHICS [26] strongly focus on making legitimate specifications, which are acceptable to a complete community. A drawback of this method, however, is that it is typically oriented towards large-scale, waterfall-like information systems development projects, and lacks the advantages of knowledge-based specification support.

A Web-based RENISYS tool is under development that supports the legitimate user-driven specification process. A first prototype will be tested in a case on group report authoring.

References

[1] S. Talbott. *The Future Does Not Compute: Transcending the Machines in Our Midst*. O'Reilly & Associates, Inc., 1995.

[2] H. Rheingold. *The Virtual Community: Homesteading on the Electronic Frontier*. HarperPerennial, 1993.

[3] A. De Moor. Toward a more structured use of information technology in the research community. *The American Sociologist*, 27(1):91–101, 1996.

[4] S.R. Tilley and D.B. Smith. Documenting virtual communities. In *Proceedings of the 14th American International Conference on Systems Documentation (SIGDOC '96), Research Triangle Park, NC, October 20-23*, pages 43–49. ACM, 1996.

[5] A. De Moor. Applying conceptual graph theory to the user-driven specification of network information systems. In *Proceedings of the Fifth International Conference on Conceptual Structures, University of Washington, Seattle, August 3–8, 1997*, pages 536–550. Springer-Verlag, 1997. Lecture Notes in Artificial Intelligence No. 1257.

[6] J. Bowers. Making it work: a field study of a "CSCW network". *Information Society*, 11(3):189–207, 1995.

[7] R. Stamper. Signs, information, norms, and systems. In B. Holmqvist and P.B. Andersen, editors, *Signs at Work*. De Gruyter, Berlin, 1996.

[8] B. Ramesh and V. Dhar. Supporting systems development by capturing deliberations during requirements engineering. *IEEE Transactions on Software Engineering*, 18(6):498–510, 1992.

[9] L. Salchenberger. Structured development techniques for user-developed systems. *Information & Management*, 24(1):41–50, 1993.

[10] L.J. Arthur. Quantum improvements in software system quality. *Communications of the ACM*, 40(6):46–52, 1997.

[11] T. Winograd and F. Flores. *Understanding Computers and Cognition - A New Foundation for Design*. Ablex Publishing Corporation, 1986.

[12] J.F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[13] T. Horstmann and R. Bentley. Distributed authoring on the web with the BSCW shared workspace system. *Standard View*, 5(1):9, 1997.

[14] A. De Moor and G. Mineau. Handling specification knowledge evolution using context lattices. In *Proceedings of the Sixth International Conference on Conceptual Structures, ICCS'98, Montpellier, France, August 10–12, 1998*, pages 416–430, 1998.

[15] S. Hoppenbrouwers, A. De Moor, and H. Weigand. A discourse-based concept definition process for user-driven specification. Technical report, Infolab Research Report, 1998.

[16] J. Breuker, R. Winkels, and A. Valente. A core ontology for law. In *Proceedings of NAIC'97*, 1997.

[17] T. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer, 1994.

[18] G. van Heijst, A. Schreiber, and B. Wielinga. Using explicit ontologies in KBS development. *International Journal of Human-Computer Studies*, pages 183–191, 1997.

[19] J. Habermas. Technology and science as "ideology". In *Toward a Rational Society: Student Protest, Science, and Politics*. Beacon Press, 1971. (Originally published by Suhrkamp Verlag, Frankfurt am Main. 1968).

[20] N. van der Rijst and A. de Moor. The development of reference models for the RENISYS specification method. In J.F. Nunamaker Jr. and R.H. Sprague Jr., editors, *Proceedings of the 29th Hawaii International Conference on System Sciences, Maui, January 3–6, 1996*, pages 455–464. IEEE Computer Society Press, 1996.

[21] F. Dignum, J. Dietz, E. Verharen, and H. Weigand, editors. *Proceedings of the First International Workshop on Communication Modeling 'Communication Modeling - The Language/Action Perspective', Tilburg, The Netherlands, July 1-2, 1996*. Springer eWiC series, 1996. http://www.springer.co.uk/eWiC/Workshops/CM96.html.

[22] I. Hawryszkiewycz and A. De Moor. A workflow approach to designing cooperative systems. In *Proceedings of the Third International Conference on the Design of Cooperative Systems (COOP'98), Cannes, France, May 26-29, 1998*, 1998.

[23] H. Weigand, E. Verharen, and F. Dignum. Integrated semantics for information and communication systems. In R. Meersman and L. Mark, editors, *Database Application Semantics*. Chapman & Hall, 1997.

[24] J.-J.Ch. Meyer. A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic*, 29(1):109–136, 1988.

[25] T.W. Malone, K.-Y. Lai, and C. Fry. Experiments with Oval: A radically tailorable tool for cooperative work. *ACM Transactions on Information Systems*, 13(2):177–205, 1995.

[26] R. Hirschheim and H.K. Klein. Realizing emancipatory principles in information systems development: The case for ETHICS. *Management Information Systems Quarterly*, 18(1):83–109, 1994.