Requirements Eng (2001) 6:75–96

Requirements Engineering

Making Workflow Change Acceptable

Aldo de Moor and Manfred A. Jeusfeld
Infolab, Tilburg University, Tilburg, The Netherlands

Virtual professional communities are supported by network information systems composed from standard Internet tools. To satisfy the interests of all community members, a user-driven approach to requirements engineering is proposed that produces not only meaningful but also acceptable *specifications. This approach is especially suited for workflow systems that support partially structured, evolving work processes. To ensure the acceptability, social norms must guide the specification process. The RENISYS specification method is introduced, which facilitates this process using composition norms as formal representations of social norms. Conceptual graph theory is used to represent four categories of knowledge definitions: type definitions, state definitions, action norms and composition norms. It is shown how the composition norms guide the legitimate user-driven specification process by analysing a case on the development of an electronic law journal.*

Keywords: Breakdown; Knowledge representation; Legitimate user-driven specification; Social norms; Virtual communities; Workflow modelling

1. Introduction

Collaborative work is increasingly being done in a distributed fashion. It is no longer only carried out in the classic hierarchical organisation, where detailed orders are given from the top down to groups of employees. Rather, teams of collaborating professionals from within and across organisations are expected to set their own goals and organise their own ways of working. This collaboration requires having people from many different organisations and locations work together intensively on complex tasks. Distributed computer technologies are therefore useful for supporting these professional networks. Especially commonly available Internet-based network information tools, ranging from mailing lists to a wide range of sophisticated Web applications, have great potential to increase the efficacy of collaboration. However, matching complex collaborative requirements with these tools is not a trivial task. The requirements engineering process therefore deserves careful support.

The *virtual professional communities* in which such collaboration is to take place are defined as networks of professionals whose collaboration for realising shared goals is mostly or completely computer enabled. To model the community's activities, we see the work processes being carried out in such communities as *workflows*. However, despite the many promises offered by network technologies, it turns out to be very hard in practice to make them successfully support the workflows of these communities. One of the main reasons is that an ongoing co-evolution of network requirements and supporting information tools is needed for adequate collaboration in these dynamic networks to be realisable.

In order for the members of a virtual professional community to collaborate productively, it is not sufficient to merely make available a set of standard Internet information tools. Instead, an integrated information system supporting the communal workflows needs to be constructed out of these standard technological components. The selection and configuration of these tools should be dependent on the collaborative context in which they are being used. Active user participation in the specification process of such continuously evolving socio-technical systems is needed. As the required changes cannot be predicted, a specification process should be started as soon as a user

Correspondence and offprint requests to: Aldo de Moor, Infolab, Tilburg University, PO Box 90153, 5000 LE, Tilburg, The Netherlands. Email: ademoor/jeusfeld@kub.nl

encounters a *breakdown*, which is any problem related to the work or its technological support that is experienced by a user during its performance.

To increase the legitimacy of changes in the information system of a professional community, proposed modifications should not only be meaningful with respect to the existing system requirements, but also *acceptable* to all community members. To systematically support such a *legitimate user-driven* specification process, specification *methods* are needed.

Traditionally, the system development phase has been decoupled from the use phase of a system. System developers elicit requirements from users, then implement a system according to these requirements, and deliver that system to the users. In our case, however, we are interested in methods that support the continuous co-evolution of both the requirements and the implementation. We believe that such an approach is technically feasible if limited to workflow systems that are built from reusable Internet tools. Workflow systems are interesting research objects for studying the acceptability of requirements changes since they target distributed teams. Moreover, they provide flexible facilities to adapt them to those changes.

1.1. Towards Acceptable Specification Change

The members of virtual professional communities, like their counterparts in normal communities, are subjected to shared *social norms*. To ensure the acceptability of specification changes, these norms should guide both the operations of such a network and the specification processes in which the network is being defined. As virtual professional communities are generally egalitarian in nature, the norms that guide these processes cannot be imposed from above, but need to be defined by the community itself.

For acceptable specifications to be produced via a method, a way of capturing these norms in a formal representation needs to be found. *Composition norms* are what we call the formally represented social norms guiding the specification process. For specification changes to be acceptable, the composition norms must be used to determine which users need to be involved in what capacity in a specification process. If this can be accomplished, all community members can see the particular interests they represent sufficiently reflected in the specified goals, activities, organisational structures and information technologies of the network. Such a norm-guided specification approach should make collaboration possible, not only upon the initiation of the network, but especially when tasks, organisational structures and supporting technologies have grown in complexity.

The RENISYS (REsearch Network Information SYstem Specification) method provides such a composition norm-guided approach. Earlier work focused on the representation and organisation of norms [1,2], and on the development of an ontological framework in which specification knowledge can be expressed [3]. A full description of the recent state of method and prototype tool, as well as the underlying theory of legitimate user-driven specification, is given elsewhere [4,5,6].

1.2. Related Work

Key aspects of requirements engineering methods are the negotiation of requirements between stakeholders and the traceability of requirements.

The requirements engineering community has considered collaboration and negotiation between stakeholders for several years. Robinson and Volkov [7] draw an analogy between the systems life cycle and negotiation life cycle. The negotiation (of requirements) should be planned and tool supported just like the systems life cycle. Systems development in general aims at producing a system in teamwork whereas negotiation aims for a contract resolving conflicting goals. The main players in the negotiation life cycle are the *owner* (main stakeholder of the negotiation outcome), the *analyst* (formalises and refines the owner's goals), the *designer* (plans the interaction between the participants), and the *facilitator* (provides the tools for negotiation). The authors identify communication support, creative conflict resolution and contract monitoring as open areas of research. For the first area, software agents exchanging formalised goals are proposed. The second area is about opening up the search space of possible solutions to a conflict. This turns out to be particularly difficult when the goal models are automatically processed. Finally, contract monitoring is about the match between the contract and its enactment in the system development and usage phases. Interesting in this approach is the identification of agent roles and the view on negotiation as a structured process. The RENISYS approach covers similar roles and provides a formal framework to represent their rights and obligations. However, it does not focus on automated conflict support, but on identifying which users can play what negotiation roles in various stages of the requirements negotiation process.

A thorough empirical study of traceability models and link types is given in Ramesh and Jarke [8] and Ramesh [9]. The study covers about 25 cases from various

industries and exhibits more than 40 incarnations of traceability link types. Traceability can be applied to learn from previous system development processes and to guide the current process. Of particular interest with respect to our approach are models of rationales which record the arguments and assumptions behind a design decision. In the KBSA-ADM system [10], for instance, a truth maintenance system is incorporated to keep the network of assumptions, rationales and decisions consistent. Essentially, it helps stakeholders to discuss in a consistent way about requirements. The RENISYS method adds the dimension of legitimacy to such a network: who is allowed to propose meaningful changes and who has to be consulted in the decision making process?

A third area of related work is enterprise modelling. A prominent example is the ARIS toolset [11]. ARIS splits the co-evolution task into four levels. The *process design* level is the modelling environment (consisting of data models, control models, organisational models and function models) where all aspects of the computerised information system are abstractly represented. Below that is the *process management* level. Here, the execution of the (implemented) information system is supervised by monitoring systems. Certain parameters of the running information system can be adjusted when the monitor system exhibits unsatisfactory performance. The third level concerns *workflows*. The control model is mapped to a workflow plan and executed by a (standard) workflow management system. Finally, the *process application* level manages reusable components supporting certain steps in the workflow (e.g., a word processor for writing invoices).

There is a major drawback with applying a traditional approach like ARIS to supporting the legitimate user-driven specification of network information systems in virtual professional communities: although the meaning (semantics) of the changes is often specified in great detail, it is unclear who is authorised to make what specification changes. Changes made under the control of an external design team are often opaque to the users, and may not reflect the interests of a community as a whole. This, in turn, can lead to systems that no longer satisfactorily support their communities. Therefore, any specification change proposed to resolve a breakdown must be *acceptable* to the community as well. Only then can the legitimacy of these changes be guaranteed. So, our research question is: *How can a community adapt its information system without violating its social norms?*

The objective of this article is to show how the norm-guided specification processes supported by the RENISYS method can produce acceptable specification changes of (workflow-enacting) network information systems. The method is improving on the *agreement* dimension mentioned by Pohl [12], by providing a language to express and evaluate so-called composition norms. These norms specify permissions, obligations and prohibitions of the community members in the discourse about system changes.

In Section 2 we present our view on workflow modelling as a process that should be focused on interpretation instead of on representation. Section 3 outlines the RENISYS method. Section 4 introduces the knowledge representation approach used. In Section 5 we illustrate the method by discussing the support it provides in subsequent steps of the legitimate user-driven specification process and by analysing a simplified case of the development of an electronic law journal. Section 6 contains some conclusions.

2. Workflow Modelling

Many definitions exist of the workflow concept. One definition conceives of a workflow as a unit of work that happens repeatedly in an organisation of work [13]. However, this interpretation does not clearly capture two important aspects of workflows: (1) that they are coordinated and (2) that they can be supported by information technology. A more explicit definition along these lines is that a workflow is an automated organisational process, which means that the coordination, control and communication of activities are automated, but that the activities themselves can be either automated or performed by people [14]. However, this definition needs to be relaxed, as in virtual professional communities much of the coordination and control is done by human actors. Therefore, we define a *workflow* as a recurring unit of work of which the coordination, control and execution can be partially or completely automated.

2.1. Workflow Management Systems

To automate or *enact* workflow processes, workflow management systems are required. These are systems that allow for the design, execution and management of business processes and activities in a network [15]. Two kinds of workflow management systems can be distinguished, based on the degree to which the workflows are structured. Most current workflow systems are transaction focused, which means that they create structures to implement and enforce frequently recurring processes. Ad hoc applications, on the other hand, focus more on supporting creative knowledge activities. Their main aim is to provide some but no complete control to make sure that tasks, responsibilities,

etc. are delivered [16]. Such deliverables and control structures are of a more indeterminate kind than those available in transaction-focused systems. Thus, in ad hoc workflow management systems more attention needs to be paid to the triggering and evaluation of workflows by human beings.

Most workflow-enacting information systems currently either focus on unstructured or on structured processes. However, the processes in between, the *partially structured workflow processes*, often lack support [14]. This is definitely the case in virtual professional communities, in which very creative, unstructured work is linked in complex ways to, for instance, formal document management procedures. In order to enact this whole spectrum of workflows, one should be able to model the different kinds of dependencies that these modes of work entail.

2.2. A Basic Workflow Model

Workflow modelling methods can be of two categories: activity-based (also known as input–process–output (IPO)-based) and conversation-based approaches [14,17]. In activity-based approaches, the focus is on the objects being used and produced in a process and on the task and process dependencies, whereas in conversation-based approaches the commitments created in conversations between participants play a central role.

A typical example of the first approach is the ExSpect method [18]. Using the Petri net formalism, in ExSpect complex tokens can be moved between input and output channels and stored via transitions. The method has proven to be especially useful for the modelling, simulation and analysis of the logistical workflows of business organisations.

Two well-known examples of conversation-based workflow modelling methods are the ActionWorkflow approach [19] and the DEMO specification method [20]. In the ActionWorkflow approach, conversations are structured around the action workflow loop. In this loop, a performer agrees to perform some action to the satisfaction of the customer who proposed it. In the DEMO (Dynamic Essential Modelling of Organisations) method, the core concept is the transaction, in which two actors, the initiator and the executor, agree on an essential action to be carried out, in what is called an actagenic conversation. The actors agree that this action has been satisfactorily performed by producing a fact in a factagenic conversation.

In order to specify the partially structured workflows prevailing in virtual professional communities, both modelling approaches are incorporated in our model of the workflow. From the activity perspective, each workflow has a set of input and output objects. From the conversation point of view, each workflow has an initiator and executor. However, in addition to the customer/performer, initiator/executor roles of respectively ActionWorkflow and DEMO, we distinguish a third role: the *evaluator*. This role is necessary, as in loosely organised networks often the participants who are responsible for approving the result of a workflow are different from the ones who started the process or produced the result. The resulting model of a single workflow is shown in Fig. 1. More detailed workflow models that include aggregation and inter-workflow dependencies are required in real-world cases, but for the sake of argument this simple model suffices here.

Both activity- and conversation-based workflow modelling methods still face some serious drawbacks [17]. First, their models are often either too rigid or too lax. What they should offer is the possibility to just capture the constraints that require satisfaction. Second, they are fragile, in the sense that the models become irrelevant when unforeseen changes occur in the context to which they apply. For instance, in an editorial environment, publication and review processes are often defined either very broadly, leaving too many degrees of freedom for the user needing guidance, or too detailed, making the system inflexible and hard to change. Especially in virtual professional communities, such as research environments, accurate workflow models are crucial, but it is often very difficult to predict their structure in advance [21].

These factors may have contributed to the fact that so far only little concrete guidance has been given for the redefinition of complex workflows, which is necessary for them to better meet evolving user needs [17,22]. Another factor contributing to the current problems is that there has been insufficient involvement of users in the modelling process, whereas workflow management systems should allow users and implementors to jointly define workflow processes [13].

Summing up, many of these problems are caused by current methods paying too much attention to the *representation* of detailed requirements and designs, instead of focusing on the organisation of the human

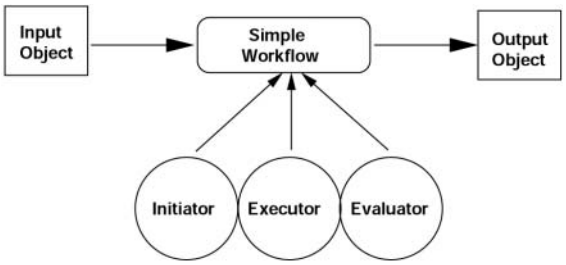

Fig. 1. A basic workflow model.

interpretation process of the complex and sometimes chaotic context of work. Instead of regarding specifications as directly implementable accounts of work processes, they should be considered as heuristic devices that can be used to discuss and reach agreement on who is to be involved in their specification process in what particular way [23]. In the next section, the RENISYS specification method is outlined, which aims to deal with these problems by developing such an interpretive approach.

3. The RENISYS Method

In the RENISYS project, a theory on legitimate user-driven specification has been developed, as well as a specification method that has been partially implemented by means of a prototype web server. Theoretical work related to the project has been published elsewhere [1–6]. We first give a summary of the developed approach in this section, before focusing on the norm-guided specification process in more detail.

In the project, an analysis was first made of the need for network information systems in research networks, which form an important class of virtual professional communities. A model was developed of Internet-enabled research collaboration supported by common information tools, such as mailing lists and web applications. The core of this model is a reference framework, consisting of three domains, which give different perspectives on the same reality. Key entities in all three domains are workflows, as well as the actors that play roles in these workflows, and the objects that are produced in them. The first domain, the *problem domain*, models the workflows, called *activities*, that are needed to accomplish the goals of the community. The *human network* describes the organisational structures and workflows of the community, e.g., group structures and discussion and decision-making processes. Together, these domains form the *usage context* of the information system. In the *information system*, the required information and communication (I/C) processes of the network are represented. These processes (e.g., editing a text or sending a mail) manipulate information objects. These required I/C processes are then mapped to the I/C processes that are enabled by the information tools comprising the information system. As indicated before, we do not consider the specification process in which this modelling takes place to happen only once. Instead, it is regarded as a continuous form of process composition [24], in which users gradually define and refine their requirements and mappings to the supporting information tools.

To define the characteristics of the user participation in the specification process, a study was made of existing user-centred specification approaches. These were classified along two dimensions: the roles that users play as *sources* and as *modellers* of specification knowledge. A class of specification methods which is still largely undeveloped is that of legitimate user-driven specification methods, which can be characterised as methods in which the users themselves control the modelling process and where groups of users, rather than individual users, act as sources of specifications.

Using this background, we next outline the actual specification approach, which focuses on the handling of breakdowns.

3.1. Breakdown Handling

As soon as a user becomes aware of a breakdown in his work or supporting tools, the breakdown needs to be formulated in terms of knowledge definitions to be changed. These changes should be worked out and accepted in a form of group conversation. The breakdown-handling process is therefore subdivided into three stages:

1. *Breakdown awareness:* The individual user becomes aware of his breakdown, invokes the specification tool and classifies the breakdown.
2. *Breakdown formulation:* The individual user identifies those problematic knowledge definitions which, in his view, need changing.
3. *Breakdown resolution:* The group of relevant users produces a set of knowledge definition changes that are legitimate, i.e., meaningful and acceptable, to the community as a whole.

In Fig. 2 the specification method that supports this breakdown-handling process is outlined: when a user becomes *aware of a breakdown*, he can call the RENISYS tool from the application he is currently using, for example by clicking a link in the web application currently being used. This *calling information tool* can then be considered as the starting point for breakdown formulation. For instance, if an editor is editing a report using some web page, the likelihood is high that the problems experienced are related to the workflows (i.e., editing processes) being supported by this calling information tool (assuming that a web page can be regarded as such a tool). Next, the user classifies the breakdown, for instance indicating that it is a breakdown having to do with the tool support provided. RENISYS subsequently supports the individual user in the *breakdown formulation* process by suggesting definitions from the *knowledge base* that could be

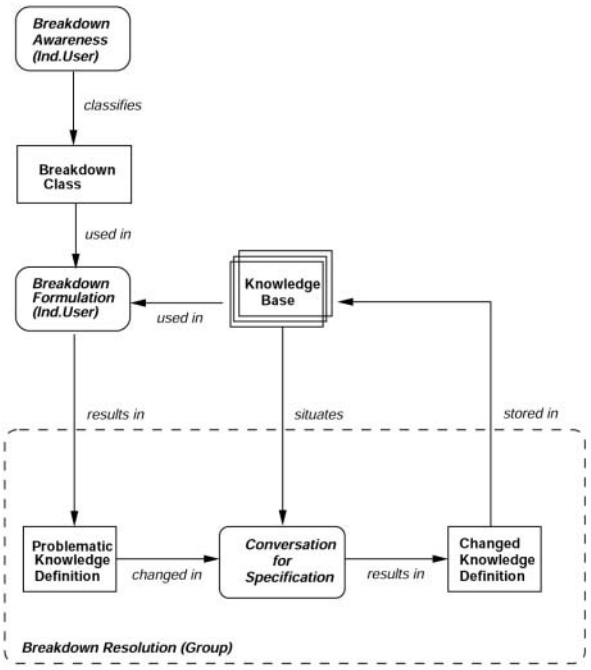

Fig. 2. The RENISYS specification method.

problematic. These suggestions are partially based on the knowledge about the workflows supported by the calling information tool, and partially on the breakdown class. In the above example, RENISYS asks the user if the edit process definition needs to be changed. The breakdown formulation stage results in one or more *problematic knowledge definitions* that need to be changed. For each problematic definition, one or more *knowledge definitions* from the knowledge base are changed in a *conversation for specification*. Definitions from the knowledge base can also be used to situate the conversation for specification, for example, by showing the knowledge context of the definition to be changed, and by selecting which users to involve in the conversation. In Section 3.1.2 a concrete example of the handling of breakdowns using RENISYS is given.

Whereas the breakdown awareness and formulation stages are conducted by the individual user facing the breakdown, the breakdown resolution process is considered to consist of a group conversation for specification. From our point of view, theories from the Language/Action Perspective (LAP) school, such as speech act theory and Habermas's theory of communicative action, are well suited to model these conversations. In this perspective, language is not only usable to *say*, but also to *do* something (for an introduction to the language/action approach, see Dignum et al. [25]). Speech act theory is used to represent the specification process as a conversation about a knowledge definition change. However, this theory alone does not allow for the adequate challenging of background assumptions, which is essential if specification changes are to be acceptable to all members of the community. Van Reijswoud's Transaction Process Model (TPM) [26] builds on both speech act theory and the theory of communicative action to structure communication acts in business processes. He distinguishes between successful transactions, which lead to the intended result, discussion processes, in which unclarities can be resolved, and discourse processes, which help to explore the context of concepts used in the other processes. We have adapted the TPM to a Specification Process Model (SPM) to represent conversations for specification as processes of rational discourse, in which such context assumptions can be challenged. An extensive description of the make-up of these conversations is given in de Moor [4]. In this paper, we do not repeat how the conversations themselves are to be modelled and supported. Instead, we focus on the equally important problem of how to determine who should be involved in these conversations and in which capacity.

A major remaining problem is how to ensure that only those users are involved in a conversation for specification to whom a knowledge definition change is relevant. In other words, a conversation for specification needs to be situated, in the sense that the *context* of the knowledge definition to be changed determines its settings. This context should help to find out both which users to involve and what other knowledge definitions are needed in order to make the change. Key concepts in modelling the contexts of conversations for specification are composition norms. Whereas *action norms* guide the operational workflows of network participants (i.e., defining that an editor is permitted to *carry out* the edit process), *composition norms* describe the acceptable workflow changes they can make (i.e., defining that a steering committee is required to *change the definition* of the edit process). Composition (and action) norms have one of three *deontic effects*, which indicate the obligations they create for the actor to which they apply: the compositions they describe are either permitted (i.e., the actor *may* be involved), required (the actor *must* be involved), or forbidden (the actor *may not* be involved). A permitted composition (norm) is an example of a *privilege*, a required composition is called a *responsibility* and a forbidden composition is classified as a *prohibition*.

To represent and reason about composition norms, and to handle norm conflicts, we make use of two deontic theories, namely Stamper's semiotic theory [27] and dynamic deontic logic [28]. Using this interpretation of norms, a context model has been developed, in which

both an internal and an external conversation context are distinguished. The *internal conversation context* determines which knowledge definitions are related to the knowledge definition being changed, depending on the particular kind of specification process being carried out. Composition norms that are applicable to the various users determine the *external conversation context* of a conversation for specification. Using this context, it can be determined which users can legitimately control the conversation, in the sense of initiating, executing or evaluating it. In this paper, we focus on the external conversation context, as we want to describe *who* to involve in specification changes. In de Moor [4] a full description is given of how to calculate the external conversation context out of a given set of composition norms. Here, we only informally illustrate how this is to be done in Section 5.

We next illustrate the prototype tool that has been developed to support the specification process. The architecture of the RENISYS tool is first described, after which its functionality is demonstrated with an example.

3.1.1. Tool Architecture

The RENISYS specification tool supports the users playing different (individual or group) roles in the specification process outlined in Fig. 2, by generating different kinds of web pages. The specification tool consists of a web server that can be accessed by any common web browser, such as Netscape or Explorer. The server is composed of three interrelated servers: a knowledge base, a page generator and a script server.

The *knowledge base* stores and makes accessible all specification knowledge definitions and auxiliary information. It has been implemented as a Postgres server. Such a server can manage a number of relational databases, each of which contains one or more tables. In these tables tuples of information are stored.

The *page generator* creates the web pages that provide the interface with the user. The generator has been implemented as a web server that allows for the dynamic generation of web pages interpretable by any web browser. A dynamic web page is similar to an ordinary web page, but, in addition, its variables can be set at runtime.

The *script server* provides the scripts that enable complex procedures, like those required for the calculation of norm dynamics.

By accessing web pages called *forms*, users issue various kinds of *specification commands*. The forms are interpreted by the page generator, which may send one or more *SQL commands* to the Postgres server. This server then carries out the requested operations on the tables of its databases, and returns the *SQL results* to the page generator. This server uses the results to generate a new *web page* that the user can view with the browser.

Instead of sending a set of SQL commands to the Postgres server, the page generator can also send a *complex command* to the script server. By this we mean a command that requires one or more special procedure or function scripts to be executed. These scripts themselves may also issue SQL commands to obtain or update the data they require for their calculations. After the commands have been executed, the script server returns the command results to the page generator.

3.1.2. Tool Functionality: A Use Case

The functionality of the tool is best illustrated by an example. Assume that John, who is a reviewer of an electronic journal, is using a web browser to fill in review comments on a paper, and feels dissatisfied. His problem is that he would actually like to be able to discuss a paper with other reviewers and the editor, instead of doing the current individual blind review.

After John has called RENISYS from the application he is currently working with (i.e., the 'Review Paper' web page), and has identified himself, he is presented with the 'Problem Awareness' page (Fig. 3). He then has to indicate what is the most likely cause of his breakdown. One of four options needs to be selected:

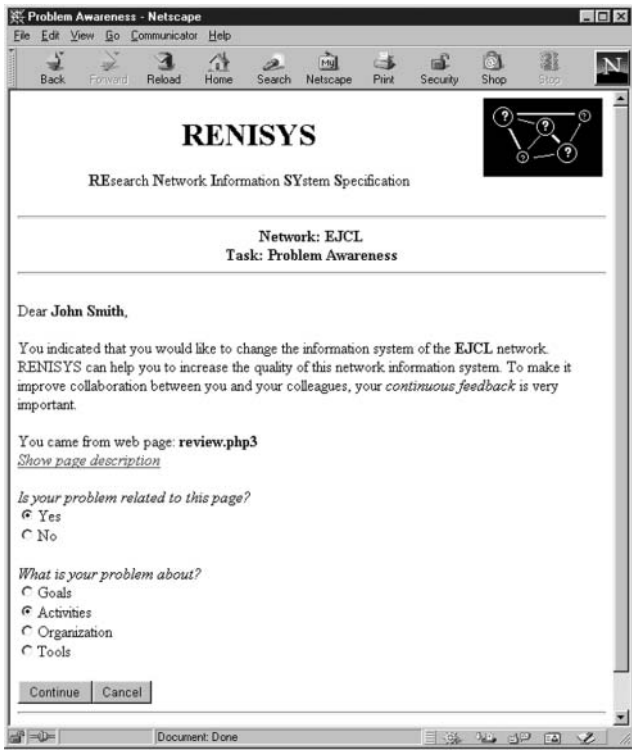

Fig. 3. The RENISYS tool: classifying the problem.

goals, activities, organisation or supporting tools of the network. John selects *activities* since he wants to change the review process.

In the next page (not displayed), the main parts of the knowledge definitions linked to the selected breakdown class are presented. In the example, let us say that two activities are mapped to the 'Review Paper' web page: the *Edit* and the *Review* process. John selects the *Review* process as the *core breakdown entity*. John is then also shown all the knowledge definitions related to the core breakdown entity, from which he selects the *problematic definition* that he wants to be created, modified or deleted. In this case, since the core breakdown entity is a workflow, there are three related kinds of knowledge definitions: first, there is the workflow itself (*Review* process). Second, there are the related action norms (all the norms indicating who can initiate, execute or evaluate the workflow). Third, there are the related workflow mappings. These are mappings that show by which the concrete information and communication processes workflows are actually supported. After considering these options, John selects the *Review* process as the problematic definition.

In the next RENISYS page (not displayed), John indicates that he wants a new activity, the *Paper Discussion* process, to be created, and explains in free text what is his problem and what he suggests as the solution. This change is an example of a *type creation* specification process (see Section 4 for an overview of all specification processes).

RENISYS calculates which composition norms apply to this so-called *active specification process*, and as a result of this calculation selects the users who should be invited to initiate, execute and evaluate the knowledge definition change. When it is their turn, these users are invited by email to access the relevant 'conversation for specification' page that is part of the specification tool. Here, they produce a new, tentative *Paper Discussion* type definition. Once this preliminary definition has been accepted by the evaluators, the *implementor* of the journal web server, for example the system administrator, is notified of the definition change. He is sent all the comments made by the users involved in the specification process, so that he knows what to change. The moment he has implemented the changes, John and all the users involved in the definition change are invited to evaluate whether the implemented changes indeed are what they expected them to be.

This, in a nutshell, is a description of the specification process supported by RENISYS. In Section 5.1 this basic functionality is explained in more detail. The current implementation of the tool is a research prototype. At present, only a subset of the specification processes (e.g., type creation processes) is fully supported. However, work is underway to develop a robust version of the tool that can be used for a large variety of real-world applications. This version will also include a more refined definition presentation format plus a short tutorial on the basics of legitimate user-driven specification.

The above example shows the essence of legitimate user-driven specification. The main advantage is in the correct enactment of the social norms that should guide the evolution of the system. Using RENISYS, the stakeholders can be assured that the changes to the system do not violate their legitimate interests.

In this article, we concentrate on the roles that the knowledge definitions play in the specification process, as both initialising and being legitimately changed in these conversations. In the next section, Section 4, we introduce the representation of the specification knowledge categories distinguished in RENISYS.

4. Knowledge Representation

The intended output of a specification process is a changed knowledge definition. The specification process consists of a conversation for specification and an embedded *definition process*, in which the actual knowledge definition change is carried out. We distinguish 12 different *definition processes* (three types of specification processes for each of four categories of knowledge definitions, discussed below). The semantics and properties of these definition processes are less relevant to understand the legitimacy of the overall specification process. They are discussed in detail in de Moor [4].

There are three kinds of specification processes: *creations*, *modifications* and *terminations* of knowledge definitions. We distinguish four categories of knowledge definitions: type definitions, state definitions, action norms and composition norms. As explained earlier, *action norms* define which actors can be involved in what workflows, whereas *composition norms* describe who can make changes in these and derived knowledge definitions. To explain the ontological meaning of the terms used in the definitions, *type definitions* are necessary. For example, a type definition of a workflow can represent that there must be at least one input and one output object. *State definitions* are required to express entities in the actual world, e.g. the particular users of a specific tool.

In order to represent, reason about and use the knowledge definitions, conceptual graph theory has been selected as the knowledge representation formalism. We first briefly give the rationale for our use of this

theory and introduce some of its core elements. In the next section we present our formal representation of the various specification knowledge categories.

4.1. Conceptual Graph Theory

In this section, we first introduce this theory and then use it to describe the semantics of the knowledge definition categories.

4.1.1. Why Conceptual Graph Theory?

The representation language of choice must be rich enough to allow the efficient expression of complex definitions. Since the users themselves are the specifiers, it is important that their definitions can be easily transformed into a (pseudo-) natural language format. At the same time the representation language must be sufficiently formal and constrained to allow for meaningful specification inferences to be made. Conceptual graph theory has these properties, and has been used to represent knowledge definitions and the operations that can be applied to them [1].

Conceptual graph theory (CGT), introduced by John Sowa [29], provides a knowledge representation formalism for typed first-order logic, which has been applied to many different domains, from natural language processing to information systems specification. Conceptual graphs can be represented in both a graphic and a linear notation. CGT 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 [30]. The formal structures and operations of the theory have been shown to be useful for representing and processing terminological knowledge in concrete applications; see, for example, Angelova and Bontcheva [31].

Users do not have to understand conceptual graphs. The graphs are used to describe and theorise about the knowledge specification processes, which can be implemented using dedicated conceptual graph tools or by other techniques. For example, a graph can be stored as a tuple in a particular database table. However, conceptual graphs are not directly presented to the users. For presentation purposes, the tool can carry out more or less simple transformations to fill out slots in predefined natural language sentences, which can be either assertions or requests. In this way, users can interact with the specification tool in a semi-natural way, while the formality of the graphs ensures that the semantics of and inferences applied to the specification knowledge are well understood. Thus, the graphs described in the next sections are only concise summaries of possibly much more complex implementations and presentations. Note that in the current implementation of the tool the presentation of the definition is still quite primitive. Based on the results of actual tool usage and results from conceptual graph research, we intend to refine the format of the presented information considerably in the near future.

4.1.2. A Brief Introduction to Conceptual Graph Theory

As CGT is relatively unknown, we give a brief introduction of conceptual graphs in linear notation. Conceptual graphs are constructed out of concepts and conceptual relations.

- Concept

A concept has two fields: a type and a referent field. The fields are separated by a colon and surrounded by square brackets. The type field contains a type label that is part of a type hierarchy. The referent field designates a particular entity with the mentioned type. The referent field is optional: if it is not given, the concept is considered to be a generic concept, by default quantified with the existential quantifier.

A referent can be a generic marker, denoted by `*`, or an individual marker. If the referent is a generic marker, then the concept is a generic concept. The generic marker may be followed by a variable identifier (lower case), e.g. `*x`. These named generic markers are useful for cross-referencing concepts. Individual markers identify individual concepts. They consist of a number sign followed by some identification number or constant (upper case), e.g. `#123` or `#John`. Sowa also distinguishes various set referents, which we do not use here. All definitions using set referents can be translated into simple definitions using simple referents. Using set referents would introduce theoretical complexity, which is unnecessary for our specification purposes and would distract from the main point of this paper.

A concept has the following format:

```
[Type : Ref]
```

An example of a concept is `[List_Owner : #John]`, which says that John is a list owner.

- Conceptual Relation

A conceptual relation links two or more concepts. Each conceptual relation has a relation type, surrounded by parentheses, and one ore more arcs, represented by arrows, each of which must be linked to some concept. If a relation has *n* arcs, it is called *n-adic*. In our research, we only make use of dyadic relations. A dyadic relation has the following representation:

```
[Type1 : Ref1] → (R_Type) → [Type2 : Ref2]
```

Generally, the relations can be read, in the direction of the arrows, as '*source-concept* has a *relation* to the *destination concept*'.

An example of a relation is

```
[Buyer : #Mary]→(Poss)→[Book : #War_ and_Peace]
```

which, literally translated, means 'There is a buyer Mary who possesses the book *War and Peace*'.

- Conceptual Graph

A conceptual graph is a combination of concept nodes and conceptual relation nodes. A graph can consist of a single concept, but every arc of every conceptual relation must be linked to some concept.

To represent such a graph, one of its concepts is chosen as the head of the graph. If more than one relation is connected to a concept, the dash symbol '–' can be used to separate the common concept from the rest of these relations. A conceptual graph is ended by a period.

The general format of a conceptual graph is based on this pattern:

```
[Type1 : Ref1] –
  (R_Type1) → [Type2 : Ref2]
  (R_Type2) → [Type3 : Ref3] –
    (R_Type3) . . .
```

A *nested graph* is a special kind of conceptual graph, which has as its referent a set of conceptual graphs. These graphs are said to occur in the context of the type of concept. A nested graph with a singleton graph referent, which is the kind of graph we use here, has the following format:

```
[Type : [Graph]].
```

We use these nested graphs to represent specification knowledge definitions. The general graph format for specification knowledge definitions is

```
[K : [Definition]].
```

Here, the knowledge category K ε {Type, State, AN, CN}, whereas Definition represents a specification knowledge definition in graph format. As the graph representation for each category of knowledge definition differs, they will each briefly be discussed next.

4.2. Specification Knowledge Categories

For each knowledge definition category, the role and structure of the definition are presented.

4.2.1. Type Definitions

A *type definition* is an ontological definition. An ontology is an explicit specification of a conceptualisation, which itself is an abstract, simplified view of the world that needs to be represented for some purpose [32]. In our case, this purpose is the specification of network information systems. 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. Ontological definitions provide a definition of the meaning of an entity. Such knowledge can be represented in different ways. For our purpose, type definitions are most suitable.[1]

Type definitions provide a clear canonical core for definition knowledge, which is useful in eliciting and checking required properties of specification entities. The basis for an ontology consisting of type definitions is a *type hierarchy*. The type of each entity used in any state, ontological or normative definition must be included in this type hierarchy.

A type definition has the following format:

```
[Type : [td : *x] → (Def) → [tg : *x] dif(dt)].
```

Here, t_d is the *defined type*, t_g is the *genus* (the supertype), and **dif**(d_t) is the *differentia* of the type definition. The differentia consists of a subgraph that specialises the genus to the defined type.

An example of a type definition is the following one, which represents that an editorial process is an activity that results in a publication:

```
[Type : [Edit : *x]→(Def)→[Activity : *x] –
  (Rslt) → [Publication]].
```

4.2.2. State Definitions

A *state definition* asserts something about one or more *particular* entities in the universe of discourse. Such a definition will normally contain at least one instantiated entity of which the value is known, e.g. stating that John is a list owner of some mailing list. Still, there may be cases when a state definition contains no such

[1] Two knowledge representation techniques often used to describe the meaning of entities are schemas and type definitions. *Schemas* can be used to represent stereotyped or prototypical definitions of entities. A schema contains a number of slots with default values, describing properties of the entity. These default values can be overruled in case of exceptions. *Type definitions*, on the other hand, only represent *essential* properties that define the meaning of what it requires to be an entity of a particular type. We use type definitions instead of schemas to represent the meaning of entities, because we want ontological definitions only to contain *typical*, not just *prototypical* knowledge (cf. [33], p. 373).

instantiation, for instance when it is known that there is some list owner, but it is not (yet) known which person plays this role.

A state definition has the following format:

```
[State: [Definition]].
```

`Definition` is a conceptual graph formed by a combination of concepts and relations. This graph must be a specialisation of some type definition in order to be meaningful.

The graph definition of the example just mentioned would be

```
[State : [List_Owner : #John]→(Poss) -
  [Mailing_List : *]].
```

4.2.3. Action Norm Definitions

Action norms describe the roles that users can play in the control of workflows through actor roles. An *action norm* regulates the behaviour of users at the operational level by saying that an action is either permitted `(Perm_Action)`, required `(Req_Action)` or forbidden `(Forb_Action)`.

An action norm (definition) is represented as follows:

```
[AN: [Actor] ← (Agnt) ← [Control] -
  (Obj) → [Workflow]].
```

The following action norm says that an editor is permitted to execute the edit process:

```
[Perm_Action: [Editor] ← (Agnt) ← [Exec] -
  (Obj) → [Edit]].
```

4.2.4. Composition Norm Definitions

A *composition norm* regulates the behaviour of users at the specification level by saying that it is either a permitted composition `(Perm_Comp)`, a required composition `(Req_Comp)` or a forbidden composition `(Forb_Comp)`.

A composition norm (definition) is represented as follows:

```
[CN : [Actor] ← (Agnt) ← [Control] -
  (Obj) → [Specify] → (Rslt) -
    [Definition]].
```

`[Specify]` stands for some specification process, i.e., a creation, modification or termination process of the knowledge definition. `[Definition]` stands for some definition in one of the four knowledge categories, i.e., type, state, action, or composition norm definitions.

The following composition norm says that an editor is permitted to initiate the modification process of the edit process (type):

```
[Perm_Comp : [Editor] ← (Agnt) -
  [Init] → (Obj) → [Modify] → (Rslt) -
    [Type : [Edit : *x]]].
```

5. Supporting the Legitimate User-Driven Specification Process

In this section, we illustrate how the web-based RENISYS supports the legitimate user-driven specification process. We first show how the method, making use of the knowledge definitions introduced in the previous section, concretely supports the users in their specification process steps. We then show how specification processes consisting of these steps can be repeatedly applied to support the evolution of the network information system of a realistic community, by using material from the case of the development of an electronic law journal. A similar analysis of the specification evolution of a research network on sustainable development was done in de Moor [4].

5.1. Specification Process Steps

In Section 3.1.2 the functionality provided by the RENISYS method and tool was briefly illustrated showing the hypothetical example of the handling of user John's breakdown during the review process. To resolve this breakdown, he proposed the creation of a new type of paper discussion process. In the current section, we illustrate the concrete support provided by RENISYS by exploring this example in more detail, subsequently looking at the support for the various specification steps in the breakdown awareness, formulation, and resolution stages.

Breakdown Awareness

- *Step 1: Invoking RENISYS*

In general, the user will invoke the RENISYS tool while working in another tool, for instance, some web application. He is then asked if his problem is related to the web page from which RENISYS was called. In RENISYS, mappings can be made from particular web pages of these invoking tools to specific knowledge definitions in its specification knowledge base. This is useful to limit the amount of definitions presented to the user. If the problem is indeed related to the calling page, then only the mapped definitions will be shown as potentially problematic definitions. For example, if a web page allows a user to edit a document, then only the definitions related to the editorial process are shown to the user.

Example: John, who was working on the 'Review Paper' page and who wants to create a completely new review discussion process, indicates that the problem is indeed related to the page he invoked.

- *Step 2: Classifying the Breakdown*

The complete set of legitimate knowledge definitions is not presented to the user all at once. First, the breakdown is classified, using classes that are related to the work situation and that the user can easily understand. For instance, in the prototype tool, the user can choose from the following breakdown classes: goals, activities, organisation and tools. Each of these classes links to different subsets of knowledge definitions. For instance, the breakdown class *Activities* comprises type definitions which have as their defined type *Activity*. If necessary, the breakdown classes can be changed in future versions of RENISYS; however, the four underlying knowledge categories – type definitions, state definitions, action norms and composition norms – will remain the same.

Example: John, who wants to create a new paper discussion process, selects *Activity* as the breakdown class, because he is not satisfied with the current properties of the activity *Review Paper*.

Breakdown Formulation

- *Step 3: Selection of Core Breakdown Entity*

Once the breakdown class has been chosen, the breakdown formulation process starts. First, all knowledge definitions mapped to the selected breakdown class (and, if selected, calling page) are retrieved. Depending on the conceptual class, *parts* of these definitions are then shown, from which the user has to select the *core breakdown entity*. Note that on the RENISYS web pages the query to be selected is not presented in raw graph format. For instance, in the case of an activity breakdown, asking the user to select a core breakdown entity is posed as: 'Please select the activity type most closely related to your problem.' The activities to be selected are then obtained from the type definition graphs that describe activities, and displayed in a standard pulldown list.

Example: John has an activity breakdown. In this case, type definitions are retrieved that define activities. Since John indicated that his problem was linked to the calling page, only those activity-defining type definitions related to the 'Review Paper' web page are used, which are those definitions describing review and editorial processes. Thus, the pulldown list now only contains the *Review* and *Edit* labels. Since the new paper discussion process is part of the review process, John selects the *Review* activity as the core breakdown entity.

- *Step 4: Selection of Problematic Definition*

For each type of core breakdown entity, there is a set of *related definitions*. For instance, for *activities*, the related definitions are the action norms and the workflow mappings (which are type definitions). The appropriate definitions can be retrieved from the knowledge base by making the appropriate conceptual graph projections. For action norms, such a projection retrieves all norms in which the *Workflow* component is a subtype of the selected activity. The core breakdown entity and its related definitions are then presented to the user on the web page in an easily readable format. From this set, the user selects the most *problematic definition*. If needed, this can be repeated.

Example: John needs to select a problematic definition from: the core breakdown entity, the action norms related to this process, and the workflow mappings that indicate by which information and communication processes this workflow is supported. The core breakdown entity was identified as the *Review* activity. One of the retrieved action norms related to the *Review* process, could be:

```
[Perm_Action : [Editor] ← (Agnt) ← [Exec] –
  (Obj) → [Edit]].
```

This norm is presented to the user in a pulldown list as 'An editor is permitted to execute the editorial process'. An example of a workflow mapping is: 'The Review Process is a Group Discussion Process that is enabled by a Mail Redistribution Process.' John, since he wants to create a new kind of review process, selects the Review activity as the problematic definition.

- *Step 5: Scheduling of the Problematic Definition*

Now that a problematic definition has been identified, the user needs to determine the action to be performed on it. If the user wants a critical discussion to be started on it, the definition is added to the *discourse agenda*. This discourse process is explained in detail in de Moor [4]. If, on the other hand, the user wants the definition to be *changed*, it is added to the *specification agenda*. In that case, the appropriate specification process type needs to be indicated: either to *create* a new definition based on the problematic definition, to *modify*, or *terminate* it. Besides this scheduling process, the user also makes a free-format *problem description*, in which he explains why the definition needs to be criticised or changed. Once the definition scheduling process has finished, another problematic definition may be selected and scheduled, if necessary.

Example: John indicates that he wants to create a new *Review* activity: the *Paper Discussion* process. In the problem description, he explains why such a refinement

is needed. RENISYS then classifies this request as a *type creation* specification process.

Breakdown Resolution

- *Step 6: Selection of Users to Involve*

The proposed change of the problematic knowledge definition is made in the *active specification process*, which consists of three *compositions*: the *initiation*, *execution* and *evaluation* of the specification process. At the heart of the specification process is the *definition process*, in which the actual definition change is made. In the case of a (workflow) type creation process, this definition process would include identifying the supertype, the input and output objects of the workflow, etc. To select which users to legitimately involve in the different compositions, the *applicable norm sets* need to be calculated. For each user and composition (i.e., user John and composition 'execution of *Review*-type creation'), a separate applicable norm set is calculated.

In de Moor [5], the algorithms to do the complex conceptual graph calculations required to determine the applicable norm sets are presented, involving generalisation hierarchies of norm graphs. The norms in these sets can have conflicting *deontic effects*, e.g. one saying that it is forbidden, another one that it is permitted for user John to execute the creation of type definitions of the review process. To handle these conflicts, a norm conflict resolution mechanism is needed. Ours uses a slight variation of standard dynamic deontic logic [34]. An example of how conflicting norms could apply to the same user is that John has these two norms in his applicable norm set for the execution of the review type creation process: a general norm saying that it is forbidden for users to change their own workflow processes, as well as a more specific norm saying that it is permitted for reviewers to be involved in the changing of such workflows.

For each applicable norm set, its *resultant deontic effect* is calculated, which says if it is permitted, required or forbidden for a particular user to either initiate, execute or evaluate the particular specification process in which the problematic knowledge definition is to be changed.

Once the resultant deontic effects for all applicable norms sets have been calculated, all legitimate initiators I, executors X and evaluators E of the activity specification process are known.

Example: In the example, John has one prohibition and one privilege in his applicable composition norm set for the composition 'execution of the *Review*-type creation process'. The resultant deontic effect would be that it is forbidden for John to be involved in the execution of this process (which would consist of modifying a type definition template for the *Review* supertype), since prohibitions have precedence over privileges in our logic:

$$\mathbf{de_r}(D_{CN\_APPL(John,\ Exec\_Create\_Type(Review))}) = Forb$$

On the other hand, for a user Jane, the resultant deontic effect might be that she is permitted to be involved in the execution of this definition process (see Fig. 4).

- *Step 7: Supporting the Conversation for Specification*

Users who have been selected in the norm calculation process receive an email when their input is required. Initiators are invited if the user who has the breakdown (the *problem experiencer*) does not have initiator privileges himself. If this is the case, a legitimate initiator must judge whether it is worth starting a breakdown resolution process.

Various RENISYS web pages are used to support the group conversation in which the specification process is carried out. In this conversation, the definition to be

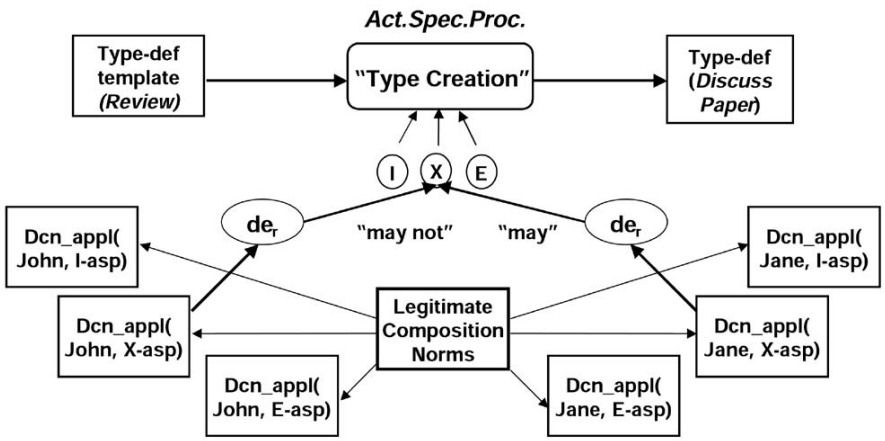

Fig. 4. Support for a legitimate type creation process.

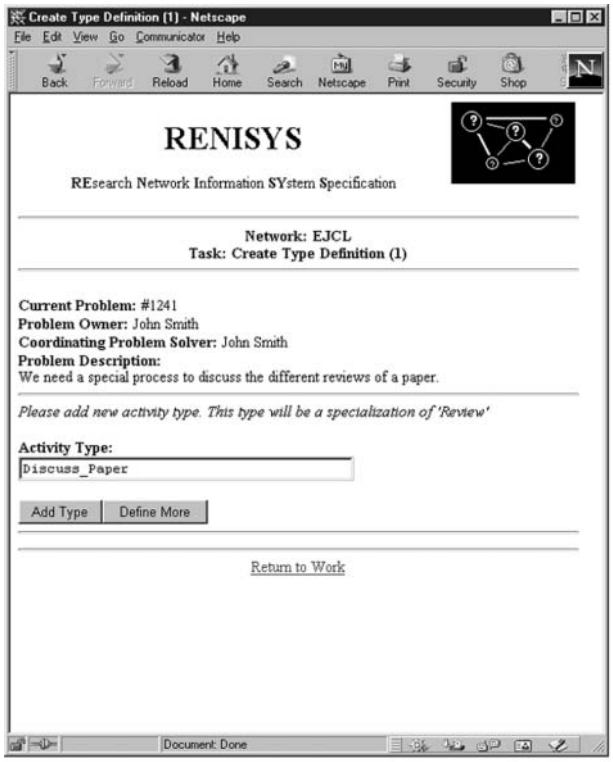

Fig. 5. The RENISYS tool: a type creation page.

changed is initiated, worked out, evaluated and, if necessary, further discussed. These procedural aspects were discussed in detail elsewhere [4,6].

At the heart of each conversation is the definition process in which the actual knowledge definition change is produced. In Section 4 we distinguished 12 such processes (one for each combination of specification process and knowledge definition category). Each of these definition processes comes with its own set of customised web pages. An example of a type creation page is shown in Fig. 5.

In this subsection, we showed in detail the functionality that RENISYS provides, as well as the role that knowledge definitions play in the coordination and performance of the specification process in which a knowledge definition is changed. Next, analysing a concrete case, we show the role that sequences of such specification processes can play in the evolution of the virtual professional community and its supporting network information system.

5.2. Case: The Development of an Electronic Law Journal

Single knowledge definition change processes, as illustrated in the previous section, are the basic building blocks of network information system evolution. We next show how the repeated use of these processes can be used to foster the growth of virtual professional communities.

IWI, a Dutch organisation stimulating new ways of distributing scientific information, funded a project to create an *Electronic Journal on Comparative Law* (EJCL[2]). 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, being done completely electronically, making use of the web. The project started in spring 1997 and ended in summer 1998.

The initially basic set of requirements defined by the users (e.g., editors or authors) gradually evolved in scope and complexity. Furthermore, the 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 [35], supporting advanced group work, was integrated into the network information system. Because of the clear legitimate user-driven specification aspects, the EJCL project was an interesting case to test and refine the theory underlying the RENISYS method.

Although the EJCL conversations for specification largely took place in face-to-face mode, we assume that the kinds of composition norms discovered there are similar to those needed in virtual conversations. In fact, their explicit representation is even more important there: they can serve as the main drivers of the specification process, since the normal social cues that abound in physical meetings are lacking. Future empirical research will have to demonstrate to what extent composition norm-guided change processes in virtual and physical professional communities are indeed comparable.

5.2.1. The Ontological Framework

Before a virtual professional community can start working with its network information system, and is able to specify its changes, a set of meaningful knowledge definitions is needed. Such definitions can be in the form of a *reference model*. Based on a typology of virtual professional communities, such a model can be used to select clusters of definitions that are most likely to satisfy the requirements of the particular community of users. If necessary, they can be modified by the users before taking effect. We described the use of such reference models in system specification in more detail in Van der Rijst and de Moor [36]. Since the purpose of

[2]http://law.kub.nl/ejcl/

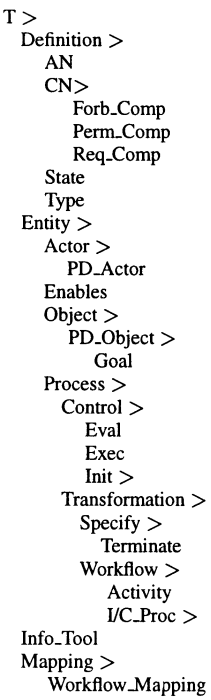

Fig. 6. The type hierarchy for the example.

this article is to show how users can manage their own workflow *change*, instead of to describe all their potential requirements in detail, we will not use a full reference model here. Instead, we base ourselves on the RENISYS *ontological primitives*. Since space is lacking to completely describe the ontologies, we will only introduce those concepts necessary to represent the knowledge definitions made in the example. The RENISYS ontological framework was described in more detail in de Moor and Weigand [3]. The type hierarchy composed from the elements of the framework relevant to the current example is given in Fig. 6. These concepts will be introduced in the definitions that follow.

5.2.2. Supporting the Network Life Cycle

Viable virtual professional communities require many knowledge definitions, which are defined in sequences of breakdown-handling processes over a longer period of time. Virtual professional communities are networks that are not immediately fully operational, but go through several *life cycle* stages to mature [37]. These we call the *network conception*, *construction* and *operation stages*. Additionally, a fourth, *network termination stage*, could also be distinguished. To illustrate, we describe the first two stages of the life cycle: the network conception and construction.

5.2.2.1. Network Conception

Although the legitimate user-driven specification process is in principle ongoing, it has to get started at a particular moment in time. This *network conception* stage is crucial, in the sense that the initial knowledge definitions produced determine to a large extent the success of future specification processes. Definitions created in this stage have a special status. Since they are used to get the network started, the definitions cannot be created in a legitimate way yet. In a sense, this situation resembles a fundamental problem in law, where the conception of a legal system is always fraught with problems: who is allowed to define a law when no law yet exists to govern this legal conception process?

However, this lack of legitimacy of the initial definitions in effect is less serious than it may seem. First, participation in virtual professional communities is essentially voluntary, so that prospective members can refuse to participate if these initial definitions are not to their liking. Second, because of the strongly developed participatory definition change mechanisms, problematic definitions can be criticised or changed later on.

In order to produce the conception definitions, a special actor, the *network conceiver*, is appointed. This actor only has temporary existence. It can produce any definition deemed necessary, without being constrained by composition norms. However, after these definitions have been created, the network conception stage comes to an end, and the network conceiver role is terminated. From then on, any definition change should be legitimate.

Definitions to be produced in this stage include at least:

- some type definition of a network constructor who is to be responsible for making the construction stage definitions.
- one or more subjects playing this role (state definitions).
- the compositions' norms that describe which compositions the network constructor is permitted, required and forbidden to make. A subset of these norms could be *constitutional norms*, being those composition norms that should not be modifiable by the users. Such norms are not studied here, however.

The project to create an electronic journal is sponsored by the IWI organisation. The network conceiver makes the following definitions:[3]

[3]The verbal description of each knowledge definition is followed by the subject who is the definer, as well as by the definition identifier. This identifier consists of a knowledge category indicator ('T' = type, 'S' = state, 'AN' = action norm, 'CN' = composition norm) followed by a number.

– *A sponsor is an actor in the problem domain (NC,T1).*
– *IWI is a sponsor of the project (NC,S1).*

- (T1) `[Type : [Sponsor : *x] → (Def) - [PD_Actor : *x]].`
- (S1) `[State: [Sponsor : #IWI]].`

The network conceiver defines some initial composition norms. First, to prevent users from removing restrictions on their specification rights, they should not be able to change their own forbidden compositions.

– *No actors in the network should be able to change the forbidden compositions in which they themselves play a role (NC,CN1).*

- (CN1)

```
[Forb_Comp : [Actor : *x] ← (Agnt) -
  [Control] → (Obj) → [Specify] → (Rslt) -
    [Forb_Comp : [Actor : *x] ← (Agnt) -
      [Control] → (Obj) → [Specify] → (Rslt) -
        [Definition]]].
```

Furthermore, the sponsor should be able to change all definitions:[4]

– *A sponsor may change any knowledge definition (NC,CN2).*

- (CN2)

```
[Perm_Comp : [Sponsor] ← (Agnt) -
  [Control] → (Obj) → [Specify] → (Rslt) -
    [Definition]].
```

The sponsor is responsible for handling changes in the goals of the network:

– *A sponsor must handle the specification process of any network goal (NC,CN3).*

- (CN3)

```
[Req_Comp : [Sponsor] ← (Agnt) -
  [Control] → (Obj) → [Specify] → (Rslt) -
    [State : [Goal]]].
```

Thus, these knowledge definitions precisely specify how the sponsor may currently be involved in the evolution of the socio-technical system that is formed by the electronic journal community and its supporting network information system. The composition norms ensure that other users cannot change the goals of the network without the sponsor's agreement. Or, put more positively: the sponsor will always be invited into discussions about goal changes (and is expected to participate in them, since the last composition norm concerns a responsibility, which means that the actor *must* be involved). Note that the composition norms themselves are only used to determine who is to participate in these discussions (conversations for specification, see Fig. 2), the inner workings of which are outside the scope of this paper.

[4]Apart from its own forbidden compositions. However, since the sponsor has no need to change them, it is not affected by CN1.

5.2.2.2. Network Construction

To reduce the risk of making illegitimate knowledge definitions, the set of network conception definitions should be minimal in size. The definitions required for the development of the network should be legitimately produced by the network participants themselves as soon as possible. Thus, after the conception definitions have taken effect, the *network construction* stage begins. In this stage, the network participants define the knowledge definitions needed to get the network going. User involvement should be relatively simple to accomplish, since willingness to participate is mostly high at the beginning of the life of the network.

If the construction stage is complex, it can be subdivided. One natural way to partition the network construction stage is according to the main activities or business processes. Different subgroups of users, covered by the composition norms defined in the previous stage, can work out the definitions for these focal points of network development.

The construction stage of the electronic law journal network consists of various substages. First, there is the definition of the project organisation. Then, the overall publishing process is defined to consist of a number of main activities. The definition of one of these activities, the paper submission process, is then described.

(1) Defining the Project Organisation

The project organisation is defined by IWI, the sponsor. The goal of the network is to produce an issue of the electronic journal:

– A journal issue is an object in the problem domain (IWI,T2).
– There is an issue of the *Electronic Journal on Comparative Law* (EJCL) (IWI,S2).
– The goal of the network is to produce this issue (IWI,S3).

- (T2) `[Type : [Journal_Issue : *x] → (Def) - [PD_Object : *x]].`
- (S2) `[State : [Journal_Issue : #EJCL]].`
- (S3) `[State : [Goal : #EJCL]].`

5.2.2.3. Norm Dynamics Calculation

We illustrate the calculation of the norm dynamics by taking a closer look at the specification of definition S3. At the time of its specification, *legitimate* composition

norms (CN1–CN3) are in force. Since there is at least one user (IWI) playing an actor role (Sponsor) that matches with the actor component of each of these norms, all three norms are *invoked*. If the composition part (i.e., control process plus specification process) of one of these norms matches with one of the compositions of the *active specification process* (the process in which the current knowledge definition, i.e. S3, is to be produced), that involved norm becomes an *active norm*. The specification process for definition S3 is the 'creation of a goal state definition'. The active norms for this process are CN2 and CN3. These norms also form the *applicable* norm sets for user IWI for all active compositions, which means that they cover the initiation, execution and evaluation of the active specification process for that particular user. Of these norms, CN3 has preference, since in our norm conflict resolution semantics a required composition has a higher priority than a permitted composition (see de Moor [4]). The *resultant deontic effect* for user IWI is therefore that it *must* handle the (initiation, execution and evaluation of the) creation process of a goal state definition when triggered.

The sponsor is not going to coordinate the project itself, but appoints a project team:

– *A project team is an actor in the problem domain (IWI,T3).*
– *There is a project team PT1 (IWI,S4).*

- (T3) `[Type: [Project_Team : *x] → (Def) - [PD_Actor : *x]].`
- (S4) `[State: [Project_Team : #PT1]]`

– *The project team is permitted to make any definition change (IWI,CN4).*

- (CN4)
```
[Perm_Comp : [Project_Team] ← (Agnt) -
  [Control] → (Obj) → [Specify] → (Rslt) -
    [Definition]].
```

However, the sponsor does not want the project team to be able to change the sponsor's right to overrule any knowledge definition that it does not agree with:

– *The project team may not make any changes in the composition norms which specifically concern the sponsor (IWI,CN5).*

- (CN5)
```
[Forb_Comp : [Project_Team] ← (Agnt) -
  [Control] → (Obj) → [Specify] → (Rslt) -
    [CN : [Sponsor] ← (Agnt) -
      [Control] → (Obj) → [Specify] → (Rslt) -
        [Definition]]].
```

Furthermore, the goals of the network are determined by the sponsor and should not be changed by the project team:

– *The project team may not change the goals of the network (IWI,CN6).*

- (CN6)
```
[Forb_Comp : [Project_Team] ← (Agnt) -
  [Control] → (Obj) → [Specify] → (Rslt) -
    [State : [Goal]]].
```

Finally, IWI appoints a project coordinator.

– *A project coordinator is an actor in the problem domain (IWI,T4).*
– *There is a project coordinator PC1 (IWI,S5).*

- (T4) `[Type: [Project_Coord : *x] → (Def) - [PD_Actor : *x]].`
- (S5) `[State: [Project_Coord : #PC1]].`

The project coordinator is made specifically responsible for evaluating that any required functionality is adequately enabled, which in RENISYS is defined by both so-called support definitions and required implementations. A *support* definition assigns a specific user and tool to a workflow mapping, thus representing how some required functionality for that user is to be enabled. A *required implementation* definition assigns, for all users, some information tool to a particular required information or communication process. An example of such a definition is that a mailing list server is to be accessible to all users for the redistribution of sent mail.

– *The project coordinator is required to evaluate all support definition changes (IWI,CN7).*

- (CN7)
```
[Req_Comp : [Project_Coord] ← (Agnt) -
  [Eval] → (Obj) → [Specify] → (Rslt) -
    [State : [Support]]].
```

– *The project coordinator is required to evaluate all required implementation definition changes (IWI,CN8).*

- (CN8)
```
[Req_Comp : [Project_Coord] ← (Agnt) -
  [Eval] → (Obj) → [Specify] → (Rslt) -
    [State : [Req_Impl]]].
```

(2) Defining the Publishing Process

The project team, once operational, starts with the definition of the publishing process, which is the main activity needed to accomplish the goal of having a published journal. It is allowed to do so because of permitted composition CN4 and because none of the forbidden compositions CN1, CN5 and CN6 applies.

The project team defines the publishing process in two partial type definitions:

– *The publishing process is an activity in which a journal issue is produced (PT1,T5).*
– *The publishing process consists of an edit process, a review process, a submission process, ... (PT1,T6).*[5]

- (T5) `[Type: [Publish : *x] → (Def) - [Activity : *x] → (Rslt) → [Journal_Issue]].`
- (T6)
```
[Type : [Publish : *x] → (Part) -
        [Edit]
        [Review]
        [Submit]
        ...].
```

Next, the project team appoints a technical committee that is to take care of defining the technological support for enabling the required functionality.

– *A technical committee is an actor in the problem domain (PT1,T7).*
– *There is a technical committee TC1 (PT1,S6).*

- (T7) `[Type : [Techn_Comm : *x] → (Def) - [PD_Actor : *x]].`
- (S6) `[State : [Techn_Comm : #TC1]].`

The project team gives the technical committee the responsibility to determine which information tools are to enable the work done in the network. In RENISYS, these are described by support and required implementation definitions:

– *A technical committee is required to handle any support definition change (PT1,CN9).*

- (CN9)
```
[Req_Comp : [Techn_Comm] ← (Agnt) -
  [Control] → (Obj) → [Specify] → (Rslt) -
    [State : [Support]]].
```

– *A technical committee is required to handle any required implementation definition change (PT1,CN10).*

- (CN10)
```
[Req_Comp : [Techn_Comm] ← (Agnt) -
          [Control] → (Obj) → [Specify] → (Rslt) -
            [State : [Req_Impl]]].
```

(3) Defining the Paper Submission Process

The technical committee analyses all publishing activities to define what information tools can best be used to support them. One of these activities is the paper submission process (see T6). Submitting a paper basically consists of transferring a file from the author to the site of the journal, which results in the following workflow mapping (i.e., a mapping between workflows from the different domains in the reference framework, which is used to define required functionality):

– *The paper submission process (problem domain) is an interaction (human network) enabled by a send-file process (information system) (TC1,S7).*

- (S7)
```
[State : [Workflow_Mapping : #34] → (Part) -
        [Submit]
        [Interaction]
        [Send_File]].
```

Thus, S7 is an instance of the workflow mapping type definition which shows how an activity (submit) is ultimately to be implemented as a send-file process in the information system.

To enable this required functionality, the technical committee proposes that all users should have access to an FTP server, a tool that enables basic file transfer:

– *To enable the sending of files in the paper submission process, an FTP server*[6] *should be used (TC1,S8).*

- (S8)
```
[State : [Req_Impl : #42] -
        (Inst) → [FTP_Server]
        (Obj) → [Workflow_Mapping : #34]].
```

Because of CN4, the project team is permitted to change any (state) definition. Thus, each time some state change request is made, the project team can decide for itself whether or not to become actively involved in the specification process. Regarding definitions S7 and S8, the project team has no problems, and it approves of them. Because of CN8, the project coordinator is asked to approve of S8 as well, which he does.

However, after having given it some further thought, the project coordinator has a problem with this definition. In his view, an FTP server is not really suitable for submitting papers. Among other things, file transfer is relatively cumbersome and insecure using this tool. Instead, he proposes to use a BSCW server (see Horstmann and Bentley [35]). This tool has been optimised for file distribution processes, as it enables advanced, user-friendly and secure file transfer. Therefore, the project coordinator requests to modify state definition S8 into:

– *To enable the sending of files in the paper submission process, a BSCW server should be used (PC1,S8).*

- (S8)
```
[State: [Req_Impl : #42] -
        (Inst) → [BSCW_Server]
        (Obj) → [Workflow_Mapping : #34]].
```

[5] Each of these activities themselves also has type definitions, which are not listed here.

[6] We assume that the information tool type definitions, as well as the definitions that describe which IC processes they enable, have already been defined. These could, for example, be retrieved from a standard tool functionality library.

Because of CN8, the technical committee is involved in the handling of this requested state definition modification process. In turn, they have their own objections, however. The main problem they see is that the BSCW tool, despite its user-friendly interface, is too complex to learn because it offers so much functionality. No agreement is initially reached between the project coordinator and the technical committee, while they are executing the definition process. However, another actor, the project team, is also permitted to be involved in this specification process (CN4). So far, it has not actively participated, but now that a specification discourse has been started it plays a mediating role and the conflict is resolved. All actors now approve of the proposed state modification, making it legitimate. To alleviate the concerns of the technical committee, great care will be taken to facilitate the BSCW learning process. To this purpose, a set of FAQs will be developed, and personal instruction sessions will be held for authors who do not feel comfortable with it. These agreements are explicitly entered in the solution description attached to definition S8.

5.3. Discussion

By analysing the, still largely face-to-face, legitimate user-driven specification process in virtual professional communities such as in the EJCL case, insights into the key change processes and required knowledge definitions, notably composition norms, have been obtained. The users are not directly presented with the bare graph definitions presented in the example. Instead, these definitions are used by the RENISYS specification tool to select and notify the relevant users, and to construct the options on the web pages through which the users interact. Vice versa, data entered by the user via a web page can be represented in such an abstract format. A prototype has shown the proof of concept that the method indeed can be implemented. It is currently being tested in larger-scale experiments. However, since the objective of this paper is to explain the ideas of legitimate user-driven specification at the more abstract level, the properties of the tool are not discussed further here.

The choice for conceptual graphs as a uniform representation formalism was made because of its expressiveness, its pattern matching and advanced graph generalisation facilities. The knowledge definitions and the conversation processes in which they are changed have a strict formal semantics. Via these semantics, the resultant deontic effect of the composition norms applicable to a user in a particular specification process stage can be computed, using norm and state definitions and precedence rules to resolve conflicts between norms. Based on this information, authorisations for users to get involved in conversation acts are derived. Since the purpose of the current paper is to discuss the role of legitimate user-driven specification in workflow modelling, the algorithms are not included here, although they are described at length in de Moor [4].

A potential problem arises when more than one user is permitted to make a change. In the current conversation process support module, we assume that a change can be legitimately made if just one of the permitted users is involved. Alternative policies, e.g. that all permitted users must agree, could be represented by changing the conversation process rules.

In general, a breakdown can be resolved by more than one change (in our case study, the submit functionality could be enabled by both FTP and BSCW). The EJCL team decided to implement just one change (BSCW). Still, it would have been equally legitimate to implement both or to remove the submit function from the specification. RENISYS does not make any suggestions about which alternative solution is best. Instead, it assembles the relevant users affected by the change and supports them in formulating and deciding upon the knowledge definition changes that are most relevant – in their eyes – to the resolution of the particular breakdown.

The case study showed that the initiation of the knowledge base with the proper initial knowledge definitions is a crucial step. Through composition norms, users have the opportunity to express – and protect – their interests in a formal way. Great care must be taken to ensure the safety of the composition norms in the sense that their actual deontic effects equal their intended deontic effects. On the one hand, users should not be able to ‘overspecify’, so that they can change more than they actually should. On the other hand, enough flexibility for future changes must be guaranteed by the defined norms allowing for the necessary changes to indeed be possible. In this case, for example, the sponsor has to safeguard his control over the goal without restricting the project team in its ability to design the relevant composition norms. It took several iterations to find a safe starting set of composition norms that prevents actors of the project team from removing or disabling the goal (of establishing the electronic law journal), while at the same time allowing them to make changes in the activities (and their technological support) via which the goals were to be accomplished.

6. Conclusions

Workflow management systems are powerful instruments to enact and guide cooperation in virtual professional communities, provided that they are carefully designed and that the interests of the members of the virtual community are taken into account. It was not our intention to propose another workflow modeling language. Instead, we have worked out a perspective on the acceptability of workflow changes in order to make them legitimate. Without such a perspective, a design team can only answer questions such as 'Have all *specified* user requirements been fulfilled?' With a legitimate user-driven approach like ours, also the question 'Do the specifications match the *authentic* requirements of the community?' can be more affirmatively answered.

In the RENISYS specification method, we have developed an approach to engineering the requirements of this particular class of workflow systems (as opposed to traditional systems) by interpreting social norms, in order to make only legitimate changes to the system specifications. These are changes that are not only meaningful, but also acceptable to all members of the community. In the method, the members of the virtual community control the specification of their own composition norms, which are formal representations of their informal social norms. The specification process starts with a set of initial knowledge definitions, including composition norms. System development is regarded as a series of conversations for specifications, which do not violate the communal composition norms, between selected members of the virtual community. The formal representation of norms allows their deontic effect on the various users to be computed at any time, while the informality of the actual discussions is guaranteed.

We strongly believe that the norm-guided specification process will increase the feeling of ownership that the users have with their workflow system and will promote their active participation in the change process. Discussing composition norms has the additional advantage of increasing community cohesion by making users aware of the social norms that often invisibly guide their joint work.

Another advantage of the legitimate user-driven approach is that the focus is on identifying relevant human actors, instead of on developing detailed functional specifications. This allows a large amount of tacit knowledge to be brought to bear on any specification change. Besides reducing the time required on the specification process, this is also likely to ensure that specifications will become more varied and tailored. It is to be expected that two different communities will develop socio-technical solutions to their problems that are quite different, compared to standard approaches that do not so much take the social context of the network information system into account.

It may be argued that we introduce a complex machinery to handle relatively simple functionality changes. It should be understood that our focus is on optimising acceptability, not technical functionality. The quality of the technical solutions still depends fully on the technical expertise of the implementors, about which the method does not make any judgement. For ensuring these aspects of system quality, other, more traditional system development methods could be used, if desired. However, the complexity we introduce is essential to safeguard the interests of the community members and ensure their active involvement in the specification process. It might be compared to a democratic system, which, although it is a sometimes costly and complex way of decision making compared to more autocratic approaches of governance, has the enormous advantage of respecting the wishes of the community and engendering true, rather than forcefully imposed acceptance of sometimes painful decisions.

Traditional specification approaches, like ARIS, focus on optimising a workflow system for requirements that are hierarchically defined. The authority basis for changes is founded in that hierarchy. Such an approach is valid for systems that require central control of their development and maintenance, i.e., safety-critical systems for power plants. Virtual professional communities, however, are not governed by such a hierarchy, but instead should allow their interests to be balanced by their unique social norms. Although such norms are always present in any community, they are usually not made explicit. We claim that an active use of social norms that have been explicitly defined is essential in the design and evolution of workflow systems for virtual professional communities. Such communities cannot be formed – or maintained – by a workflow system being imposed upon them. Instead, the community must be able to create a normative framework that establishes the rules of acceptable changes to their workflow system, a framework that is often unique to the community at hand.

Existing requirements engineering approaches emphasise the 'correctness' of mapping user requirements to systems. Although users are regarded here as the ultimate measure for the correctness of the system – it has to fulfil what the user wants – the users themselves are rarely active players in those scenarios. In our approach, however, legitimacy of changes to the (workflow) system is key. By precisely answering the question of who is to define a change requirement, users can become more actively involved in a way that is more

acceptable to the community as a whole. Legitimacy is thus added as a new dimension to requirements engineering. We argue that this is not only enhancing the acceptability of changes. It also may make discussions about change more efficient, since the right subset of all users can be computed from the composition norms in the knowledge base. This is also particularly relevant for geographically distributed human networks. Note that we do not claim that our approach should replace existing specification approaches, for example from the field of requirements engineering. In our opinion, ours could be complementary to these methods. It would be interesting to see if a concrete coupling between RENISYS and one or more of these approaches can be achieved.

Much work is still needed in order to provide adequate support of legitimate user-driven specification processes in virtual professional communities. Typologies and reference models of virtual professional communities can help to reduce the burden of setting up new network information systems. The current specification tool prototype needs to be further developed, as at the moment it provides only rudimentary conversation support and can deal with only simple graphs. Much theoretical and empirical research also remains to be done. For instance, would it be better to start with few generic (strong) composition norms or with many specific (weaker) composition norms? Besides the electronic law journal case, we have started further case studies to gain the empirical evidence required to answer such questions.

References

1. de Moor A. 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, WA, 3–8 August 1997. LNAI 1257. Springer, Berlin, 1997, pp 536–550
2. de Moor A, Mineau G. Handling specification knowledge evolution using context lattices. In: Proceedings of the sixth international conference on conceptual structures, ICCS'98, Montpellier, France, 10–12 August 1998, pp 416–430
3. de Moor A, Weigand H. An ontological framework for user-driven system specification. In: Proceedings of the 32nd Hawaii international conference on system sciences (HICSS-32), Maui, Hawaii, 5–8 January 1999
4. de Moor A. Empowering the user: a method for the legitimate user-driven specification of network information systems. PhD thesis, Tilburg University, The Netherlands, 1999
5. de Moor A. Composition norm dynamics calculation with conceptual graphs. In: Proceedings of the eighth international conference on conceptual structures (ICCS2000), Darmstadt, Germany, August 2000. LNAI 1867. Springer, Berlin, 2000, pp 522–535
6. de Moor A. The initialization of conversations for specification: a context of social norms. In: Proceedings of the fifth international workshop on the language-action perspective on communication modelling (LAP2000), Aachen, Germany, 14–16 September 2000. Aachener Informatik Berichte 2000-06, 2000, pp 1–20
7. Robinson W, Volkov V. Supporting the negotiation life cycle. Commun ACM 1998;41(5):95–102
8. Ramesh B, Jarke M. Towards reference models for requirements traceability. CREWS Report 99-13, Technical University of Aachen, Germany, 1999
9. Ramesh B. Factors influencing requirements traceability practice. Commun ACM 1998;41(12):37–44
10. Ramesh B, Dhar V. Supporting systems development by capturing deliberations during requirements engineering. IEEE Trans Software Eng 1992;18(6):498–510
11. Scheer A-W. ARIS. In: Bernus P, Mertins K, Schmidt G (eds). Handbook on architectures of information systems. Springer, Berlin, 1998, pp 541–565
12. Pohl K. The three dimensions of requirements engineering: a framework and its applications. Information Syst 1994;19(3): 243–258
13. Schäl T. Workflow management systems for process organizations. Springer, Berlin, 1996
14. Sheth A, Georgakopoulos D, Joosten S, Rusinkiewicz M, Scacchi W, Wileden J, Wolf A. Report from the NSF workshop on workflow and process automation in information systems. SIGMOD Rec 1996;25(4):55–67
15. Abbott KR, Sarin SK. Experiences with workflow management: issues for the next generation. In: Furuta R, Neuwirth C (eds). Proceedings of the conference on computer supported cooperative work, Chapel Hill, NC, 22–26 October 1994. ACM Press, New York, 1994, pp 113–120
16. Khoshafian S, Buckiewicz M. Introduction to groupware, workflow, and workgroup computing. Wiley, New York, 1995
17. Klein M. Challenges and directions for coordination science. In: Proceedings of the second international conference on the design of cooperative systems (COOP'96), Juan-les-Pins, France, 2–14 June 1996, pp 705–722
18. Van der Aalst WMP, Van Hee KM, Houben GJ. Modelleren en analyseren van workflow: een aanpak op basis van petri-netten. [Modelling and analysing workflow: an approach based on Petri-nets.] Informatie 1995;37(11):744–753
19. Medina-Mora R, Winograd T, Flores R, Flores F. The action workflow approach to workflow management technology. Information Soc 1993;9(4):391–404
20. Van der Rijst N, Van Reijswoud V. Comparing speech act based modeling approaches for the purpose of information system development. In: Proceedings of the 3rd European conference on information systems, Athens, 1–3 June 1995, pp 353–365
21. McClatchey R, Vossen G. Workshop on workflow management in scientific and engineering applications. SIGGROUP Bull 1997;18(3):20–23
22. Sheth A. From contemporary workflow process automation to adaptive and dynamic work activity coordination and collaboration. SIGGROUP Bull 1997;18(3):17–20
23. Robinson M, Bannon L. Questioning representations. In: Proceedings of the second European conference on computer-supported cooperative work, Amsterdam, 25–27 September 1991, pp 219–233
24. Fitzpatrick G, Welsh J. Process support: inflexible imposition or chaotic composition? Interact Comput 1995;7(2):167–180
25. Dignum F, Dietz J, Verharen E, Weigand H (eds). Proceedings of the first international workshop on communication modeling: 'Communication modeling – the language/action perspective', Tilburg, The Netherlands, 1–2 July 1996. Springer eWiC series [http://www.springer.co.uk/eWiC/Workshops/CM96.html]
26. Van Reijswoud V. The structure of business communication: theory, model and application. PhD thesis, Delft University, 1996
27. Stamper R. Social norms in requirements analysis: an outline of MEASUR. In: Requirements engineering: technical and social aspects. Academic Press, New York, 1994, pp 107–139
28. Meyer JJCh, Wieringa RJ. Deontic logic: a concise overview. In: Meyer JJCh, Wieringa R (eds). Deontic logic in computer science: normative system specification. Wiley, Chichester, 1993, pp 3–15
29. Sowa JF. Conceptual structures: information processing in mind and machine. Addison-Wesley, Reading, MA, 1984

30. Lehmann F. CCAT: the current status of the conceptual catalogue (ontology) group, with proposals. In: Ellis G, Levinson R (eds). Proceedings of the third international workshop on PEIRCE: a conceptual graphs workbench, University of Maryland, 19 August 1994
31. Angelova G, Bontcheva K. DB-MAT: knowledge acquisition, processing and NL generation using conceptual graphs. In: Eklund PW, Ellis G, Mann G (eds). Conceptual structures: knowledge representation as interlingua. LNAI 1115. Springer, Berlin, 1996, pp 131–134
32. Gruber TR. Toward principles for the design of ontologies used for knowledge sharing. Technical report KSL 93-04, Knowledge Systems Laboratory, Stanford University, 1993
33. Luger GF, Stubblefield WA. Artificial intelligence and the design of expert systems. Benjamin Cummings, Redwood City, CA, 1989
34. Meyer JJCh, Wieringa R (eds). Deontic logic in computer science: normative system specification. Wiley, Chichester, 1993
35. Horstmann T, Bentley R. Distributed authoring on the web with the BSCW shared workspace system. Standard View 1997;5(1):9
36. Van der Rijst N, de Moor A. The development of reference models for the RENISYS specification method. In: Nunamaker JF Jr, Sprague RH Jr (eds). Proceedings of the 29th Hawaii international conference on system sciences, Maui, 3–6 January 1996. IEEE Computer Society Press, Los Alamitos, CA, 1996, pp 455–464
37. Kreiner K, Schultz M. Informal collaboration in R&D: the formation of networks across organizations. Organ Stud 1993;14(2):189–209