

# Formalizing the Evolution of Virtual Communities

Aldo de Moor

STARLab, Vrije Universiteit Brussel, Belgium

Hans Weigand

Infolab, Tilburg University, P.O.Box 90153, 5000 LE Tilburg, The Netherlands

e-mail: [ademoor@vub.ac.be](mailto:ademoor@vub.ac.be), [weigand@uvt.nl](mailto:weigand@uvt.nl)

corresponding author: Hans Weigand, tel. +31 13 4662806 fax. +31 13 4663069

December 3, 2005

## Abstract

Collaboration increasingly takes place in virtual communities using the Internet. These communities are socio-technical systems that tend to evolve strongly and become more complex over time. To ensure that changes to these complex socio-technical systems are meaningful and acceptable to the community as a whole, the relevant members of the community need to be involved in their specification. The RENISYS method conceptualizes community specification processes as conversations for specification by relevant members. It supports this process in two steps. First, it uses formal composition norms to select the relevant community members who need to be involved in a particular conversation for specification. It then uses a formal model of conversations for specification to determine the acceptable conversational moves that the selected community members can make, as well as the status of their responsibilities and accomplishments at each point in time. By combining composition norms with conversations for specification, the specification processes can be precisely tailored to the specification support needs of the community.

**Keywords:** virtual communities, conversations for specification, specification process model, theory of communicative action

## 1 Introduction

With the rise of the Internet, virtual communities are gaining importance as a new business model for online collaboration, as demonstrated by the proliferation of trading and education communities. In an increasingly networked society, with ever more need for global, and flexible ways of professional interactions, virtual communities are natural candidates to fill collaborative gaps in traditional, hierarchical

organizations. With the advent of more user-friendly and powerful web applications, business is also discovering the power of virtual communities, e.g. [28].

In this section, we will discuss the specific challenges for Community Information Systems and their specification process on the basis of general literature and a case study from an electronic journal community. This is followed by an introduction to the RENISYS specification method that aims to support evolving virtual communities in their conversations for specification. Conversations for specification heavily depend on the human factor and social dynamics, but some structural elements can be formalized, and the overall objective of this article is basically to show both how this can be done and for what purpose.

## 1.1 Virtual Communities

What is a virtual community? Communities are not just aggregates of people, temporarily interacting. A community has been defined as a group of people who share social interactions, social ties, and a common 'space' [28]; as a social network of relationships that provide sociability support, information, and a sense of belonging [57], and as a set of relationships where people interact socially for mutual benefit [44]. The key seems to be strong and lasting interactions that bind community members and that take place in some form of common space [58]. A virtual community differs from other communities only in that its common space is cyberspace. Virtual communities therefore describe the union between individuals or organizations who share common values and interests using electronic media to communicate within a shared semantic space on a regular basis [4].

Virtual communities are complex social systems enabled by a complex set of information technologies. A good way to conceptualize a virtual community is therefore to see it as a socio-technical system [34]. Well-designed socio-technical systems can provide communities with the energy necessary for healthy social development, as well as technical effectiveness [42]. Socio-technical design aims to optimize two systems jointly: (a) the technical system, in which the objective is to maximize task accomplishment and (b) the social system in which the objective is to maximize the quality of the working life of system users [53]. The social system contains many complex social constructs, such as goals, workflows, organizational structures, and social norms. This social system is supported by a technical system, increasingly consisting of sets of standard information tools, such as mailers, databases, and many different kinds of web applications [38]. The development of community information systems requires a careful process to ensure its *sociability*. This means that for successful community IS development it is essential to plan and develop social policies supporting the community's purpose and which are understandable and acceptable to members [34].

## 1.2 The Specification of Community Information Systems

The information systems development process consists of an analysis process, a design process, and an implementation process. In the analysis process, a perceived real-world system is transformed into a conceptual model of the information system.

During the design process, this conceptual model is translated into a model of the information system, which in the implementation process is turned into a (partially) machine-executable implementation. Involvement of community members, who are domain experts, not technical experts, is mostly in the analysis stage, in particular in the requirements specification process in which relevant real world phenomena are mapped onto concepts of the specification language [6].

Communities are characterized by emergent behaviour. The evolution of their socio-technical system does follow an overall life cycle, from birth to maturity and even death. Each community has its own, unique evolving mix of people, processes, and technologies, but this mix can only be planned, if at all, at a very high level and requires much customization in the various stages of the lifecycle [20, 58].

Almost identical Community Information Systems in practice are configured very differently across communities, and communal requirements and their enabling information technologies typically co-evolve strongly, leading to many breakdowns [27, 61]. Active user participation in the specification process of such continuously evolving community information systems is very important, since community members have the most detailed knowledge about when breakdowns in work arise and how they can be resolved. Also, such involvement leads to a stronger sense of ownership, which is an important indicator of community success [34]. Furthermore, it has been shown that members becoming actively involved in community moderation and standard, i.e. norm setting, is a necessary condition for the virtual social networks to become self-sustaining [1].

Coordination of user-driven community IS specification efforts is not trivial, however. There is a lack of strong central control by management, an IS department or trained and committed software engineers, as is the case in more standard business workflow modelling situations. This leads to many problems like sketchy requirement definitions and lack of documentation [25]. Specification support to prevent or reduce the impact of such problems is thus necessary. In general, distributed systems development requires process-centered software development environments that support strategies for dividing the work, assigning work to different developers and to indirectly coordinate their actions [5]. However, accomplishing such governance in a user-driven community environment is not trivial.

Governance in virtual communities is a complex process. First of all, it is situated in the sense that each online community has to work out its own system of governance [36, 45]. Communities are unique social constructs that require a subtle process of organizing themselves in order to be sustainable, which may differ from community to community. Second, research findings indicate that virtual communities, because of their voluntary nature, are more democratic and less authority-driven than other organizational forms [7, 34]. Rather than being regulated by imposed rules, they develop their own set of shared group norms. Thus, self-governance is key to communities, meaning that their change processes are governed by their own, communal norms instead of by legalistic rules [45]. However, such norms can easily be jeopardized by the ephemeral nature and rapidly expanding membership of many Internet-based communities, including online journals. Therefore, making these norms explicit is important, even more so than in physical communities. Self-governance in communities should work best when they are mature and have

developed sophisticated norms. On the other hand, permitting self-governance at an early stage may give communities the freedom and autonomy needed to establish these advanced norms [45].

Summarizing, communal decision making, also on specifications, is thus not a simple process of top-down control, but much more a subtle negotiation process between many stakeholders in the community, governed by their own, evolving norms. Since interests of stakeholders need to be safeguarded, it is very important that not only *user-driven*, emergent specification behavior is supported, but also that the specifications made are *legitimate*, in the sense of being both meaningful and acceptable to community members. In this article, we present the RENISYS method, a formal method supporting the legitimate user-driven specification of community information systems. First, however, we introduce a case study of an electronic journal which has been used in the development and validation of the method.

### 1.3 Case: The Development of the Electronic Journal of Comparative Law

Virtual communities which focus on scholarly communication in the form of e-journal publication are prime examples of socio-technical systems. Their Internet technologies continuously have to be carefully calibrated with the social infrastructure of scholarship, taking into account how the nature of the publication process is changed, both in terms of the social structure and the underlying dynamics of knowledge itself [26, 19]. E-journal communities can be classified as communities of practice, which are institutionalized, informal networks of professionals managing domains of knowledge [20]. Such networks are characterized by evolving governance structures and the need for legitimate authority, such as editorial roles. Since there are professional interests at stake, changes to the socio-technical system need to be very carefully made. In this article, we focus on one case of an e-journal. This case has been monitored for the past seven years, and has lead to detailed observations of its dynamics [14, 15]. We will use some of these observations to illustrate our method.

IWI, a Dutch organization stimulating new ways of distributing scientific information, funded a project to create an *Electronic Journal of Comparative Law* (EJCL<sup>1</sup>). 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 in a completely electronic way, using the web. The project started in spring 1997 and ended in summer 1998. After that, the journal has continued to evolve into a successful publication.

Around the journal, a complex virtual community emerged, in which stakeholders like editors, reviewers, authors, and project team members have worked together and changed their socio-technical system over the years. The initially basic set of requirements, defined after long deliberations by the users themselves, gradually evolved in scope and complexity. In this article, we use real and hypothetical examples based on specification situations observed in the case study.

The development of EJCL from 1997 onward consisted of four distinct stages:

---

<sup>1</sup><http://www.ejcl.org>

1. *Set-up*: Construction of initial system by the project team with representatives of stakeholders (law librarians, comparative legal scholars and IT specialists).
2. *Launch*: Promotion of the web site via conferences (printed brochures, oral presentations), mailing lists, and personal contacts.
3. *Internal Growth*: Increasing readership, articles, and issues.
4. *External Growth*: (a) Building up own network of contacts directly interested in the journal, (b) establishing connections with related initiatives through links and cooperation.

In the Set-up stage, the objective was to create a comparative law e-journal with an international editorial board. A project team, consisting of staff from Tilburg University and Utrecht University libraries, computer centres and law faculties was put together. The legal scholars within the project team acted as a preliminary editorial board. During this stage, two actors coordinated the definition of the socio-technical system: the project manager and the future editor-in-chief. Socio-technical requirements were elaborately analysed by the project team at the very beginning of the ECJL. The initial workflows and the website of the EJCL were based on key functionalities of other electronic journals. There was a considerable variation in the degree and kind of involvement in the specification process by the various actors, such as project team leader, scholars, librarians, technical experts, and consultants. For example, whereas in the beginning of the Set-up stage, everybody was involved in all decisions, later on, for efficiency reasons, key technologies were mostly proposed and evaluated by technical experts and the project leader only. These changes in involvement led to new norms governing the change process. In future situations, these norms were used to select the participants to be involved in dealing with the particular change required.

In the current, External Growth stage of the journal, change management practices are different. The assistant editor, the advisor to the board, the editor-in-chief, and one Dutch editor therefore review the EJCL once or twice a year in an informal, face-to-face meeting. All change decisions in relation to the EJCL must be worked out and proposed unanimously by the review team to the editorial board. The (international) editorial board always has the right to change or even overrule those proposals. Everybody else, for example authors, are allowed to make suggestions for changes. All suggestions are taken into account during review sessions held by the review team. Without complaints, the changes proposed by the review team will be carried out, otherwise a person from the review team will discuss the problem with its owner to find a solution.

Across the stages, it was observed that most changes in e-journal development concern generic issues, but require specific solutions. For example, all e-journals need to address the issue of citations. In law communities, citations often occur in the form of footnotes at the bottom of each page. Law journals typically do not use a double-blind review process as is more common in IS journals. Such issues were not all laid out in advance, but were raised as problems to be solved in the initial development stage of the journal.

One aim of the journal was to let all interactions, for example between the editorial board members, be completely electronically mediated. Most of the operational tasks indeed are done over the Internet. However, especially when dealing with changes to the socio-technical system, face-to-face conversations are still needed. From interviews with key members of the community, such as the editor-in-chief and project leader, it turns out that electronically-supported change management is much desired, for example to involve international editorial board members more actively in the evolution of the community. E-mail in their opinion is not sufficient, since it is incapable of dealing with the complex dependencies between specifications, especially over time, and does not enforce the norms governing change processes. If this is already the case with a small and relatively well-organized community like EJCL, the need for systematic specification support will be much larger in large, emerging virtual communities such as for example in open source development, e-business, or research communities. RENISYS is one approach to provide such support.

## 1.4 The RENISYS Specification Method

The RENISYS method conceptualizes community specification processes as conversations for specification by relevant members. It support this process in two steps. First, it uses formal composition norms to select the relevant community members who need to be involved in a particular conversation for specification. It then uses a formal model of conversations for specification to determine the acceptable conversational moves that the selected community members can make, as well as the status of their responsibilities and accomplishments at each point in time. By combining composition norms with conversations for specification, the specification processes can be precisely tailored to the specification support needs of the community.

The rationale, characteristics of, and methodological support for the legitimate user-driven specification process are described in detail in [10, 11, 13]. In the current article, we focus on the formal foundation that enables the selection of relevant members and the support of the conversations for specification. Many methods for community-centered IS development are informal, providing guidelines and heuristics for the various stages of the development process. However, formal methods have a number of advantages: they reduce ambiguity, can serve as a contract and resolve conflicts over the interpretation, allow for reasoning about properties, and are a prerequisite for the application of all kinds of analysis techniques [47]. Still, to engineer truly useful solutions, the right balance between informal and formal approaches needs to be found [35]. In our approach, we therefore use formal representations and reasoning in the selection of community members, and coordination of the specification process, but not for building complex models of the universe of discourse itself. We leave it up to the community members themselves to interpret the domain the semantics of the specifications made in their conversations, since they have the tacit knowledge and intelligence to do so. In this way, the number and complexity of specifications can be kept to the essential minimum, and can formal overkill be prevented.

The RENISYS (**RE**search **NE**twork **IN**formation **SY**stem **SP**ecification) method

is based on these insights and provides concrete support in ensuring that only legitimate changes can be made to specification knowledge. In [14], we showed how to support the process in which specification knowledge definitions are changed. We distinguish four categories of specification knowledge in RENISYS: *type definitions* form an ontology of concepts, while *state definitions* capture states-of-affairs. Two categories of normative knowledge are distinguished: *action norms* describe the acceptable operational (workflow) behaviour, whereas composition norms represent which (specification) change processes are acceptable to the various community members. In other words, action norms regulate the work to be done in a community, while composition norms prescribe how to adapt the socio-technical system in which the work gets done. Composition norms therefore play a central role in coordinating the work of the distributed community members. As these norms are defined by the community itself, they ensure the legitimacy of changes to the community information system.

Specification changes are made in so-called *conversations for specifications*. In these conversations, selected community members take conversational turns in initiating, executing, and evaluating the processes in which knowledge definitions are changed. In [11, 13], we showed how composition norms can be used to initialize these conversations, so that only those members participate who can legitimately do so. In the current article, we focus on formalizing the semantics of the role that composition norms play in ensuring the legitimacy of specification conversations. Such formalization results in the logic needed to design computer support for (1) selecting the relevant conversation participants, and (2) the facilitation of those conversations, including the systematic handling of authorizations and commitments made during those conversations.

The purpose of the article is first to show how the norms can be used to select conversation participants by describing the composition norm status dynamics. We then give a formalization of the conversations for specification. We use an extended version of standard dynamic deontic logic to ensure composition norm conflict resolution, as well as to determine the exact conversational moves participants may or must make. Throughout the article, we will use examples taken from the case of an evolving electronic journal community.

In Sect. 2, we introduce conversations for specification. Sect. 3 shows how composition norms can be used to calculate the legitimate conversation participants. Sect. 4 presents a formal model of the conversation for specification, which can be used to develop more sophisticated conversation support. Sect. 5 discusses related research and we end the article in Sect. 6 with conclusions.

## 2 Conversations for Specification

The use of individual speech acts is insufficient to coordinate meaningful work-related communication. To do so, larger units of communicative interaction are needed, which are called *conversations*. In Sect. 2.1, we discuss how the system specification process can be seen as a form of conversation. Sect. 2.2 grounds such conversations in theories from the Language/Action Perspective. In Sect. 2.3, we introduce our Specification Process Model. The way that the conversation for spec-

ification is embedded in a normative context is described in Sect. 2.4.

## 2.1 Conversations

Every legitimate change to a virtual community requires discussion, whether it concerns social aspects such as its policies, or technical aspects such as which tool to use [34, 58]. Such discussion, however, is not open-ended, but is a goal-oriented conversation.

We adopt a somewhat restricted view on conversations, seeing them as series of interrelated communicative acts aimed at defining and reaching a goal [17]. We therefore define a *conversation* as a self-contained unit of communication to accomplish certain specification objectives, like the specification of a new type of article submission workflow. Evidence for the effectiveness of predefined conversation models is ambiguous [2]. We therefore require a conversation to be only partially structured in the sense that main specification process entities are predetermined, although the format of the utterance acts in which these entities are defined is relatively free. There are many types of work-related conversations, one of which is the conversation for action, in which the goal is to coordinate explicit cooperative action [60]. This kind of conversation is the basis for the well-known Coordinator and ActionWorkflow modelling methods [30], in which conversations are used, for instance, to coordinate order and delivery processes by customers and performers. Fig. 1 represents a conversation for action as a state transition network [61, 60]. Many different types of conversations, but especially the conversation for action, can play a role in the specification process.

Figure 1: A State Transition Diagram of a Conversation for Action [60]

The system specification process is often triggered by breakdowns in work [61, 27]. For example, a breakdown experienced by author John could be that he finds the editorial process of his submitted article taking too long.

As a consequence of the occurrence or anticipation of breakdowns, new semantic distinctions are always emerging. The generation and interpretation of these distinctions should be treated as an activity based on conversations that can be designed and facilitated through the computer [60]. To provide support for breakdown-initiated conversations, a conversation framework is needed that combines specialized as well as more general conversation patterns [24]. We call such a conversation aimed at making specification changes a *conversation for specification*. To position this type of conversation in a theoretical embedding, we now turn to theories from the Language/Action Perspective.

## 2.2 Language/Action Perspective Theories

Communication is not just about exchanging information. An essential aspect in conversations for specification are the mutual commitments that community members make. Communication therefore is also a crucial instrument to execute and

coordinate organisational work [55]. The theoretical foundation of this point of view is provided by the work done in the Language/Action Perspective (LAP). LAP is grounded in Austin's and Searle's speech act theory, and Habermas's theory of communicative action, as well as the work done by Winograd and Flores on LAP-based IS-design. [3, 41, 59, 61]. LAP takes the fundamental position that language is not only used for exchanging information as in reports, statements etc. but also to perform actions, e.g. promises, orders, declarations. The conventional perspective on information systems stresses the contents of messages rather than the way they are exchanged and the effects they have. In contrast, the Language-Action Perspective emphasises what people do by communicating, how language is used to create a common basis for communication partners, and how their activities are coordinated through language.

Speech act theory as defined by Austin and Searle claims that language not only describes the world, but is also used to create and coordinate actions in the world. For example, by making an assertion about something one claims that this something is true in the world. Often, such speech acts come in sequences: a request is followed by promise in which the hearer makes a commitment to the speaker.

An important limitation of Searle's original speech act theory is that it stresses the conversational role of the *speaker*, while ignoring the role the *hearer* plays in the success of speech acts. This makes it hard to know whether a hearer does something because she, fully informed and unpressured, accepts the speech act made by the hearer, or because she is, for example, insufficiently knowledgeable or forced in some way to accommodate the speaker. In virtual communities this would be a most undesirable situation, as legitimacy of the changes produced in conversations for specification is of the greatest importance. These communities require information system development methods that do take the hearer into account as well.

Habermas's theory of communicative action provides a comprehensive conceptual framework in which to ground such methods that focus on the role of the hearer as well. One concrete contribution of his work are the rules of discourse that give guidelines for how a conversational process should be structured so that discursive equality, freedom, and fair play are guaranteed. For Habermas, the essence of communicative action is that the parties are oriented at mutual understanding of a situation for the purpose of coordinating actions. The situation definition includes what is the case, but also what should be done and what is desirable. Communication proceeds by parties making claims in series of related speech acts. For example, an assertion is not just to convey some information, but also claims that the situation is such and such. As the orientation is towards mutual understanding, the other party can accept the claim, but can also challenge it, which means that a rational discussion is started in which arguments on validity claims pro and con can be exchanged.

Summarizing, what we need to support conversations for specification is a natural language-like discussion process based on a communicative action-like conversation coordination mechanism. This provides us with a universal approach for facilitating legitimate user-driven specification, which is independent of the specific context in which it is taking place. The coordination mechanism must make clear in any conversational state which (finite) set of *conversational actions* or *moves* are possible

Figure 2: The Transaction Process Model [50]

[61, 39]. In addition to the state diagram modelling techniques such as used in the conversation for action approach (see Fig. 1), theory-grounded *conversation protocols* are therefore needed that prescribe the allowed conversational moves for the participant whose turn it is to speak. The Specification Process Model embedded in the RENISYS method provides such a protocol [10, 13].

### 2.3 The Specification Process Model

The Specification Process Model (SPM) operationalizes the conversation for specification. It is derived from Van Reijswoud’s Transaction Process Model (TPM) [50]. The TPM is a communication model that presents the possible conversational moves in a business communication process. The model is represented as a state transition diagram, similar to Fig. 1, in which the states represent *transaction states* and the transitions are caused by *transaction acts* (Fig. 2). These acts are subdivided into two categories: communication acts and objective acts. A *communication act* (represented by CAx in the figure) is an utterance by a participant that causes a transaction process transition. An *objective act*, the purpose of the transaction, is the act that changes the objective world. Objective acts do not need to be further modelled, as the actual activities that change the objective world are not part of the communication process. Of course, they are embedded in this process, but the objective acts themselves are aimed at the *production*, not at the *planning* or *validation* of the results.

Besides being able to model successful communication processes, the TPM also allows for the representation of discussion and discourse, as proposed in the theory of communicative action. The model therefore consists of three layers. In the *success-layer*, a regular transaction process is described. The *discussion and failure-layer* allows for the discussion of validity claims. The *discourse-layer* contains discourse with the purpose of restoring background conditions, such as the questioning of assumptions.

The TPM in Fig. 2 is to be regarded as the generic model, showing all possible paths. When used to model actual communication situations, different paths through the model may be traversed. If there are no problems, a conversation for specification will only need the success-layer. However, the other two layers are useful to get the conversation back on track in case of communicative problems. The TPM can not only be used to model actual situations, as they happen in practice, but also normatively, how communication *ought to* go, as is the case in, for example, best practices. What exactly these paths are is still the subject of ongoing research.

Although the TPM forms the basis for the modelling of the conversation protocols needed in RENISYS, there are certain differences in terminology and application. We therefore use the term *Specification Process Model* (SPM) for the conversation model used in RENISYS. The main differences with the TPM are that the transaction is renamed into *specification process* and that the *evaluator* role is added. This role is to approve of the proposed specification change. Furthermore, the purpose

of the specification process is no longer an objective action, but a *definition process*. Communication acts and transaction states are renamed into the more precise terms *conversation acts* and *conversation states*. A complete overview of all conversation acts making up the SPM is given in [10]. For each of these acts, it is discussed there which conversational roles can perform as speakers and hearers of the act.

In the SPM, we formalize conversations as little as possible, in order to provide flexibility and not to cognitively overburden users. Thus, although a user can start a discussion to, say, question the sincerity of another user’s conversation act, the initiator does not need to formally indicate why he does so. The reason for this is that RENISYS *enforces* the legitimacy of specification processes by only inviting those participants to take part in some conversation for specification who are justified to do so. Once they have been selected, they are free to discuss in the way they like.

To illustrate the use of the SPM, we represent a basic *successful specification process* aimed at the modification of an editorial workflow definition. It is an instantiation of the generic model presented in Fig. 2. In this simple example, there is no need to go into the discussion or discourse layers. This representation is similar to those given in [50, p.95], showing four of the in total 23 conversation acts.

In the example, the sequence of conversation acts and definition processes is the following:

<i>Act</i>	<i>Description</i>	<i>Resulting State</i>
<b>CA<sub>1</sub>:</b>	I: C <sub>legit</sub> [propose(directive)<mod_type_def(edit),now>]	Directed
<b>CA<sub>2</sub>:</b>	X: C <sub>legit</sub> [promise(commisive)<mod_type_def(edit),now>]	Committed
<b>DP:</b>	X: DP <sub>legit</sub> [define(execute)<mod_type_def(edit),now>]	Executed
<b>CA<sub>3</sub>:</b>	X: C <sub>legit</sub> [rep_compl(decl)<mod_type_def(edit),now>]	Decl.(Completion)
<b>CA<sub>4</sub>:</b>	E: C <sub>legit</sub> [decl_success(decl.)<mod_type_def(edit),now>]	Decl.(Success)

Table 1: The conversation acts and definition process in a successful type creation process.

Here, the initiator in a directive asks the executor(s) to modify the existing editorial workflow (type) definition. The executor (which can consist of more than one person, such as a task force or working group) promises to do this. In the execution phase, this person or group then performs the actual definition process, for example by defining subtasks to streamline the editorial process. An outcome of the definition process could be to add a subtask *assess-abstract* to the editorial process, to ensure that submitted articles are within the scope of the journal. Once completed, the executor presents the modified editorial process definition to the evaluator(s). If they approve the proposed change, the specification process has been successfully completed.

## 2.4 The Context of Conversations for Specification

One major criticism of the application of speech act theory in systems development is that it is not able to represent what people really do, as it would provide models

Figure 3: The Normative Context of Conversations for Specification

that are too rigid and simplistic to capture the complexities of actual work practices [2, 51].

Thus, in real social practice, the complex world beyond the representations must somehow be considered. In other words, it is not just important to produce definitions, but also to understand the *situatedness* of the conversations in which the definitions are produced, the way in which the definitions are represented and how they are understood by the people who use them [60, 9, 46]. Thus, a fundamental problem has not been addressed by the TPM (and by the SPM, so far): how to make the link between the specific 'social/organizational and work situations' and the conversations for specification? In other words, the following question needs to be addressed:

*Who are to be the initiators, executors, and evaluators of these conversations and what should be on their agendas?*

To this purpose, it is important that the *context* of the specification conversation is taken into account [9]. The RENISYS conversation context model captures this context. The conversation context consists of two parts, the external and the internal conversation context [13]. The *internal conversation context* consists of the knowledge definitions which are related to the knowledge definition being changed. The *external conversation context* consists of the knowledge definitions needed to select the users who can legitimately be involved in a particular conversation for specification.

The internal conversation context gives meaning to the definition being changed, as it situates the definition in a web of semantically related definitions, that are already meaningful to and accepted by the community. For example, the type definition of the editorial workflow may include links to the submission and review workflows, and to submitted and edited papers as input and output objects, respectively. Each of these concepts in turn have their own definition. Much research has been done on this part of the context, for example in the areas of ontologies, data mining, and knowledge management.

The focus in this article is on the external conversation context, however, as it is used to determine the relevant users to involve in definition change processes. The external conversation context is thus a key element in the facilitation of community evolution. In our approach, it consists of the set of composition norms.

Fig. 3 outlines our formalization approach for the evolution of virtual communities. Knowledge definitions are changed in conversations for specification. Composition norms determine who should be the community members to involve in particular conversations. In Sect. 3 we show how the composition norms can be used to calculate who can legitimately play what conversational roles. In Sect. 4, we then model the conversations for specification themselves. This formalization is useful to develop system support for legitimate conversations, as for example authorizations for conversational moves can now be precisely determined.

### 3 Selecting the Conversation Participants

In this section, we examine the precise meaning of composition norms, and the role that they play in selecting conversation participants. As described earlier, RENISYS distinguishes four categories of knowledge definitions: type definitions, state definitions, action norms and composition norms. In [14], we gave formal definitions of all categories using conceptual graph theory. In this article, we will focus on the composition norms, of which we give a formal definition in Sect. 3.1. In Sect. 3.2, we describe how to calculate which composition norms are applicable to a particular specification change request. Sect. 3.3 shows how to deal with conflicting norms. In Sect. 3.4, we demonstrate how composition norm hierarchies can be used in optimizing calculations.

#### 3.1 A Formal Definition of Composition Norms

Composition norm definitions define acceptable specification behaviour. First, some sets of entities needed for the formal specification of these norms are defined. At the heart of RENISYS is an ontological framework, which specifies entity types, such as core domain and specification process concepts. The types defined by the ontology are partially ordered in a type hierarchy. A comprehensive description of the ontological framework is given in [10] and not repeated here. An example of a type hierarchy is shown in the example given in Sect. 3.2.4.

To specify composition norms, we need to consider some subsets of the set of entities. Entities are placed in these sets if their type conforms with the type of the set. The set of users includes all individual members of the community. The set of actors designates the roles the users play, e.g. editor or reviewer. Control processes are either initiations, executions, or evaluations of, in this case, specification processes. Specification processes consist of the creations, modifications, or terminations of knowledge definitions. A composition is a control process coupled to a specification process. A composition could be the initiation of the modification process of the editorial workflow type definition, for example.

**Definition 1**  $\mathcal{E}$  is the set of entities. Subsets relevant to the specification of composition norms are: the set of users  $\mathcal{U}$ , the set of actors  $\mathcal{A}$ , the set of control processes  $\mathcal{CP}$ , the set of workflows  $\mathcal{W}$ , the set of specification processes  $\mathcal{SP}$ .

The set of compositions  $Comp$  consists of all possible combinations of control processes and specification processes, represented by the powerset of their Cartesian product:  $Comp = \mathcal{P}(\mathcal{CP} \times \mathcal{SP})$ .

□

An example of a composition would be the initiation of the creation of a state definition (e.g. who is the editor of a particular journal).

One additional construct needed to define composition norms is the so-called *deontic effect*. This designates which effect a norm has on a specification process. It can be either *permitted*, *required*, or *forbidden*. The related norms are called privileges, responsibilities, and prohibitions, respectively.

**Definition 2** The set of deontic effects  $\mathcal{DE} = \{Perm, Req, Forb\}$ .  $\mathcal{ID}$  is a set of unique identifiers.  $\mathcal{D}_{CN}$  is the set of composition norms.

A *composition norm*  $d_{cn} \in \mathcal{D}_{CN}$  is defined as:

$$d_{cn} = \langle id, de, a, cp, sp \rangle$$

with  $id \in \mathcal{ID}$ ,  $de \in \mathcal{DE}$ ,  $a \in \mathcal{A}$ ,  $cp \in \mathcal{CP}$ ,  $sp \in \mathcal{SP}$ .

For each norm  $d_{cn}$ ,  $\langle cp, sp \rangle$  is called its *norm composition part*.

**de** is a function from  $\mathcal{D}_{CN}$  to  $\mathcal{DE}$ , which assigns a deontic effect  $de$  to each composition norm  $d_n$ . For all  $d_{cn} \in \mathcal{D}_{CN}$ :

$$\mathbf{de}(d_{cn}) = \begin{cases} Perm & \text{if } d_{cn} \text{ is a privilege (somebody } \textit{may} \text{ do something).} \\ Req & \text{if } d_{cn} \text{ is a responsibility (somebody } \textit{must} \text{ do something).} \\ Forb & \text{if } d_{cn} \text{ is a prohibition (somebody } \textit{may not} \text{ do something).} \end{cases}$$

□

The specification process that is part of the composition norm definition of the example has a complex structure, which is cumbersome to denote in the current syntax. In [14], we developed a formal representation of the knowledge definitions that are the object of specification processes, using conceptual graphs. For the purposes of this article, it suffices to treat most of this internal structure of the specification process as a black box. We therefore use a special notation to indicate that specification processes are not completely worked out by underlining its main elements: the process itself and the knowledge definition to which it applies. The predicate ('Create\_Type') represents the type of the specification process, the term ('Edit') is the definition that is the result of the specification process.

Next, we give some examples of composition norms observed in the EJCL case [10].

### **Example**

This norm from the Set-up stage says that a responsibility of the project coordinator was to evaluate all changes of support-definitions (those definitions that concern which information tools support what workflows):

$$\langle \#7, Req, Project\_Coordinator, Eval, \underline{Specify\_Type(Support)} \rangle$$

The following composition norm from the External Growth stage represents that it is a responsibility of the review team to control (i.e. initiate, execute, and evaluate) all change processes of any type of knowledge definition. It is thus much broader in scope than the previous norm, and will apply to more change situations:

$$\langle \#11, Req, Review\_Team, Control, \underline{Specify\_Type(Definition)} \rangle$$

To guarantee checks and balances, the editorial board in addition always has the right (i.e. privilege) to evaluate whatever the review team decides:

□

Composition norms only become useful when applied in the handling of change situations. Next, we show how to calculate their applicability to a particular situation.

## 3.2 Calculating Applicable Composition Norms

Composition norms play different roles in different specification processes. A composition norm may have an effect on one particular change request, while not being applicable to others.

An *active specification process* is a specification process in which the community is currently involved.

To calculate what role a composition norm plays in the active specification process that deals with a specific change request, one should know its *status* with respect to the process. This status differs depending on the roles that the users play in the specification process and the degree to which the norm matches with the process itself.

Sect. 3.2.1 defines the axioms we use to define the meaning of and operations on composition norms. Sect. 3.2.2 introduces a typical scenario in journal management. We use the scenario to illustrate the concepts introduced in the remainder of the article. In Sect. 3.2.3, we describe the two matching processes needed to calculate the status of any composition norm: the user match and the composition match. Sect. 3.2.4 shows the actual calculation of the norm status using these matching processes.

### 3.2.1 Norm Axioms

In RENISYS, the norm categories distinguished are permissions, responsibilities, and prohibitions, as discussed earlier. In order to describe the formal semantics of these categories and their relations, we turn to deontic logic.

Deontic logic is the logic used to reason about norms. Several branches of this logic have been developed in the past decades (for an overview see [31]). One of these branches, dynamic deontic logic (DDL), is of particular interest to us, as it allows for the direct integration of deontic and dynamic constraints, which is important when describing norm-constrained (specification) processes. It is simpler than the standard temporal logics, is efficient in describing the normative context of actions and is a good basis for the formalization of communication.

DDL introduces actions  $\alpha$ , and a violation atom  $V$  which indicates a violation of some deontic constraint (i.e. norm). Complex actions can be created out of atomic actions using a series of operators, which we do not discuss here, as we are only interested in simple actions (and compositions). Furthermore,  $[\alpha]\phi$  means that the performance of  $\alpha$  necessarily leads to a state in which  $\phi$  holds.

In deontic logic, three *deontic modalities* are distinguished: an action can be permitted ( $P$ ), obligatory ( $O$ ), or forbidden ( $F$ ). These modalities are similar to

the RENISYS deontic effects: permitted (*Perm*), required (*Req*), and forbidden (*Forb*), respectively.

In DDL, the semantics of the deontic modalities are reduced to the semantics of Dynamic Logic as follows:

- (D1)  $F\alpha \equiv [\alpha]V$
- (D2)  $P\alpha \equiv \neg F\alpha$
- (D3)  $O\alpha \equiv F(\neg\alpha)$

This means that  $\alpha$  is forbidden if and only if performing  $\alpha$  leads to a *violation*;  $\alpha$  is permitted iff  $\alpha$  is not forbidden;  $\alpha$  is obligatory iff not performing  $\alpha$  is forbidden. More recent developments in the logic of obligations can be found, for example, in [49] and [18], however, for our purpose the current basic logic suffices.

### 3.2.2 Scenario: Revising the Editorial Process

A transparent and high-quality editorial process is a key business process for scientific journals in general, and for e-journals that want to establish a professional reputation in particular. The scenario presented here is typical for e-journal communities. It is loosely based on experiences in EJCL, where the editorial process has been considerably revised over time, as more experience was obtained about the specific needs of the computer-mediated comparative law community, and should at least partially be recognizable in other e-journal cases.

The journal has been running for a while, so far focusing on publishing standard journal articles. However, now that the journal has reached a relatively mature state, it wants to innovate and offer related types of publications, such as technical reports. One important and sensitive task is to define a sound editorial process for the reports. It should be based on the standard editorial procedure, but differs in some respects. Assume the community information system has two users, say John, a journal editor, and Jack, who is a reviewer. How, if at all, should they be involved in handling this change request?

The *active specification process* in this case is the creation of the editorial workflow of a report:  $Create\_Type(Edit\_Report)$ . Assume that the current set of legitimate composition norms  $\mathcal{D}_{CN}$  consists of these definitions:

- $\langle \#58, Perm, Editorial\_Board, Init, \underline{Terminate\_State(Reviewer)} \rangle$
- $\langle \#59, Req, Editor, Exec, \underline{Modify\_Type(Review)} \rangle$
- $\langle \#60, Perm, Actor, Control, \underline{Specify(T)} \rangle$
- $\langle \#61, Req, Editor, Control, \underline{Create\_Type(Edit)} \rangle$
- $\langle \#62, Forb, Journal\_Editor, Eval, \underline{Create\_Type(Edit)} \rangle$
- $\langle \#63, Req, Reviewer, Init, \underline{Create\_Type(Edit)} \rangle$
- $\langle \#64, Perm, Reviewer, Control, \underline{Create\_Type(Edit\_Report)} \rangle$

- $\langle \#65, \text{Forb}, \text{Reviewer}, \text{Eval}, \underline{\text{Create\_Type}}(\text{Edit\_Report}) \rangle$

Composition norm #58 indicates that the editorial board may start the removal process of a particular reviewer of a journal. Norm #59 expresses that an editor must modify the review workflow definition, if prompted. Norm #60 is a very generic norm, saying that any actor may control any specification process ( $T$  stands for all types of definitions). Such a generic norm typically is defined at the conception of a community, when its information system is still small in scope and only few users and actor roles have been defined. Norm #61 says that an editor must control the creation of new types of editorial workflows. However, according to norm #62 a journal editor is not allowed to evaluate such newly created workflow types. This norm could be introduced to ensure that such an editor cannot manipulate the results of her own work processes. Norm #63 says that a reviewer is responsible for starting the creation of a new editorial workflow, for example when he or she is no longer satisfied with the way reviews are being handled. Norm #64 permits a reviewer to control (i.e. initiate, execute, and evaluate) the creation of report edit workflow types. Finally, norm #65 says that a reviewer is not allowed to evaluate a newly created report edit workflow definition. Such a privilege could instead be granted, for instance, only to the editorial board.

The types of the various elements of these norms are ordered using the following type hierarchy (which consists of the relevant parts of the ontological framework on which RENISYS is based, combined with some additional, domain-specific types, such as ‘Journal\_Editor’). Note that PD\_Actor stands for problem domain actor:

```

PD_Actor >
  Editor >
    Journal_Editor
    Editorial_Board
    Reviewer
Control >
  Init
  Eval
Activity >
  Edit >
    Edit_Report
  Review
Specify >
  Create_Type
  Modify_Type
  Terminate_State

```

Next, we need to calculate the composition norm sets applicable to the current change request.

### 3.2.3 Composition Norm Matching Processes

Like any specification process, the active specification process consists of three compositions: its initiation, execution, and evaluation. These compositions are called *active compositions*.

In order to model the composition norm status dynamics, first two matching processes need to be defined: user matches and composition matches.

A *user match* is a match between a user and an actor part of some composition norm. This is defined as that at least one of the actor roles that the user plays is a subtype of the actor part.

A *composition match* is defined as a match between an *active composition* and the composition part of some composition norm, called the *norm composition part*. Such a match implies that the active composition must be a specialization of the norm composition part, as the norm should cover the active composition. A composition matches with, i.e. is governed by some composition norm, if the norm composition part is more generic than the composition to which it is to be applied.

**Definition 3** Let there be a composition norm  $d_{cn} = \langle id, de, a, cp, sp \rangle \in \mathcal{D}_{CN}$ . For this norm, the *actor part* is  $a$  and the *norm composition part*  $comp(d_{cn}) = \langle cp, sp \rangle$ .

Let there be a user  $u \in \mathcal{U}$ . There is a *user match* between  $u$  and actor part  $a$ , denoted as  $u \theta_u a$ , if  $u$  plays a role  $r \in \mathcal{A}$ , and this actor role is a subtype of  $a$ .

$sp_a$  is the active specification process. The set of active compositions  $Comp_A = \{\langle Init, sp_a \rangle, \langle Exec, sp_a \rangle, \langle Eval, sp_a \rangle\}$ .

Let there be some active composition  $comp_a \in Comp_A$ . There is a *composition match* between  $comp_a$  and norm composition part  $comp(d_{cn})$ , denoted as  $comp_a \theta_n comp(d_{cn})$ , if  $comp_a$  is a specialization of  $comp(d_{cn})$ . This means that the types of all elements of the active composition are subtypes of the corresponding elements of the norm composition part.

□

### Example

The active specification process  $sp_a = \underline{Create\_Type(Edit\_Report)}$ . One active composition is the initiation of this process:  $\langle Init, \underline{Create\_Type(Edit\_Report)} \rangle$ . For user John (the journal editor), there is no user match with norm #58, since the only role that John plays (*Journal\_Editor*), is not a subtype of *Editorial\_Board*. There is a match with, for example, norm #59, however, since *Journal\_Editor* is a subtype of *Editor*.

For the active composition  $\langle Init, \underline{Create\_Type(Edit\_Report)} \rangle$ , there is no composition match with the first two norms. It does match with the norm composition part of composition norm #60, however, since *Init* is a subtype of *Control*, *Create\_Type* is a subtype of *Specify*, and *Edit\_Report* is a subtype of the most generic type  $T$ .

□

### 3.2.4 Composition Norm Status Calculation

At any time, a composition norm base contains the set of *legitimate* norms  $\mathcal{D}_{CN}$ . A legitimate composition norm is *invoked* if there is at least one user with whom

the norm matches. Invoked norms become *active* if they match with the active specification process. For each combination of user and active specification process composition, a set of *applicable norms* exists, which determines what is the acceptable specification behaviour for that user and composition.

We say that a (legitimate) composition norm becomes an *invoked composition norm* if there is a user match between some user and the norm actor.

**Definition 4** The set of invoked composition norms  $D_{CN\_I} = \{d_{cn} = \langle id, de, a, cp, sp \rangle \in \mathcal{D}_{CN} \mid \exists u \in \mathcal{U} : u \theta_u a\}$

□

**Example**

The set of legitimate norms  $\mathcal{D}_{CN} = \{\#58, \dots, \#65\}$ . Norm #58 is not an invoked composition norm, because none of the roles John and Jack play are subtypes of *Editorial\_Board*. All other norms have at least one user match, and are thus in the invoked set of norms. Thus, the set of invoked norms  $D_{CN\_I} = \{\#59, \dots, \#65\}$ .

□

Whereas the invocation of legitimate norms depends on which users are participating in the community, the activation of the invoked composition norms depends on the currently *active specification process*. An invoked composition norm is also an *active composition norm* if at least one of the *active compositions* making up the active specification process matches with the norm composition part of the invoked norm.

**Definition 5** The set of active composition norms  $D_{CN\_A} = \{d_{cn\_i} \in D_{CN\_I} \mid \exists comp_a \in Comp_A : comp_a \theta_n comp(d_{cn\_i})\}$

□

**Example**

Invoked norm #59 is not an active composition norm, because none of the active compositions is a specialization of its norm composition part  $\langle Exec, \underline{Modify\_Type(Review)} \rangle$ . The remaining invoked norms are active norms, because at least one of the active compositions is a specialization of the norm composition part. Thus, the set of active norms  $D_{CN\_A} = \{\#60, \dots, \#65\}$ .

□

Active norms do not equally affect all users. We call an active composition norm *applicable* to a particular user for a particular active composition if (1) the user matches with the actor part of the norm and (2) the active composition matches with the norm composition part.

**Definition 6** The set of applicable composition norms for user  $u$  and active composition  $comp_a$ ,  $D_{CN\_APPL(u, comp_a)} = \{d_{cn\_a} = \langle id, de, a, cp, sp \rangle \in D_{CN\_A} \mid u \theta_u a \wedge comp_a \theta_n comp(d_{cn\_a})\}$

□

Figure 4: Composition Norm Dynamics Example

**Example**

To calculate, for instance,  $D_{CN\_APPL}(John, Exec\_Create\_Type(Edit\_Report))$ , we first look at the user matches. John, the journal editor, has user matches with norm #60, #61, and #62. The active composition  $\langle Exec, \underline{Create\_Type(Edit\_Report)} \rangle$  also matches with the norm composition parts of norm #60 and #61. However, it does not match with the norm composition part of norm #62, since  $Exec$  is not a subtype of  $Eval$ .

Summarizing, the applicable norm sets for this active specification process are:

- $D_{CN\_APPL}(John, Init\_Create\_Type(Edit\_Report)) = \{\#60, \#61, \#62\}$
- $D_{CN\_APPL}(John, Exec\_Create\_Type(Edit\_Report)) = \{\#60, \#61\}$
- $D_{CN\_APPL}(John, Eval\_Create\_Type(Edit\_Report)) = \{\#60, \#61, \#62\}$
- $D_{CN\_APPL}(Jack, Init\_Create\_Type(Edit\_Report)) = \{\#60, \#63, \#64\}$
- $D_{CN\_APPL}(Jack, Exec\_Create\_Type(Edit\_Report)) = \{\#60, \#64\}$
- $D_{CN\_APPL}(Jack, Eval\_Create\_Type(Edit\_Report)) = \{\#60, \#64, \#65\}$

All norm sets are depicted in Fig. 4. The subsets depicted within the set of active norms represent the various sets of applicable norms. In [11], we showed how the norm status dynamics of this example can be calculated making use of conceptual graph theory.

□

### 3.3 Composition Norm Conflict Resolution

Assuming that a set of norms applicable to a particular user for some active composition has been calculated, the next question is what is their *resultant deontic effect*? This is the overall deontic effect on the specification behaviour of a particular user in a particular active composition. Each set of applicable composition norms thus has its own resultant deontic effect.

If all norms are of the same category, then the deontic effect of this category applies. *Norm conflicts*, however, may occur, if norms are of different categories. There are two ways to handle such conflicts.

The first approach is to model norm conflicts as logical *inconsistencies*. In this case conflicting norms necessarily need to be modified, so that any inconsistency is removed before they can take effect.

This conflict prevention approach has the advantage that sets of norms are always theoretically consistent and that ambiguities about which deontic effect should prevail cannot occur. However, it comes with some serious drawbacks. First, it may not always be clear which norms need to be changed for an inconsistency to be resolved, and, second, no agreement may be reached on which norm has to give in.

In the meantime, no changes can be made until the status quo is broken. This problem is only worsened when a norm change causes a cascade of other norm changes. This means that a much needed knowledge definition change may not be allowed to take effect, even though the specifiers have the privilege to do so. Furthermore, the complexity of norm changes and their interrelationships may be so large that required change effectively comes to a halt.

The second approach to norm conflict resolution is to allow the occurrence of norm inconsistencies, and to determine the resultant deontic effect by means of *norm priority rules*. This can be classified as a defeasible norm reasoning approach, in which contradictory norms are not seen as true contradictions, but require some kind of ordering strategy to make the inconsistent set of norms consistent [31]. The main advantage is that norm changes take effect immediately and that the *autonomy* of the specifiers is guaranteed: users with permission to make a change can always do so, independent of what norms apply to other users. Another advantage is that norm changes can be kept more localized: only if users think the effects of some norm change on their own work to be really unacceptable will they have to start conversation for specification to undo it. A disadvantage of this approach is that acceptable behaviour may be different from what is expected, as the deontic effect of a specified norm may be overruled by a norm with higher priority.

However, if such norm conflicts are made sufficiently visible, they form a good basis for rational discourse about alternative norm specifications, while at the same time ensuring the independence of specifiers and the continuity of network operations.

In line with the semantics of the deontic effects defined in the previous section, the following rules are used to determine the *resultant deontic effect*:

1. If in a set of applicable norms all norms are of the same category, then the resultant deontic effect equals that of the norm category.
2. If at least one of the norms is a prohibition, then the resultant deontic effect equals 'forbidden'.
3. If none of the norms is a prohibition, and there is at least one responsibility, then the resultant deontic effect equals 'required'.

These rules are formalized in the following definition:

**Definition 7** Let  $u$  denote a user,  $comp_a$  an active composition, and  $D_{CN\_APPL(u,comp_a)}$  a set of applicable norms.

The resultant deontic effect  $\mathbf{de}_r$  is a function from  $\mathcal{P}(D_{CN\_APPL(u,comp_a)})$  to  $\mathcal{DE}$ , which assigns a deontic effect to each set of applicable norms.

Let  $d_{cn1}, d_{cn2} \in D_{CN\_APPL(u,comp_a)}$   
with  $d_{cn1} = \langle id_1, de_1, a_1, cp_1, sp_1 \rangle, d_{cn2} = \langle id_2, de_2, a_2, cp_2, sp_2 \rangle$ .

$$\mathbf{de}_r(D_{CN\_APPL}(u, comp_a)) = \begin{cases} \mathbf{de}(de_1) & \text{if } \forall d_{cn1}, d_{cn2} \in D_{CN\_APPL}(u, comp_a) \mid \\ & \mathbf{de}(d_{cn1}) = \mathbf{de}(d_{cn2}) \\ Forb & \text{if } \exists d_{cn1} \in D_{CN\_APPL}(u, comp_a) \mid \\ & \mathbf{de}(d_{cn1}) = Forb \\ Req & \text{if } (\neg \exists d_{cn1} \in D_{CN\_APPL}(u, comp_a) \mid \\ & \mathbf{de}(d_{cn1}) = Forb) \wedge \\ & (\exists d_{cn2} \in D_{CN\_APPL}(u, comp_a) \mid \\ & \mathbf{de}(d_{cn2}) = Req) \end{cases}$$

□

In this way,  $\mathbf{de}_r$  is well-defined for each set of applicable norms.

### Example

The following are the resultant deontic effects of the applicable norm sets calculated in the example that was given earlier in this section:

- $\mathbf{de}_r(D_{CN\_APPL}(John, Init\_Create\_Type(Edit\_Report))) = Forb$
- $\mathbf{de}_r(D_{CN\_APPL}(John, Exec\_Create\_Type(Edit\_Report))) = Req$
- $\mathbf{de}_r(D_{CN\_APPL}(John, Eval\_Create\_Type(Edit\_Report))) = Forb$
- $\mathbf{de}_r(D_{CN\_APPL}(Jack, Init\_Create\_Type(Edit\_Report))) = Req$
- $\mathbf{de}_r(D_{CN\_APPL}(Jack, Exec\_Create\_Type(Edit\_Report))) = Perm$
- $\mathbf{de}_r(D_{CN\_APPL}(Jack, Eval\_Create\_Type(Edit\_Report))) = Forb$

Thus, John is not allowed to initiate new types of report editorial workflow processes, while Jack is required to do so, and so on. Fig. 6 depicts the norm conflict resolution process for the initiation of this active specification process. Similar figures could be drawn for the execution and evaluation stages.

□

## 3.4 Composition Norm Hierarchies

The complexity of norms is caused not only by the interaction between different norm categories, but also by the *specificity* of the norms.

To determine the genericity of a norm, we introduce the concept of the *body* of the (composition) norm. These are all the norm elements that are included in the type hierarchy: the actor, control process, and specification process part. The deontic effect is thus excluded from the body, as it is orthogonal to the definition to which it applies.

All norms are implicitly ordered in a generalization hierarchy. We say that a norm is more *generic* than another norm if its body is a generalization of the body of the other norm. A generic norm affects more specification processes than a specific norm, in the sense that it affects at least the same and possibly more combinations of actors and compositions than a more specific norm. Conversely, if some norm applies, all more generic norms also apply.

**Definition 8** On each set of composition norms a partial ordering  $\mathcal{N}_{\leq}$  is defined. The top element of the hierarchy is represented by the  $\diamond$  symbol.

Figure 5: Composition Norm Hierarchy for the Example

A composition norm  $d_{cn1} = \langle id_1, de_1, a_1, cp_1, sp_1 \rangle$  is at least as or more generic than composition norm  $d_{cn2} = \langle id_2, de_2, a_2, cp_2, sp_2 \rangle$ , written as  $d_{cn1} \geq d_{cn2}$ , iff  $(a_1 \geq a_2) \wedge (cp_1 \geq cp_2) \wedge (sp_1 \geq sp_2)$ .

If  $d_{cn1}$  is at least as or more generic than  $d_{cn2}$ , then  $d_{cn2}$  is at least as or more specific than  $d_{cn1}$ , written as  $d_{cn2} \leq d_{cn1}$ .  $d_{cn1}$  is a *generalization* of  $d_{cn2}$ , while  $d_{cn2}$  is a *specialization* of  $d_{cn1}$ .

If  $d_{cn2}$  applies to a particular user  $u$  and active composition  $comp_a$ , and  $d_{cn2} \leq d_{cn1}$ , then  $d_{cn1}$  also applies to  $u$  and  $comp_a$ .

□

### Example

In Fig. 5, the partial ordering for the abovementioned example is presented.

We see, for instance, that norm #61 is both a specialization of norm #60, and a generalization of norm #62. This means that wherever the specific norm #62 applies, norms #60 and #61 also apply.

□

The hierarchical dependencies can be used to limit the amount of recalculations needed in case of composition norm changes. If a new composition norm is added to the set of legitimate norms, for example, it surely does not affect an applicable norm set if it is a specialization of an existing norm that itself is not in the particular applicable norm set.

**Definition 9** (Proposition) Let there be a new composition norm  $d_{cn\_n} \in \mathcal{D}_{CN}$ , a user  $u \in \mathcal{U}$ , and an active composition  $comp_a \in \mathcal{Comp}_A$ , then  $\forall D_{CN\_APPL}(u, comp_a)$  :

If  $\exists d_{cn} \in \mathcal{D}_{CN}, d_{cn\_n} \leq d_{cn} \wedge d_{cn} \notin D_{CN\_APPL}(u, comp_a)$ , then  $D_{CN\_APPL}(u, comp_a)$  does not change.

*Proof*

If the situation is such that  $d_{cn\_n}$  is applicable, then a more generic norm would also be applicable. So if at least one more generic norm is not applicable, then  $d_{cn\_n}$  cannot not be applicable.

□

### Example

Assume norm #60 does not exist (since it is the most generic norm, it always holds), that  $D_{CN\_APPL}(u, comp_a) = \{\#64, \#65\}$ , and that the new norm  $d_{cn\_n}$  is a specialization of norm #63. Since norm #63 is not in  $D_{CN\_APPL}(u, comp_a)$ ,  $d_{cn\_n}$  also cannot be.

□

Figure 6: Norm Conflict Resolution in the Example

Such a look-up approach is only one example of the kind of optimizations that can be made using properties of composition norm hierarchies. Especially with large numbers of norms and participants, such properties can generate significant efficiencies. More advanced properties are conceivable and will be the subject of future research.

## 4 A Formal Model of the Conversation for Specification

In the previous sections, a formal approach was described to *select* the users who can legitimately control the active specification process. Now, we study the *process* in which the specification changes are made, that is, the conversation for specification. The process describes *when* users are authorized to perform the conversation acts of the active specification process. These acts are described in full detail in the Specification Process Model (SPM) in [10]. Knowing the authorizations is essential in order to give particular users access to the proper specification options at the right moment in time.

A conversation for specification can be modeled as a state diagram where the edges correspond to conversational acts (cf. Fig. 2). For each state, it can be determined which conversational roles may or must execute which conversational acts. This structure is introduced in Sect. 4.1 and 4.2. However, this conversational structure has its limitations: it does not give precise semantics of conversational acts, and therefore the coherence between the conversation protocol and the conversational acts remains obscure. Therefore, in Sect. 4.3 we provide more detailed conversation semantics.

### 4.1 Conversation Roles

In each Conversation for Specification, three conversation roles are distinguished: the initiator, executor, and evaluator. The users that can play these roles are those for whom the resultant deontic effect of their respective applicable norm sets is either ‘permitted’ or ‘required’.

**Definition 10** The set of conversation roles  $\mathcal{CR} = \{I, X, E\}$ .

The set of users permitted to initiate the active specification process  $I_{(sp_a)} = \{u \in \mathcal{U} \mid \mathbf{de}_r(D_{CN\_APPL}(u, comp_a)) \in \{Perm, Req\}\}$ , with  $comp_a = \langle Init, sp_a \rangle$ .

$X_{(sp_a)}$  and  $E_{(sp_a)}$  are the sets of users permitted to execute and evaluate the  $sp_a$ , respectively, and are defined analogously.

$CR_{(u, sp_a)}$  is the set of conversation roles played by user  $u$  in active specification process  $sp_a$ . This set includes  $I$  if  $u \in I_{(sp_a)}$ ,  $X$  if  $u \in X_{(sp_a)}$ ,

$E$  if  $u \in E_{(sp_a)}$ .

□

**Example**

The sets of users permitted to control the active specification process  $sp_a$  are:

- $I_{(Create\_Type(Edit\_Report))} = \{Jack\}$
- $X_{(Create\_Type(Edit\_Report))} = \{John, Jack\}$
- $E_{(Create\_Type(Edit\_Report))} = \emptyset$

The sets of conversation roles played by the users are:

- $CR_{(Jack,Create\_Type(Edit\_Report))} = \{I, X\}$
- $CR_{(John,Create\_Type(Edit\_Report))} = \{X\}$

□

## 4.2 Conversation Acts and States

In Sect. 2.3, we showed how the Specification Process Model is composed of a set of interrelated conversation states and acts. Conversation acts, such as "requesting a knowledge definition change", have the effect of pushing the conversation to a next state (Fig. 2). First, we give a formal definition of the structure of conversation acts and states.

**Definition 11** The set of conversation act identifiers  $\mathcal{ID}_{CA} = \{CA1, \dots, CA23\}$

The set of conversation act labels  $\mathcal{L}_{CA} = \{ \text{'Requesting knowledge definition change'}, \dots, \text{'Reporting completion of knowledge definition change'} \}$

The set of conversation states  $\mathcal{CS} = \{CS0, \dots, CS12\}$

$\mathcal{CA}$  is the set of conversation acts. With  $id_{ca} \in \mathcal{ID}_{CA}, S, H \in \mathcal{CR}, l_{ca} \in \mathcal{L}_{CA}, ss, rs \in \mathcal{CS}, l_{rs} \in \mathcal{L}_{CS}, S$  and  $H$  being the conversation role (I, X, or E) of speaker and hearer,  $ss$  being the *starting state*, and  $rs$  the *resulting state*: we define the conversation act  $ca \in \mathcal{CA}$  as  $\langle id_{ca}, S, H, l_{ca}, ss, rs \rangle$ .

□

**Example**

These are the full descriptions of the first 4 conversation acts, plus the execution of the actual definition process. A description of all 23 conversation acts is given in [10].

Conv.Act#	S	H	CA-label	SS	RS
CA1	I	X	Requesting knowledge definition change	CS0	CS1
CA2	X	I	Committing to knowledge definition change	CS1	CS2
DP	X	X	Making knowledge definition change	CS2	CS3
CA3	X	E	Reporting completion of knowledge definition change	CS3	CS4
CA4	E	X,I	Accepting completion of knowledge definition change	CS4	CS5

□

Conversation states are identified by a number, but what do these conversation states stand for? Basically, the conversation state aims at capturing two kinds of information: (a) it determines who is the *player-to-move* (which is not a specific user, but a conversational role); and (b) it determines the *performable conversation acts*. For example, the performable acts in state CS1 (after a knowledge definition change request) are CA2 (commit), CA5 (request justification), and some others (see Fig. 2). This is captured by the following definition.

**Definition 12**  $cs(now)$  is the current conversation state.

For  $cs(now)$ , the set of performable conversation acts  $CA_{P(cs)} = \{ca = \langle id_{ca}, S, H, l_{ca}, ss, rs \rangle \in \mathcal{CA} \mid ss = cs\}$ .

At the start of  $sp_a$ ,  $cs(now) = CS0$ . After a conversation act  $ca \in \mathcal{CA} = \langle id_{ca}, S, H, l_{ca}, ss, rs \rangle$  has been performed,  $cs(now) = rs$ .

□

### Example

Assume the specification process of the report editorial workflow has begun, and the current conversation state  $cs(now) = CS1$ . The set of performable conversation acts  $CA_{P(cs)} = \{CA2, CA5, CA7, CA8, CA19\}$ . Say that act CA2 ("committing to knowledge definition change") has been performed. As a result, the current conversation state  $cs(now) = CS2$ .

□

## 4.3 Conversation Semantics

The conversational state is characterized by which conversational acts are possible from there and whose turn it is. However, from a socio-technical perspective, it is important to know what the meaning is of that state in terms that matter to the community. For example, at some point an obligation is created for certain actors to perform a knowledge definition change. But when? When the Conversation for Specification is started? Or after the request being made? And when is the obligation fulfilled? To get this kind of knowledge, we must know the effects of the conversation acts on the social world. The practical benefit of this is that it allows the community members to know, at each point in time, who is responsible or authorized for what. To achieve this, the state diagram model is not sufficient.

In [56], a formal language called  $L_{ill}$  is described with which an integrated semantics for information and communication systems can be expressed. It is an extension of Dynamic Deontic Logic and the semantics of speech acts is described using preconditions and postconditions. For example, the postcondition of an authorized request is that the Hearer is obliged to perform the requested action. Pre- and postconditions have been used also in agent communication languages such as KQML and FIPA-ACL (see [8] for references). For example, the precondition of KQML's *tell* message states that the sender believes what it tells and that it knows that the receiver wants to know that the sender believes it. The postcondition of sending the *tell* message is that the receiver can conclude that the sender believes the content of the message. In a similar vein, FIPA-ACL uses feasibility preconditions and rational effects. There have been critical discussions about this approach

Figure 7: Dynamics of Knowledge Definition States

(see [8] for an overview). Some have argued that the semantics should not be based on mental states, but on social commitments [43]. Others have tried to ground the semantics in the notion of sign conventions [23]. The semantics that we propose here is in line with these latter two approaches in the sense that we agree that the effect on the social world should be at the core. This is also in accordance with Habermas’s theory of communicative action. Where our approach differs from the latter two is the Habermasian assumption that intersubjective truth (common ground) is established by a joint act of speaker and hearer.

The semantics that we propose in this article is built on this assumption. As we have seen in section 2.2, the theory of communicative action is based on the notion of validity claim. Speakers make claims, and when these claims are accepted or conceded, they turn into common ground.<sup>2</sup> In this way, coordination can be achieved. The general scheme is as follows.

**Definition 13** Inference scheme for communication semantics.

For all  $\phi$  being a well-formed formula,  $i, j$  being conversational roles

$$[claim(i, j, \phi); accept(j, i, \phi)] agreed_{i,j}(\phi)$$

In words: when  $\phi$  has been claimed and has been accepted, then it is agreed. For example, when some user claims that the editors should change the reviewing process, and the editors accept this claim, (only) then there is an agreed upon obligation.

□

The fact that  $\phi$  has the status ”agreed” does not say anything about internal beliefs. Depending on whether the hearer is convinced of the sincerity and trustworthiness of the speaker, he will infer (or not) that  $\phi$  is believed by the speaker and believe it himself. However, this inference is not critical for the conversation process, because what counts for other community members is what is agreed upon.

Using the basic communication semantics above, we are able to describe the effects of conversational acts once we know which claims are made with a certain conversation act. Among the many claims that could be made by a speaker when performing a conversational act, we distinguish the following essential categories:

- authorization claims - when performing a conversational act, the speaker claims that he is authorized to perform the act. In section 3, we have seen how this can be calculated. If such a calculation is available, it provides a strong backing for the claim.

---

<sup>2</sup>In a more refined account, a difference can be made between claiming and suggesting. Both lead to agreement. However, in the case of a claim, the speaker supposedly has made his choice, and tries to convince the other party to agree, whereas in the case of a suggestion, the speaker merely says: if you make this choice, you already have my agreement. This has certain consequences for the argumentation process in the case of a discussion

- claims about actor obligations - what a user or set of users should do. Obligations can be in one of the following states: *created, cancelled, violated, fulfilled*.
- claims about actions to be performed - the things to be achieved in the specification process, that is, the knowledge definitions. Knowledge definitions (KD) can be in one of the following states: *desired, intended, started, finished, approved*(see Fig. 7). These phases correspond roughly to the well-known action cycle of Norman [33].

This categorization is not meant to be exhaustive, but it covers the most important cases in our context. For each of the *claim* types, there is also a corresponding *accept* action.

### **Example**

We consider again the creation of the editorial workflow of a report: *Create\_Type(Edit\_Report)*, that we indicate here by kd. The conversation act that starts the specification process is the request of a knowledge definition change(CA1). When Mary sends a request to the editors to create the new workflow, in fact she performs a CA1 and makes the following claims:

- *authorized(Mary,CA1)*- Mary, being the Speaker, is permitted to perform this request. As CA1 is by definition a conversational act that is performed by the Initiator role, this claim is equivalent to the claim that Mary has the Initiator role.
- *created(obligation(Editor,kd))* - the *Editor* role is obliged to perform the knowledge definition change (that is, prepare a change proposal to be implemented after approval.
- *intended(kd)* - the action state of the specification process knowledge definition kd is "intended" ("to be done")

□

A request of a knowledge definition change is made after a breakdown has been observed or an opportunity has been recognized. At this point, the desirability of the knowledge definition change has been discussed already, so the Conversation for Specification starts when the kd already has the status "desired". The speaker (with the initiator role) claims that the composition is to be performed now (intended), and claims that the hearer (with executor role) is obliged to perform it. Both claims can be challenged by the hearer, but if they are accepted, they lead to an obligation for the executor and a state change of the action itself (from intended to started).

The obligation claims and composition claims are closely related, because it would be odd when an composition is considered to be intended, but no one is responsible for the execution, or vice versa. However, these odd situations can happen in complex settings. For example, when John after having committed withdraws and his obligation is cancelled. Or when the request to the editors is made in a situation where no user has been assigned the *Editor* role, so that the claim

”created(obligation(*Editor*,kd))” is void. By separating the two claims, it also becomes possible to accommodate the situation that the hearer accepts one claim but challenges the other.

CA1, by attempting to create an obligation on the part of the hearer, is a typical request (the semantics are a refinement of the generic speech act semantics of request). Whether it is a *legitimate* request, can only be determined against the background of the normative context (Fig. 3), and it is also this background that determines the net effect. For example, the request creates an obligation on the part of the executor role, but we only know who is to be involved in the execution by consulting the composition norms.

### **Example**

The intended response of CA1 (request) is CA2(commit). This action is performed by one of the hearers (a turn taking takes place), and consists of a commitment to execute the knowledge definition change. Let us suppose that John as editor takes up Mary’s request and commits. The authorization claim of this commitment (CA2) is:

- authorized(John,CA2) - the hearer is authorized to make a commitment. As CA2 is by definition a conversational act that is performed by the Executor role, this claim is equivalent to the claim that John has the Executor role. In our example, John is editor, and as we have seen in section 3.2, it can be calculated that he has the Executor role.

Furthermore, this conversational act contains *acceptances* corresponding to the claims made by Mary (CA1) above. So CA1 and CA2 together have at least the effect that ”created(obligation(*Editor*,kd))” and ”intended(kd)” are ”agreed”. Note that John could also have responded differently. For example, if he thinks that this workflow definition should not be undertaken, he will challenge Mary’s claim ”intended(kd)”, and Mary should start a discussion backing up her claim. Note that there is no guarantee that she will prove her claim (it can not be calculated in the way John’s authorization can be calculated), but she may be able to provide arguments that convince John.

If John commits, and everything goes well, the obligation will move later to the status ”fulfilled” and kd will move from the status ”started” to ”finished” and ”approved”. If it does not go well, for example, because John does not produce a workflow definition in time, the obligation will move later to the status” cancelled” or ”violated”.

□

In this way, the effects of each conversational move can be described in terms of claims and accepts about authorizations and about the status of obligations and knowledge definitions.

A remark is due on the meaning of the obligation. We assume that it follows basically the logical properties of Deontic Dynamic Logic. What complicates the

picture is that the obligation is assigned to a conversational role, and behind this role, there is typically a set of users. So the obligation on the conversational role must be interpreted as a collective obligation [37]). A collective obligation does not distribute to the members of the group. For example, if the obligation is the editorial board has to provide a type definition, then it does not mean that each member of the board has the individual responsibility to provide the type definition. However, it is the case that when one of the members, or any subset of the board, does the job, the collective obligation is fulfilled. In specification processes, this kind of collective obligations is quite common. However, we do not think that the relationship between collective and individual obligations can be specified on an abstract logical level, as different groups will use different decision processes. For example, one board may have the rule that such a job is always to be done by two of its members. These kinds of rules can be taken into account as additional norms.

## 5 Related work

In this section, we discuss our results in the context of other approaches.

### 5.1 Workflow modeling

The focus of this article is on composition norms, which govern specification behaviour. There is a direct link with workflow modelling methods. Much of the current workflow modelling literature focuses on developing representations of enterprise models or control flows, e.g. [40, 48]. However, such approaches focus on optimising workflow systems that are hierarchically defined. This does not work for virtual communities that are not hierarchically governed, however. Communities should have the diverse interests of their members balanced by their unique social norms. If virtual communities are not to have their information systems imposed upon them, their composition norms need to be made explicit and used to create acceptable workflow specifications [14].

Besides for workflow modeling processes, similar ideas could be worked out for checking the legitimacy of operational processes, such as workflow enactment. The idea of generalization hierarchies could also prove useful to create more advanced forms of role-based access control (RBAC) models, in which permissions are assigned to users, and users assigned to appropriate roles [32].

### 5.2 Argumentation

The semantics of conversations is usually described in terms of speech acts. We have extended this framework to account for the interplay of communicative action and the shared background of knowledge definitions. The semantics of communicative actions is given in terms of claims, and these claims get their backing from the shared norms in the community. The effects of the communicative actions appeals to the shared background as well: once a claim is accepted, its content is added to the "agreed".

The next step not worked out in this article is to account for the possibility of a rational discussion by linking to argumentation theory. The discussion layer is recognized by Van Reijswoud in speech act theory [50], but he does not offer a formal semantics. Argumentation is a process in which dialogue partners challenge claims of the other, and defend their own claims by building up a line of reasoning from their claim down to definitions in the common ground, or to claims that the partner is willing to accept at that moment. Argumentation theory [52] can provide a quite natural semantics to the discussion processes. Its practical utility could be that it helps to structure the discussion process in order to avoid non-constructive discussion patterns such as repetitions.

### 5.3 User-centered design

To position RENISYS in the field of specification methods for community information systems, we give a brief motivation here. A more detailed account is given in [13, 12].

Community IS development (CISD) is a form of user-centered design [34]. User-centered design is the aim of a wide range of approaches, such as participative/participatory design, cooperative design, joint-application design, and customer-centered design. In participatory design, for example, end-users, in conjunction with developers, explore the affordances and constraints of information tools for supporting specific work practices [4].

The question is how to theoretically ground a legitimate user-driven specification method. Traditional methods are often grounded in the functionalist paradigm, assuming little change and that there is one objective reality. In contrast, the neo-humanist paradigm holds that knowledge is socially constructed in a process of human interaction, and that there is a natural tendency towards change and conflict [22, 21]. This paradigm is especially suited for modelling CISD, because in virtual communities, stakeholders with many conflicting interests need to work together to construct and balance their models of their own work processes and supporting information technologies. In neo-humanist system development approaches, the removal of communication distortions in specification processes is essential. This happens in a process of *rational discourse*, in which claims made throughout the systems development process are critically evaluated.

From a neo-humanist point of view, user-centered methods can be classified along two dimensions. The first dimension is the *user control*. It concerns the role that users play as modellers of specifications. The modelling roles can be mostly or completely played by external analysts (as in traditional methods) or by the users themselves. We call the first category *user-assisted* and the second category *user-driven* specification methods.

The second dimension concerns the *legitimacy* of the specification method. Methods that focus on obtaining specifications from individual users are called *individualistic* methods, while those that concentrate on the selection of the relevant stakeholders affected by a specification change can be called *legitimate* specification methods.

Mainstream user-centred specification methods can be classified along these two

dimensions. Prototyping is an example of a user-assisted/individualistic specification method. Tailorable tools [29] are examples of user-driven/individualistic methods. Socio-technical methods such as Soft Systems Methodology and ETHICS [22] are typical user-assisted/legitimate methods. They pay much attention to rational discourse, but are not very good at managing evolutionary change by end-users. Furthermore, their specification processes are hard to automate, due to lack of formal semantics. RENISYS is an example of method that aims to fill the gap of user-driven/legitimate specification methods.

## 6 Conclusions

Virtual communities, such as e-business platforms and research networks, are crucial instruments for collaboration in today's networked and globalizing society. These communities are prone to extensive change, because of rapid evolution of their requirements and technological developments. However, often change processes are not successful, because of their complexity, cost, and unclarities about dependencies and responsibilities.

Traditional specification approaches focus on optimising a workflow system for requirements that are hierarchically defined, often by an anonymous analyst. The authority basis for specification changes is found in that hierarchy. Such approaches are valid for systems that require central control of their development and maintenance, i.e. transaction processing systems for banks. Virtual communities, however, are not governed by such a hierarchy, but instead should allow the interests of their members to be balanced by their unique social norms. To reduce these problems, systematic methodological support is needed for the required legitimate user-driven specification process. In this article, we outline one such approach: the RENISYS method.

To develop adequate legitimate user-driven specification methods, the formal semantics of the specification process prevailing in virtual communities must be clearly understood. First, trust is essential for collaboration in these communities to occur. Only when the rationale for selecting and authorizing members in particular change processes is clearly defined, will such trust be possible. Second, change processes are very complex, because of the many different dependencies between process elements. Well-defined formalizations can help in the administration and facilitation of the change process, and the resolution of any breakdowns. Thus, formalizations can be used to structure specification conversations. Third, many variations of the legitimate user-driven support methodology are conceivable, for example in dealing in different ways with norm hierarchies and conflicts, or in the authorizations members get in the various stages of the specification process. Clear formalizations help to adapt the methodology without generating inconsistencies and incompleteness. The resulting semantics form a firm basis in which to ground methodological functionality for participant selection and conversation for specification support.

Besides working out the formal structure and role of composition norms, we have also formalized the basic semantics of the required conversations for specification. This we did by extending the traditional speech act framework to account for the

interplay of communicative action and given community norms. This is achieved by defining communicative acts in terms of claims. An advantage of the communicative action semantics is that it can deal quite naturally with rational discussion (not worked out in this article). Discussion processes are vital for thriving communities, and the efficiency and effectiveness of these discussions processes may be improved by adopting certain rationality principles. Current discussion tools usually lack formal semantics.

## 6.1 Is it worth the costs?

It may be argued that we introduce a complex machinery to handle relatively simple functionality changes. However, it should be understood that our focus is optimizing acceptability, not functionality. The quality of the solutions still fully depends on the contributions of the participants in the conversation for specification. For ensuring technical quality of specifications, other, more traditional specification methods could be used, if desired. However, the complexity we introduce is essential to safeguard the interests and ensure the active participation of community members. The approach embodied in RENISYS might be compared to a democratic system. Although democracy is a sometimes costly and complex way of decision making compared to more autocratic approaches of governance, it has the crucial advantage of respecting the wishes and sensitivities of the community. In this way, true, instead of forcefully imposed acceptance of sometimes painful decisions can be achieved, essential for the sense of ownership and long-term prospering of the community.

RENISYS was developed using a base case of a relatively small and well-organized professional online community, which also had a - funded - project stage. Many of the specifications could therefore be done face-to-face. As soon as complexity increases, however, human oversight of normative dependencies disappears and automated support by methods like RENISYS will be necessary. Experiments with large-scale, distributed, emerging and more fuzzy communities should demonstrate the power of this approach. We conjecture that one reason that many distributed communities, especially among stakeholders with adversarial interests, do not take off, is especially because of the lack of such support [16].

It could be argued that the method is too complex for the average user to understand. However, the underlying complexity of norm and conversation authorization calculations does not need to be presented to the user at all. He or she will see an interface in which options are presented in a clear way, for example a pulldown-list of permitted actions [14]. By showing options, legitimate participants, and verbal versions of the norms that apply, members can become more aware of and involved in the governance of their community.

## 6.2 Further research

A prototype of the method was implemented, and has been described in [14]. The method has been used to analyze evolution processes in various professional virtual communities, for example in the domain of e-commerce [54]. We are currently applying the method to a range of cases and plan to operationalize it into various

tools. We are doing several longitudinal case studies to distill the composition norms involved in successes and failures of community information system evolution, amongst others in research and gaming communities. Moreover, a software company providing large and advanced ERP-systems for supermarket retail management is planning to develop a workflow maintenance module based on the RENISYS method. In this way, the many stakeholders involved in the supermarket supply chain will be able to deal with their strongly evolving requirements in a trusted way.

Of course, many other interpretations of the basic idea that users should be actively and legitimately involved in the definition of their *own* socio-technical system are conceivable. We are convinced, however, that the formalization of the legitimate user-driven specification process presented here, can help in the development of a new class of robust methodologies and applications crucial in the Internet-age.

## Acknowledgments

The authors wish to thank Manfred Jeusfeld, Mike Papazoglou, and the anonymous reviewers for their helpful comments on earlier versions of this article.

## References

- [1] D. Andrews. Audience-specific online community design. *Communications of the ACM*, 4(45):64–68, 2002.
- [2] E. Auramäki and K. Lyytinen. On the success of speech acts and negotiating commitments. In *Proceedings of the First International Workshop on Communication Modelling, the Language/Action Perspective (LAP'96), Oisterwijk, The Netherlands, July 1-2, 1996*, pages 1–12, 1996.
- [3] J. Austin. *How to Do Things with Words?* Clarendon Press, London, 1962.
- [4] L. Bannon. From requirements as texts to requirements as constructions - emphasizing use, process, and iteration in systems development. In *CAiSE 96 - Workshop W1: Requirements Engineering in a Changing World, Crete, May 20-21, 1996*, 1996.
- [5] P. Barthelmeß. Collaboration and coordination in process-centered software development environments: A review of the literature. *Information and Software Technology*, 45:911–928, 2003.
- [6] J.F.M. Burg. *Linguistic Instruments in Requirements Engineering*. PhD thesis, Free University of Amsterdam, 1997.
- [7] L. Carotenuto (coord.). Communityspace: Toward flexible support for voluntary knowledge communities. In *Changing Places Workshop, London, April 1999*, 1999.
- [8] B. Chaib-draa and F. Dignum. Trends in agent communication language. *Computational Intelligence*, 18(2):89–101, 2002.

- [9] G. De Michelis and M.A. Grasso. Situating conversations within the language/action perspective: The Milan Conversation Model. In R. Furuta and C. Neuwirth, editors, *CSCW '94*, pages 89–100. ACM, 1994.
- [10] A. de Moor. *Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems*. PhD thesis, Tilburg University, The Netherlands, 1999. ISBN 90-5668-055-2.
- [11] A. de Moor. Composition norm dynamics calculation with conceptual graphs. In *Proceedings of the Eighth International Conference on Conceptual Structures, ICCS2000, Darmstadt, Germany, August 14–18, 2000*, 2000.
- [12] A. de Moor. Evaluating methods for community is development. In *Proc. of the 7th CAiSE/IFIP WG8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD-2002), Toronto, May 27–28, 2002*, pages 152–159, 2002.
- [13] A. de Moor. Language/action meets organisational semiotics: Situating conversations with norms. *Information Systems Frontiers*, 4(3):257–272, 2002.
- [14] A. de Moor and M.A. Jeusfeld. Making workflow change acceptable. *Requirements Engineering*, 6(2):75–96, 2001.
- [15] A. de Moor and S. van Erp. Community dynamics in an online law journal. In *Proc. of the 17th Bled eCommerce Conference, Bled, Slovenia, June 21 - 23, 2004*, 2004.
- [16] A. de Moor and H. Weigand. Effective communication in virtual adversarial collaborative communities. In *54th Annual International Communication Association Conference: Communication in the Public Interest, New Orleans, May 27-31, 2004*, 2004.
- [17] J.L.G. Dietz. Modelling business processes for the purpose of redesign. In *Business Process Re-Engineering: Information Systems Opportunities*, pages 233–242. North-Holland, 1994.
- [18] V. Dignum, J.J. Meyer, F. Dignum, and H. Weigand. Formal specification of interaction in agent societies. In *2nd Goddard Workshop on Formal Approaches to Agent-Based Systems, Maryland*, pages 401–415, 2002.
- [19] B.R. Gaines. Dimensions of electronic journals. In T.M. Harrison and T. Stephen, editors, *Computer Networking and Scholarly Communication in the Twenty-First Century*, pages 315–339. State University of New York Press, 1996.
- [20] P. Gongla and C.R. Rizzuto. Evolving communities of practice: Ibm global services experience. *IBM Systems Journal*, 40(4):842–862, 2001.
- [21] R. Hirschheim, H. Klein, and K. Lyytinen. *Information Systems Development and Data Modeling - Conceptual and Philosophical Foundations*. Cambridge University Press, 1995.

- [22] R. Hirschheim and H.K. Klein. Realizing emancipatory principles in information systems development: The case for ETHICS. *Management Information Systems Quarterly*, 18(1):83–109, 1994.
- [23] A.J. Jones and X. Parent. Conventional signalling acts and conversation. In *AA-MAS Workshop on Agent Communication Languages and Conversation Policies, Melbourne, 2003*, 2003.
- [24] F. Kensing and T. Winograd. The language/action approach to design of computer-support for cooperative work: A preliminary study in work mapping. In R.K. Stamper, P. Kerola, R. Lee, and K. Lytinen, editors, *Collaborative Work, Social Communications and Information Systems*, pages 311–331. IFIP, 1991.
- [25] W.J. Kettinger and C.C. Lee. Understanding the IS-user divide in IT innovation. *Communications of the ACM*, 45(2):79–84, 2002.
- [26] R. Kling and L. Covi. Electronic journals and legitimate media in the systems of scholarly communication. *The Information Society*, 4(11):261–271, 1995.
- [27] R. Kling, G. McKim, J. Fortuna, and A. King. Scientific laboratories as socio-technical interaction networks: A theoretical approach. In *Proceedings of AMCIS 2000, August 10-13, Long Beach, CA, 2000*.
- [28] R.V. Kozinets. E-tribalized marketing? the strategic implications of virtual communities of consumption. *European Management Journal*, 17(3):252–264, 1992.
- [29] T.W. Malone, K.-Y. Lai, and C. Fry. Experiments with Oval: A radically tailorable tool for cooperative work. *ACM Transactions on Information Systems*, 13(2):177–205, 1995.
- [30] R. Medina-Mora, T. Winograd, R. Flores, and F. Flores. The ActionWorkflow Approach to workflow management technology. *The Information Society*, 9(4):391–404, 1993.
- [31] J.-J.Ch. Meyer and R.J. Wieringa. Deontic logic: A concise overview. In J.-J.Ch Meyer and R. Wieringa, editors, *Deontic Logic in Computer Science: Normative System Specification*, pages 3–15. John Wiley & Sons Ltd., 1993.
- [32] C.-J. Moon, D.-H. Park, S.-J. Park, and D.-K Baik. Symmetric RBAC model that takes the separation of duty and role hierarchies into consideration. *Computers & Security*, 23:126–136, 2004.
- [33] D.A. Norman. *The Design of Everyday Things*. Doubleday, New York, 1990.
- [34] J. Preece. *Online Communities: Designing Usability, Supporting Sociability*. John Wiley & Sons, New York, 2000.

- [35] S. Purao and D.P. Truex III. Information systems research: Relevant theories and informed practice. In S. Purao, D.P. Truex III, D. Wastell, A.T. Wood-Harper, and J.I. deGross, editors, *Supporting Engineering of Information Systems in Emergent Organizations*. Kluwer, 2004.
- [36] H. Rheingold. *The Virtual Community: Homesteading on the Electronic Frontier*. HarperPerennial, 1993.
- [37] L. Royakkers and F. Dignum. From collective to individual commitments. In *Proc. of the 7th International Conference on Artificial Intelligence and Law*, pages 192–193. ACM Press, 1999.
- [38] S. Sawyer. A market-based perspective on information systems development. *Communications of the ACM*, 44(11):97–102, 2001.
- [39] T. Schäl. *Workflow Management Systems for Process Organizations*. Springer Verlag, 1996.
- [40] A.-W. Scheer. ARIS. In P. Bernus, K. Mertins, and G. Schmidt, editors, *Handbook on Architectures of Information Systems*, pages 541–565. Springer-Verlag, Berlin, 1998.
- [41] J.R. Searle. *Speech Acts - An Essay in the Philosophy of Language*. Cambridge University Press, 1969.
- [42] B. Shneiderman. ACM’s computing professionals face new challenges. *Communications of the ACM*, 45(2):31–34, 2002.
- [43] M.P. Singh. A social semantics for agent communication languages. In F. Dignum and M. Greaves, editors, *Issues in Agent Communication*, pages 31–45. Springer-Verlag: Berlin, 2000.
- [44] M. Smith. Tools for navigating large social cyberspaces. *Communications of the ACM*, 45(4):51–55, 2002.
- [45] Harvard Law School (student authored). The law of cyberspace. communities virtual and real: Social and political dynamics of law in cyberspace. *Harvard Law Review*, 112(7):1586–1609, 1999.
- [46] J.R. Taylor. The limits of rationality in communication modeling: A socio-semiotic reinterpretation of the concept of ‘speech act’. In *Proceedings of the Third International Workshop on Communication Modelling, the Language/Action Perspective (LAP’98), Steningevik, Sweden, June 25-26, 1998*, pages 35–46, 1998.
- [47] W.M.P. Van der Aalst. Information and process integration in enterprises: Rethinking documents. In T. Wakayoma, S. Kannapan, C.M. Khoong, S. Navathe, and J Yates, editors, *Three Good Reasons for Using a Petri-Net-Based Workflow Management System*. Kluwer, 1998.

- [48] W.M.P. Van der Aalst, A.H.M. ter Hofstede, B. Kiepuszeswki, and A.P.B. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14:5–51, 2003.
- [49] L. van der Torre. *Reasoning about Obligations: Defeasibility in Preference-Based Deontic Logic*. PhD thesis, Erasmus University Rotterdam, 1997.
- [50] V. van Reijswoud. *The Structure of Business Communication: Theory, Model and Application*. PhD thesis, Delft University, 1996.
- [51] E. Verharen. *A Language-Action Perspective on the Design of Cooperative Information Agents*. PhD thesis, Infolab, Tilburg University, 1997.
- [52] D.N. Walton and E. Krabbe. *Commitments in Dialogue. Basic Concepts of Interpersonal Reasoning*. State Univ of New York Press, Albany, NY, 1995.
- [53] R. Weber. *Information Systems Control and Audit*. Prentice Hall, 1999.
- [54] H. Weigand, A. de Moor, and W.J. van den Heuvel. Supporting the evolution of workflow patterns for virtual communities. In *Proc. of HICSS-33, Maui, January 4-8, 2000*.
- [55] H. Weigand, F. van der Poll, and A. de Moor. Coordination through communication. In *Proc. of the 8th International Working Conference on the Language-Action Perspective on Communication Modelling (LAP 2003), Tilburg, The Netherlands, July 1-2*, pages 115–134, 2003.
- [56] H. Weigand, E. Verharen, and F. Dignum. Integrated semantics for information and communication systems. In R. Meersman and L. Mark, editors, *Database Application Semantics*. Chapman & Hall, 1997.
- [57] B. Wellman. Computer networks as social networks. *Science*, 293:2031–2034, 2001.
- [58] E. Wenger, R. McDermott, and W.M. Snyder. *Cultivating Communities of Practice*. Harvard Business School Press, 2002.
- [59] S.K. White. *The Recent Work of Jürgen Habermas: Reason, Justice, and Modernity*. Cambridge University Press, 1988.
- [60] T. Winograd. A language/action perspective on the design of cooperative work, report no.CSLI-87-98. Technical report, Center for the Study of Language and Information, Stanford University, May 1987.
- [61] T. Winograd and F. Flores. *Understanding Computers and Cognition - A New Foundation for Design*. Ablex Publishing Corporation, 1986.