Legitimacy Checking in Communicative Workflow Design

Aldo de Moor (AdeMoor@uvt.nl)[1] and Hans Weigand (H.Weigand@uvt.nl)[1]

Tilburg University, The Netherlands

Abstract. Communicative workflow modelling is key to describing, analyzing, and designing business processes in virtual collaborative networks, such as present in e-commerce. To make workflow models meaningful and acceptable to all partners, their legitimacy needs to be checked. To this purpose, the underlying norms must be made explicit. A key class of communicative workflow models is captured by our extended workflow loop. Using this loop as the basic unit of analysis, we introduce the concept of workflow loop norms, grounded in, amongst others, internal control theory. Workflow loop schemas are used to represent workflow situations, allowing for actual or proposed situations to be matched with the norms. Using these constructs, we outline our legitimacy checking process for workflow designs, and illustrate it with a case.

1 Introduction

In today's networked organizations, organizational hierarchies are rapidly becoming less relevant for structuring business processes. Intra-organizational, self-organizing teams, learning organizations, and inter-organizational e-business alliances are emerging in which fixed power and communication structures no longer suffice. To make sense of the increasing organizational complexity and dynamics, and to design more adequate supporting information systems, a workflow view on organizational interactions is helpful. A good example is the increasing prominence of supply chain modelling [10]. Thus, workflow modelling is becoming increasingly important as a structured way of describing, analyzing, and designing the collaborative business process.

Many workflow models exist, ranging from Petri-nets representing logistical or production workflows [1] to approaches that capture more of the organizational semantics of business processes [16]. One class of workflow modeling approaches takes a communications view, grounded in the Language/Action Perspective (LAP), as originally introduced by Winograd and Flores [20]. In contrast to data-oriented

methods like those based on state-transition or UML interaction diagrams, LAP modeling is based on the notion of *communicative action*, which implies that workflows are seen as communicative acts grounded in social relationships and focused on organizational coordination. For example, a request for a certain good not only aims at the performance of a particular action, but is also an action in itself. A successful request creates a commitment for the party that has promised to deliver a service, thus changing the social world. LAP workflows are represented as *communication loops* between business *roles*. For example, in the ActionWorkflow approach [14], *customers* and *performers* go through four communication acts: in the *preparation* act, a customer asks a performer to do something, at the end of the *negotiation* stage, the performer promises to do this, in the *performance* act the performer reports that he has done so, and, finally, with the *acceptance* act the customer reports that she is satisfied. In the DEMO (Dynamic Essential Modeling of Organizations) approach [9], an *initiator* and an *executor* subsequently (inter)act in similar *order*, *execution*, and *result* stages.

From a coordination perspective, communication loops are more than sequences of communicative acts. LAP imposes a certain normative structure on communication processes [19]. For instance, ActionWorkflow modeling requires communication loops to be complete, or "closed". This means that all stages of a communication loop must be followed, none can be skipped. Another example of a communication norm, implicit in DEMO, is that the initiator of a transaction must also be the person who evaluates the success of that transaction. It can be argued, however, that, especially in complex network organizations, such evaluative activities should be delegated to third parties. Also, many workflows spawn other workflows, making the coordination of their interdependencies soon very complex. An additional factor is the strong co-evolution of social and technical requirements in the modern organization [13]. The resulting dynamics in organizational requirements, structures, and behaviour greatly increases the complexity of workflow analysis.

In order to ensure successful organizational performance, the workflow specifications of electronic business networks need to be legitimate, implying that they are both meaningful and acceptable to all

partners [6]. Such legitimacy is essential to create a sense of trust and ownership in the network and its supporting IT infrastructure. Central to this analysis are the communicative norms that apply to the workflows of a particular community. To ensure the legitimacy of workflow models, it is important that (1) underlying norms are made explicit and (2) actual or proposed workflow models are checked against these norms. However, the complexity of the communicative norms resulting from their subtle definition, compounded by delegation, recursion, and evolutionary aspects, makes manual analysis very hard to perform. In this chapter, we therefore propose a formal approach to the legitimacy checking of communicative workflow loops, as this helps to deal with representational and reasoning complexities.

In Sect. 2, we first define the *extended workflow loop*, our basic unit of analysis. Sect. 3 introduces our concept of workflow loop norms, and shows how they are firmly grounded in internal control theory, among others. In Sect. 4, we present *workflow loop schemas* as a concise way of representing workflow situations. Sect. 5 outlines the method for the legitimacy checking of extended workflow loops. We end the article with discussion and conclusions. In the various sections, we illustrate the ideas with a typical business case.

2 The Extended Workflow Loop

In [19], we extensively explained extended workflow loops and their norms. In Sect. 2 and 3, we give a brief summary of these ideas.

To define the extended workflow loop, we take the *service relationship* between two actors as its starting point. In most cases, this is a symmetric relationship where a *service* or object of value is exchanged against some (generally financial) compensation. This is thus a form of a contractual relationship, whether there is a written contract or not. The service has a *provider* and a *beneficiary*, which usually but not necessarily coincide with the *performer* and the *customer* of the service, respectively. Service relationships are typically found on the organizational borders, but they may also be explored within organizations.

To activate a service relationship, we start with describing a *service loop*, that is, the communication around the service. This service

loop should not be identified with the service relationship, although it necessarily follows from it. From DEMO, we take the workflow loop *roles* of *initiator* and *executor*, to which we add the *evaluator* role. Instead of using the specific DEMO (order, execution, result) and ActionWorkflow (preparation, negotiation, performance, and acceptance) workflow loop phases, we distinguish two, more neutral pairs of communicative acts, each pair defining a conversation: a *request* is followed by a *commit* (an *actagenic* conversation), and a *report* is followed by an *accept* (a *factagenic* conversation). Furthermore, we distinguish three workflow tasks: *initiation* (I), *execution* (X), and *evaluation* (E). The initiation is the preparation of a request. The execution is the actual performance of the service, around which the communication loop revolves. The evaluation is the assessment work that must be done before the service report can be accepted. All four communicative acts and three workflow loop tasks are examples of *workflow loop acts*.

The service relationship is fundamental, but it can be complemented with *delegation* relationships. In this paper, we will focus on delegation on the side of the provider, but delegation at the customer's side is possible as well. Any workflow loop act can be delegated to an *agent*, who then becomes the *performer*. However, as the provider, the delegating actor, the *principal*, still keeps a responsibility to the customer, that is, the service relationship itself is not delegated. From internal control theory, we derive the distinction in operational and control tasks. We define the functional role of principal to be responsible for the control tasks (initiation and evaluation), and the agent for the operational (execution part). The resulting control loop is very similar to the service loop, which makes it possible to view them as two types of communication loops.

Fig. 1 presents the extended workflow loop model. Notice that the agent has two executor roles, but there is a slight difference between the X-role of the agent in the service loop and the X-role of the agent in the control loop. From a control perspective, the agent's performance may consist of these executive tasks making up the service, but also of the conversations with the beneficiary (so his overall performance in the service loop is what counts as X in the control loop).

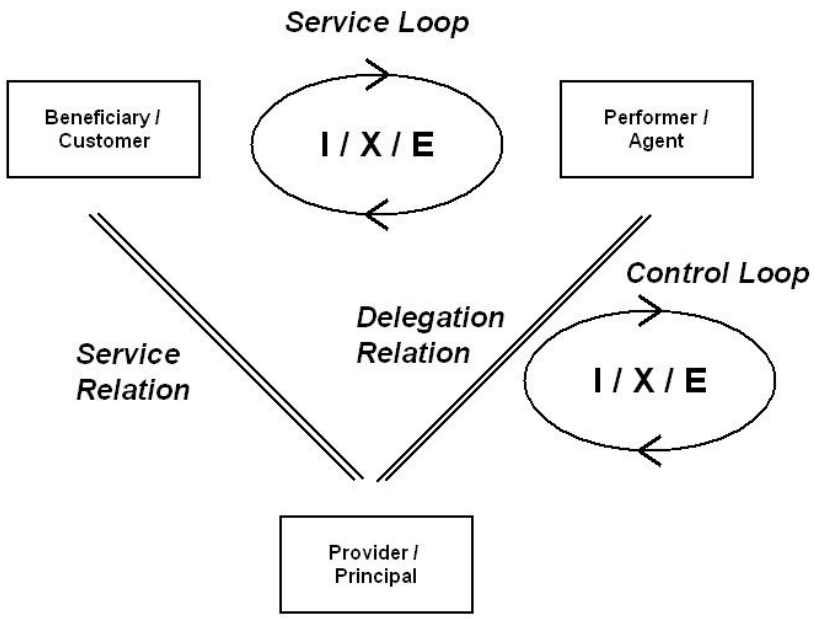

Fig. 1. The Extended Workflow Loop Model

3 Extended Workflow Loop Norms

Internet-age information systems are much more communication than computation systems. There are many applications that support complex collaborative communication processes, like discussion, group decision making, and group authoring. The semiotics of these systems are often much more complex than of traditional information systems, particularly because the intended semantics and pragmatics are not under the control of one single organization, and therefore often remain un(der)defined. This entails that often the meaning of information produced and responsibilities for system use and specification are not clear.

3.1 Norms for Organizational Communication

Effective organizational communication presupposes that the communicative partners agree on certain norms: not only syntactical norms on the language that is used, but also norms related to the semantics

and pragmatics (what are the intended and perceived effects of and on senders and receivers?) of the communication. For instance, a major customer of a company may expect a priority treatment and receive an immediate response to his request for repairs.

Not all organizational communication norms need to be explicit. Many norms are implicit or tacit knowledge: everybody is aware of their existence, but they cannot or should not be formalized [3]. For example, in the scientific community, there is a tacit norm that participants are prepared to defend the claims that they make, whereas a director in a multinational company may expect his staff to follow orders. If such norms are not observed, miscommunications may occur that can have serious effects on the efficacy of the organization.

Having said this, sometimes organizational communication norms do need to be made explicit, if they are to act as clear decision making rules for staff members. This may also be the case with breakdowns [20], for example when two communicating parties disagree on the meaning of a term or responsibility. Then, the rules of action that seemed clear turn out to be really norms that can be violated [15], and explicit discourse may be needed to determine the proper course of action.

There are many types of organizational communication norms. Our focus on workflow loop norms proper is motivated by the need to find relatively generic principles for inclusion into system designs. By making information systems more legitimate in this way, we contend that organizational communication can be significantly improved. Of course, other, more specific categories of communication norms may be needed as well. One additional class of norms, for example, concerns social norms defining the effectiveness and efficiency of internal organizational communication aimed at motivating or informing employees [8]. However, in collaborative situations, the workflow loop norms provide a solid foundation for such more refined normative analysis.

3.2 Workflow Loop Norms Implicit in LAP

One important source of stable workflow loop norms is internal control theory. This theory provides normative guidance in complex organizational structures, when there are delegated task structures which

allow agents to establish commitments on behalf of the organization. Delegating an activity does not mean that the responsibility for this activity is delegated as well. Instead, it introduces a control task for the principal that delegated the task to the agent. This involves communication: since in most cases, the principal responsible for that control task cannot personally observe the performance of operating tasks, she must rely on documentary evidence (evidence function). At the same time, to protect himself, the executing party (the agent) must be able to prove the completion of an activity (preventative function).

An extensive literature related to internal control theory exists, often drawing from accountancy research. For example, Chen [5] lists a set of principles like "An operational task and its corresponding control task should be segregated into two different organizational positions and into two different agents". Bons [4] notes that control tasks can be divided into two categories: control tasks that make direct statements about the operational tasks (such as witness reports), and control tasks that evaluate the resulting document and draw conclusions based on them. Many principles can be found in the literature, but more interesting at the moment is to see how they can actually be used to construct workflow loop norms.

In [19], we proposed an approach to analyze workflows using communicative norms based on such internal control theory norms. We identified the following list as a first approximation of the basic implicit norms underlying LAP:

1. For any activity, a distinction must be made between the *operational task* and the *control* task. These two tasks are executed by different roles and different subjects.
2. If an operational task exists, there should be a corresponding *initiating* control task and the operational task should follow the initiating task.
3. If an operational task exists, its corresponding *evaluative* control task should exist as well and should always follow the operational task.
4. The initiating task should contain a request for action from a role (initiator) independent of the role performing the task
5. The role issuing the initiating task (initiator) should be the same as the role responsible for the (evaluative) control task.

6. The initiating task should be closed with a commitment (promise) from the role performing the operational task.
7. The evaluative control task should be furnished by supporting documents. The supporting documents should originate from the role performing the operational task.
8. The evaluative control task should be closed with a performative statement from the role performing the evaluative control task.
9. The performative statement of the evaluative control task should be received by the role that performed the operational task.

Based on our own extended workflow loop model and principles from internal control theory, we then generalized and refined those implicit LAP norms. We formalized the basic concepts from our extended workflow loops as well as the norms, resulting in a formal ontology and a set of extended workflow loop norms expressed in terms of this ontology. In the next section, we present an adapted version of this ontology.

3.3 The Extended Workflow Loop Ontology

A normative analysis quickly becomes complex. To deal with this complexity, formal representation and reasoning can be of great help. Having introduced the concepts underlying the extended workflow loop, we are now ready to start our formalization. The formalization consists of two main parts: (1) a *formal ontology* to precisely define the meaning of the extended workflow loop concepts, and (2) a set of *extended workflow loop norms* based on these definitions. We present both the ontology and formal representation in the remainder of this section.

First, we present our extended workflow loop ontology, adapted from [19].

A **service** is a tuple <Service Type, Performer, Object, Beneficiary>, where *Service Type* is some predicate designating a service type. The *Object* is the object of value. The object can be immaterial. *Performer* and *Beneficiary* are actor roles with respect to the service.

A **service relation** is a tuple <Service, Customer, Provider>, where *Customer* and *Provider* are actor roles, and *Service* is some service as defined above.

The **service loop** for a certain service is a tuple <Service, Init, Exec, Eval, Acta, Facta>, where *Init*, *Exec* and *Eval* are tasks and *Acta* and *Facta* are conversations consisting of pairs of communicative acts (*Request* and *Commit*, and *Report* and *Accept*, respectively).

A **delegation relation** is a tuple <Delegation, Principal, Agent>, where *Delegation* stands for one or more of the delegated tasks or communicative acts that make up a workflow loop. *Principal* and *Agent* are Actors. For the time being, we omit what exactly has been delegated (that is, task roles and/or conversations).

A **control loop** for a certain delegation relation is a tuple <Delegation, Init, Exec, Eval, Acta, Facta>. The *Exec* task of the control loop of a delegation is usually done by the Agent of the delegation, whereas the *Init* and *Eval* tasks are done by the Principal, unless these have been delegated as well. Both the service and control loop distinguish the conversational roles of *Initiator*, *Executor*, and *Evaluator*, which are defined in terms of the workflow loop acts.

3.4 Formalizing Extended Workflow Loop Norms

The purpose of the ontology is to define precisely the extended workflow loop norms. In [19], we presented a rather comprehensive set of extended workflow loop norms. These formal norms follow from the implicit set listed in Sect. 3.2. Here, we take only one two (simplified) norms to illustrate the main contribution of this paper: developing an operational method for legitimacy checking of workflow schemas. Other norms can be addressed in similar ways.

XWL Norm 1:

```
∀s:service_loop(s.initiator=s.service.customer
∧ s.evaluator=s.service.customer)
```

This norm requires that the customer of a service (as defined in the service relation) is both initiator and evaluator of the service loop, and thus reflects a strong commitment to customer orientation.

XWL Norm 2:
$\neg\exists$s:service_loop(s.executor=s.evaluator)

This is a rather strong norm, that says that nobody should evaluate their own work. This principle is becoming increasingly important, given the current focus on transparency and accountability of bookkeeping practices.

4 Workflow Loop Schemas

To apply workflow loop norms, we need to distinguish between actual and deontic states. Norms indicate deontic ("soll") states: how the world should (not) be. However, equally important are the actual or proposed states on which the norms are to have their regulatory deontic effect. We call these actual states the *workflow situation*. Each workflow situation comprises one or more extended workflow loops. Each extended workflow loop is represented by two workflow loop schemas: a service loop schema and a control loop schema.

4.1 The Schema Structure

In order to model current or proposed *workflow loop situations*, we introduce the concept of *workflow schema*. Schemas can be used to organize knowledge that represents complex situations or objects in a domain [12]. Here, we use a schema to decompose a workflow loop into its constituent acts. To these acts, the various roles and subjects playing these roles are mapped.

Table 1 shows the basic structure of a workflow loop schema. The first row represents the *workflow loop acts*. There are seven possible acts: the init-task (to prepare the request), the request, the commit-act, the execution task, the reporting act, the evaluation task, and the acceptance of the results.

The second row shows the *domain roles* that carry out the workflow loop acts. These roles are often defined in the specific domain in which the workflows are carried out. The labels are thus domain-dependent.

The third row describes the *conversational roles* (initiator, executor, or evaluator) that perform the workflow loop acts. In general, there is a strong correlation between the workflow loop acts and the conversational roles that perform them. For instance, in DEMO the role performing the evaluative task is always the initiator. However, as our analysis of workflow loop norms has shown, such constraints often need to be relaxed in the complexity of real work situations. In many networked organizations, for example, it is not the initiator, but a separate evaluator role that performs the evaluative task, for reasons of efficiency or to implement checks and balances.

The fourth row captures the complexity introduced in the extended workflow loop: a *workflow loop role* can act as a customer, beneficiary, provider, principal, or agent (the agent being the performer). As many mappings to the other roles are possible, this separate row is justified. For example, in case of delegation, the customer role requesting a service is not necessarily the beneficiary.

The fifth row describes the *subjects* performing the workflow loop acts. In DEMO, for example, no specific constraints are imposed on which subjects perform the acts. From an internal control perspective, however, many constraints (i.e. prohibitive norms) are often demanded, such as that the evaluation of performance cannot be done by the same subject who has executed the work.

4.2 Modelling Workflow Loop Situations

In Table 1, we use the workflow loop schema to model a simple workflow loop situation: a baker promises his customer to bake a bread, upon request, and then bakes and delivers the bread himself. In this case the customer explicitly asked the baker to bake the bread. No preparation of the request was needed, so the init-task remains empty. This situation has been modelled using standard LAP semantics, with no explicit evaluator role distinguished. As there is no delegation, only the provider and customer roles need to be defined. It is clear that in this situation there are only two subjects performing the subsequent workflow loop acts. For clarity, these subjects have labels here similar to the domain roles that they play. However, normally there is not necessarily a one-to-one mapping between subjects and domain roles, as one subject can play more than one such role, for example.

The simple workflow loop can be modelled using a single workflow loop schema as follows[1]:

Workflow loop: Simple Pizza Delivery

WL Act	**Init**	**Reqst**	**Commit**	**Exec**	**Report**	**Eval**	**Accept**
Dom. Role	-	Cust.	Baker	Baker	Baker	Cust.	Cust.
Conv. Role	-	I	X	X	X	I	I
WL Role	-	Cust.	Prov.	Prov.	Prov.	Cust.	Cust.
Subject	-	#cust	#bak	#bak	#bak	#cust	#cust

Table 1. Using a Workflow Loop Schema to Represent a Simple Workflow Situation

Note that there are two customer-roles here: a domain role (somebody buying a bread) and a workflow loop role (the LAP role).

To illustrate how workflow loop schemas can also be used to model the more complex workflow situations typical of extended workflow loops, we now represent the scenario about the extended pizza delivery case described in [19].

The complex pizza delivery case

In the case of a pizza baker who originally bakes and delivers his pizzas himself, the communication between him and a customer can be easily modelled using a standard workflow loop (within a contract relation, but we will focus here on the baker as performer). Now the baker hires a boy to deliver the pizza to the house of the hungry client for him. Then there exists an agency relation between the baker and the boy: the baker plays the manager/principal role, the boy the employee/agent role. The workflow loop now seems distorted, since the new pizza delivery workflow loop performer is no longer one subject. Say the hungry client calls the baker on the phone. In an actagenic conversation, part of the workflow loop, the baker agrees to bake and

[1] Note that current LAP semantics are still unclear about the precise differences between and constraints on the roles. Our current mappings in the tables presented in this paper are therefore still tentative. However, structuring semantics in these workflow loop schemas is an important first step in identifying semantic unclarities or gaps. Here, we aim to demonstrate the principle of legitimacy checking, but do not claim that the examples are necessarily the best possible interpretation.

deliver a pizza. After calling the boy, the baker orders the boy to bring the pizza to the client. The boy takes the pizza, drives to the house, rings and starts a factagenic conversation in which the hungry client accepts the pizza, perhaps after having signed a note. The boy returns to the baker and reports the succesful delivery, possibly with handing over the note as evidence.

Note that there are now a control loop and a service loop, together forming the extended workflow loop. We therefore require two – related - workflow loop schemas to represent the complex workflow loop norms (including delegation) implicit in this case:

Service loop schema: Complex Pizza Delivery

WL Act	**Init**	**Reqst**	**Commit**	**Exec**	**Report**	**Eval**	**Accept**
Dom. Role	-	Cust.	Baker	Boy	Boy	Cust.	Cust.
Conv. Role	-	I	X	X	X	E	E
WL Role	-	Cust.	Prov.	Agent	Agent	Cust.	Cust.
Subject	-	#cust	#bak	#boy	#boy	#cust	#cust

Control loop schema: Complex Pizza Delivery

WL Act	**Init**	**Reqst**	**Commit**	**Exec**	**Report**	**Eval**	**Accept**
Dom. Role	Baker	Baker	Boy	Boy	Boy	Baker	Cust.
Conv. Role	I	I	X	X	X	E	E
WL Role	Perf.	Princ.	Agent	Agnt	Agnt	Princ.	Princ.
Subject	#bak	#bak	#boy	#boy	#boy	#bak	#cust

Table 2. Using a Workflow Loop Schema to Represent a Complex Workflow Loop Situation

These tables should be mostly self-explanatory after the previous introduction. The workflow loop roles in the service loop schema are distributed among customer, provider, and agent. In the control loop schema, there is now an Init-task (namely, the baking of the pizza) which has to be done before the baker can request the boy to deliver the pizza. The baker is the performer of this role, as the baking is not part of the principal-agent relation. In the example, we abstain from the difficulties added by more than one subject playing a particular role of some act. Such set-theoretical issues need to be addressed in future work, though.

5 A Method for Legitimacy Checking

The basic idea underlying the method is that the extended workflow loop schemas, as well as the workflow norms, can be defined as semantic networks in the form of conceptual graphs. The workflow loop norm graphs put selectional constraints on the workflow schema graphs, in other words, on the actual or proposed workflow loop situation. These norms define what schema elements are required or forbidden[2]. Legitimacy is checked by projecting the workflow norm patterns on the workflow schema definitions. If required patterns are matched with situations(i.e. have projections), and forbidden patterns have no matches (i.e. have no projections), no norm violations occur. The present set of workflow schema definitions, and thus of the workflow situation, is thus legitimate. Next, we first formalize the workflow schemas using conceptual graphs. We then present our method for legitimacy checking.

5.1 Conceptual Graph Theory

In Sect. 3, we formalized the norms by defining an ontology and norms using first order logic. We now formalize the concept of schemas, in order to provide precise semantics and be able to reason about their properties. To this purpose, we use conceptual graph theory. Two important advantages of conceptual graphs are that they allow for the efficient construction of generalization hierarchies of graphs, and that they can represent contextualized or nested definitions. These properties are needed to efficiently check normative knowledge definitions. We will give a brief introduction of conceptual graph theory, as it is relatively unknown. The theory is explained in much more detail in [17].

Conceptual graphs are constructed out of concepts and relations.

Concepts

[2] In a full normative analysis, privileges (norms that define *permitted* behaviour, would also need to be taken into account. For simplicity, these norms are not considered here.

A *concept* has two fields: a type and a referent field. It has the following format:

`[`*`Type`*`: `*`Ref`*`]`, an example being `[Customer: #John]`

The *type* field contains a type label that is part of a type hierarchy. The *referent* field designates a particular entity with the mentioned type. This field is optional: if not specified, the concept is considered to be generic,and is by default existentially quantified.

A referent can be an individual marker or a generic marker. An individual marker can be a number sign, followed by some constant, e.g. `#John`. A generic marker, denoted by `*`, indicates a generic concept. It may be followed by a variable identifier, e.g. `*x1`. This allows one to refer to a specific, but as of yet unidentified entity. These named generic markers are useful for cross-referencing concepts in graphs. A co-referent concept is indicated with a question mark, e.g. `?x1`.

Conceptual Relations

A conceptual relation links two or more concepts. Each conceptual relation has a relation type, surrounded by parentheses. It also has one or more arcs, represented by arrows, each of which must be linked to some concept. A dyadic relation has the following representation:

`[`$Type_1$`: `Ref_1`] -> (`*`R_Type`*`) -> [`$Type_2$`: `Ref_2`]`

Generally, the relations can be read, in the direction of the arrows, as ‘the *source concept* has a *relation* to the *destination concept*’. An example of a conceptual relation could be:

```
[XWL: #Pizza_Delivery] -> (Part) -> [Service_Loop]
```

which states that the extended workflow loop for pizza delivery has some (yet unspecified) service loop.

Conceptual Graphs

A conceptual graph is a combination of concept nodes and conceptual relation nodes. It can also consist of a single concept node. To represent such a graph, one of its concepts is chosen as its head. If more than one relation is linked to a concept, the dash symbol '-' can be used to separate the common concept from the rest of these relations. A conceptual graph is ended by a period. Many examples of these graphs are given in the remainder of this section.

5.2 Outline of the Method

The method for legitimacy checking of workflow loop norms is an adaption of the method used in [7] to match required and enabled web service functionality.

The method consists of the following steps:

1. Define an extended workflow loop ontology
2. Represent the workflow loop norms in patterns
3. Represent the workflow situation in workflow loop schemas
4. Calculate the match between situations and norm patterns
5. Interpret the matching results

Define the Extended Workflow Loop Ontology First, we define the extended workflow loop ontology as a type hierarchy, with accompanying type definitions. All entities followed by a '>' sign are supertypes of the indented entities that follow. The hierarchy and type definitions follow from the previous discussion.

The extended workflow loop type hierarchy

```
Entity >
   Actor >
      Conv_Role >
         Evaluator
         Executor
         Inititiator
      Dom_Role
      WL_Role >
         Agent
         Beneficiary
```

```
      Customer
      Performer
      Principal
      Provider
Comm_Loop >
   Service_Loop
   Control_Loop
Conversation >
   Acta
   Facta
Object
Relation >
   Del_Rel
   Serv_Rel
Schema >
   Comm_Loop_Schema >
      CL_Schema
      SL_Schema
Service
Speech_Act
Subject
WL_Act >
   Comm_Act >
      Accept
      Commit
      Report
      Request
   Task >
      Eval
      Exec
      Init
XWL
```

There is also a small relation type hierarchy, defining such roles as (Agnt), (Part), and (Conv_Role), which will be omitted here.

The semantics of the concept types are given by the following type definitions:

- Workflow loops revolve around the performance of services. A service is of a particular type, has some object, and is done by a performer for a beneficiary.

```
[Service: *x] -> (Def)-> [Entity: ?x] -
   (Obj) -> [Object]
   (Agnt) -> [Performer]
   (Ptnt) -> [Beneficiary].
```

Notice that such a (meta) type definition indicates the ontological properties that a concept *must* have. This definition may be contracted, by replacing the genus [Entity] by the defined type [Service] and then dropping the (Def) relation.

- An extended workflow loop consists of both a service and a delegation relation as well as a service loop and a control loop.

```
[XWL: *x] -> (Def)-> [Entity: ?x] -
   (Part) -> [Serv_Rel]
   (Part) -> [Del_Rel]
   (Part) -> [Service_Loop]
   (Part) -> [Control_Loop].
```

- A service relation is about a service between a customer and a provider; a delegation relation is about some workflow act between a principal and an agent.

```
[Serv_Rel: *x] -> (Def) -> [Relation: ?x] -
   (Obj) -> [Service]
   (Agnt) -> [Customer]
   (Agnt) -> [Provider].

[Del_Rel: *x] -> (Def) -> [Relation: ?x] -
   (Obj) -> [WL_Act]
   (Agnt) -> [Principal]
   (Agnt) -> [Agent].
```

- Both a service loop and a control loop are communication loops. Each of these loops has a communication loop schema. This consists of an Init-task, a Request-act, and so on. Each of these workflow loop acts is done by a domain role, a conversation role, a workflow loop role, and a subject. However, in our notation we do not distinguish a separate (Subject) relation. Instead, when applicable, we use the subject as an identifier in the other roles, represented by the referent.

```
[Comm_Loop_Schema: *x] -> (Def) -> [Schema: ?x] -> (Part) -
   [Init] -
      (Dom_Role) -> [Dom_Role]
      (Conv_Role) -> [Conv_Role]
      (WL_Role) -> [WL_Role]
   [Request] -
```

```
   ...
[Commit] -
   ...
[Exec]
   ...
[Report]
   ...
[Eval]
   ...
[Accept]
   ...].
```

- Each communication loop contains two conversations, an actagenic and a factagenic conversation. Although for completeness we define their semantics, in this paper we do not further address how conversations and larger communicational structures can be used in legitimacy checking.

```
[Acta: *x] -> (Def) -> [Conversation: ?x] -> (Part) -
   [Request]
   [Commit].

[Facta: *x] -> (Def) -> [Conversation: ?x] -> (Part) -
   [Report]
   [Accept].
```

Represent the workflow loop norms in patterns To represent the norms, we define two kinds of patterns: *required patterns* and *forbidden patterns*. Such explicit coding of organizational norms is common in workflow systems [11].

Each workflow loop norm is translated into a set of required and forbidden patterns. Each required pattern has a *selection condition*. This condition determines if a schema should be matched with it during the calculation of the match between workflow loop schemas and norm patterns. We call a required pattern *relevant* for a schema if the schema satisfies this selection condition. For instance, the required pattern representing XWLN #1, introduced in Sect. 3.4, is relevant to all service loop schemas in which a customer is involved.

Here we show how norms are represented as patterns for the extended workflow loop norms XWLN #1 and #2.

According to XWLN #1, *the customer of a service must be both the initiator and the evaluator of the accompanying service loop*. Relevant ontological concepts are: the service loop (schema), customer, initiator, evaluator, and subject. XWLN #11 has only one required and no forbidden patterns.

XWLN #2 said that the evaluator of a workflow loop cannot be the same as its executor. This norm has one forbidden pattern.

- *Required patterns*:

For XWLN #1, there is one required pattern #rp1:

```
[SL_Schema] -> (Part) -
   [WL_Act] -
      (WL_Role) -> [Customer: *x]
   [WL_Act] -
      (Conv_Role) -> [Initiator: *x]
   [WL_Act] -
      (Conv_Role) -> [Evaluator: *x].
```

For all service loop schemas, the customer, initiator, and evaluator roles must be played by the same subject *x.

- *Forbidden patterns*:

For XWLN #2, there is one forbidden pattern #fp1:

```
[Comm_Loop_Schema] -> (Part) -
   [WL_Act] -
      (Conv_Role) -> [Executor: *y]
   [WL_Act] -
      (Conv_Role) -> [Evaluator: *y].
```

This pattern states that the conversational roles of executor and evaluator may not be played by the same subject within any communication (i.e. service or control) loop schema.

Represent the workflow situation in workflow loop schemas Key to the definition of the workflow situation are the workflow schemas.

Auxiliary definitions, such as a list of the services and service relations also need to be defined, but will be omitted here.

There are two workflow loop schemas, each of which is a specialization of the workflow loop type definition given in the previous section. We have not represented domain roles, as they are not relevant in this case. In fact, *the* baker, *the* boy and *the* customer are regarded as subjects only. Instead of using the abstract subject notations #s1, #s2, and #s3 we use the more comprehensible subject identifiers #cust, #bak, and #boy in the graphs. However, in other cases, the domain role may indeed be important. Many formal norms, for example, are domain-dependent: a manager may, *an* employee must, etc. In future work, we will investigate the role of this additional role complexity in our schemas.

The service loop schema #sl1 of the case is:

```
[SL_Schema: #sl1] -> (Part) -
   [Request] -
      (Conv_Role) -> [Initiator: #cust]
      (WL_Role) -> [Customer: #cust]
   [Commit] -
      (Conv_Role) -> [Executor: #bak]
      (WL_Role) -> [Provider: #bak]
   [Exec] -
      (Conv_Role) -> [Executor: #boy]
      (WL_Role) -> [Agent: #boy]
   [Report] -
      (Conv_Role) -> [Executor: #boy]
      (WL_Role) -> [Agent: #boy]
   [Eval] -
      (Conv_Role) -> [Evaluator: #cust]
      (WL_Role) -> [Customer: #cust]
   [Accept] -
      (Conv_Role) -> [Evaluator: #cust]
      (WL_Role) -> [Customer: #cust]].
```

The control loop schema #cl1 of the case is:

```
[CL_Schema: #cl1] -> (Part) -
   [Init] -
      (Conv_Role) -> [Initiator: #bak]
      (WL_Role) -> [Performer: #bak]
   [Request] -
      (Conv_Role) -> [Initiator: #bak]
```

```
      (WL_Role) -> [Principal: #bak]
   [Commit] -
      (Conv_Role) -> [Executor: #boy]
      (WL_Role) -> [Agent: #boy]
   [Exec] -
      (Conv_Role) -> [Executor: #boy]
      (WL_Role) -> [Agent: #boy]
   [Report] -
      (Conv_Role) -> [Executor: #boy]
      (WL_Role) -> [Agent: #boy]
   [Eval] -
      (Conv_Role) -> [Evaluator: #bak]
      (WL_Role) -> [Principal: #bak]
   [Accept] -
      (Conv_Role) -> [Evaluator: #cust]
      (WL_Role) -> [Principal: #cust]].
```

Note that – contrary to the service loop – the Init-task is now performed by the baker: baking the bread is a necessary preparatory act for the baker to be able to request the boy to deliver it. The Eval-task of the baker could, for instance, consist of regularly checking with the customer if the deliveries are in time, either face-to-face in the shop or by phone.

Calculate the match between workflow loop schemas and norm patterns S is the set of all workflow loop schemas. RP is the set of all required patterns, FP is the set of all forbidden patterns:

S = { #sl1, #cl1 }, RP = { #rp1 }, FP = { #fp1}.

- Project all required patterns on all the workflow loop schemas s ∈ S. A schema is in RM iff it matches with all of its relevant required patterns:

RM = { #sl1, #cl1 }

- Project all forbidden patterns on all the workflow loop schemas s ∈ S. FM = the set of schemas matching *any* of these patterns:

FM = ∅.

Interpret the matching results If $\forall s \in S$: $s \in RM$ and $s \notin FM$, then the workflow situation is legitimate, otherwise it is illegitimate (or at least its legitimacy has not yet been decided). In the example, the situation is legitimate. If such a situation is illegitimate, the conflicting pattern(s) must be dealt with by redefining one or more of the workflow loop definitions. The proposed definitions must be checked in turn by running the calculation again. How exactly this interpretation process is to occur, depends on (1) the type of workflow loop definitions causing the violation, (2) the type of norm being violated, and (3) the meta-norms governing what should be done in case of violation. No uniform interpretation approach can thus be given. In future research, we intend to develop interpretation classifications to structure such norm conflict resolution processes.

6 Conclusions

In our information society, the quality of the communication in and between organizations is becoming a critical success factor. To design and maintain effective communication systems, we need more than communication modeling. We should also be able to check the quality of the communication models. Thus, there must be norms that define legitimate, i.e. meaningful and acceptable organizational communication.

In modern rational organizations and networks, not only the communication structures themselves must be legitimate, but also the processes in which these norms are generated. This means that communication norms may have to be made explicit, and become the subject of rational discourse.

These two considerations provided the motivation for this chapter on the role of legitimacy checking in communicative workflow loop design. First, we have shown an analysis of communication norms based on the Extended Workflow Loop model. Using this model to define a workflow loop ontology and accompanying norms, we have described a practical method of legitimacy checking. The method uses the notion of workflow loop schemas in which various elements of the communication workflow loops are integrated. It was shown how such a schema can be represented using Conceptual Graph Theory. This

makes it possible to delegate the norm checking to reasoning tools. Note that the norms themselves are not yet necessarily legitimate; for this the definition process of the norms should be embedded in a social process in which the communication structures and norms can be discussed and challenged by relevant stakeholders, if necessary [6].

One important application of the legitimacy checking method is communication diagnosis [18], which works in a bottom-up fashion. The goal of diagnosis is to model the current situation and to analyze actual or potential flaws by linking them to communicative norm violations. The diagnosis should result in recommendations for improvement. In the case of workflow redesign, the reengineering process description should indicate how a legitimate situation can be reached from a currently illegitimate situation by redefining workflow structures that violate the communication norms. This reengineering process itself must also be legitimate.

The current trend in information system development is a move away from detailed and formal methodologies [2]. Formalization is not a goal in itself; what is needed is rationalization. A more contingent approach is therefore needed. What is most problematic in current-day elaborated methodologies, is, in our view, the lack of attention to systematic involvement of users – as stakeholders – in collaborative system (re)design. Admittedly, user involvement and stakeholder analysis have been gaining prominence for a long time. Some approaches use brainstorming sessions, for instance. However, this is still far off from encouraging rational discussion. A rational discussion also allows participants to challenge the norms that underly the design choices. A legitimacy checker as discussed in this paper can be instrumental in such a process. If embedded in a carefully designed social interaction process, it can be an example of a useful application of formal methods in practice.

7 References

1. van der Aalst, W.M.P. Three Good Reasons for Using a Petri-Net-Based Workflow Management System. In *Proc. of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96), Cambridge, MA, November 1996*, pp. 179-201.

2. Avison, D. and Fitzgerald, G. Where Now for Development Methodologies? *Communications of the ACM*, **46**(1):78-82, 2003.

3. Baumard, P. *Tacit Knowledge in Organizations*. London: Sage, 1999.

4. Bons, R. *Designing Trustworthy Trade Procedures for Open Electronic Commerce*. PhD thesis, Erasmus University, Rotterdam, 1997.

5. Chen, K.T. *Schematic Evaluation of Internal Accounting Control Systems*, PhD thesis, University of Texas at Austin, 1992.

6. de Moor, A. and Jeusfeld, M.A. Making Workflow Change Acceptable. *Requirements Engineering*, **6**(2):75-96, 2001.

7. de Moor, A. and van den Heuvel, W.J. Making Virtual Communities Work: Matching their Functionalities. In *Proc. of the 9th International Conference on Conceptual Structures, Stanford, July 30-August 3, 2001*, pp. 260-274.

8. de Wit, B. and van Delden, M. Performance Analysis of Internal Communication: a Research Approach for Assessing the Quality of Policy and Motivating Communication (transl. from Dutch). In de Ridder, J.A. and Seisveld, K. (eds.) *The Quality of Communication in Organisations in Theory and Practice (transl. from Dutch)*, Amsterdam: Cramwinckel, 1996.

9. Dietz, J. and Van Reijswoud, V. *DEMO Modelling Handbook v2*, Delft University, 1998.

10. Kuo, D.C.L. and Smits, M. Performance of Integrated Supply Chains: An International Case Study in High Tech Manufacturing. In *Proc. of the 36th Hawaii International Conference on System Sciences, Hawaii, 2003*.

11. Liu, K., Sun, L., Dix, A. and Narasipuram, M. Norm Based Agency for Designing Collaborative Systems. In *Information System Journal*, **11**(3):229-247.

12. Luger, G. and Stubblefield, W. *Artificial Intelligence and the Design of Expert Systems*. Redwood City, CA: Benjamin-Cummings, 1989.

13. Lyman, P. How is the Medium the Message? Notes on the Design of Networked Communication. In Stephen, T. (ed.) *Computer Networking and Scholarly Communication in the Twenty-First-Century University*. New York: State University of New York Press, 1996, pp. 39-52.

14. Medina-Mora, R., Winograd, T., Flores, R. and Flores, F. The ActionWorkflow Approach to Workflow Management Technology. In *The Information Society*, **9**(4):391-404.

15. Orlikowski, W. and Yates, J. *Genre Systems: Structuring Interaction through Communicative Norms*. Boston: MIT, 1998.

16. Schäl, T. *Workflow Management Systems for Process Organizations*. Berlin: Springer Verlag, 1996.

17. Sowa, J.F. *Conceptual Structures : Information Processing in Mind and Machine*. Reading, MA: Addison-Wesley, 1984.

18. van der Poll, F., Weigand, H. and de Moor, A. Communication Diagnosis of a Financial Service Process. In *Proc. of the Seventh International Workshop on the Language-Action Perspective on Communication Modelling (LAP-2002), Delft, The Netherlands, June 12-13, 2002*, pp. 118-134.

19. Weigand, H. and De Moor, A. Workflow Analysis with Communication Norms. In *Data and Knowledge Engineering*, [in press].

20. Winograd, T. and Flores, F. *Understanding Computers and Cognition : a New Foundation for Design*. Norwood, NJ: Ablex Pub. Corp., 1986.