The Dynamics of Organizational Coordination

D.T.T. van Eijck and G.J. de Vreede
Delft University of Technology
School of Systems Engineering, Policy Analysis and Management
P.O. Box 5015, 2600 GA Delft
The Netherlands

Abstract

Coordination is a key issue in the design of organizational structures and processes. In traditional design methods, a way of modeling that captures both interactor communication and dynamic aspects, is lacking. Based on the dynamic modeling approach, we propose a body of concepts that meets these two requirements. We use a case study to introduce and illustrate these concepts. Furthermore, we formulate a way of transforming these concepts into a simulation model. We conclude by stating some unsolved problems and directions for further research.

1. Introduction

In our post-industrial society, organizations are faced with a constant need to adapt their structures, processes, and technologies in order to meet new demands from their environment, [5,16,19,20]. Organizations have to reorganize their current way of working in order to survive [7,8,14,29].

Reorganizing (in fact 'organizing') can be pragmatically defined as the process of dividing business processes into tasks and distributing (groups of) tasks and related responsibilities among one or more individuals or groups. In order to guide and monitor the business process as a whole these tasks, often carried out in parallel, have to be coordinated. Hence, a need to reorganize the current way of working implies a need to reorganize the way in which an organization achieves the coordination of its business processes.

(Re)designing coordination can be regarded as a very complex activity. There are many possible courses of action and many variables. Moreover, options for organizational change are difficult to test in reality, meaning that the effects of these changes cannot be known in advance. To overcome this problem, many different approaches have been suggested to support redesigning the coordination of business processes, see Van Meel et al. [23] for an overview.

We build on the work of Sol [28,29] who describes a process of problem solving that constitutes the core of a simulation based approach for (re)designing organizations and the coordination of their business processes. Crucial to this approach is the use of empirical dynamic models of the organization. First a model for understanding is developed that correctly describes the current situation at a suitable level of detail. Then, after analysis of this model, one or several models for design may be built: derivatives of the model for understanding that depict possible organizational changes. These models are valid (i.e. have predictive value) provided that they do not drastically deviate from the basic structure of the model for understanding.

To support the construction of both the models for understanding as well as the models for design, this paper describes a body of concepts for modeling organizational coordination. The next section first discusses the notion of coordination within organizations and known approaches to modeling organizational coordination. The concepts we propose for modeling organizational coordination are then discussed in Section 3. These concepts are illustrated in a case example of an organization that operates in a geographically dispersed way. Section 4 describes how the concepts from Section 3 can be translated into simulation models that support the quantitative analysis of the situation under investigation. The paper concludes with a summary of our findings and directions for further research in this area.

2. Organizational coordination

Malone and Crowston [21] state that 'coordination ... can occur in many kinds of systems: human, computational, biological and others.' The notion of coordination addressed in this paper concerns the structuring and adjustment of operational activities in organizations. We denote this as organizational coordination.

2.1 Defining organizational coordination

Organizational coordination is a subject that is widely described in literature on organization design. Simon [27] speaks of coordination as 'the process of informing each as to the planned behavior of others.' Galbraith [12] sees organization design as the search for coherence between the goals or purposes for which an organization exists, the people that do the work and the patterns of division of labor
and interunit coordination. Mintzberg [24] describes three coordinating mechanisms that are concerned with coordination: mutual adjustment, direct supervision and standardization. Thompson et. al [32] define coordination from a sociological perspective. More recently, Malone and Crowston [21] define coordination as managing dependencies between activities. They mention actors and (interdependent) activities as key concepts. Within the scope of this paper, we restrict to a definition of organizational coordination that focuses on the coordination of operational activities:

Organizational coordination is the mutual influencing of working processes of two or more organizational actors in order to attain a certain objective.

This definition is in line with the definition of Malone and Crowston, but as we focus on operational activities in an organization, we refer to working processes instead of activities and mutual influencing with a certain objective instead of managing. Three concepts are of importance here:

- Multiple actors (defined as organizational units that may take decisions)
- The mutual influencing of working processes
- The attaining of an objective

We chose for mutual influencing instead of mutual adjustment because the latter term is often used to describe only one specific coordination mechanism, Mintzberg [24]. The concept of attaining objectives or goals is important because coordination problems are often caused by the attainment of different goals by actors. By assuming that actors are able to exchange information about their work processes at any moment, they can carry out work in parallel and combine their products in the end. Actors may inform each other about the starting or ending times of their tasks in order to synchronize. Actors may issue instructions to other actors in order to get things done. We submit that the definition as given above is a good starting point for modeling organizational coordination.

2.2 Approaches to modelling coordination

The essence of modeling organizational coordination is not only modeling what activities actors perform, but mostly how they cooperate and influence each other's activities. In this respect we want to be able to model important organizational phenomena that relate to coordination like: parallel work processes, synchronization of work processes, and control (management) of work processes. We can formulate two requirements that have to be met by models describing organizational coordination:

1. They have to capture time-aspects, i.e. they have to be dynamic models.
2. They have to capture the communication between the actors involved in the coordination process.

In literature several modeling approaches can be found that seem to meet these requirements. The Petri net formalism [26] is one of the first approaches to modeling concurrency and synchronization. Based on the Petri net formalism, a substantive number of modeling approaches have been formulated, see e.g. (Sol and Van Hee, [30] Sol and Crosslin, [31]). However, two reasons make Petri nets less suitable for modeling organizational coordination. First, they are poorly communicable to people that are not trained in reading them. Second, Petri nets tend to grow very large and incomprehensible, if the modeled situation gets slightly complex.

A mechanistic view on organizational coordination is offered by diplans, a formal graphical language for expressing plans, see Holt [15]. The strong characteristic of diplans models is that they explicity the coordination effort in a certain situation. The disadvantages are like those of Petri nets: they are poorly communicable and quickly grow very complex and incomprehensible.

Task structures as proposed by Bots, [4] offer a means of modeling the activities (tasks) and their relationships of individual actors. However, coordination that involves two or more actors at the same time cannot be modeled straightforwardly, although the formalism offers a construct for modeling situations where coordination with other actors is required. Task structures have proven to be easy communicable.

Dur [9] proposes task structures from a different perspective: in his task structures, task can be performed by different actors, but all involve the same item (e.g. a form or product being processed). Because of this focus on items, it is not possible to describe coordination tasks in which no item is being processed. Furthermore, it is not possible to use single task structures to describe the coordination of business processes that involve more than one class of items. Like Rots' task structures, they have proven to be easy communicable though.

Up to now we have elaborated on organizational coordination in general and shown different ways of modeling it. In the simulation based approach we discussed in the introduction, two modeling steps are taken in order to arrive at models for understanding or for design, see Sol [28]. The first step is called conceptualization and results in a conceptual model. A conceptual model offers the concepts in terms of which the modeller describes a situation, see [4].

The second step is called specification, resulting in an empirical model (usually a simulation model). The specification activity concerns the transformation of the
conceptual structures into a simulation model and the addition of empirical data from the actual situation. We proceed in the next section by giving a set of concepts for the conceptualization task that meet the requirements described in this section and overcome the disadvantages with the other approaches we discussed. In Section 4 we will illustrate how these concepts can be transformed into simulation code.

3. Concepts

In this section we propose a body of concepts for constructing a conceptual model of organizational coordination. We introduce a set of basic building blocks and perspectives for modeling organizations, business processes and organizational coordination. Before describing the concepts, we introduce a case study that is used as an example to illustrate the concepts.

3.1 An illustrative case study

The case study concerns the Dutch telecom network supplier. The focus is on the effects of information and communication technology on distributed and mobile workplaces. By using simulation and animation models, we analyzed the coordination within a department of service engineers. We first describe the organizational setting before we introduce the concepts.

A service engineer solves problems that occur at clients of the telecommunications network. At the beginning of his workday, he phones to the department and collects the problems that have occurred and drives to the first client. He solves the problem and notes the amount of used materials and time on a service notice. The client signs it and gets a copy. If the engineer has no more jobs to do, he calls the department again to collect new assignments. Then he drives to the next client.

Some problems occur with this method. First of all, the phone calls of the engineers to the department take too long, which results in undercapacity at the phone desk and long waiting times. Second, the service notices are sometimes misunderstood because of poor handwriting. This can result in the inability to send a bill to the customer.

To overcome these problems, the service engineers, are now equipped with a portable personal computer, a modem and a small printer. At the phone base, a computer is installed that is connected to the central telecom computer which contains the problems to solve. An engineer does not have to call the office any more, he calls the computer instead. This way, he gets the assignments much faster. The service note problem is solved as well because the service notes (of which a copy is held in the computer) are printed instead of handwritten. Both the old and the new situation (including the coordination between service engineer and telephone desk) can be modeled with our body of concepts. We demonstrate this by taking the case study's new situation as an example. Since the the purpose of this case study is to illustrate the concepts we propose, we will not discuss the results of the case study in great detail. They are reported in van Eijck and Attema [10].

3.2 Building blocks: objects

We adopt a systems approach as the basis of our body of concepts. Not just only because is has proven to be a useful approach in organization science (Kramer and de Smit, [18]), but merely because it can be used as a frame of reference for developing analysis and modeling concepts, see Checkland, [6]. We follow Sol [28] in defining a system: ‘A system is a nested structure of entities’. We, however, do not use the term entity here because it is generally replaced by the term ‘object’ in the literature. Hence we define a system as a nested structure of objects.

We note that, according to Sol [28], a delineation of a part of the world as a system is a matter of choice and that this part of the world can be described by the attributes and actions of its objects, see also Holbaek-Hanssen et al. [17].

In order to work together in an organization, objects need to communicate. In the case study, the service engineers need to communicate with the dispatchers at the phone desk and indirectly, they communicate with the billing office. Hence, we see an organization as a network of communicating objects and therefore a system. The description of such a system is given by a description of each of its objects. The properties of an object are described by its attributes and actions. The attribute part contains data elements describing the state of the object. The action part contains the tasks and decisions that reflect the possible behavior of an object. We refer to Booch [3] for further properties of objects in the object oriented approach.

For our concepts we have to describe the relevant objects in organizations, how these objects interact and coordinate their activities, how they are part of the organizations' processes. These processes can be seen as systems. We define a business process as a sequence of activities carried out by interacting objects, to obtain an operational result. Hence, an organization comprises of a system of linked sub-systems, the sub-systems being business processes.

Within each business process we identify three kinds of objects: Nodes, Links and Items. A node can be either an actor (e.g. a service engineer) or a repository (e.g. a database) Both repositories and actors inherit the properties of node. When a node is an actor, its action part, apart from inherited actions, is not empty and it can establish state changes in the system. When a node is a repository, its action part is empty, apart from inherited actions, and it contains
data that can be examined or changed by actors. Repositories may serve as a coordination mechanism: the job database is the node between the dispatchers and the service engineers and serves as mechanism that coordinates the work of the engineers and the dispatchers.

An object that has a non empty action part can alter its own attributes. Altering attribute values of other objects must be done by means of communication, i.e. sending an item. An item contains information (in its attribute values) that triggers an action (or more precise: a reaction) in the receiving object; it does not carry out any actions itself. We use the term 'item' instead of 'message' because an item may represent information as well as physical matter. The arrival of physical matter at an actor may also trigger actions.

The construct that connects the nodes in such a way that they can exchange items is called a link. A link is specified by its attributes and may carry out actions related to the kind of technology it represents. It routes items from node to node. It may also change attributes in items (e.g. a link may behave erroneously and cause distortion). The configuration of the nodes and the links in the organization corresponds to the perception of the modeller of the analyzed situation. All objects that take part in a modeled system inherit the properties of one of these three basic classes. We state four specializations of the three basic object classes that are convenient in modeling organizations. They inherit from NODE and ITEM respectively. Together, the seven object classes form the basic elements of conceptual models of organizations and the coordination of their business processes. A formal description of the seven object classes is given in table 1.

Using these object classes as basic building blocks, we can define three perspectives on organizations and the way they coordinate their business processes. These perspectives are discussed in the next three sections.

3.3 The organization perspective

Having defined the building blocks, we now introduce the way of constructing an organization model for analyzing coordination and the way of representing it. In constructing the organization model, the first step is to set the system boundaries. Then, the relevant nodes, items and links have to be identified and described according to the object class definitions as given above. Finally, a graphical representation of the model has to be made. We illustrate this way of working by applying it to our case study:

Step 1: Set system boundaries and reduce problem: In our case study, we choose to model the service engineers, the dispatchers and the billing department. We do not model the clients that call in with their problems, but we assume that jobs arrive regularly from the central job database. We also leave out the systems that are repaired by the engineer; we assume a probabilistic repair time.

Step 2: Identify relevant objects in terms of actors, repositories, links, messages and products: The entries in table 2 show all objects identified in the case study. There are no products, all the items are messages.

Step 3: Describe objects in object classes that inherit from the predefined ones: In table 3, an object definition is given, of one example object of each predefined class.

Step 4: Make graphical representation of the organization model: For the graphical presentation of the model, we use the icons as given in Figure 1. Using this representation for the case study results in an organization model like figure 2.

<table>
<thead>
<tr>
<th>object class NODE</th>
<th>object class LINK</th>
<th>object class ITEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifying attributes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>actions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>receive item</td>
<td></td>
<td></td>
</tr>
<tr>
<td>process item</td>
<td></td>
<td></td>
</tr>
<tr>
<td>send item</td>
<td></td>
<td></td>
</tr>
<tr>
<td>change attribute value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>object class ACTOR</td>
<td>object class REPOSITORY</td>
<td>object class MESSAGE</td>
</tr>
<tr>
<td>inherits from NODE</td>
<td>inherits from NODE</td>
<td>inherits from ITEM</td>
</tr>
<tr>
<td>attributes</td>
<td>attributes</td>
<td>attributes</td>
</tr>
<tr>
<td>Identifying attributes</td>
<td>Identifying attributes</td>
<td>Identifying attributes</td>
</tr>
<tr>
<td>Status attributes</td>
<td>Status attributes</td>
<td>Information contents</td>
</tr>
<tr>
<td>actions</td>
<td>actions</td>
<td>actions</td>
</tr>
<tr>
<td>work process</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Formal description of object classes

854
Boots [4] also uses tasks and decisions to model organizational activities. However, he attaches the tasks and decisions to a single actor. Dur [9] did it the other way around and attached actors to tasks, but limited the process to only one item class. We go one step further and allow more items to flow through the process and also allow decisions to be carried out by a specific actor. We propose the following object classes:

Table 2: Objects in the telecom case study

<table>
<thead>
<tr>
<th>Object class</th>
<th>Inheriting from</th>
<th>Attributes</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engineer</td>
<td>Actor</td>
<td>Name, Status</td>
<td>Build connection, transmit data</td>
</tr>
<tr>
<td>Job dispatcher</td>
<td>Message</td>
<td>Number, Address, Description, Status</td>
<td></td>
</tr>
<tr>
<td>Administrator</td>
<td>Repository</td>
<td>Job database</td>
<td></td>
</tr>
<tr>
<td>Modem</td>
<td>Link</td>
<td>Speed</td>
<td>Build connection, transmit data</td>
</tr>
</tbody>
</table>

Table 3: Examples of object classes from the case study

The business process perspective

The organization perspective gives a rather static view on the objects that are concerned in an organization. We introduce the business process perspective to be able to model the dynamics of the organization concerned. We use the term business process to denote a sequence of tasks and decisions, carried out by one or more actors, leading to a certain operational result (e.g., serving a customer, making or selling a product, etc.). The conceptual model of such a business process is called a business process model.

Figure 2: Organization model of the case study

Step 1: Choose system boundaries and identify processes: Based on the system boundaries for the organization model, the system boundaries of the business process model can be chosen. Depending on the problem that has to be solved, certain processes may be included or excluded for further investigation. In our case study, we have to identify performance improvements due to a new coordination mechanism, the modem and database. Therefore the boundaries of the business process model are set to the entrance of the job in the system and the eventual billing of the job.

Sometimes it is hard to identify the actual processes in an organization. Starting from the organization model, we propose two methods of identifying processes for the business process model. The first method is to evaluate the flow of one item. In many cases, especially in the logistic and administrative world, a business process can be defined as the routing and transformations an item undergoes from the moment it enters the system to the moment it exits the system. In fact, the modeller sits on an item, watching the tasks and decisions that are carried out during the item's
flow. The modeller notes the pre- and postconditions and the operational results after activities are carried out. He can also identify the actors that are responsible for decisions and activities. The second method is to watch the activities of each actor. The activities that contribute to a business process should be connected to the activities of other actors. The link objects are good starting points for this kind of analysis. In this way, a chain of activities will appear. This chain will either end or cross a system boundary. Both methods require interviews with actors that take part in the actual situation. Using the first method, we identified the following tasks and decisions in the case study:

Step 2: Identify pre- and postconditions and used repositories per task and decision: We argued that organizations can be modeled as networks of communicating objects. We defined coordination as the mutual influencing of work processes of actors. Hence, to coordinate, actors must be able to influence individual tasks or decisions of other actors. Actors communicate through the exchange of items. This means that we need a mechanism of inter task communication. The theory of Petri nets gave us the idea to use pre- and postconditions to establish this. A task’s and a decision’s attributes contain the pre- and postconditions, and the action part contains the transition between pre- and postconditions. Preconditions must be met before a task can be executed, postconditions are met after a task is executed. Fulfillment of the preconditions may require items that are available outside the actor, for example in a repository. The postcondition may require that items are sent to a place outside the actor. The transition may account for the altering of attributes or the transformation of an item. The transition between pre- and postconditions may implicitly model the objective that is attained by executing the task. The construct of pre- and postconditions and transitions for executing tasks is used to model inter task communication. In this way we are able to model synchronization, parallelism and control, the three important coordination phenomena we put forward in section 2.2.

In the case study, an example of a precondition is that at least one job should be available in the local job database before a service engineer can do a modem session. The postcondition of doing a modem session is the fact that a service engineer can go to a client. Apart from the pre- and postconditions, we have to identify the repositories that are used to communicate and store items. The result of this step for the case study is already visible in figure 2.

Step 3: Identify decision rules or probabilities per decision: The result of a decision may be expressed by a probability or a decision rule. The decision rules can be of the form IF condition THEN decision. If no exact rules can be identified or if the decision is based on a probabilistic value, a stochastic variable may be used for the decision.

Step 4: Make a graphical representation of the business process model: We use the symbols as depicted in figure 3 to make a graphical representation of the business process model. The arrows can be enhanced with probabilities or the outcomes of decision rules, the tasks and decisions may be combined with the repository symbols of Figure 1 to denote the location of the items that are used while executing a task or decision. The actor should always be noted in task or decision symbols. In our case study, the service engineer is part of a business process, carried out by the service department of the telecom company. This business process can be depicted by the business process structure of figure 4. Note that we added the repository symbols of the organization model to give a better insight.

Figure 3: Process structure icons

We explicitly note that ending up with a business process structure that is a good representation of the real situation in an organization, is not as simple as given above. During the process of designing the business process structure, a number of reductive steps have to be done in order to keep complexity within manageable bounds. It is very important to realize that the choice of which processes to incorporate into the business process model should be problem driven.

3.5 The single actor perspective

The third perspective on the conceptual modeling of organizational coordination is the behavior of the individual actors and their role in the business processes. We use task structures as proposed by Bots [4] to model the work processes of actors. As an example we take the service engineer of the case study. We identify the following steps:

Step 1: Identify and draw task structure per actor: The work process of a service engineer is carried out every day he is at work. He gets his jobs from the job database (via his carphone and modem) and drives to the clients to solve the problems that are listed in the job description. This can be represented by the task structure of figure 5.
Step 2: Identify time delay and variable assignments per task: In order to capture the dynamics of the situation, a time delay has to be specified for each task. This time delay may be deterministic or probabilistic. Also for each task, the variable or attribute assignments have to be determined. For example, the task “input billing info” has a time delay that is determined by a normal distribution and assigns values to specific variables of the job that is done (e.g., used parts, time consumed).

4. Simulation

In this section we demonstrate how the modeling concepts of the previous section can be translated into simulation models. Such simulation models can be used to analyze an organization’s business processes and the coordination required for these processes. Although outside the scope of this section, we mention that for such a quantitative analysis, empirical data such as process times, capacities of actors, and interarrival times of items, also have to be incorporated into the simulation model. A simulation model is constructed in three steps, according to the three modeling perspectives: the organization perspective, the business process perspective, and the single actor perspective. We start this section with a brief description of the simulation language we used.

4.1 Concepts of the simulation language

In the case study we used SIMAN/Cinema, a process oriented language that allows processes to be executed in parallel, see Pegden et al. [25]. Strong features of SIMAN/Cinema are its possibility to separate empirical data from the structure of the simulation model, its support for statistical analysis and its animation facilities. A simulation model written in SIMAN/Cinema can be thought of as a collection of items flowing through a piece of simulation code. Items can be created and disposed of. Items can be sent to other pieces of code by conditional branch statements. Code that is restricted to one physical location can be assigned to a station. Items may be routed between stations. Stations may contain queue/resource combinations that act on items. For the construction of a simulation model from our modeling concepts, we use the SIMAN/Cinema statements given in table 4. There are many more statements but only the ones stated in table 4 are necessary for our purposes.
4.2 The basic structure: the organization model

For each object in the organization model we propose a conversion to the Siman/Cinema language. We do not specify the item construct as they can be simply translated into Siman entities with attributes.

**Actors** can be modeled with the STATION statement and a QUEUE/RESOURCE combination. An actor is seized by an incoming product or message and performs (part of) its work process. An actor has only limited capacity, so the items will be queued up if the actor is busy. This results in the following code for an actor:

```
ActornameST STATION, Actorname;
QUEUE, ActornameQ;
SEIZE: ActornameR;
BRANCH, 1: ALWAYS, ActornameWP;
ActornameRA RELEASE: Actorname;
BRANCH, 1: ALWAYS, ActornameLI;
```

In this example, Actorname stands for the name of the actor. This name can be extended with a Q to denote the queue, with an R to denote the resource, with WP to denote the address where the code for the actors work process is located, with RA to denote the address where the item returns after the work process has been accomplished and with LI to denote the address where the link interface is located. This construct is more complex when an actor shows autonomous behavior, because the actor should then be kept 'alive', by a control item (see van Eijck and de Vreede, [11]). The repository construct can also be modeled by a station. The only task of a repository is to receive and supply items. This can be done by the following code:

```
RepnameST STATION, Repname;
BRANCH, 1: IF, Supply,
RepnameSP IF, Request,
RepnameRQ:
RepnameSP QUEUE, RepnameQ: DETACH;
RepnameRQ BRANCH, 1: IF, NQ(RepnameQ)==0, RepnameLI;
ELSE, RepnameRQA;
RepnameRQA SEARCH, RepnameQ: Condition;
REMOVE, J, RepnameQ, RepnameLI:DISPOSE;
```

Because items are not placed in any order in a queue, the queue must be searched first to find the item that is requested. The RepositorynameLI address points to the place where the link interface is located. After having completed a work process, the actor may send items away to links objects. An actor, however, may have more than one link (one for each actor or repository it is connected to). So in one way or another, the destination of the item should be determined. This can be done anywhere, so the easiest way is to reserve an attribute in the item where its destination is stored. This may be altered during a work process or somewhere else. It may also be stored in a global variable or determined instantly when sending the item. The following code can be used for sending an item to a link object:

```
ObjectnameLI ROUTE:O, Destinationexpression;
```

The duration is zero, any time delay in transport is arranged for by the link object. The destination can be determined from the item, from a variable or local expression or can have a standard value. The destination **must** be a link object. The **Link** object is modeled by the following code:
This construct simulates the possible working or failure of the link object. The LinknameDELAY variable stands for the time it takes to send an item to its destination (note that a link has only one origin and one destination). If a link is inactive, the item waits (LinknameDELAY2) and tries again (this construct can be replaced by other mechanisms, e.g. the item can be disposed if it's not sent). The actual failure of links can be generated by stochastic generators.

With these concepts it is possible to simulate the basic structure of the organization model. We have not yet specified the way in which the tasks of actors are performed. This is done in the following section.

4.3 Adding dynamics: actor behavior

The business process to be modeled is reflected in the tasks that are carried out by actors. This means that the work processes of the relevant actors should be specified. In an actor, the item is sent to the initial task or decision of a work process by the following code:

ActornameWP BRANCH,1: ALWAYS, Name of initial task or decision;

This may also be a conditional branch, determining the initial task from the attribute values of the incoming item. It might even be that items have to be copied to accomplish a workprocess. We now need simulation code for the task and the decision objects. Tasks have two properties: they take time and they may influence attribute and variable values. Decisions also take time and they may influence the sequence of tasks to be performed. A task is modeled by the following piece of simulation code (note that this code always belongs to an actor STATION):

Taskname ASSIGN: Variable1 = Valuel;
VariableN = Valuen;
Attribute1 = Valuel;
AttributeM = Valuen;
DELAY: TasknameDUR;
BRANCH,1: ALWAYS, Nexttaskordecision;

The assignment of attributes may also include a destination attribute that determines the flow of the item after the work process is completed. In case of a task that has no successor tasks, the last line of code in this construct is replaced by:

BRANCH,1: ALWAYS, ActornameRA

A Decision can be modeled by the following piece of simulation code:

Decisionname DELAY: DecisionnameDUR;
BRANCH,N: IF,WITH, and/or ALWAYS constructs;

In the branch statement, items may jump to other tasks and decisions according to the decision rules or probabilities specified in the decision object. N represents the maximum number of items that exit the decision for one incoming item.

For establishing coordination that cannot be achieved by item routing, we have to specify the pre- and postconditions for the tasks and decisions in the work processes. Preconditions may be expressions that operate on global variables or attributes. We model this by assigning values to attributes and variables that may be tested later on (the conditions are implicitly specified by the values of attributes and variables). The code for the task and decision object needs to be extended with code for precondition testing by using a simple conditional branch statement like the following (note that this code precedes the specified code for tasks or decisions, so the labels Taskname and Decisionname are replaced by TasknameA and DecisionnameA):

Taskname BRANCH,1: IF, Preconditions met,
TasknameA: IF, NOT Precond. Met,
TasknameWT;
TasknameWT DELAY: Duration of test cycle;
BRANCH,1: ALWAYS, Taskname;
TasknameA ........ Rest of code for task or decision

5. Conclusions

The concepts we have presented in this paper, have turned out to be sufficiently powerful to model the business process and its coordination in the case study. Other experiences support these findings, see de Vreede and van Rijck [33]. We extended object orientation with steps and techniques for modelling coordination. We demonstrated how our concepts can be successfully translated into a simulation model. Similar translations can be done to other languages like Simula [2] and its simulation library, DEMOS [3], GPSS [13], and Simscript [22] as these simulation languages use concepts that are similar to the Siman Cinema approach. This means that our modeling concepts are not restricted to the Siman/Cinema language.
Some situations are still hard to model and need more powerful concepts. Especially in the single actor perspective we can think of situations that cannot be modeled in a straightforward manner. Two examples:

- **Interruption:** Interrupting an actor, for example when a telephone rings, is still hard to model with our set of concepts. This problem requires adaptation of the actor (or the node) object.
- **Parallel task execution** by one actor (e.g. receiving a telephone call and writing a note) is still impossible. Using the task structures as proposed by Bots [4], the actor is restricted to executing one task at a time.

We believe that the concepts can be further extended to a method for modeling business processes. Using the concepts within a design methodology, combined with automated support tools, the modeller can concentrate on the conceptual modeling of a specific situation and leave the making of an empirical model to an automatic model generator (that can do this job much faster). The goal of further research is therefore to come up with an integrated approach to address improving organizational coordination. This involves three directions for the coming two years:

- Developing concepts that solve the above problems.
- Developing a set of modeling guidelines (a design methodology) for employing the presented body of concepts.
- Designing and developing automated support for translating the concepts into executable simulation code.

6. References