Towards A Model for Understanding Cooperative-work in Developing Information systems

Lahouaria Bendoukha

Hamburg University, Computer Science Department
Working Group for Software Development
Vogt-Koelln-Str. 30
22527 Hamburg, Germany
Bendoukh@informatik.uni-hamburg.de

ABSTRACT
The understanding of the organizational constraints and working practices within projects is a requirement in developing information systems for use in an organization. Our purpose is to improve particularly the understanding of cooperative work in organizations. Two objectives are considered: (a) taking into account the three main dimensions of a cooperative work: C*ollaboration, Coordination and Communication* (b) taking into account the integration of both technical and social aspects in software. According to those objectives, the model we propose characterizes a work organization in terms of network of dependencies among entities which we define in this paper. The basic idea consists in capturing the cooperation aspects by keeping track of entity dependencies, and by controlling entity change propagation through these dependencies.

Keywords
Information systems, cooperative work, requirements analysis, organization work, entity dependency, entity versioning

1 INTRODUCTION

A cooperative work allows several users to cooperate around a common task through a shared space. This general definition is differently specified according to several authors. The most important fields considered in the literature are: the multidisciplinary field computer supported cooperative work (CSCW), business processes and information technology. Other fields are also investigated in the literature: information systems, databases, tool integration and configuration management. There has been an increasing attention on supporting human activities characterized by a high degree of cooperation. Technology mediated workgroups systems are sometimes called CSCW systems or groupware. In general, the cooperation involves three interrelated aspects, i.e., communication, collaboration and coordination.

Research in software engineering and in information system development have increasingly recognized the need to understand work practices in order to explore and discuss with the organizational users the different ways in which computing systems can help improve their work processes. They have been interested in particular into complex cooperation contexts in organizations. Cooperative work techniques become very important in organizations as well as in the information systems community. The development of information systems is itself becoming performed in a cooperative manner.

At the same time, research on *ontology* is becoming increasingly widespread in the computer science community, promising solutions to the major problem in the evolving world of the New Economy which is the difficulty of establishing a partnership among different enterprises and making them sharing a common understanding of a given application domain.

In our research group researchers have studied questions related to the new requirements arising for software engineering when considering complex cooperations in organizations, as is demonstrated by the examples of hospital and bank information systems research projects[1] grounded in the tradition on STEPS (Software Technology for Evolutionary Participative Systems design)[2] and application-oriented systems Tools&Materials [3].

In those projects, complex cooperation contexts in organizations are characterized by joint tasks[5]. Joint tasks are understood as tasks in an organization consisting of a large number of individual tasks and responsibilities. They are performed by workers from different departments and professions in close cooperation. The necessary coordination takes place through explicit or implicit arrangement and has to provide the space for situative adaptation.

This work emphasizes more on coordination aspect in complex cooperations. That means that the understanding of the application domain is from the point of view of task coordination which manages dependencies among activities.

Since crucial point in using cooperation techniques is the transition from the current work situation to the future system, we believe that analyzing only joint tasks is not enough for outlining consequences for the design of the future system, which should support the three well-known cooperation dimensions: collaboration, coordination and communication aspects [4]. Indeed, the cooperation techniques proposed in the topic of this research project cover the coordination between activities by making explicit also articulation activities. We think that if we take into account requirements dealing with information integration, and not only tasks, several forms of cooperation become invisible by means of such techniques.

In this paper, we investigate the need to go further and to enrich the already proposed cooperation techniques based on joint tasks [5] by giving more semantics and improving thus, the answers to the crucial question “why” about activities-dependencies.

An ontological model providing modeling constructs and techniques for capturing not only coordination but also information sharing and communication evolving the concerned persons too is needed.

The remainder of this paper is organized as follows. Section 2 introduces the main problems encountered in describing cooperative work within a software development methodology. According to new objectives, we discuss in this section the limitations of cooperation pictures and purpose tables used as cooperation techniques in research projects in our research group. Section 3 introduces our approach to describe more knowledge dealing with cooperative work than when doing with the techniques presented in section 2. In section 4 we define the core of a user-customizable strategy change propagation model dealing with, not only the consistency problems by update, but gives also the concepts and techniques to understand the cooperation. A brief illustration is also given concerning the cooperative work in the university lecture-planning application. We then conclude with perspectives for this work.

2 DOCUMENT TYPES FOR COOPERATIVE WORK

The aims of STEPS as well as WAM approaches lie in evolutionary and software development, based on a cyclical process model, in the support of a participative communication and learning process for the developers and users [6]. User participation here essentially means integrating users actively in the development and further development of software [7], and evolution of system development means improving the cooperation between developers and users [6].

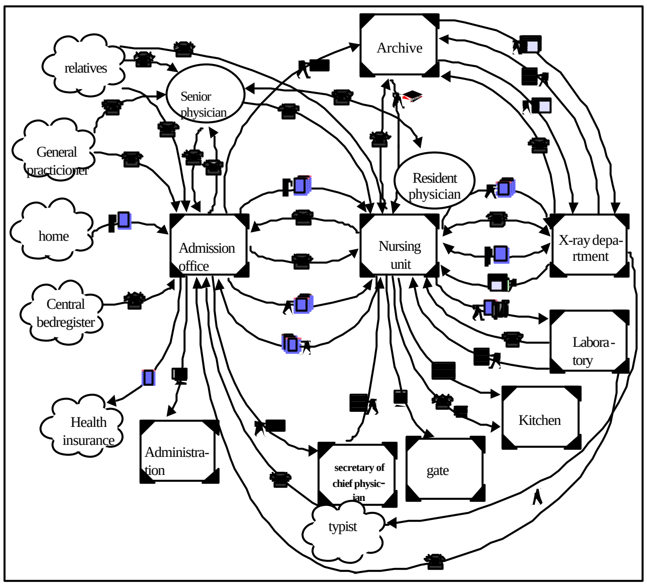
The basic principle is then the integration of both social as well as technical aspects. Analysis and design systems is accomplishable by application-driven documents such as scenarios and glossaries which are successfully used for an individual workplace in different stages of the development process.
They are also called application-orientation which means that analysis, design and construction of software is firmly based on the tasks and the way of dealing with them in everyday work situations [8].

Starting with the hospital domain, a good example is the „admission of a patient“ in a hospital which arises a lot of times each day. It involves administrational and organizational tasks as well as medical and caring work with the patient [6].

When the cooperative work was analyzed it became necessary to extend the former repertory of document types [5] such as Scenarios and glossaries, which describe the individual activities and tasks within cooperation. The overall perspective was missing. To bridge this gap, Cooperation Pictures provide a graphical visualization of concrete cooperation in order to accomplish a joint task. They portray the participating workplaces or roles and the cooperation between them by using every-day pictograms or graphics linked by arrows (see. Fig. 1).

With this technique the purpose of the exchanged documents or of the explicit coordination processes could not be understood. This is why Purpose Tables as a representation technique are also needed [6]. In this table, the cooperative task is concerned with "Who –does What - with What or Whom - for What purpose".

The focus of this table is to identify the different purposes or implications of each individual task (see Fig. 2).



Places and roles are visualized by symbols with names. Cooperations are represented by annoted arrows. In the hospital context, different kinds of cooperation were identified: the delivery of documents by the hospital staff, phone calls, data exchange via computer and the patient making his/her way to the different units of the hospital.

Fig.1 Cooperation pictures "Admission of a patient" in the hospital domain (from [5])

Some Single Activities of an Order Entry	**Purpose/Implications**
Physician writes the order on the physicians order form	It is documented who ordered the test at what (forensic, quality assurance). To kick on the implementation of the test
Physician puts the order entry sheet in the nurse’s mail basket.	Nurse is alerted that she has to act. She knows what is planned with her patient.
Nurse enters patient’s name, other relevant data and the type of test on the order entry sheet	Nurse prepares the order entry sheet in order to believe the physician of such burdens.
Nurse enters the test with pencil on the patient’s flowsheet.	It is documented for every member of the care team and physicians when the examination was ordered and to which further examinations he is scheduled.

Fig. 2 A part of the Table of the registration for an X-Ray examination (from [6])

I think that cooperation pictures and purpose tables as technique for cooperation description do not give much support for teams at the level of ambition of their approaches STEPS and WAM they are heading for. They give a very abstract view, where the joint (cross-departmental) tasks are described independently from each other. Purpose tables describe particularly the cooperation between persons in the sense ”who do what with whom/what and why” but for each individual task separately. It is clear that is difficult and very tedious to describe with a such static structure the complex and evolutionary networks of knowledge about an organizational work.

Cooperative Work Properties in the University Lecture-planning Application

We have identified more real examples from the application of *University lecture-planning* [9]. The process is based on two running processes which must occur in different time. The first process concerns the lecture-planning for the after the next semester delivering a temporary lecture-table and the second process for the next semester which provide a definitive table. Eventual updates are possible. The process of a semester planning is characterized by a great part of a cooperative work. The planning as well as the cooperation occur on four levels:

a. Each lecturer in the department introduces his (her) proper Courses for a semester. The involved persons are all informed of the contents, goals and the organization aspects (time, place,...)
b. The independent planed courses are introduced in the department lecture-planning.
c. Department lecture-plannings are themselves introduced in a common institute lecture-planning.

d. The common university lecture-planning which contains all courses in all institutes is elaborated in this higher level.

The group of persons working together represent a multiple cooperation. They cooper not only in a same level but between the four levels too [9], for instance: the responsible of institute lecture-planning, the responsible of department lecture-planning, institute counsel, department director, department secretariat, department lecturers, responsible of institute rooms /time, etc...

The necessary planning documents concerning e.g., the short-term lecture-planning which are created during the elaboration of the lecture planning are: institute lecture-table (IN), department lecture-tables (DE), commented courses catalogue (CCC), capacity report, etc.
During the elaboration of the institute lecture-table an incalculable number of versions for both institute and department lecture-tables are created, which are distinguished just with their update date. It is not clear when a new version of a document is created and when it deals just with a change of date. This induce to serious misunderstanding problems because we need often to work on different versions simultaneously. All planning documents are permanently updated. The decisions are took from responsible persons respectively the functional roles.

An ulterior change at the level of the lecture from a lecturer conduces to series of changes which represent a chain of reactions propagated on all levels of planning. Changing e.g. a title of a lecture must be also executed on department and institute lecture-tables as well as eventually on the capacity report. The change propagation operations appear more tedious when lectures must be deleted because of an overcapacity.

For simplicity reasons, we don't represent in this paper the cooperation pictures nor the purpose table. According to the scenario of the task dealing with the elaboration of the university lecture-planning table, we understand that an important requirement that involves the application objects, is that they should be time-stamped. This is a consequence of the fact that, during the evolution of the application model, objects may be created, terminated or changed, without forgetting the former states of these objects.

An other property of those application objects involves causal dependencies between the several objects. For example, the termination of one object causes the termination of all activities which specify operations on this object. A creation, termination or change of an object may cause another object to be created, terminated or changed.

As cooperation requirements we identify many questions which stay open such as:

- What would be the effect of a task completion on the object on which the task acts and vice versa?
- What are the pertinent states of accomplishing a task or/ and of the object which need to be notified?
- Who are the concerned persons to be notified?
- A complex object may participate in several tasks and a task could act on several objects. A person may carry out a task within more than as a functional role he (she) possesses, and a task could act on several objects. What consistency problems may arise?
- What are the consequences of eventual change at the level of functional roles for a given person ?
- What would be the effect on accomplishing certain tasks or on the change of communication with other persons ?
- Which objects are shared in accomplishing a given task, which one are been changed by other tasks and which are necessary for the continuity of a task carry out by another person?
- Which persons are concerned in accomplishing a given task?
- When a task is started by one person and taken over by another (which is frequent in the (hospital), how to make the delegation of assignment or responsibility explicit?
- We describe what would be done to all patients but there is no mean to describe the particularities for a

patient?

- How to deal with the above requirements knowing that it's furthermore, very hard to determine how long it will take to complete a task?

The major limitation of cooperation picture and purpose table techniques is that they seem to describe *snapshot* knowledge, i.e., they reflect information valid at a certain point of time, but lack the ability to preserve the history of the information. An evolving information system, on the contrary, must be *conservative* in the sense that it should not forget anything. We need modeling constructs and techniques for capturing not only coordination but also information sharing and communication between the concerned persons too. Taking into account Information sharing beside tasks coordination will improve the understanding of the daily cooperative work and will give more details issues concerning the future system.

In this purpose we try to consider explicitly aspects dealing with human intervention and the objects dealing with the task. The basic idea of our approach consists in capturing the cooperation by keeping track not only activities-dependencies but also object- and person-dependencies. The evolution of those dependencies are taken into consideration by controlling *change propagation* through the entities. We use techniques for change propagation from versioning research area. Indeed, some common properties to both versioning and cooperation modeling are classified in [10], bringing out implicit cooperation support in version modeling.

3 TOWARDS A META-MODEL FOR THE REPRESENTATION OF COOPERATIVE WORK

What is an Ontology ?

An *ontology* describes concepts and relationships among them, related to a particular domain, task, or application [11]. Ontologies can be used to share knowledge about specific aspects of the real world between individual agents in an unambiguous way. Since information integration is a major area for ontologies, the *portion* of the meta-model we propose in this paper is considered as an ontology providing a set of generic concepts.
A detailed description of the complete meta-model will appear in a next paper. It deals, in addition, with intentional aspects such as *belief*, *intention*, *goal*, etc. which we must take into account.

Collaboration Aspect as a Key for Understanding the Cooperation

We focus on objects and the operations on these objects. The support of object evolution and the eventual resulting change propagation is an important issue when supporting cooperative work. We think that starting from the collaboration aspect, if we master the support of object evolution, we will then succeed to describe easily also the other cooperation aspects. In other words, the same techniques provided for object evolution support allow to explore not only the collaboration aspect but also the possible relationships crossing the coordination and the communication aspects.

In order to analyze work organization, we identify three views according to three types of knowledge: objects, persons and tasks. The work organizational is then characterized in terms of the network of three types of networks according to the three views. The structures that are provided from the conceptual "ontologization" allows to guide the knowledge acquisition process. During a cooperative construction this will facilitate the negotiation.

Person
Person is an active entity, able to make decisions and carry out tasks by performing activities in order to change the state of affairs.
A *person* is identified by his (her) name. Each person has his (her) proper privileges and obligations to achieve a task according to the *functional role* (s) he (she) possesses (because we consider only roles of persons, we omit in the rest of the paper the term functional).
A *role* is then a characterization of the activity of persons according to their task. *Persons* are then characterized by the role(s) they possess. "Student" in a University application, "Doctor", "Nurse", "Client" in a Hospital application, "Client" in a Bank application or also "Designer", "project manager" or "administrator", "Programmer", "Tester", "Quality assurance engineer", "Maintainer" are examples of functional roles characterizing persons.
In developing an Information system, roles are fundamental to identify the different categories of persons (users) with their responsibilities, privileges and access rights. Persons evolve in their work. This is why, the acquisition (and the loss) of roles from persons evolves also with the time. That means that the responsibilities, privileges

and access rights for a person could change which make the relationship between a person or roles also evolving. Cooperating *Persons* are organized according to some structure with relationships, and tasks assigned. The responsibilities of each person depends on his (her) functional role (according to a given *task*) or composite roles (according to e.g. the *organization*)

Tasks

A *task* is the obligation, the privilege or the purpose perceived for a *Person*. A given person is obliged or allowed to perform tasks according to the role he/she possesses. They are classified into functional roles [12] by a collection of tasks for which a person or a group of persons is responsible. In the university lecture-planning application, the integration of the different lecture planning delivered from the different departments, into a single institute planning, is an example of one institute headmaster tasks. Affecting different tasks to a group of designers is an example of a project manager task. *Tasks* are achieved by performing (person) activities. The operations which are executed to perform activities describe the part of the functional meaning in a task.

Objects

An object is an entity which carry on information(s). It refers to an entity in the application domain or to its equivalent modeled in computer [12]. The objects are for example a form (a document) in an office, a bank account, an agreement (contract). They are characterized through attributes and operations.

The entities: person, task, object are themselves related via dependency relationships (see Fig.3): "acts on", " is responsible of", "carry out".

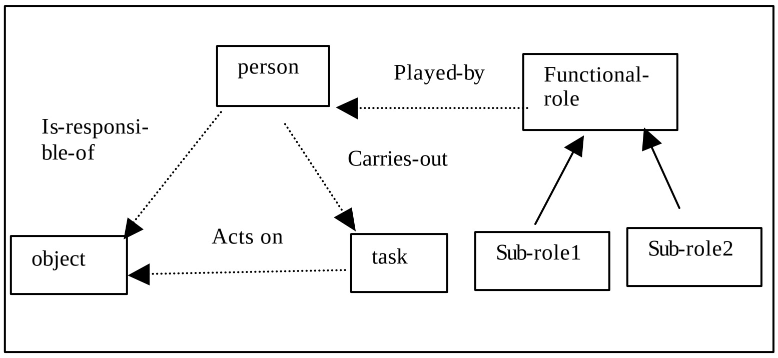


Fig.3 A portion of a meta-model relating the generic concepts: person, task, object

Fig.3 shows that a given person, possessing a given functional role, is responsible to perform a task. The object represents even the goal of the task in a sense that the activity associated to the task produces the object or transforms it into a desirable state (specified by the person responsible). Each role played by a person supplies a particular view of a set of capabilities of this person.

Network of Dependencies

The entities person, object and task are considered each one as complex entities, tightly interconnected one another. We speak then about *object-dependencies*, *task-dependencies* and *person-dependencies* (see fig.4). Dependence relations could be composition, inheritance, association, etc.

An entity is said *elementary* when we consider its proper description, i.e. independently from the references which the entity possesses with others entities and independently from the references which have the other entities with the entity in question.

An entity is said to be complex when it is composed from entities which are called the components of the complex entity and from the references between these components w hich are called dependency relations [13].

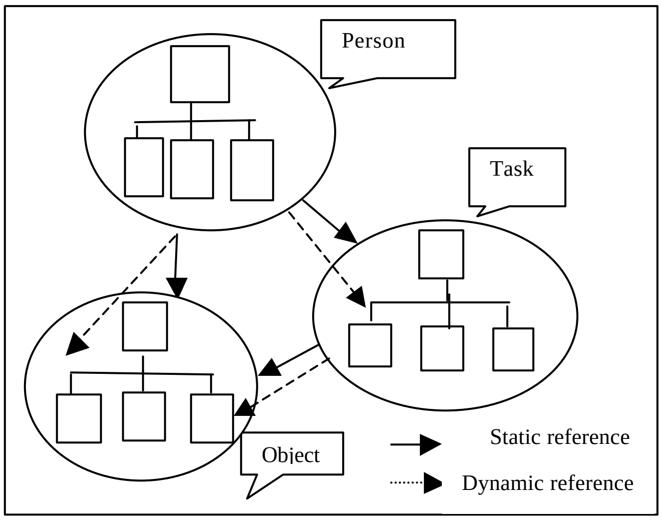


Fig.4 Object-dependencies, Task-dependencies and Person-dependencies

A person could have several roles at the same time. The same role could characterize several persons. A *Person* possessing several roles at the same time is considered as a complex entity, in the sense that this person is characterized by all the obligations and the privileges identified by all roles he (she) possesses. Sometimes, in our work, we shall use the term *composite role*. A composite role for a given person contains several other roles.

Person-Dependencies describe the relationships (hierarchies) into a complex person entity, i.e., the relationships between a person with his (her) roles.

Tasks and objects are also complex entities composed from entities which are called the components and from the references between the components called dependency relations.

The abstract example presented in Fig.4 is illustrated in the concrete application of the university lecture-planning process (see Fig.5).

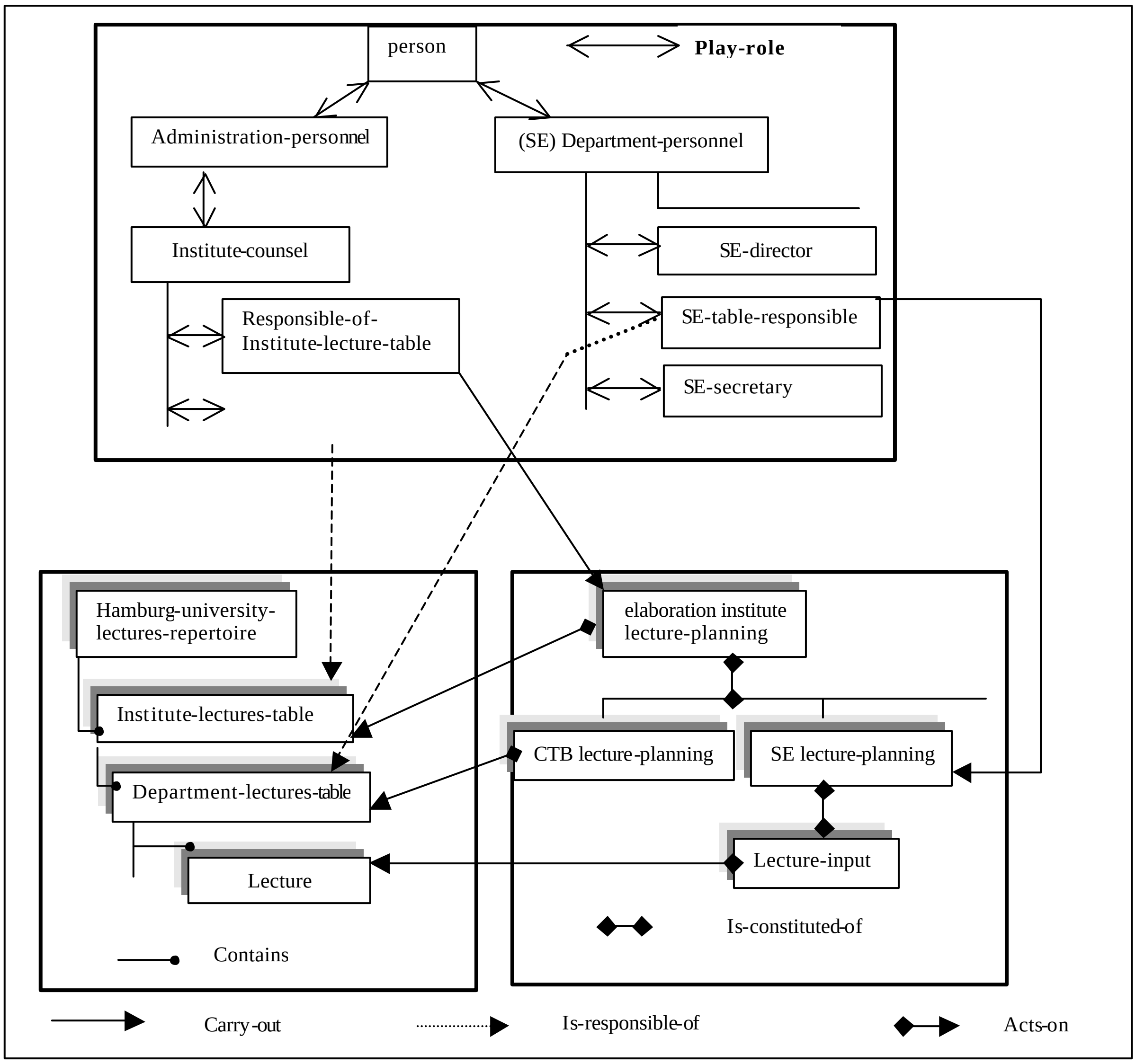


Fig.5 Information structures dealing with a portion of the cooperation process concerning only two departments: (SE)Software engineering department, (CTB) computer science theory bases department

4 EVOLUTION OF THE META-MODEL

We consider the evolution of the meta-model due to the changes in the universe of discourse (a part of the organization which is reflected in the information system). We address some requirements dealing with this evolution according to the needs identified in lecture-planning application (section 2).

- It must be possible to provide several versions for an object (e.g. department lecture-table)
- It must be possible to determine dynamically who is responsible of which object version
- It must be possible to determine dynamically in which activity is a given object version updated.

Modeling techniques borrowed from database research area are natural solutions to those crucial problems. Particularly, we see that version modeling and cooperation support bring several common problems to researchers in these two areas[10].

Bringing out Implicit Cooperation Support in Version Modeling

The reflection on our objectives led us naturally to think about version concept. The advantages of the concept of version have already been illustrated in a huge amount of research papers in different research areas. We focus on the particular interest of versioning for cooperation support. We mean by implicit cooperation the fact that human activity is not explicitly understood. In order to make explicit the correspondences between the both areas: cooperation and version modeling, we try to classify some common properties to them.

- *Both Version and Cooperation Modeling are Application- or Work-oriented*: it is impossible to formalize a version entity evolution. The task of determining what constitutes a version of an entity and the semantics for creating and retrieving versions is always left to the user.

- *The Time and the Space* are Key *Factors for both research Areas*: in version modeling, this is materialized by the well-known different type of version evolution: revisions, alternatives, equivalencies, variants and merging versions. On the other hand, the usual topology dimensions of groupware systems are also based on the temporality (synchronous or asynchronous) and the localization of the participants (at the same or different place).

- *Version Types Deal with the Evolution of all Key Concepts in Cooperative Work*: versions are used to model evolution of different type of entities from objects to processes, and from simple to complex entities. Version modeling community suffer unfortunately of several difficulties caused by this diversity. We believe that this diversity is precisely needed for the support of cooperative work. Those unrelated purposes are all found to be the necessary elements in cooperative work. Indeed, If we consider the key concepts in a conceptual model for the development of CSCW system[14], such as objets, tools, subjects, community, rules, work processes, constructions, we realize that a such description of cooperative work is a good area for concretion of the diverse version modeling.

- *Implicit Cooperation Support in Complex Object Versioning:* in database area, the problem of maintaining the consistency of evolving complex entities is well overcame. These applications are supposed implicitly to be in interaction with several types of users. Some techniques are proposed to control change propagation between related entities. They are concerned, by the way, for instance, in which entity versions which are linked to a given entity are propagated. We think that these techniques, when adapted to cooperation requirements, could be of a great interest for cooperative applications.

Object Version Propagation Techniques

Change propagation techniques are concerned, by the way, for instance, in which entity versions which are linked to a given entity are propagated. According to Katz [15] , many ambiguous reference situations could be clarified if the dependency relations between different entities are clearly described between the (versions of) entities depending from each other. Our considerations start from the idea to consider the ability to propagate operations (for instance, version creation and destruction) and the characteristic of this propagation as properties related to the *dependency relations* which interconnect the entities, and not the entities themselves [16]. The interconnections define the interactions between entities, and in particular, their existential dependencies.

We precise that in databases, object version propagation concerns the automatic management of object creation and destruction when objects are too complex to be dealt with manually. According to the cooperation requirements, the propagation techniques could have more semantic and could be of great interest, since they allow the online versioning and the awareness.

Furthermore the key problems for the control of version propagation are the same as those dealing with consistency problems in a cooperative work: (a) L*imitation of the scope of propagation*: it is not always advantageous, when a version of an object O is created to propagate version creation in the whole graph of the objects linked to O by dependence relations. The model must avoid the proliferation of useless versions. The best example from the lecture-planning application deals with the update at the level of the institute planning-table which do not need to be propagated to all the departments planning-tables rather to notify and eventual change propagate to only the concerned departments, (b) V*ersion referencing*: when two objects A and B are linked by a relation r, their respective versions may or may not be linked. If the versions are linked, the reference may then be static or dynamic [13], (c) *management of ambiguous propagation cases:* propagation is said to be ambiguous when more than one path can lead from one object to another.

Our approach is largely inspired from the object version propagation model defined in [13]. We consider this model as a basis to an evolutionary model dealing with the evolution of the ontology. We then generalize it by taking into account explicitly not only objects but also persons and tasks. Three specific relations are taken into account: "is performed by", "carry out" and "acts on".

Entity evolutions

Person evolution

a person may loose or earn new functional roles. That means that a given person may have more or less rights in order to be responsible of specific objects or also to carry out a given task in the organization. This change is propagated to the relation between *person* and *functional role*.

task evolution

we are interested particularly in the most pertinent status of the degree to fill up a task, which are: at the beginning: "activated", during a process: "working" and at the end of the process: "finished". Task-dependency specifies how the task is to be performed. In the application of university lecture-planning, we consider each task for elaborating lectures-table having three states: request(at the beginning), in progress (during) and elaborated (at the end).

Object evolution

different types of object versions are considered in the sense of [15].

Cooperative-views

We introduce what we call cooperative-view to allow the identification of the different allowed combinations between the three entities. It describes a *state of affairs* in the sense of [17]. It means a shared context in which a set of objects and the operations performed on those objects are visible to a set of persons. Each object is thus described according to the trellis it belongs to.

Cooperative-views are able to evolve; each time at least one entity evolves. We speak then of cooperative-view versions. A cooperative-view version consists thus of versions of each entity in the cooperative-view.

Propagation Strategies

Based on the dependency relationships between the three entities, object versions control the coordination and the state of the tasks which create or update it as well as the selection of only the persons who are concerned. This is not enough if we consider that the dependency relationships are also able to evolve dynamically.

Propagation strategies describe what the behavior of a given relation should be regarding version propagation. This behavior is defined for an operation, e.g. version creation of one of the extremities of this relation. The propagation strategy is composed of one or more propagation rules for each operation. The rules represent in a way conditions to be attained in order to achieve purposes such as represented on a purpose-table.

Each relation can propagate according to four propagation directions (FORWARD, BACKWARD, BI-DIRECTIONAL or NONE) and two propagation modes (RESTRICTED or EXTENDED)[16].

Forward direction: operation triggered on the source of a relation is propagated to the relation if the mode is restricted and to both the relation and the destination of the relation if the mode is extended.

Backward direction: operation triggered on the destination of a relation is propagated to the relation if the mode is restricted and both to the relation and the source of the relation if the mode is extended.

None: when an operation is associated with the propagation direction none, it is not propagated via this relation.

Application to the University Lecture-planning Process

By modeling the university lecture-planning process (see section 2) with WFMS (Workflow management systems) such ARIS and COSA [9], one important limitation with such modeling is that it is not clear, e.g., who has controlled the department lecture-table and which version he has used. The different relationships between all entities at different level of abstraction are very dynamic. E.g., the documents institute lecture-table and department lecture-table may have versions according to a specific update operation. The versioning of those documents leads to ambiguities and then inconsistencies in the sense of which entities are now related or not to the new version.

This problem is crucial in this application where it is necessary to deal with different versions at the same time. Besides the fact that the documents (objects) could be updated and conserved, it is necessary to conserve also the historic of the process for accomplishing a task, e.g., the states concerning the task of elaborating a lecture-planning of an organizational-unit (university, institute or department): request (activated), in progress and elaborated.

Propagation rules define the operations to be carried out for version propagation. They describe, in a declarative manner, the different propagation mechanisms that can be used.

Rule R1

Direction: forward

Mode: extended

Event: X‘<- create-version and update (X, „lecture(i)“)

Condition: let Ri be a relation such that X ∈ R(i).source and Y ∈ R(i).destination and S(i) be the strategy for R(i)

Actions:
R(i)‘ <- derive R(i)
Y‘ <- create-version (Y)
R(i)‘.source <- X‘
R(i)‘.destination <- Y‘
notify (responsible-of-department-lecture-planning(i))
Update(department-lecture-table(i))

Rule R2

Direction: backward

Mode: extended

Event:Y’<- update (lecture, L)

Condition: let R(i) be a relation such that X ∈ R(i).source and Y ∈ R(i).destination And S(i) be the strategy for R(i)

Actions:
R(i)’ <- derive R(i)
X‘<-create-version-with-update (X,lecture, L)
R(i)‘.source <- X‘
R(i)‘.destination <- Y‘
Notify (reponsible-departement-lecture-Planning)
Request (department(i)-lecture-table-update)

Rule 3

Direction: forward

Mode: extended

Event: X’<- create-version(X,state,”request”)

Condition: let R(i) be a relation such that X ∈ R(i).source and Y ∈ (i).destination and S(i) be the strategy for R(i)

Actions:
R(i)’<- derive R(i)
Y’<- create-version (Y, state,”request”)
R(i)’. Source<- X’
R(i)’.destination<- Y’
notify (responsible-of-department-lecture-planning(i))

Strategy S1

Has-as-rules: (R1, R2)

Is-the-default-strategy-for: “contains”

Strategy S2

Has-as-rules: R3

Is-the default-strategy-for: „is-constituted-of“

Fig.6 Examples of rules and strategies definitions

5 CONCLUSION

We propose a portion of a meta-model dealing with a wide range of cooperative work business (or software) processes. We aim to contribute to the understanding of the very complex network structure of an organizational work in order to facilitate the anticipation of the future system design. This portion will be completed by considering the intentional aspects such as: belief, intention, goal, etc.
Certain versioning problems evoked here are already more or less treated in CSCW research area[18]. We are interested particularly in the supply of the solutions to those problems for system development. The meta-model with its property to support entity evolutions could be used during the whole cycle of system development, i.e. dedicated to the analyst, designer as well as the end-user.
Some cooperation requirements are identified from real applications. Accordingly, The model we propose characterizes an organizational work in terms of network of networks of dependencies among three type of entities: objects, persons and tasks. These dependency networks are constantly changing. The capture of this changing is a key problem to understand the cooperation. Some common properties to both version and cooperation modeling are classified. This justify our adaptation of version object change propagation techniques based on dependency relationship and to extend them by considering not only entities which are objects, but also persons and tasks entities.

The different entity-dependency models and their evolution contribute in the description of the three main aspects of the cooperation: collaboration, coordination and the communication.

The future perspectives to this work concern the completion with intentional aspects, the definition of a taxonomy of propagation strategies and a detailed study concerning the integration of the model into an evolutionary and participative software development methodology.

ACKNOWLEDGEMENTS
The author is grateful to Prof. Dr Christiane Floyd for helpful discussions and constructive comments as well as to the referees for their valuable suggestions.

REFERENCES

[1] I. Wetzel, *Information Systems Development with Anticipation of Change Focussing on Professional Bureaucracies*, Proc. Of Hawai'i International Conference on System Sciences, HICCS-34, Maui, Januray 2001.
[2] C. Floyd, *STEPS-Projekthandbuch*, Universität Hamburg, 1992 (in German).
[3] U.Bürkle, Gryczan, G., Züllighoven, H, *Object-oriented System Development in a Banking Project: Methodology, experience, and conclusions*, Proc. Human-Computer Interaction, Volume 10, Numbers 2 & 3, pp: 296-336, Lawrence Erlbaum Associates Publishers Hillsdale, England, 1995.
[4] C. (Skip) Ellis, J. Wainer, *A conceptual Model of Groupware*, ACM CSCW 94 Conference on Computer Supported Cooperative Work, Chapel Hill, pp. 79-88, North Carolina, USA, 1994.
[5] A. Krabbel, I. Wetzel, H. Züllighoven, *On the Inevitable Intertwining of Analysis and Design: Developing Systems for Complex Cooperations*, Proc. G. van der Veer, A. Henderson, S. Coles (eds.): DIS'97 Designing Interactive Systems: Processes, Practices, Methods, and Techniques, Conference Proceedings, pp.205-213, Amsterdam, The Netherlands, August 1997.
[6]A. Krabbel, *Entwurf, Auswahl und Anpassung aufgabenbezogener Domänensoftware*, Dissertation zur Erlangung des Doktorgrades am Fachbereich Informatik der Universität Hamburg. Apr. 2000 (in German).
[7] C. Lilienthal, H. Züllighoven, *Techniques and Tools for Continuous User Participation*, in: J. Blomb erg, F. Kensing, E. Dykstra-Erickson (Eds.): PDC'96 Proceedings of the Participatory Design Conference, pp. 153 – 159, Cambridge, Massachusetts, USA. 13 - 15 November 1996.
[8] H. Züllighoven, G. Gryczan, A. Krabbel, I. Wetzel, *Application-Oriented Software Development for Supporting Cooperative Work*, Proc. Human-Computer Interaction. Ergonomics and User Interfaces, Volume1. Edited by Hans-Jörg Bullinger, Jürgen Ziegler. S. 1213-1217, 1999.
[9] C. Döhring, *Die universitäre Lehrplanung als kooperative Anwendung: Möglichkeiten und Grenzen von ARIS-Prozeßmodellierung und COSA-Workflowrealisierung*, Diplomarbeit, Universität Hamburg, Fachbereich Informatik, August 2000 (in German).
[10] L. Bendoukha, *Version-Based Description of Cooperation aspects: Motivations and Objectives*, In Proc. Of International Workshop on Open Enterprise Solutions: Systems, Experiences, and Organizations (OES-SEO), pp. 93-94, Rome, 14-15 September 2001.
[11] N.Guarino, *Formal Ontology and Information systems*, In N. Guarino (ed.), Formal Ontology in Information systems, IOS Press, Proceedings of the 1st International conference, Trento, Italy, 6-8 June 1998.
[12] C. Floyd, H. Oberquelle, *Softwaretechnick und Software-Ergonomie*, Lecture for semester 2000/2001 (in German).
[13] C. Urtado, C. Oussalah, *Complex Entity Versioning at two Granularity Levels*. Information systems Vol.23, N°. ¾,pp. 197-216, 1998.
[14] Kari Kuutti, *Debates in IS and CSCW Research: Anticipating System Design for Post-Fordist Work*, in Information Technology and Changes in Organisational Work, W. Orlikowski, et al., Editors, Chapman & Hall, pp. 287-308, London, 1996.
[15]R. H. Katz, *Towards a unifying framework for version modeling in engineering databases*, Proc. ACM computing surveys. vol. 22 no. 4 December 1990.
[16] J. Rumbaugh, *Controlling propagation of operations using attributes on relations*, In proceeding of the OOPSLA'88 conference, pp 285-296, September 1988.
[17] M. Uschold, *Building Ontologies: Towards Unified Methodology*, In proc. Of experts systems 96, the 16th Annual conference of the British Computer Society Specialist Group on Expert Systems, held in Cambridge, UK on 16-18 December 1996.
[18] D. Hicks, *The Role of Version Control in CSCW Applications*, In Proc. of the ECSCW'95 Workshop, GMD-Forschungszentrum Informationstechnik GmbH.David Hicks ... (eds.), Stockholm, Sweden, September 10, 1995.