Coordination Design

Hilda Tellioğlu
Institute of Design and Assessment of Technology
Vienna University of Technology
Argentinierstr. 8/187, A-1040 Vienna, Austria
hilda.tellioğlu@tuwien.ac.at

Abstract

Coordination design (CoorD) is about designing the coordination work. It looks for coordination patterns in work practices, identifies coordinative constructs and tries to identify structures in activities carried out. It describes how to provide elements necessary to produce computer support for coordination work. Coordination design is integrated in a system design process, starts at the same time and ends before implementation. It is interdisciplinary and contains several engineering techniques like requirement analysis or modeling as well as methods from other disciplines like ethnography. It uses coordination rules to describe the temporal and logical order of tasks performed in a cooperative work setting. Coordination design is participatory, contextual, situative, object-oriented, model-driven and rule-based. It consists of several steps: contextual inquiry, task analysis, domain modeling, rule creation and deployment of coordination rules. This paper introduces coordination design and shows its methodology in detail.

1. Introduction

To establish collaboration within a workgroup several communication, coordination and cooperation mechanisms are necessary. No matter whether computerized or paper-based, these mechanisms have a great impact on the work processes and must therefore be designed carefully. Research carried out in the field of computer supported cooperative work (CSCW) is mainly focused on understanding and providing a real support for the workers, and this both on technical and organizational level. Some researchers investigated artifacts for coordination of work. However, it is still not clear yet what the role of formal constructs [21] in cooperative work settings is and how formal constructs can be considered in the design of CSCW systems. Some other researchers tried to identify and classify coordination processes [11]. Etcheverry et al. defined a topology of coordination problems and tried to solve them by means of well-known coordination forms [12]. They provided a catalogue of coordination patterns based on coordination situations identified by Malone and Crowston [15].

In the field of coordination programming there are several coordination models and languages which are mostly categorized by distinguishing between data-driven (like Linda [1], Sonia [7], Laura [28], Ariadne [13] etc.) and process-oriented coordination models (like Ideal Worker Ideal Manager – IWIM [4], MANIFOLD [6] [5]) [24, p.35ff]. The focus of these approaches is to overcome the complexity of providing coordination mechanisms on a technical level. Most of them offer a mechanism to implement coordination on a specific technical platform without considering how the cooperative work is in fact carried out.

In software engineering there are several methods and techniques for the gathering and analysis of requirements. Some of them are object-oriented and use modeling languages to illustrate the work domain with its artifacts and use cases [14]. Unfortunately, in both system and software design not enough attention has been given to coordination issues from a methodological point of view. There are no methods or sets of methodologies specialized for analysis of coordination patterns in a collaborative work environment. It is not clear how to cope with the complexity of finding out what domain workers really do, what the routine day-to-day work is, when and how contingencies occur, how actors deal with unexpected situations and how they overcome the complexity of work, especially if it is collaborative. An important question is how to make the implicit knowledge of domain workers – especially about how to coordinate their work – accessible for others like systems designers. In particular, it is a challenge to integrate computer-based coordination mechanisms into the information system to develop. This paper tries to answer these open questions by introducing the design approach called coordination design.

In the next section we will show the theoretical framework coordination design is based on. The Section 3 describes methodologies integrated in coordination design be-
2. Theoretical Framework

The most cited coordination theory is developed by Malone and Crowston [15] by focusing on an interdisciplinary study of coordination. They identified common key concepts of coordination and tried to give a simple and general definition of it: “Coordination is managing dependencies between activities. ... , if there is no interdependence, there is nothing to coordinate” [15, p.90]. This means that to achieve any type of support for coordination of people, different kinds of dependencies must be characterized and coordination processes must be identified.

The second thread of theories that are relevant to our work is originated by Schmidt and Simone [23]. They introduced the concept of coordination mechanisms – initially called mechanisms of interaction [19] [20] [18] – as a “generalization of phenomena described in empirical investigations of the use of artifacts for the purpose of coordinating cooperative activities in different work domains” [p.155]. Their theory is mainly based on the use of artifacts for coordination purposes in a cooperative work setting.

By means of ethnographic field studies Schmidt and Simone tried to understand how cooperating actors apply coordinative constructs in their work, what these constructs are and how such constructs are supported by artifacts. Specialized artifacts like time tables, schedules, catalogues, classification schemes for large repositories etc. help reducing the complexity of articulation work and alleviating the need for ad hoc deliberation and negotiation [23]. Schmidt and Simone call these specialized artifacts, the mentioned conventions and procedures coordination mechanisms.

A coordination mechanism is “a protocol that, by encompassing a set of explicit conventions and prescribed procedures and supported by a symbolic artifact with a standardized format, stipulates and mediates the articulation of distributed activities so as to reduce the complexity of articulating distributed activities of large cooperative ensembles” [22]. A coordination mechanism consists of a coordinative protocol and an artifact [23].

Coordination mechanisms are aligned with other mechanisms because they alone serve only limited purposes [23, p.182]. For complex work arrangements we need usually more than one coordination mechanism. Due to serious inherent limitations of coordination mechanisms based on paper artifacts, computational coordination mechanisms are defined. A computational coordination mechanism is “a computer artifact that incorporates aspects of the protocol of a mechanism of interaction so that changes to the state of the mechanism induced by one actor can be automatically conveyed by the artifact to other actors in an appropriate form as stipulated by the protocol” [25].

3. Coordination Design (CoorD)

Coordination design (CoorD) is about designing the coordination work. It is mainly based on the coordination theory developed by Malone and Crowston [15] and uses concepts like coordinative constructs defined by Schmidt and Simone [23].

- CoorD is a conceptual framework: It looks for coordination patterns in work practices – no matter under which circumstances and in which environment. It identifies coordinative constructs and tries to define structures and orders in activities carried out. The main goal is to create a domain-specific coordination mechanism (with its coordinative protocols and artifacts) that can be attached to the IT system-in-use. The coordination mechanism must be computational, malleable and linkable.

- CoorD is also a methodology-based approach: It describes how to provide the elements necessary to produce computer support for coordination work. It is integrated in a system design process, starts at the same time and ends before implementation. For coordination design, processes consist of resources, activities and dependencies [15]. Dependencies can be described by using temporal relations between tasks. Allen defined seven primitive relations between pairs of tasks [3]. These descriptive, mutually exclusive relations can be applied over time intervals T1 and T2: T1 equals T2 when they start and end at the same time. T1 starts T2 when both start at the same time, T1 ends before T2, T1 finishes T2 when T1 starts after T2 and both end at the same time. T1 meets T2 when T1 starts when T1 ends. T1 overlaps T2 when T1 starts before T2 ends and T1 ends before T2. T1 is during T2 when T1 starts after and ends before T2. T1 is before T2 when T1 ends before T2 starts. Allen also used a set of axioms to create a temporal logic based on these relations [2]. Raposo and Fucks extended Allen’s seven primitives by introducing active, passive, blocking and resource management interdependencies. They created a model by proposing Petri-Net-based coordination mechanisms to deal with these task interdependencies [17]. Furthermore, they developed coordination components and used them in an implementation of a multi-user videogame [16]. CoorD uses task interdependencies defined by Allen [2] [26].

- CoorD is a bottom-up approach. It is important to offer computer support for workers in their daily work and not only to provide managers with work-related data which they can use for management purposes. As a
consequence of this principle designers must consider wishes, expectations, improvisations and constraints of domain workers in the design of coordination processes by letting them to participate in design activities.

- **CoorD is participatory.** To enable participation communication and cooperation between different communities of practice must be established. How is this to manage when actors talk different languages even if they have a common understanding of what the work is about? The language is a crucial factor in achieving a cooperation between actors with different backgrounds. A formal language or a model can help to cope with these differences. For instance, UML (Unified Modeling Language)\(^1\) can be used to create intermediaries to communicate the work domain and qualities of the system-to-design. It helps to exchange contextual information and domain knowledge from users to coordination designers. Furthermore, technical information about systems needed in the design process, about the infrastructure used, about possibilities and restrictions of the technology can be transferred easily to the users.

- **CoorD is not only participatory, it is also interdisciplinary.** It contains several engineering techniques like requirement analysis or modeling and methods from other disciplines like ethnography.

- **CoorD is situative.** It is possible to adapt or modify the coordination rules depending on changing situations or contingencies. The adaptation can be done by users. They simply modify the rules and setup the system considering the changes in task dependencies.

- **CoorD is rule-based.** It uses coordination rules to describe the temporal and logical order of tasks performed in a cooperative work setting. It makes no sense to try to design coordination processes for a real work environment without being aware of the work context, of the complexity of cooperative work, of ways and conventions applied to overcome the unexpected situations. CoorD creates use cases to capture work activities and to understand cooperative arrangements. In this sense, CoorD is contextual.

- **Object-oriented system design is most suitable for CoorD because CoorD is also object-oriented.** CoorD helps to conceptualize domain artifacts in form of objects and relations between artifacts in form of associations. CoorD uses data dictionaries and domain models to represent and connect objects (the coordination entities).

- **CoorD is model-driven.** Changes in the domain model have a direct impact on rules. A real-time rule modification is a challenge for a coordination mechanism, especially when modifications can also be done by domain users. CoorD tries to provide designers and users with useful methodologies and techniques.

![Figure 1. Interfaces between the phases of coordination design.](image)

Coordination design is carried out in several steps:

- Contextual inquiry
- Task analysis
- Domain modeling
- Rule creation
- Deployment of coordination rules

Some of the phases listed above are tightly connected. They influence each other. For instance, when additional data is captured in contextual inquiry, all related tasks need to be analyzed again. Or when questions occur during the task analysis, these need to be answered by going back to the field, observe participatorily or interview the right persons. Figure 1 demonstrates relations between the phases of CoorD.

In the following subsections the phases are described in detail.

### 3.1. Contextual Inquiry

"Contextual inquiry uncovers who customers really are and how they work on a day-to-day basis." [14]. Not only in contextual design [14] but also in other design approaches like participatory design, user-centered design, customer-oriented design it is very important to know enough about the use context and real users. Designers have to know how users perform their work. Coordination designers need additionally to know all about task dependencies, relations between activities, temporal and logical orders in carrying out..."
work activities. By interviewing representatives of all types of users, by observing actors at work especially when they cooperate or coordinate their work, by gathering and analyzing relevant artifacts coordination designers try to understand the users, their needs and intentions in cooperative work environments. For this purpose, coordination designers create use cases. Use cases are written stories told initially by users. They include actions, tasks, task orders, exceptional situations etc.

That goes without saying that use cases need to be defined correctly, their influence to the whole design process is huge. Domain knowledge must be well communicated between users and designers. Since systems designers are in most cases not familiar with domain-specific work processes, users from the work domain are expected to bring their knowledge into the project. Unfortunately it is not always easy to get a unique definition of actors’ work practices and the artifacts they use. They sometimes tell slightly different stories about their work and how they carry it out. Even the concepts they use within the same company can vary.

3.2. Task Analysis

After creating use cases, the initial and final states of coordinative artifacts must be determined. This involves all related entities and their use cases. For all use cases considered as relevant for coordination purposes each possible initial state must be analyzed. For this purpose state diagrams can be used.

By means of activity diagrams tasks to be performed can be modeled. Activity diagrams show tasks’ temporal and logical order. The way tasks are ordered and whether they are executed parallel or sequential are very important factors for coordination purposes. By designing the execution of tasks and forcing the system – and through it its users – to carry out the work in a specific order, a coordination mechanism can be established.

3.3. Domain Modeling

Domain modeling is about creating a class diagram to show system entities and associations among them. By using UML the system’s ontology can be designed first as a data dictionary. Data dictionary describes the concepts and their relations to each other. After identifying the classes (ontological entities) and their attributes relations can be modeled. The result is a domain model which can be modified several times. Before the coding starts, designers create the class diagram based on the refined domain model. Associations and specializations must be defined explicitly.

The system-to-design has to guarantee that system ontologies are not violated during processing, which means that the relations between entities are correct in all states of the processing. It must be avoided that inconsistencies appear although the common ontology is shared by several actors and accessed at the same time.

3.4. Rule Creation

After creating a domain model and dependencies between its entities, it is necessary to define the constraints that the system has to fulfill. Relations, associations and hierarchies between entities include restrictions and norms. These need to be considered in the design process. To achieve this, mechanisms are required that enable the coordination of the access to the shared ontology.

It is important to make all constraints explicit in a domain model. The violation of these constraints must be avoided by the system. To achieve this, we introduce the concept of rules that derive from relations defined in the domain model. Rules express (hidden) dependencies between ontological entities. They describe what needs to be guaranteed by the system in order to achieve consistency and correctness in the domain model. Rules emerge when users talk about dependencies between their artifacts. Rules need to be refined and communicated between users and coordination designers until all are convinced of their definiton, necessity, correctness and usefulness.

Schmidt defines formal organizational constructs like procedures, workflows, process models [21]. Their representation help to regulate the routine coordinative activities in an organization and enable cooperation settings to perform more reliably and efficiently. Formal constructs can be introduced to deal with increased complexity of work that normally occurs when the scale of cooperative work is increased [10] [9] [8]. Rules are formal organizational constructs. Rules are not causal, they are instrumental and convey constraints. They “offer a precomputation of interdependencies among activities” and provide instructions to actors [21, p.7].

3.5. Deployment of Coordination Rules

Rules must be deployed in order to be verified and evaluated. At the end of this phase coordination rules are implemented and integrated into the overall IT system.

By analyzing activity diagrams drawn for each use case and considering all rules written in a natural language we can define coordination rules. Coordination rules show the main temporal dependencies between tasks. In this sense, coordination rules are rules that are derived by analyzing the temporal and logical order of tasks that must be performed in order to implement a use case in an application.

---

2 The write access (including create, modify and delete) needs to be coordinated because it may change the domain model.
Coordination rules are written in a formal language or in pseudocode.

To associate ontology entities with coordination rules, tasks are used. Certain tasks are assigned to certain entities. Figure 2 shows the relation between entities, tasks and coordination rules in a database. For any entity in the ontology tasks are defined that perform actions on them like create, modify, delete, assign, remove assignment etc. Coordination rules are defined on tasks showing temporal interdependencies between them like equals, starts, finishes, meets, overlaps, during and before.

**Figure 2. Associations between ontological entities, tasks and coordination rules.**

### 4. Conclusions

This paper shows coordination design as an approach for designing coordination work. It contains several phases which are strongly interrelated. Figure 3 summarizes the workflow in CoorD. It also shows the entities created and used by several activities.

One of the main goals of this work is to provide such methodologies and techniques which make designers to coordination designers. They learn how to design a mechanism and a system respectively enabling the transformation of coordination work from spoken or paper-based interaction to artifact based interaction. CoorD shows how this transformation can be carried out. Task dependencies articulated by domain users can be formalized by coordination designers. And the result is a set of coordination rules. Coordination rules are rules that describe the temporal and logical order of tasks performed in a cooperative work setting. These are used to coordinate semantic dependencies between work activities carried out by different actors.

CoorD described in this paper is applied to develop CoMex – a computational coordination mechanism which is described somewhere else [26] [27]. CoMex is a running prototype that needs to be further developed.

### References

Figure 3. The workflow in coordination design.


