The Pennsylvania State University

The Graduate School

College of Information Sciences and Technology

DESIGNING FOR AWARENESS PROMOTION IN DISTRIBUTED,

COMPLEX COLLABORATIVE ACTIVITIES

A Dissertation in

Information Sciences and Technology

by

Bo Yu

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

Spring 2013

The thesis of Bo Yu was reviewed and approved[*] by the following:

Guoray Cai
Associate Professor of Information Sciences and Technology
Thesis Advisor, Chair of Committee

Alan M. MacEachren
Professor of Geography

Mary Beth Rosson
Professor of Information Sciences and Technology

Xiaolong Zhang
Assistant Professor of Information Sciences and Technology

Abstract

One of the major challenges to support complex, distributed geo-collaborative activities is to integrate effective coordination mechanisms to manage different types of work dependencies. This study focuses on event-based awareness mechanisms to support the management of dependencies by giving team members an awareness of what each other are doing or have done so that participants can adjust and coordinate their work in a more flexible way. Unlike existing event-based mechanisms, this study (1) relies on a deep understanding of the variety and dynamics of dependencies in these activities to distribute awareness events, and (2) provides explicit visualization and interaction support for the interpretation of awareness events.

The central idea of the approach to modeling dependencies and awareness events is based on a formal structure of collaborative activities. Rooted in the SharedPlans theory [42], the activity model treats a collaborative activity as an evolving shared plan situated in a set of physical and mental contextual factors. Such a model of collaborative activities then can be used to (1) interpret awareness events according to how they are related to the current activities, and notify the relevant users of the events based on the identification of dependencies between the events and users' current focuses.

Then this study illustrates the visualization and interaction provided by the activity model through a design scenario, where a couple of first responders in an emergency response team can use it to help generate and interpret awareness information to manage dependencies. To investigate the impacts of complexities in geo-collaborative activities on the actors' ability to interpret awareness events, an experiment is being designed and performed. The general hypothesis is that the effectiveness of our visualization and interaction design to support awareness interpretation is correlated to the level of complexity of collaborative activities.

Table of Contents

List of Figures

List of Tables

List of Corrections

Chapter 1

Introduction

1.1 Problem Scope

Support for awareness is one of the most active research areas in computer supported cooperative work (CSCW) [26, 86, 76]. At its essence, awareness refers to the ability of collaborators to understand each others' activities and relate them to a joint context [76]. This context is essential for collaboration as it is used to ensure that individual contributions are relevant to the groups activity as a whole, and allow groups to coordinate the process of collaborative working [26].

One general design concern for awareness systems is that awareness must be achieved with minimal attention and effort from the participants of teamwork [59]. Awareness falls into the category of 'the articulation work' that is required for coordination but not the primary goals for collaboration [85]. Taking extra time and effort to achieve awareness can interrupt the current line of action, and therefore damage team performance [40]. As a result, a large number of awareness mechanisms and tools have been proposed in the literature, aiming at promoting awareness in a relatively effortless way [76].

Although much progress has been made in designing awareness mechanisms to support team activity at relatively small and medium scales [5], it becomes a much more difficult task to promote awareness in complex and highly distributed activities [16]. The geo-collaborative activities of interest in this study are a subset of these complex collaborative setting, where awareness is unlikely to be achieved effortlessly, and hence it becomes even more important to design computational ar-

tifacts that actively promote awareness. In the rest of this section, we first describe the characteristics of collaborative settings that we consider as geo-collaboration in this study, and then present the challenges for supporting awareness in geo-collaboration that motivates our work.

1.1.1 Geo-Collaboration

The geo-collaborative activities we consider in this paper are a subset of complex collaborative setting, with the following characteristics:

1. Multiple team members are geographically distributed.

2. They are engaged in tightly interdependent activities that require effective coordination.

3. They work in dynamic setting that entails rapid and frequent changes in environment and activities.

Examples of geo-collaboration within these boundaries abound in practical applications such as crisis management, resource management, military exercises, and science explorations. For the ease of illustrating the work we present in this study, we will frequently refer to the following scenario in emergency response.

> **An Emergency Response Scenario.** A chemical factory near an urban area was exploded and caused a major pollution. To respond to this critical incident, task force is formed that includes search and rescue teams, decontamination teams, medical treatment teams, and transportation teams. Teams are dispatched and configured geographically to cover the impacted area. Each team cover a functional area of the overall mission. Search and rescue teams patrol the incident area to search for victims and report their locations and status. Discovered victims are first decontaminated (by one of the Decontamination teams) before they can be moved to other facilities. If a victim is wounded, he or she will be scheduled and transported to a medical station for treatment. All transportation needs for moving

How we define complexity: 1. distributed in geography, 2. number of activities and dependencies, 3. task-specific asymmetries among actors, 4. level of contingency

victims to treatment stations and shelters are handled by the transportation team. Although teams are working autonomously on their local tasks, they must coordinate their capacity, schedule, and priority to deal with emerging and unexpected situations in order to save and protect all the victims in an efficient fashion.

Such an emergency situation usually involves multiple individuals and organizations that are distributed in different geographic locations. Figure 1.1 shows a hypothetical distribution of tasks and teams in relation to the disaster area. Because of the distribution, workers structure their tasks so that they can work relatively autonomously within their local environment and responsibilities. In the same time, due to interdependencies among sub-activities [88], they find themselves working on a multitude of different activities, interleaving them in ways that seem best suited for getting their work accomplished given the practical pressures. Furthermore, exceptions to the planned responses are a common and critical factor in these activities [103]. What specific information is of concern and interest to a given individual is changing rapidly.
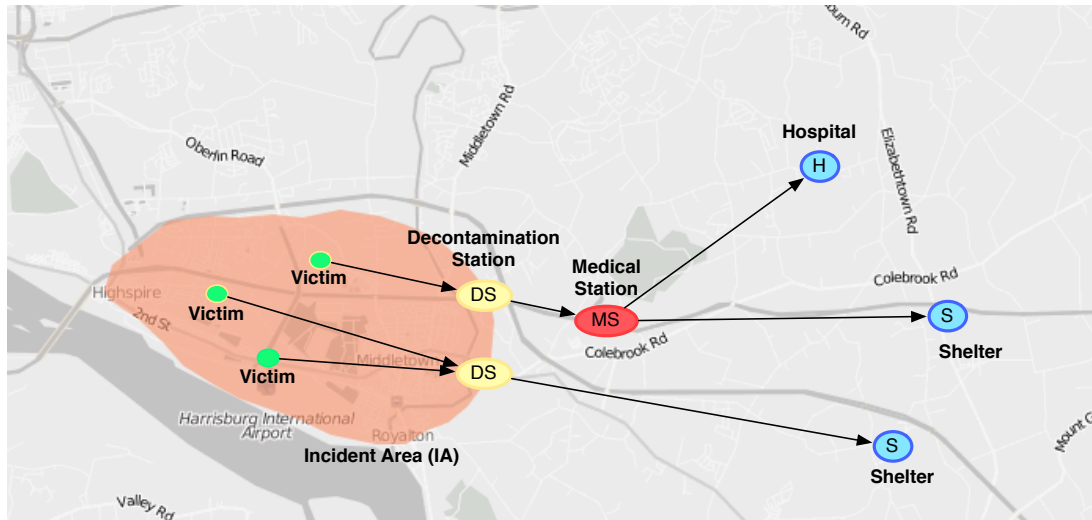
Figure 1.1. An emergency scenario

1.1.2 Challenges in coordinating geo-collaboration

This scenario clearly demonstrates the three major challenges to coordinate complex geo-collaborative activities: the complexity and contingency of collaborative activities, and the involvement of geographical space.

1.1.2.1 High level of complexity

With low degrees of complexity, the coordination of cooperative work can be achieved by means of mutual awareness and alignment [86]. As demonstrated by the body of rich empirical studies of cooperative work within CSCW [47], actors tacitly monitor each other; they perform their activities in ways that support coworkers' awareness and understanding of their work; they take each others' past, present and prospective activities into account in planning and conducting their own work; they gesture, talk, write to each other, and so on, and they mesh these interactional modalities dynamically and seamlessly . This appears effortless, because, to a competent member in the flux of doing the work and thus attuned to the changing state of the field of work what the colleagues next to him are doing is immediately meaningful; it does not require interpretation, reflection, contemplation to know why they are doing what they are doing or why they are not doing something else [86].

However, in the complex work settings that characterize modern industrial, service, and administrative organizations where hundreds or thousands of actors engaged in myriads of complexly interdependent activities, the task of coordinating the interdependent and yet distributed activities is of an order of complexity where the mutual awareness mechanisms are far from sufficient. Carstensens study of a software development project [22] clearly illustrates the problem of scaling with complexity. In previous projects the systems they had been constructing had been small and the programming work had been done by a couple of programmers. In these projects they had been able to manage their interdependencies practically effortlessly. They had been working next to each other and had practically unconstrained access to consulting each other and to monitoring each others work. At the time of the study, however, a new project had been undertaken in which the engineers were building a significantly larger system comprising many hundred

thousands lines of code. Their traditional coordinative practices were now quite inadequate. The interdependencies of their cooperative effort now transcended the local practices, and they were faced with situations that the effects of their local activities to other regions of the cooperative effort are not immediately and straightforwardly evident. To deal with the ensuing crisis, the ensemble had to develop a set of formal coordinative artifacts, such as the bug report forms, to regulate local practices.

One of the major challenges demonstrated in the scenario is characterized by the degree of *complexity* of dependencies that can exist in coordination work. With low degrees of complexity (e.g. when the number of dependencies that need to manage is limited, or their effect is only within the scope of local practices), the coordination of collaborative work can be achieved by means of human communicative and cognitive skills [87]. However, in the complex work settings as described in the scenario, multiple dependencies can co-exist at the same time, and they together form a web of dependencies where state changes of one of them can cause chain effects on others. Faced with a high degree of complexity of coordination work, collaborative actors often use a category of symbolic artifacts which, in the context of a set of procedures and conventions, stipulate and mediate coordination work and thereby are instrumental in reducing its complexity and in alleviating human effort [92]. These artifacts, together with the concomitant procedures and conventions are called 'coordination mechanisms' [87]. The first goal of this study aims at building an integrated coordination mechanism that would keep track of the state of work and to manage relations and dependencies among actors, tasks, and resources in order to reduce the complexity of coordination work in geo-collaborative activities.

1.1.2.2 High level of contingency

Developing appropriate coordination mechanisms must be based on a solid understanding of various collaborative dependencies, i.e. the capabilities of coordination mechanisms should match coordination requirements of the task at hand. In relatively static work domains such as process control or manufacturing, the set of work dependencies is largely known in advance and thus it becomes feasible to specify formal coordination mechanisms to manage them, such as routines, pre-

planning, or standardization. Increasingly, however, collaborative work is characterized by high levels of interdependence, uncertainty, and time constraints, such as cooperative design [22] or emergency response [88]. In these dynamic contexts, interdependence of work is emerging and rapidly shifting over time [33]. Due to changes and evolution of collaborative plans, dependencies may emerge, sustain, or disappear as the activity advances.

The contingency of dependencies can be clearly evidenced in the motivating scenario, where patterns of dependencies among people are in fact quite volatile, either due to the changes of environment in which the work is done (environmental uncertainty), or the changes of the task that the contributors are trying to accomplish (task uncertainty) [56]. For instance, while the decontamination process at a given decontamination station is under way, a first responder reports that five new victims have been discovered and are on their way to the decontamination station. As a result, a request to deliver an additional operator is initiated in anticipation of exceeding its maximum capacity of victims the station can handle. Suddenly, the decontamination process becomes intertwined with the process of delivering the operator, and the five new victims cannot be decontaminated until the new operator arrives at the station. In this way, such unexpected events cause seemingly un-related tasks to suddenly become interdependent.

As a result, we believe that the contingency in the interdependence of work represents a major feature of complex geo-collaborative work, but existing coordination tools have not taken this into account seriously. When a collaborative activity involves an extended period of work with volatile dependencies, the potential dependencies (i.e. the collective set of all dependencies that could potentially happen during the whole activity) are a quite large number, but the actual dependencies (i.e. those dependencies that are live and need to be coordinated) are rather a relatively small number. Coordination tools could play a more effective role if they target coordination of actual dependencies, rather than potential dependencies. However, identifying the set of actual dependencies at any given moment is difficult due to the fact that such dependencies are unknown a priori and they emerge as a consequence of the evolving activity. It would be highly desirable for collaborative tools to be able to assess the characteristics of the activities, computationally identify actual dependencies, and track their changes over time.

1.1.2.3 The uniqueness of being spatial

Coordination work varies, according to the complexity of the interdependence, that is, depending on factors such the distribution of activities in time and space, the number of participants in the cooperative ensemble, the structural complexity posed by the field of work (interactions, heterogeneity), the degree and scope of specialization among participants, and so on [87]. As a result, different types of dependency relationships have been discussed in the literature. For example, Yu and Mylopoulos [108] illustrate that actors can depend on each other in various means: they may depend on each other for goals to be achieved, activities to be performed, and resources to be furnished. In the case of task dependencies, Malone and Crowston [58] identify several common types of dependency relationships between tasks, including temporal relationships, resource-related relationships, and goal-related relationships.

In the context of geo-collaborative activities, some of dependency relationships may be directly cast on the geospatial relationships between entities. For example, in the scenario, the different rescue resources, e.g. vehicles, equipments, the victims, the response actors, and their activities are spatially distributed. The locations of these entities and spatial relationships among them can raise dependencies that need to be managed explicitly in the coordination work. A traffic jam at a given location may have no direct impact on the actor who is performing the task to take care of victims. However, the fact that the location of the traffic jam is on the path of a rescue vehicle delivering vital medical equipments to the actor may poses a dependency between the traffic jam and the actor's current work. As a result, the actor may have to switch the focus to explicitly manage this dependency before the domain task can be performed.

The nature of being spatial in these geo-collaborative activities provides a unique type of dependency relationships that need to be explicitly managed. However, existing models of dependencies lack of support for dependencies that involves spatial relationships. To understand and manage the dependencies involving spatial relationships, we must develop a deep understanding of how geographic space and human activities are related to each other, and integrate these spatially-related dependencies with other types of dependencies in a coherent framework.

1.2   Research Objectives and Approach

By setting the scene in the previous section, the overall objective of this dissertation is to addresses the major challenges in coordinating complex, dynamic, and distributed geo-collaborative activities by developing an awareness-based coordination mechanism that can utilize the knowledge of activities and dependencies.

To achieve the research objective, this study follows the design science paradigm in information systems research [50]. Knowledge and understanding of the aforementioned coordination problems in geo-collaboration and their solutions are achieved through a set of design activities, including building, evaluation, and application of the designed artifact. Figure 1.2 provides the overview of the research approach in this study.

The research starts with justifying the relevance of the research to the problem domain, i.e. geo-collaborative activities. The unique characteristics of geo-collaboration motivate this study. Then the existing knowledge about theories, models, and methods of coordination is reviewed to provide the scientific foundation of this study. Based on the grounding work in the literature, the overall design framework is developed. Such a design framework help the author to identify knowledge gaps in existing studies, and develop concrete research questions that need to be answered. Answers to these research questions are achieved through a set of design research activities. During the process of these design activities, more relevant knowledge from existing literature is applied.

Following the design-research paradigm, the expected contributions of this research are two-folded. On the one hand, the research activities in this study have the potential to extend our understanding of spatial dependencies in geo-collaboration, and provide a comprehensive approach to supporting awareness process in coordinating geo-collaborative activities. On the other hand, the proposed approach can be applied in many geo-collaborative activities, such as emergency response, transportation management, to offer solutions to important real life problems.
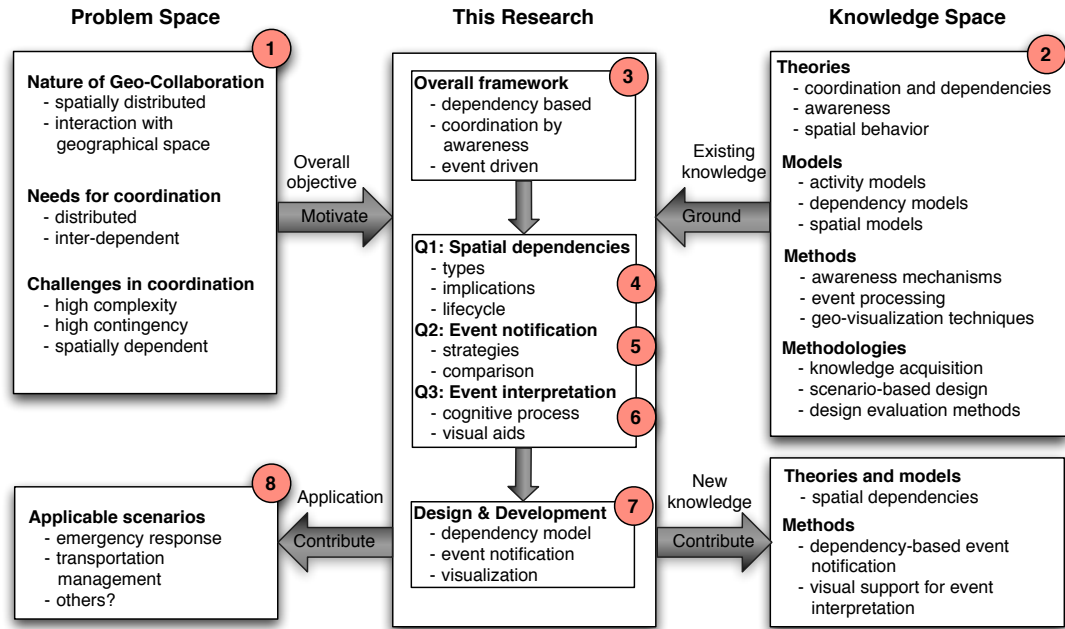
Figure 1.2. Overview of the research approach

1.3 Thesis Structure

The rest of this document is structured corresponding to how each chapter fits into the overall research framework. Chapter 2 presents the grounding work of this study, including the various studies on coordination and awareness mechanisms. Chapter 3 provides the overall design framework and relevant research questions. Chapter 4-6 provide the research activities and answers to these research questions respectively. Chapter 7 presents the design and implementation details. Chapter 8 demonstrates the use of the approach in the domain of emergency response. Chapter 9 concludes the study.

Chapter 2

Understanding Awareness

2.1 Overview

The concept of awareness has come to play a central role in both social and technical research in CSCW. However, what in CSCW labeled as 'awareness' has little in common, besides the fact that it represents some aspect of human interaction that is important for successful collaboration [86]. In a broad sense, two types of awareness can be distinguished in the CSCW research area: *social awareness* and *task-oriented awareness* [72, 86].

Social awareness addresses the availability of different kinds of information about the social context of the team members, e.g. awareness about what they are doing, if they are talking to someone, if they can be disturbed etc. *Social awareness* thus is conceived of as something that engenders "informal serendipitous interactions" [51] and "a shared space for community building" [27]. Awareness of the general social context is an important aspect of collaborative work, especially in domains where the the actors are engaged in cooperative work in a loose and broad sense, or domains where socialization is crucial [86].

However, when the tasks of collaborating actors become closely interdependent on each other, more urgent concerns need to be given to the aspect of *task-oriented awareness*. The *task-oriented awareness* focuses on practices through which actors seamlessly align and integrate their distributed and yet interdependent activities, e.g. awareness of things being done or in need of being done, of developments

within the joint effort that may be advantageous or detrimental for ones own work, of occurrence that makes ones work more urgent or leads to changes to the intended course of actions, etc. [86]. The major difference of *task-oriented awareness* from *social awareness* is that it focuses on activities performed to achieve a specific shared goal [19] and the actors being interdependent in their work [86], which lead to the unavoidable requirement for coordination.

In this study, we focuses on the **task-oriented aspect of awareness**, because the primary goal of supporting awareness in this study is motivated by the actors' being interdependent in their work and hence by the unavoidable requirements of coordinating and integrating their various actions in distributed geo-collaborative activities.

Within the scientific investigation into task-oriented awareness phenomena in collaboration, two lines of research can be identified. On one hand is the research aiming to establish conceptual understanding of the awareness phenomena from the cognitive and social aspects [82], and on the other hand is an increasing bearing on system design and development in particular technologies to promote awareness [76]. Although our work has an emphasis on the second aspect, i.e. the supportive technologies to promote awareness, we believe that a solid conceptualization of awareness phenomena in collaboration is extremely important for awareness promotion. System designers need to know what awareness might comprise and also how it is built and maintained in order to identify the specific awareness features they want to support [97].

As a result, before we move to the computational issues for awareness promotion, this chapter attempts to identify which of the existing theories and conceptualization of awareness in the literature is the most suitable for understanding the awareness phenomena in real world complex collaborative activities. A review and critique of what is currently known on the concept of awareness at both individual and team levels is presented, following which an integrated conceptual framework of awareness phenomena in complex collaborative activities is presented. In next chapter, we will show how such a conceptual framework helps us to evaluate existing computational awareness models and systems, and informs our design of the computational framework for awareness promotion.

2.2 Situation Awareness

Research into awareness at the individual level originated from the study of situation awareness (SA) in the human factors research community. Situation awareness is considered as knowledge created through interaction between a person and his/her environment, i.e. "knowing what is going on" in the situation [31]. A good general definition of situation awareness is as "the up-to-the minute cognizance required to operate or maintain a system" [1]. Although most of the situation awareness models in the literature are individual focused theories [82], it has also been well recognized as an important element in collaborative environments. For example, Gutwin and Greeenberg [45] view their workspace awareness as a specialization of situation awareness tied to the specific setting of the shared workspace. The concept of activity awareness proposed by Carroll et al. also subsumes situation awareness with an emphasis on aspects of the situation that have consequences for group work towards shared goals [19]. Hence, to understand the awareness phenomena in collaboration, it is important to start with understanding the practice of how individuals maintain and develop the situation awareness.

The human factors community has not settled on a common explanation of situation awareness, but we still can summarize some of the important characteristics that are well recognized in the literature, and also applicable to collaborative environments:

1. The products of awareness is the knowledge about the elements of the environment that is hierarchically structured [31].

2. The awareness phenomena should be seen as both product and process. As product, it is the knowledge that an actor can make use of. As process, it includes the cognitive processes through which the knowledge is achieved and developed) [1].

3. The process of achieving and developing awareness revolves around internally held, mental models, which contain activated information regarding current situations [94]. The activation of awareness information into the mental models are directed by the actor's activity [7].

We elaborate these three characteristics in the following of this section.

2.2.1 Hierarchically Structured Awareness Knowledge

Among the numerous attempts at specifying the products of situation awareness, i.e. what must be known to solve a class of problems posed when interacting with a dynamic environment [82], Endsley's three-level model [31] has undoubtedly received the most attention. The three-level model describes situation awareness as the operator's internal model of the state of the environment, comprising three hierarchical levels that is separate to the process used to achieve it [94]:

1. Level 1: *perception of relevant elements in the environment.* An actor must first be able gather perceptual information in the surrounding environment, and be able to selectively attend to those elements that are most relevant for the task at hand. At this stage, the information is merely perceived and no further processing takes place.

2. Level 2: *Comprehension of task-related elements in Level 1.* Level 2 involves the interpretation of the perceptual information from Level 1 in a way that allows an actor to comprehend or understand its relevance in relation to their tasks and goals.

3. Level 3: *Projection of the states in the near future.* Using a combination of Level 1 and Level 2 awareness-related knowledge and experience in the form of mental models, actors forecast likely future states in the situation.

Endsley's three-level model presents an intuitive description of situation awareness and has been applied in a plethora of different domains [106]. Its simplicity and the division of SA into three hierarchical levels allows the construct to be measured easily and effectively [29], and also supports the abstraction of situation awareness requirements and the development of design guidelines [82]. Furthermore, it has been extended in order to describe team situation awareness [30], and the three levels of awareness information are applicable in many collaborative situations [45].

Despite its popularity, the three-level mode has some important flaws. One of the key assumptions of the three-level model is the separation between depicting situation awareness as a product and the cognitive processes used to achieve it

[82], which leads to the inability to cope with the dynamic nature of situation awareness [94]. For example, Uhlarik and Comerford [104] suggest that the process of achieving situation awareness presented by the three-level model is both static and finite. Nevertheless, the model is also criticized by the ill-defined concept of mental models. Although Endsley's model emphasizes the critical roles of mental models in directing attention to critical elements in the environment (Level 1), integrating the elements to aid understanding of their meanings (Level 2), and generating possible future states (Level 3), the definition only includes the long-term knowledge that is formed by training and experiences, more important factors, such as the actor's goals, conceptual model of the current situation, are neglected [7].

2.2.2   The Development of Awareness

To address the dynamic nature of situation awareness, many researchers have used Niesser's perceptual cycle model [64] to clarify the cognitive components involved in the acquisition and development of situation awareness [94, 1, 45, 98]. According to the perceptual cycle model (Figure. 2.1), an actor's interaction with the world continues in an infinite cyclical nature. By perceiving the available information in the environment, the actor modifies its knowledge. Knowledge directs the agent's activity in the environment. That activity samples and perhaps anticipates or alters the environment, which in turn informs the agent. The informed, directed sampling and/or anticipation capture the essence of behavior characteristic of situation awareness.

Based upon Niessers perceptual cycle model, Smith and Hancock suggest that situation awareness is neither resident in the world nor in the person, but resides through the interaction of the person with the world [94]. Thus they viewing situation awareness as a generative process in 'an adaptive cycle of knowledge, action and information' [94]. In a similar fashion, Adams et al. [1] used a modified version of Niessers perceptual cycle model to describe how situation awareness works. They argue that the process of achieving and maintaining situation awareness revolves around internally held mental models, which facilitate the anticipation of situational events, directing an actor's attention to cues in the environment and
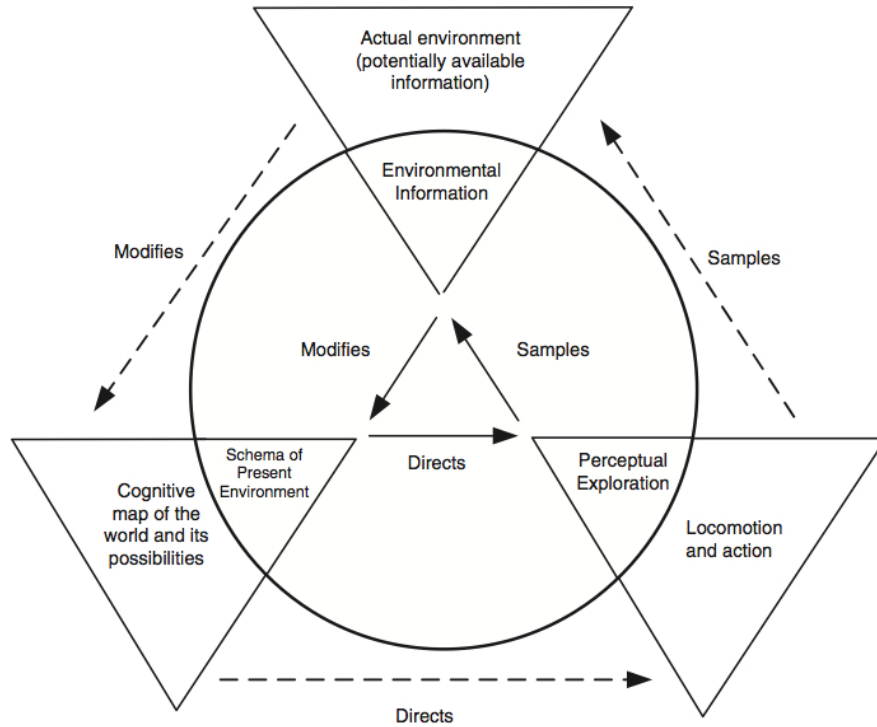
Figure 2.1. Niesser's Perceptual Cycle Model [82]

directing their eventual course of action. An actor then conducts checks to confirm
that the evolving situation conforms to their expectations. Any unexpected events
serve to prompt further search and explanation, which in turn modifies the actor's
existing model. Gutwin and Greeenberg used the perception-action cycle to ex-
plain how the awareness is maintained in a shared workspace, in which awareness
knowledge both directs and is updated by perceptual exploration of the workspace
environment [45].

Unlike the three-level model that depicts SA as a product separate from the
processes used to achieve it, models based on the perceptual cycle view situation
awareness as both process and product, offering an explanation of the cognitive
activity involved in the development of situation awareness, and also a judgment
as to what the product of SA comprises [82]. One of the key assumptions of these
models is the interplay between the awareness information and the internal men-
tal model of current situation. In the process of awareness development, some
knowledge is activated and integrated into the mental model, while some becomes

inactive or removed from the mental model. Although Smith and Hancock suggested that the adaptation of awareness information into the mental model should be goal-directed, i.e. it must reside in the task environment rather than in the actor's head [94], little detail has been given about the cognitive processes that guide the selection and interpretation of awareness information into the mental models.

2.2.3   Activity Directed Awareness Process

Bedny and Meister propose a description of situation awareness based on the activity theory in an attempt to clarify the cognitive processes involved in the interpretation of awareness information [7]. Based on the activity theory, they purport that individuals possess goals that represent an ideal image or desired end state of activity, which direct them towards the end state or methods of activity (or actions) that permit the achievement of these goals. It is the difference between the goals and the current situation that motivates an individual to engage in the awareness process and take action towards achieving the goal. They conceptualize activity in three stages: the orientational stage, the executive stage, and the evaluative stage. The orientational stage involves the development of an internal representation or picture of the world or current situation. The executive stage involves proceeding towards a desired goal via decision-making and action execution. Finally, the evaluative stage involves assessing the situation via information feedback, which in turn influences the executive and orientational components.

Instead of considering the actor's internal mental model as a whole, they suggest that the mental model is comprised of several functional blocks as presented in Figure 2.2. Each functional block has a specific role to play in the development and maintenance of situation awareness and that the blocks orientate themselves towards the achievement of situation awareness. The interpretation of incoming information (function block 1) is influenced by an individuals goals (function block 2), conceptual model of the current situation (function block 8), and past experience (function block 7). This interpretation then modifies an individuals goals and experience and conceptual model of the current situation. Critical environmental features are then identified (function block 3) based upon their significance to the

task goals and the individuals motivation towards the task goal (function block 4), which directs their interaction with the world (function block 5). The extent to which the individual proceeds to engage the task goals is determined by their goals (function block 2) and their evaluation of the current situation (function block 6). The resultant experience derived from the individuals interaction with the world is stored as experience (function block 7), which in turn informs their conceptual model (function block 8).
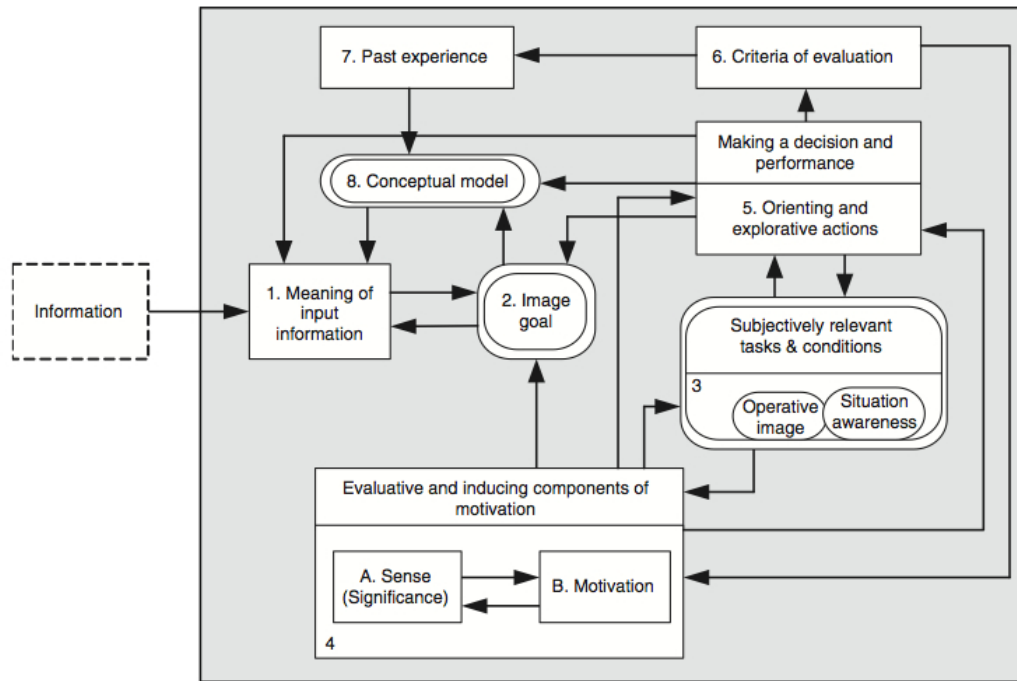
Figure 2.2. Bedny and Meister's Activity-Directed Awareness Process [7]

Based on the activity theory, Bedny and Meister clearly elucidate the functional blocks within the actor's mental model and their roles in the development of situation awareness [7]. Their model clearly shows how the actor's goals and activities play a central role to drive the process of awareness development. However, like most other situation awareness-related models, Bedny and Meister's activity theory model does not attempt to cater for, or explain the awareness phenomena in collaborative settings.

2.3   Awareness In Collaboration

The awareness phenomena in collaborative environments is indubitably more complex than situation awareness at the individual level. Salas et al. [80] point out that there is a lot more to team level awareness than merely combining individual team member's situation awareness. Beyond knowing what is going on in the environment, in their tasks, team members also need to develop an understanding of the activities of others, which provides a context of their own activities. This context is used to ensure that individual contributions are relevant to the group's shared goal as a whole [26]. This section reviews three prominent conceptualizations of awareness in collaboration that informs this study: team situation awareness, activity awareness, and distributed team awareness.

2.3.1   Team Situation Awareness

The research on team situation awareness (TSA) attempts to extend the theories and models of situation awareness to collaborative settings. Endsley et al. [30] suggest that, during team activities, situation awareness can overlap between team members, in that individuals need to perceive, comprehend and project awareness elements that are specifically related to their specific role in the team, but also elements that are required by themselves and by members of the team. Successful team performance therefore requires that individual team members have good situation awareness on their specific elements and also the same awareness for those elements that are shared. It is therefore argued that, at a simple level, team situation awareness comprises three separate but related components: individual team member's situation awareness; situation awareness of other team members; and situation awareness of the overall team (Figure 2.3).

Similar to individual situation awareness, team situation awareness should be considered as a dynamic process. Salas et al. [80] propose a framework of team situation awareness that comprises two critical processes, individual situation awareness and team processes. According to them, team situation awareness depends on communications at various levels. The perception of SA elements is influenced by the communication of mission objectives, individual tasks and roles, team capability and other team performance factors. The comprehension of awareness
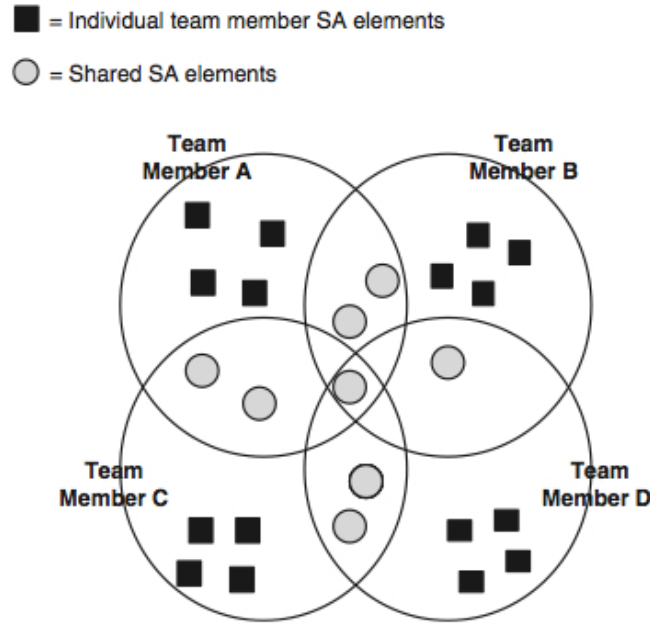
Figure 2.3. Team Situation Awareness (adapted from Endsley 1995 [31])

information (i.e. Level 2) is impacted by the interpretations made by other team members, so it is evident that SA leads to SA and also modifies SA, in that individual SA is developed and then shared with other team members, which then develops and modifies team member SA. Thus, a cyclical nature of developing individual SA, sharing SA with other team members and then modifying SA based on other team members SA is apparent.

Most attempts to understand team SA have centered on a 'shared understanding' of the same situation. Nofi [65], for example, defines team SA as: 'a shared awareness of a particular situation' and Perla et al. [70] suggest that 'when used in the sense of "shared awareness of a situation", shared SA implies that we all understand a given situation in the same way'. Shu and Furuta sugggested that TSA comprises both individual SA and mutual awareness and can be defined as 'two or more individuals share the common environment, up-to-moment understanding of situation of the environment, and another person's interaction with the cooperative task' [89].

Team situation awareness has been broadly recognized as a critical factor to understand awareness in collaborative environment, because its compatibility to

individual situation awareness, and the abstraction of individual awareness process from team processes [89, 82]. However, TSA also has many limitations.

1. First, a critical factor of team situation awareness is to define the configuration of shared situation awareness requirements, i.e. the degree to which team members understand which information is needed by which team member, or by the whole team. The identification of such shared awareness requirements is feasible in simple, small-scale collaborative scenarios However, for complex, real world collaborative scenarios, such a task becomes more intricate. In complex scenarios that involve numerous agents and artifacts working both collaboratively and dispersed geographically, viewing and assessing team situation awareness is actually too complex [82].

2. Second, the role of team processes, such as communication, mutual monitoring, in maintenance and development of team situation awareness, is only barely touched. It is recognized that an increased level of team processes will lead to enhanced levels of team situation awareness. However, the specific relationships between team situation awareness and team processes remains largely unexplained [82].

3. Last, the team situation awareness adopts the knowledge-in-common view of shared mental models [62], i.e. it focuses on how the shared understanding of the same situation is developed. However, as argued by Mohammed and Dumville, the knowledge-in-common view may be appropriate for only certain task domains and types of groups [62]. For example, in teams with high level of division of work, the distribution of knowledge and skills across the team typically is not uniform, as a result, a high level of overlapping knowledge in such teams might be inefficient.

2.3.2 Activity Awareness

To address the problem of team situation awareness that posits 'knowledge in common' as a basis for awareness, Carroll et al. proposes a new framework for understanding awareness in collaborative environment, based on the concept of 'activity awareness' [19, 20]. The major distinction between team situation awareness and

activity awareness is that, in realistically complex circumstances, instead of merely sharing relatively static and stable constructs such as knowledge in common, people share their activities [20]. In framing activity awareness, they appropriate the concept of *activity* from Activity Theory to emphasize that collaborators need to be aware of a whole, shared activity as complex, socially embedded endeavor, organized in dynamic hierarchies, and not merely aware of the synchronous and easily noticeable aspects of the activity [21].

Similar to the activity-directed SA model proposed by Bedny and Meister [7], activity awareness emphasizes the importance of using the concept of activity to structure the products and processes of awareness phenomena. The ultimate motivation of human actors to acquire and maintain awareness in the collaborative environment is to achieve their shared goals by performing their activities. As a result, the context surrounding a collaborative activity (e.g. the manner in which a shared activity is decomposed into smaller inter-related tasks, how these subtasks are assigned or adopted by collaborators, and when and how distributed subtasks are interdependent on each other), becomes the most important aspects of the situation that the team members need to be aware of [19].

By shifting the focus from shared knowledge to shared activity, activity awareness aligns the development of awareness with the development of collaborative activities. Most basically, activity awareness is achieved and developed through the join construction of common ground - shared knowledge and beliefs, mutually identified and agreed upon by members through a rich variety of communication protocols [20]. In long-term, open-ended activities over significant spans of time, the construction of shared practices, social capital, and human development become also important to enhance team member's activity awareness.

Activity awareness with its basis on Activity Theory, primarily focuses on the social aspect of the awareness phenomena, i.e. how the team members' awareness of the social context of collaborative activities is developed through common grounding, construction of shared practices, social capital, and human development. Although the authors claim that activity awareness subsumes situation awareness [19], little discussion has been given to how these higher-level social processes are connected to the cognitive processes to maintain situation awareness. Issues, such as how the state change in the external environment can lead to

a team member's internal goal change, which could furthers lead to re-planning of their activities, cannot be explained.

Furthermore, the activity awareness focuses on the sharing of activities, i.e. the importance of a common picture of the shared collaborative activities. However, such a common picture is usually distributed in the whole group, instead of in any single actor's mind [98]. Each actor in the group has their own awareness, related to the goals they are working towards. However, this seldom includes the whole picture of the collaborative activity, and only when all the actors' awareness knowledge is meshed up together, the common picture emerges. Activity awareness framework provides little support to explain how the activity knowledge is distributed across multiple actors.

2.3.3   Distributed Team Awareness

A more recent theme to conceptualize awareness in collaboration is the concept of distributed or systemic team awareness [98, 6]. Distributed team awareness approaches are borne out of distributed cognition theory [53], which describes the notion of joint cognitive systems comprising the people in the system and the artifacts that they use. Within such systems, cognition is achieved through coordination between the system units [6] and is therefore viewed as an emergent property (i.e. relationship between systemic elements) of the system rather than an individual endeavor. Distributed team awareness approaches therefore view awareness in collaboration not as a shared understanding of the situation, but rather as a characteristic of the socio-technical system itself [6]. Whilst recognizing that team members possess their individual SA for a particular situation and that they may share their understanding of the situation, distributed team awareness assume that awareness is distributed across the different human and technological agents involved in collaborative systems [98].

The main difference between distributed team awareness and other TSA and activity awareness models relates to the concepts of *compatible* and *shared* awareness. *Shared* awareness accounts suggest that efficient team performance is dependent upon team members having the same awareness knowledge. Distributed team awareness, on the other hand, postulates that, within collaborative systems, differ-

ent team members have unique, but *compatible* awareness, regardless of whether the information that they have access to is the same or different [98]. Team members experience a situation in different ways, as defined by their own personal experience, goals, roles, tasks, training, skills, and so on. So whilst some of the information required by two different team members may be 'shared' in the sense that they both need to attend to it as part of their job, their resultant understanding and use of it is different [83]. Ultimately, the picture developed by each team member is unique for themselves. *Compatible* awareness is therefore the phenomenon that holds distributed systems together. Each team member has their own awareness, related to the goals that they are working towards. Although different team members may have access to the same information, differences in goals, roles, the tasks being performed make them view it differently. In this way, each team members awareness is different in content, but is compatible in that it is all collectively required for the system to perform collaborative activities successfully.

While the distributed team awareness emphasizes the distribution of awareness, it does not discount the social interactions among different team members. The term 'transactive' awareness is used to describe the notion that distributed team awareness is acquired and maintained through *transactions* that arise from communications or other team processes [83]. A transaction in this case represents an exchange of awareness between one agent and another (where agent refers to humans and artifacts). Agents receive information, it is integrated with other information and acted on and then passed on to other agents. The interpretation on that information changes per team member. The exchange of information between team members leads to transactions in the SA being passed around. For example, an agent may perceive certain awareness element in the environment, interpret the meaning, and then pass it to another agent via a transaction. The second agent then builds its own interpretation upon the first agent's interpretation, and may start a new transaction to pass the awareness to other agents. Hence, it is the systemic transformation of awareness elements as they cross the system boundary from one team member to another that bestows upon awareness in collaboration an emergent behavior [98].

The concept of distributed team awareness has been investigated by the authors in a number of domains, including naval warfare [97], energy distribution [81], and

air traffic control [98]. The major strength of the approach is related to the systemic approach that it advocates, which is more suitable to analyze the awareness phenomena in complex, real world collaborative activities [98]. However, the main weakness, as admitted by the authors, is also related to it complexity [83]. Similar to other team situation awareness models, it uses concepts as the basic unit to analyze awareness elements, which often leads to extremely large networks in order to represent all the concepts and their relationships. A possible remedy is to integrate the distributed team awareness with the activity-directed models and switch the focus from concepts to activities to understand the awareness phenomena.

2.4 An Integrated Conceptual Model of Awareness

By reviewing the existing theories and conceptualizations of awareness, we believe that, instead of adopting one particular viewpoint, a suitable conceptual model for understanding the awareness phenomena in real world complex collaborative activities requires the integration of multiple constructs in the literature. Specifically, we identify the following requirements:

1. *The integration of individual cognitive processes and social processes.* As most theories and models of awareness in collaboration claim that individual situation awareness is still an important component in collaborative environment, the conceptual model should be able to account for both cognitive processes at the individual level and social processes at the team level, and emphasize on how these two aspects interplay with each other.

2. *The integration of compatible and transactive aspects of awareness phenomena.* We agree with the distributed team awareness approaches [83] on that, because of the differences in goals, roles, the tasks being performed make, each team members awareness is different in content, but at the same time is compatible for the team to perform collaborative activities successfully. Hence, the conceptual model should be able to account for how the awareness is distributed across multiple team members, and meanwhile can interact with each other via transactions.

3. *The integration of awareness and activity.* We resonate with the activity-directed SA model [7] and the activity awareness framework [19] to emphasizes the importance of using the concept of activity to structure the products and processes of awareness phenomena. As the purpose of human actors to acquire and maintain awareness in the collaborative environment is to achieve their shared goals by performing their activities, it is very natural to use the activities to structure their awareness requirements.

To satisfy these requirements, we propose an integrated conceptual model of awareness in complex collaborative activities, by combining multiple constructs in the literature. In general, the model has two major components: (1) an integrative model of activities, local scopes of work, and dependencies that form **the field of collaborative work**, (2) and **a set of awareness processes** built on top of it. The goal of the former is to establish the necessary knowledge that is needed to understand the content of awareness, i.e. *aware of what*, and the later focuses on the awareness processes, i.e. *how awareness is achieved and developed*. In the following of this section, we elaborate each component in more details.

2.4.1　The Field of Work

Following the activity-directed SA model [7] and the activity awareness framework [19], we focus on the concept of activity to structure the content of awareness, i.e. the field of collaboration work, built on top of three interrelated concepts (Figure 2.4):

1. We consider *activity* as the basic unit of analysis to support coordination in geo-collaboration.

2. Due to the distributed nature of geo-collaboration, the various activities have different implications on different actors, and form their respective *local scopes of work*.

3. Activities in different local scopes of work are interdependent due to various types of *dependencies* existing among them. Dependencies serve as the bridges between different local scopes of work to evaluate the implications of remote activities transcending multiple local scopes.
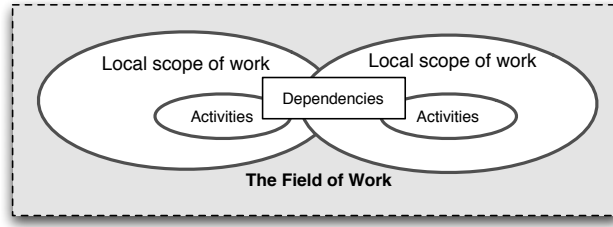
Figure 2.4. The Field of Work

2.4.1.1   Activity as the basic unit

In our approach, we subscribe to Activity Theory to conceptualize human activities
[63]. From this perspective, the basic structure of an activity can be defined as
several basic elements and mutual relationships between them.

1. **Actions** An action specifies a particular way of doing something. An action
   can be either basic or complex. A basic action can be directly executed by one
   or multiple actors. A complex action needs to be decomposed into subsidiary
   actions. When an action is specified as a sub-component of a higher action,
   this restricts the higher action to that particular course of doing. An action
   is assigned to a list of actors who are responsible to executing it.

2. **Actors** are defined as entities capable of performing actions and capable of
   making decisions on the performance of their actions.

3. **Resources** are anything that can be used in the transformation process of
   an action, including both material resource and resources for thinking. For
   example, in the case of response emergency, resources may include rescue ve-
   hicles, medical equipments, and information such as the locations of victims.

2.4.1.2   Local scope of work

Although various activities can be identified in a complex collaborative environ-
ment, each actor usually only engages in a small set of them. In fact, one of the
fundamental motivations for team work is to decouple a complex problem into a
set of smaller ones that are much easier to manage and tackle. One result of the

decoupling is that the work of actors is distributed and each actor is only interested in the states of a small set of activities, resources, and conditions that are relevant to their roles, current goals and tasks. To characterize the distributed nature of team awareness, we define the local scope of work for each actor as the set of activities, resources, and conditions that have direct or potential impact on the actors work.

The concept of local scope of work have several characteristics of the distributed nature of emergency response activities:

1. First, the local scope of work is a relative concept that changes with the actors current goals. Whenever an actors goal is changed, the local scope of work may also be changed.

2. Second, the local scope of work may cover multiple portions of the collaborative activity as a single actor may focus on several activities at the same time.

3. Third, the local scopes of two actors can overlap with each other. It is common that the same activity, resource, or condition falls into the local scopes of different actors, even though it may have different impacts on these actors. Actually, these overlapping elements across multiple local scopes of work play an important role in supporting the transactive awareness process.

2.4.1.3 Dependencies

Although activities are largely distributed and belong to local scopes of different actors, they cannot be performed without interacting with each other. Although an actor is only responsible for or interested in the activities within her/his local scope of work, the activities outside the local scope can still potentially impact her/his work through dependencies.

Dependencies have been studied by many researchers with aspect to coordination, from organizational studies ([108, 24]), multi-agent systems (MAS) ([**?**, 91, 90]), and computer-supported cooperative work (CSCW) ([74, 23, **?**]). Dependencies within a collaborative process can be of various forms. In general, we

can summarize three types of dependency relationships that have been well recognized in the literature: temporal relationships, resource-related relationships, and goal-related relationships.

1. *Temporal dependencies.* The activities of the actors might be interdependent due to the constraint of ordering them in a certain order [91]. In some case, certain activities cannot be started until others are finished (the problem of sequencing) or a certain group of activities have to be performed at the same time (the problem of synchronization).

2. *Resource dependencies.* Resource dependencies can be analyzed in terms of common objects, i.e. resources, that are used by multiple actors or involved in multiple actions. An example is when two or more users simultaneously want to alter the same part of a document in a collaborative authoring system [**?**]. These common objects constrain how each activity is performed. Different patterns of use of the common objects by the activities will result in different kinds of dependency relationships [**?**]: A *fit dependency* occurs when multiple activities collectively produce the same resource. A *flow dependency* arises whenever one activity produces a resource that is used by another activity. A *sharing dependency* arises whenever multiple activities all use the same resource.

3. *Goal dependencies.* Goal dependencies involves the subsidiary goals that might be interdependent across various actors or their tasks [91]. A goal-related dependency reflects the fact that the depender depends on the dependee to bring about a certain state in the world. Unlike the temporal or resource-related dependencies in which the depender knows what tasks are involved in these relationships, the depender in a goal-related dependency does not care how the dependee goes about achieving the goal, i.e. the dependee is free to, and is expected to, make whatever decisions are necessary to achieve the goal [108]. A goal-related dependency is usually characterized by the structural relationship resulting from goal decomposition of the whole shared goal, i.e. task A depends on the other task B because B is a means to achieve a subsidiary goal that must be satisfied in order to perform A.

Figure 2.5 illustrates how the different activities are distributed into different local scopes of work (represented as dotted circles) and how they are connected by the various dependency relationships in the motivating scenario. As we can see from the example, the different actors in the scenario, the victim manager, the decontamination manager, and the transportation manager, have very different roles to play, and therefore their local scopes vary significantly. However, due to the dependencies among their activities (For example, the decontamination manager relies on the transportation manager to deliver the victim to the decontamination station; the victim manager relies on the decontamination manager to perform decontamination on the victim), their local scopes are overlapped with each other.
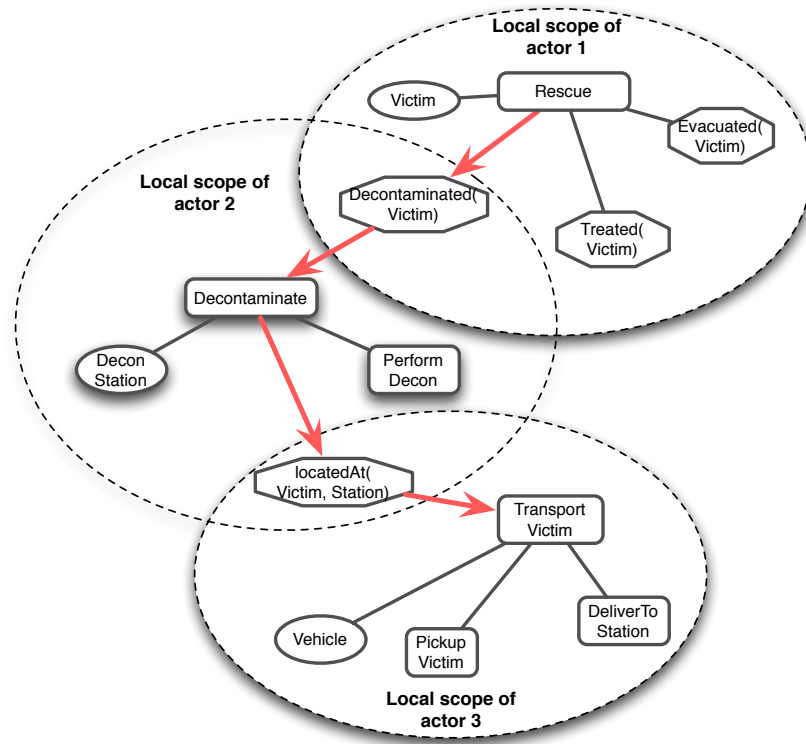
Figure 2.5. An Example: The Field of Work

2.4.2 Awareness Processes

Built upon the field of work, a set of awareness processes can be identified to describe how the awareness is acquired and developed in a cyclical way (Figure

2.10). In general, we can identify the awareness development cycles at both the individual and team levels.

2.4.2.1 The Development of Individual Awareness

At the individual level, each team member develops his/her own awareness in the similar way as described in the Neisser's perceptual cycle model [64]: the development of awareness starts with the perception of selective elements in the actor's local scope of work, which is then interpreted with the help of the actor's existing knowledge. The result of interpretation then help the actor to make certain decisions and perform actions, which then further update the actor's local scope of work (Figure 2.6). An important difference in our model from the original perceptual cycle model is that, instead of action directly stimulated by the acquired awareness knowledge, we emphasize the individual's planning process before the action performance, in which the individual needs to make decisions on what to do based on the interpretation of awareness information, such as whether he/she needs to change the goal, perform re-planning, or ask for help etc.

The individual processes need to be modified.

1. *Perception.* The process of perception is very similar to the concept in the perceptual cycle model. An actor must first be able gather perceptual information in the surrounding situation, and be able to selectively attend to those elements that are most relevant for the task at hand. The only difference is that the perception is constrained by the actor's local scope, i.e. he/she can only perceive the information about the environment, acitivities, or other objects defined in his/her local scope.

2. *Comprehension.* Comprehension refers to the mental activities where perceived awareness information is processed for the purpose of understanding its meaning in the context of the individual's goals and activities.

3. *Projection.* The process of predicting likely future states in the situation.

4. *Decision Making.* The process of decision-making involves two major tasks. The first is to elaborate the actor's plan based on the interpretation. This involves decisions such as commitment to certain activity, focus switching,
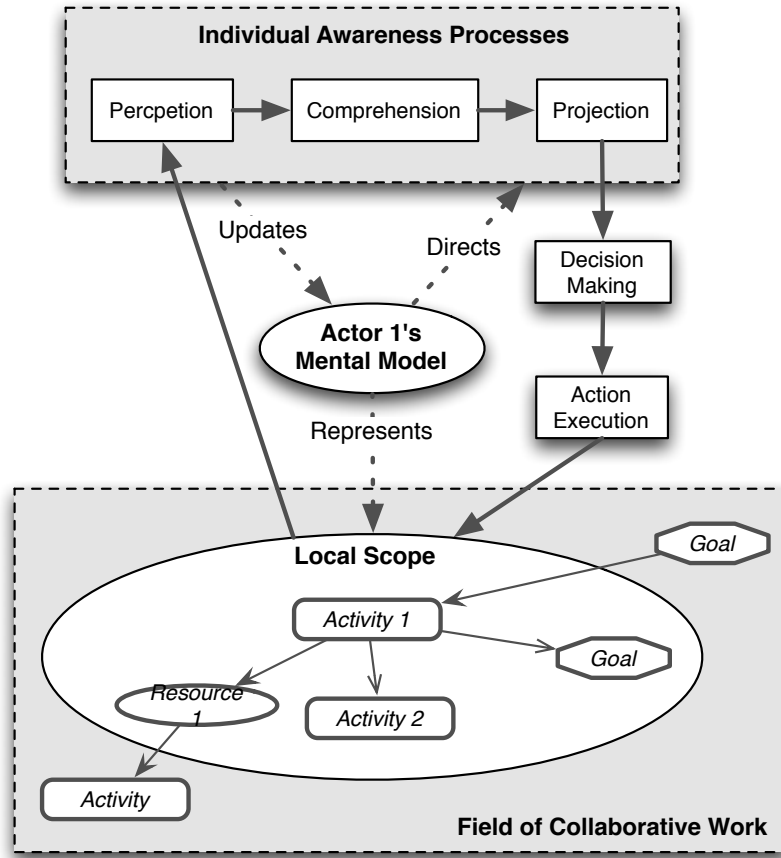
Figure 2.6. The Development of Individual Awareness

re-planning. The second task includes decisions about whether the actor
wants to propagate the interpretation to other actors, which is important for
initiating the awareness development cycle at the team level.

5. *Action.* Based on the result of decision-making process, the actor may per-
   form some action to manipulate his/her local scope, which can lead to further
   changes that will start another round of perception.

2.4.2.2 Team Awareness Propagation

Beyond the individual processes of awareness development, our conceptual model
allows the awareness propagation across multiple actors in a team. In general,
we can identify three basic trajectories of how the propagation can work: (1) via

explicit communication, (2) via implicit feed-through via action execution, and (3) via the manifestation on boundary objects in overlapping local scopes.

Elaborate on how each approach in more detail
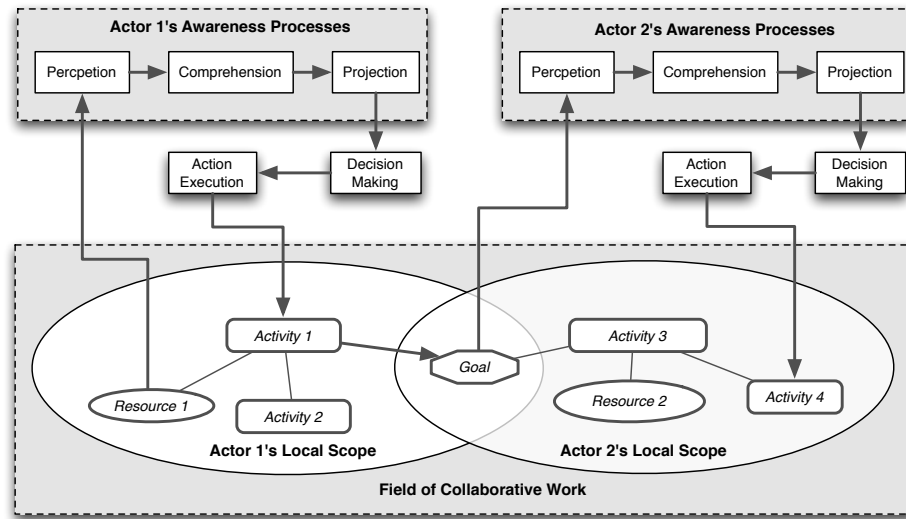
Figure 2.7. Team Awareness Propagation via Feed-through

Based on the interpretation of awareness information, an actor may decide to propagate the change to other actors by indicating how the other actor's activities can be impacted by his/her interpretation. The result of propagation is then perceivable by the other actor and may initiate the other actor's individual awareness development cycle. As the other actor develops his/her individual awareness, he/she may decide to propagate his/her interpretation back to the actor or other actors in the team, or he/she may perform actions that lead to changes in other actors' local scopes. In either way, new individual cycle may be initiated and the awareness development continues at the team level.

The development of awareness at the team level is guided by the dependencies among activities and the shared activities in overlapping local scopes. The dependencies allow the actors to cascade the awareness interpretation across multiple activities. The shared activities in overlapping local scopes enable the propagation process to cross the boundary of multiple local scopes.
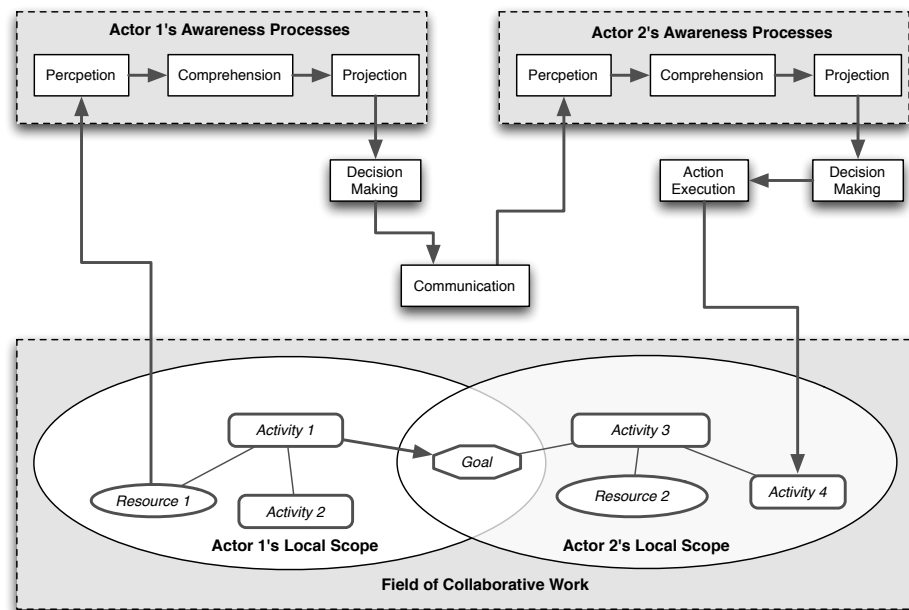
Figure 2.8. Team Awareness Propagation via Communication
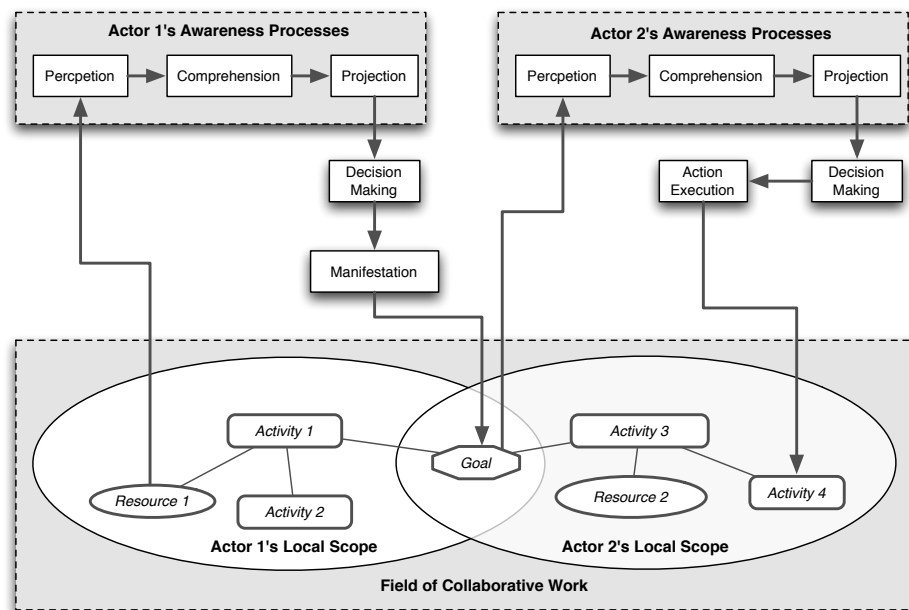
Figure 2.9. Team Awareness Propagation via manifestation

2.4.3 Discussion

Figure 2.10 shows the integrated conceptual framework by combining the afore-
mentioned components together.

Figure 2.10. The Integrated Conceptual Model of Awareness

Figure 2.11 demonstrates how the awareness processes work on top of the three
constructs. In the beginning, Actor 1 perceives some unexpected event happening
in the environment, and attempts to interpret it as whether and how it can impact
the activities within her local scope of work. After understanding the event, she
associates it with a state change of Activity 1. Furthermore, she predicts how
the state change of this activity will impact other activities because of the depen-
dencies among them. After Actor 1s interpretation, the event is likely to impact
another activity (Activity 2), and Actor 1 decides to propagate it. The propagated
event falls into Actor 2s local scope of work. Upon receiving this interpretation
from Actor 1, Actor 2 first needs to understand where this event comes from by
backtracking the interpretation, and evaluate how this event will impact his own
line of work, which leads to a new projected state change of Activity 3. The similar
process then is propagated to Actor 3s local scope of work. In this way, the teams

This
example
needs to
be revised
and elabo-
rated!

awareness of the initial external event is collaborative developed as the relevant actors gradually attach their interpretations to it.
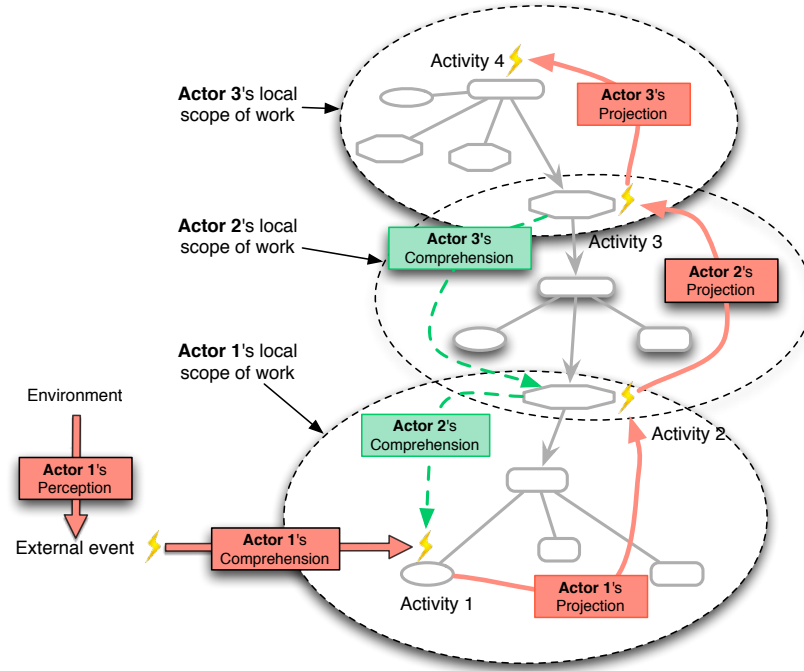
Figure 2.11. An Example: The Awareness Processes

The proposed conceptual framework shows the capabilities to satisfy the three requirements we presented in the beginning of this section.

1. First, the conceptual model provides two ways to connect the individual cognitive processes and team processes. On one hand, the connection can be implicit as team members interact with the filed of work. The development of an actor's individual awareness can manipulate the field of work, which can then be perceived by other actors in their separate individual awareness processes. In addition, the actors can explicitly propagate their interpretations to other actors within their individual awareness processes.

2. Second, the concept of local scope allows us to define how the awareness knowledge is distributed across multiple actors. The different actors often attend to different sets of activities as their local scopes. At the same time, the actors' local scopes are often overlapped due to the various dependency

relationships that may occur between activities, this provides the mechanism to allow awareness transactions among multiple actors. In this way, the compatible and transactive aspects of awareness phenomena are integrated.

3. Last, our structure of the field of work is based on the activity theory, which enables the integration of awareness and activity.

Designing for Awareness Support

Following the conceptual model of awareness phenomena in complex collaborative activities, we describe the design space of awareness support in this chapter. The goal of this chapter is to identify key design issues to support awareness in distributed, complex collaborative activities, by mapping them to the major components in our conceptual model presented in Chapter 2. We first present a framework of the whole design space with the major design issues identified. Then we discuss how these design issues have been addressed (or partially addressed) in existing awareness support systems, and the design challenges that motivate our approach.

3.1   The Design Framework

By conceptualizing the awareness phenomena in distributed, complex collaboration as a continuous development of awareness knowledge (i.e. the knowledge about the environment, activities, and their dependencies) through a variety of cognitive and social processes, the design issues for awareness support can be organized at two levels: the *individual* level and the *team* level. The former focuses on supporting the cognitive processes of individual team members to develop their own awareness, while the latter provides support for the social processes in which team members interact with each other to achieve awareness at the team level.

3.1.1   Designing for individual processes

At the individual level, the role of computer support can be understood in term of designing 'cognitive artifact' [66], which is defined as an artificial device designed to maintain, display, or operate upon information in order to aid cognition. Norman argues that an important design consideration for cognitive artifacts is the human action cycle that emphasizes the two sides of human action [66]. One side is the 'evaluation' side of action by perceiving, interpreting and evaluating the state of the environment. The other is the 'execution' side that of acting upon the environment. Norman's human action cycle matches well with the major steps of awareness development at the individual level, where the 'evaluation' side refers to the *perception*, *comprehension*, and *projection*, and the 'execution' side includes the *decision making* and *action execution*.

Perception

The achievement of individual awareness starts with the ability of individuals to perceive key features in the environment. The primary goal of awareness support in the perception process is to ensure that the awareness information is perceivable to the users. This goal can be achieved from two aspects: *selection* and *presentation* [12]. The former focuses on filtering out the information so that only the relevant information is perceived by the users, while the latter involves strengthening the stimulus to ensure that information is perceivable.

1. Support for *selection* is based on the assumption that perception is a selective process that depends on the requirements of the current working situation and the tasks at hand [31]. Not all information is of relevance to a user, hence the pool of all awareness information has to be processed to filter the relevant information [12]. Support for selection is to delegate the effort of filtering out irrelevant information to computer systems, so that human actors can focus on processing only the relevant set of information to avoid information overload.

2. Support for *presentation* is to strengthen the stimulus in the user interface to ensure important awareness information is perceivable. Existing studies

in cognition have shown that the salience of elements in the environment will have a large impact on which portions of the environment are initially attended to, and these elements will form the basis for perception [48]. As a result, the way in which information is presented via the interface will largely influence the perception process by determining which part of the environment will draw the user's attention.

Comprehension

Comprehension is to understand the meaning of perceived awareness information within the context of a user's current goals and activities [67]. In the comprehension process, new information must be combined with existing knowledge to develop a composite picture of the situation [31]. Hence, the computer support for comprehension is to help the users establish the connection between new awareness information and existing knowledge that forms the inferential framework [19]. In general, there are several ways that the comprehension can be supported by computer systems:

1. *Representation.* First, the computer system can support human comprehension by providing external representations of the existing knowledge. Instead of merely relying on the internal representations of human users, the external representation can serve as the information store, so that the internal representation at a given time can be quite sparse, perhaps containing only detailed information about their current focus [48], or pointers to locations of other important information in the external representation [57]. In this way, the limited working memory resources of human users are freed up for other aspects of cognition [57].

2. *Linking.* Second, the computer system can directly aid the linking between new awareness information and existing knowledge by presenting the awareness information along with the contextual information that is potentially relevant to understand its meaning [102]. Supporting comprehension by linking has its basis on the design principle of offloading cognitive processes onto perceptual processes [57]. By explicitly linking the awareness information

to contextual information for interpretation, some complex cognitive processes, such as searching for and activating the relevant portion of existing knowledge, can be replaced by simple pattern recognition processes [48].

Projection

Projection is the process that the individual predicts the future states of other activities based on the comprehension of awareness information. As argued by Endsley [31], this is the most difficult and taxing parts of situation awareness because it requires a fairly well developed mental model of the activities and relationships among them, and the capabilities to perform reasoning. System-generated support for projecting future events and states of the system can then target on supporting the development of the mental model by representing the activities and relationships, or supporting the reasoning processes.

1. *Representation.* Similar to the comprehension process, the projection process can be supported by providing external representations of the existing knowledge to enhance human cognition. However, unlike the representational support for comprehension that focuses on individual activity elements, the knowledge representation for projection emphasizes the various relationships and dependencies among the activity elements, so that the users can infer how the state on one activity can lead to possible changes on other activities' states.

2. *Reasoning.* The analytical reasoning is central to the projection process, through which users identify possible alternative future scenarios and the signs that one or another of these scenarios is coming to pass [101]. As a result, one of the critical requirements for supporting projection is to provide the analytics tools and techniques that allow the users to synthesize information and derive insight from it.

3.1.2 Designing for team processes

At the team level, the roles of computer in awareness support can be characterized by supporting the three basic types of team processes to propagate awareness

information as described in Section 2.4.2.2: *feedthrough*, *communication*, and *manifestation*.

Feedthrough

The process of feedthrough is to make consequences of individual activities apparent to other participants [26]. In co-located collaborative environments, this usually can be achieved without computer intervention, as collaborators can readily see each other's activities and artifacts they are working on [86]. However, in distributed collaboration, computer support becomes inevitable to enable the process of feedthrough. The role of computer to support feedthrough is to *broadcast* the effects of individual actions and make them visible to each other.

Communication

Communication is the prevalent form to propagate awareness information, in which people explicitly talk about awareness elements with their collaborators [45]. Computer mediated communication has been an important component in almost every distributed collaborative system, by providing team members a *medium* to communicate with each other remotely.

Although we consider communication as an important process for the team members to propagate awareness information, it is usually supported separately from the awareness systems as a standalone feature in collaborative applications. As a result, we do not put much emphasis on supporting communication in the following discussion.

Manifestation

Manifestation refers to a more subtle means to propagate awareness information among team members. Instead of directly performing actions to impact each other via feedthrough, or explicitly communicating with each other, the team members can make some aspects of their individual awareness visible to others, so that anyone who is interested in these aspects, or who is monitoring the field, can perceive the information. The aspects of individual awareness that can be made visible in the manifestation process can be the raw awareness information perceived by

a team member, or his/her interpretation of the awareness information during comprehension/projection, or the results of decision-making. To support the manifestation process, the computer system needs to provide the following functions:

1. *Externalization.* The computer system needs to provide the tools to allow the users to externalize aspects of their individual awareness and make them visible to other users.

2. *Visibility.* The computer system should allow the users to control the visibility of their manifested information. They can specify who can see what piece of the information they make visible.

Table 3.1 summarizes the whole design space with the major design issues for each element identified.

Awareness Process	Supporting Aspects
Perception	*1. Selection*: filtering out the information so that only the relevant information is perceived by the users *2. Presentation*: strengthen the stimulus in the user interface to ensure awareness information is perceivable
Comprehension	*1. Representation*: providing external representations of the existing knowledge to support human comprehension *2. Linking*: presenting the awareness information along with the contextual information that is potentially relevant to understand its meaning
Projection	*1. Representation*: providing external representations of activities and their relationships *2. Reasoning*: provide analytics tools and techniques to help users perform reasoning
Feedthrough	*1. Broadcasting*: making consequences of individual activities apparent to other participants
Communication	*Medium*: providing team members a medium to communicate with each other remotely
Manifestation	*1. Externalization*: providing tools for the users to externalize aspects of their individual awareness and make them visible to other users *2. Visibility*: controlling the visibility of manifested information, i.e. who can see what piece of the information

Table 3.1: Design Space for Awareness Support

3.2 The State of Art

By outlining the design space for awareness support in Section 3.1, we can use it as a framework to review existing studies and awareness systems by looking into how the design aspects in various cognitive and social processes have (or have not) been supported. Before that, we need to make a distinction between two types of awareness models for organizing and processing awareness information: *space-based* and *event-based* models, because it is the underlying awareness model that determines how the awareness processes are supported in an awareness system [41].

3.2.1 Awareness models

Most awareness systems rely on certain computational models for organizing and process awareness information. An awareness model usually provides a representation of the field of work, and specifies how awareness information is generated on top of it. In general, we can distinguish two types of awareness models commonly used in existing awareness systems: *space-based* and *event-based* models.

3.2.1.1 Space-based models

Many researchers have previously described systems to promote awareness based on the spatial metaphor [8, 78, 84, 93]. These space-based models explicitly represent the field of work as objects (which might represent actors, information, resources, or other computer artifacts) situated and manipulable in some space. Awareness is then achieved by the interaction between actors within the space, as they present themselves and attend to each other in the field [78]. Awareness information thus is implicitly embedded as perceivable properties of objects in the space.

The use of space-based model relies on presenting a 'shared space' among collaborators at any given time to provide awareness information, both implicitly and explicitly [26]. By maintaining the state of the shared work setting, people can either keep an eye on what the rest of the group is doing while doing their

individual work, or actively monitor the activities of others, to perceive awareness information [93].

In existing awareness systems, the 'shared space' can take different forms [5].

1. 'Shared physical space'. In the early work of supporting awareness, a significant effort was devoted to exploring the potential of media space technologies, i.e. an array of continually audio-video links between distributed actors, to provide a 'shared physical space' between actors in different locations [27]. As with the awareness study, the media space investigations were concerned with informal or social aspect of awareness, and in most cases task-oriented awareness was not mentioned explicitly [86].

2. 'Shared virtual space'. Rodden [78] developed the notion of 'virtual space' as a collection of computer-supported interactive spaces. Many collaborative systems offer various types of virtual spaces to support awareness, including abstract media space [69], virtual meeting rooms [12], and collaborative virtual environment [9]. Unlike traditional media spaces that aim to establish the 'shared physical space', this line of work focuses on the use of abstract representations as awareness indicators. Besides providing a kind of 'shielding' for privacy of the people in the shared space [69], a more interesting feature of using abstract representations is the capability to organize awareness information around task-oriented structures, such as office tables, meeting rooms [12], so that more task-oriented aspects of awareness can be supported.

3. 'Shared workspace'. 'Shared workspace' is a specialization of the notion of 'shared space' that emphasizes the task-oriented awareness. According to Gutwin and Greenberg [45], a 'shared workspace' is a bounded space where people can see and manipulate artifacts related to their activities. A group editor is a good example of this type of 'shared workspace', as it serves to organize activities like writing and revising, while maintaining a coherent view of the whole [26]. The awareness information in 'shared workspace' is presented by attaching it to the manipulable objects through which the task is carried out, and is achieved by user's interaction with the artifacts [45].

3.2.1.2 Event-based models

Event-based models, on the other hand, provide people with awareness of what is going on around them as expressed by discrete events [76]. Each event highlights a piece of awareness information related to certain state change in a collaborative setting for a limited amount of time. Events can be generated by sensors that are associated with actors, shared material, or any other objects that constitute or influence a cooperative environment [72]. The list of events that are supported in each awareness system can be totally different, depending on the application domain. As argued by Fuchs et al [39], we can basically imagine any kind of event, as long as it has a certain relevance when it comes to coordinating the work in a given setting.

Unlike space-based models where the awareness information is implicitly embedded as properties of objects in the underlying representation of the field of world, awareness information is explicitly represented as first-class objects in event-based models. Each event contains identifiers of the originator, the time, the state change in the collaborative setting, and the context in which the state change took place [38]. Modeling awareness information as first-class objects decouples it from the representation of field of work, which allows the awareness information refers to any change in the field of work, but not necessarily within the user's current viewpoint in the field. The user may attend to their own individual work in the filed, but still be able to receive events about other users that are outside his/her current focus. In this way, the user does not need to monitor the field of work, instead the awareness information is pushed to the user when it becomes relevant.

Because of the decoupling of awareness information from the representation of field of work, many existing event-based systems actually do not need to provide an explicit representation of the field of work [72, 36]. Even in some systems where the field of work is represented (for example, the GroupDesk [39] and the POLIAwaC systems [95] use semantic networks comprised of actors, artifacts, events, and their relations; the BSBW system [11] and Atmosphere model use hierarchical structures of workspaces, to represent the field of work), they are mainly used by the designers to specify events and manage event distributions.

Table 3.2 shows a summary of the major distinguishing features of the *space-based* and *event-based* models. In the following of this section, we will discuss

details on how the design aspects in various cognitive and social processes as shown in Section 3.1 have (or have not) been supported each of these models.

	Space-based models	Event-based models
Field of work	explicitly represented as a 'shared space',	implementation-dependent representations,
	visible to users	invisible to users
Awareness information	embedded as perceivable properties of objects in the space	explicitly represented as discrete events
	presented in the context of its origin	can be outside the user's current viewport
	users need to pull the awareness information from the field of work	awareness informaion is pushed to the users
Example Implementations	1. physical spaces (e.g. Portholes [27]) 2. virtual spaces (e.g. AROMA [69], DIVA [12], MASSIVE-2 [9]) 3. workspaces (e.g. ShrEdit [26], GroupKit [79])	GroupDesk [39], POLIAwaC [95], NESSIE [72], Elvin [36]

Table 3.2: The main distinguishing features of space-based and event-based models

3.2.2 Awareness processes in space-based models

3.2.2.1 Support for perception

Selection The first step to support perception regards the selection of the awareness information for a particular user, i.e. what awareness information should be presented? Systems adopting the *space-based* awareness model usually depend on the various relationships between objects inhabiting in the shared space to decide on this.

Benford et al.'s spatial model [8] provides a formal approach to managing awareness levels between objects inhabiting in a common spatial frame. Awareness between objects is manipulated via *focus* and *nimbus*, two subspaces within which an object chooses to direct either its presence or its attention. Then the aware-

ness information is selected based on the overlapping between the receiver's *focus* and the performer's *nimbus* in certain medium [8]. Anything about the performer happens in the overlapping area will be perceivable to the receiver.

The original spatial model has been extended in several ways to support different awareness systems. Sandor et al. [84] redefined the concepts of *focus* and *nimbus* upon a semantic network that forms a representation of the working context, and use them to build a generic model 'Aether' for supporting awareness in collaborative systems. Rodden [78] adopted an object-oriented approach to generalize the spatial model to provide notions of presence, sharing and awareness applicable to applications lacking a spatial metaphor. Simon and Bandini [93] based on the spatial mode to propose the reaction-diffusion model that can support more user adaptation and dynamic features. Instead of using *focus* and *nimbus*, the awareness information is distributed using field and sensitivity function that allow for a more flexible and compact way of representing various types of nimbi and foci.

Although various space-based models differ from each other in the specific calculation of awareness levels, the most important point emphasized in all of this work is the insistence that the selection of awareness information is a joint-product between the performer and the receiver, i.e. how the receiver directs the attention to the performer (*focus*) and how the performer projects the presence or activity to the receiver (*nimbus*).

Presentation   Presenting the awareness information to the user in space-based systems can be achieved in two ways: presenting the information in place of its origin, or presenting in dedicated awareness widgets [79].

As the awareness information is implicitly embedded as properties of objects in the shared spaces, it is natural to present the awareness information directly in place of the associated object in the general presentation of the field of work. For example, in the DIVA shared workspace [12], the color of document icons indicates the document status (e.g. green means 'modified by others'). As argued by Dourish and Bellotti [26], presenting awareness information in place prevents users from having to switch their attention focus between different information sources. However, this approach relies on the user's capability to perceive and

retrieve information from the shared space, which cannot always be guaranteed [12].

On the other hand, some space-based systems have also explored more lightweight awareness widgets for presenting awareness information [46]. For example, DIVA uses a virtual office table to display participants in a chat room, as well as visualizing their contributions over time[12]. The Babble system [32] includes a 'social proxy' showing team member's level of contribution to a threaded discussion. However, this approach has the major drawback that there is no direct connection between the awareness information and the shared space, which can be disturbing when the user has to switch the focus back forth between the different views. As a result, these lightweight awareness widgets are usually more beneficial for displaying specific aspects of social awareness, such as the collaborator's arrival, availability, involvement, but cannot adequately support task-oriented awareness [19].

3.2.2.2   Support for comprehension

Comparing with event-based systems, support for comprehension can be achieved relatively effortlessly in space-based systems because of two features: the existence of an explicit representation of the shared space, and the possibility of presenting awareness information directly in the shared space. Most existing space-based systems provide visual-spatial displays [48] of the shared spaces, which has a number of advantages to support comprehension. First, they organize information by indexing it spatially. In these displays, space in the display represents space in the field of work, so that if the representation of two items is close in the display, it is likely that those items are also close in the represented field of work. Therefore, information that needs to be related in interpreting and making inferences is likely to be represented by visual features that are close in the display [48]. Second, they provide overviews of the whole field of work, and therefore can be perceived as a whole and provides necessary background for comprehending awareness information [12]. Furthermore, as the awareness information can be directly presented in the place of its origin, the user can offload the cognitive effort to locate the related objects onto perceptual processes [57].

However, space-based systems also have limitations in supporting comprehen-

sion. First, we need to distinguish the difference between the context of *origin* of the awareness information and the context of *work* of the user who need to comprehend the information [41]. Most space-based systems present the awareness information in place of the object where the information occurs on, i.e. within the context of its origin. However, what is more important in the comprehension process is for the user to understand the effects of the awareness information on his/her own work, i.e. within the context of the user's work. Second, the visual-spatial display of the whole field of work can become a difficult or even impractical task when the complexity of the collaboration scales up. When the number of actors, artifacts, and activities increases significantly, such a representation of the whole shared space will also become less efficient to support the comprehension.

3.2.2.3   Support for projection

To our knowledge, explicit support for projection is rarely discussed in existing space-based awareness systems. Most of the space-based systems focus on representing the current state of the 'shared space', but how new awareness information will lead to future changes in the 'shared space' is usually considered as a cognitive task that is performed by human users. In addition, space-based systems represent the field of work using the spatial metaphor, i.e. the objects are connected based on their spatial relationships in the space. The dependency relationships, which are important knowledge to predict future changes, are not represented.

3.2.2.4   Support for feedthrough

In space-based systems, the process of feedthrough, i.e. making consequences of individual activities apparent to other participants, can be achieved either through the direct video-audio links between actors [27], or through the artifact mediation [99].

In media space-based systems, feedthrough can be supported by the networks of audio and video to provide rich representation of people and their immediate surroundings [27]. By seeing others through the media space, it is believed that people can get a sense of others activities so that they can infer the consequences of their activities. However, Fish et al.s study in the use of the Cruiser media space

[35] has shown that even though the media space allows the users to perceive each other's activities, it usually does not provide the visibility of the work artifacts involved in these activities. As a result, showing each other's activities does not necessarily mean the consequences of their activities can be fed through.

As a result, a more common approach to supporting feedthrough in space-based systems is through the common artifacts in the shared spaces, especially in shared virtual spaces and workspace [12]. In these systems, users' activities are organized around the common artifacts in the shared spaces. The artifacts provide the awareness information in how they manifest themselves within the space, and in how they provide the group with feedthrough, i.e., when artifacts are manipulated in some activities, they give off information that informs others the consequences of these activities [99]. However, artifact-mediated feedthrough requires the objects of user's activities can be represented as some artifacts, which may not be applicable in many real world collaboration. For example, the activity of police officers patrolling along the street cannot be easily organized around artifacts.

3.2.2.5 Support for manifestation

The basic way to support manifestation in space-based systems is annotations, which are defined as hypertext nodes that are linked to the objects in the shared spaces [111, 105]. Annotations allow the users to add their comments or interpretations on specific objects or locations in the shared space that are visible to other collaborators. In this way, annotations provide a way for the users to externalize their individual awareness and display them to others. However, annotations in space-based systems are often bound to the objects in the shared space, and therefore have limited capability to manifest other aspects of individual awareness, such as the intentions to perform activities, or state changes of certain activities.

Another more sophisticated way to support manifestation has been integrated in the MoMA system in the term of add-on awareness [93]. The system allows the user to specify field parameters in rule premises to construct add-on awareness promotion. For example, the user can define awareness behavior of the kind: if I receive specific pieces of awareness information, then I will trigger certain interpretation on it, and make the interpretation visible to other actors. The supported

manifestation in this way is a reactive process, i.e. actors react to certain changes in the space, however they cannot actively initiate the manifestation process.

3.2.3 Awareness processes in event-based models

3.2.3.1 Support for perception

Selection   The selection of awareness information in event-based systems focuses on filter out the amount of irrelevant events presented to the users. Many mechanisms that enable the filtering of events have been discussed in the literature, and they vary from each other depending on how the filters are defined.

In the GroupDesk system [39], Fuchs et al. define several filters that allowed the limit of the flow of events. On the actors side there was an individual privacy filter that allowed the actor to set privacy policies for the events gathered about him. On the perceivers side an individual interest filter allowed the perceiver to subscribe only to the events in which he or she was interested. A global filter would allow for the filtering of general conditions (e.g., to comply with organizational policies). Definition of filters in the system is based on individual event types, which consist of a mapping from a set of event types to a list of interested users. The drawback of event type-based filtering is that it requires the user to explicit his/her interest on each event type in a predetermined way, which is a tedious task that the users are not always willing to perform [44]. Furthermore, the type-based subscriptions tend to be too rigid or unable to adapt to the team development [2].

Alternatively, Elvin [36] adopts the content-based filtering of events. It describes events using a set of named attributes of simple data types and consumers subscribe to sets of events using boolean subscription expressions. When a notification is received at the Elvin server from a producer, it is compared to the consumers registered subscription expressions and forwarded to those whose expressions it satisfies. A key benefit of content-based notification is the capability to reduce the effort needed to manage event subscription. Instead of subscribing to each type of events, users can subscribe to event patterns, which can describe a set of events. In addition, content-based subscriptions allow the definition of composite events.

ENI [41] extends the content-based filtering approach and integrates the notion

of 'contexts' into the model. Context information includes locations, artifacts and applications and other information, which is linked to a specific context. ENI adds this information to existing event information in an awareness system. The model is based on two concepts of context. First, the model tries to determine the context of origin for each event. The authors suggest a context mapping mechanism that maps events gathered from sensor information against rules saved in a context database. Second, the model identifies the work context of the user who is receiving the notification. The work context is derived from the selection of shared workspaces. Based on the two types of context, then the filtering is performed on context matching, i.e. the user defines what types of event context he/she wants to receive in a specific work context. The context-based model tries to improve the event filtering by gathering additional information and allowing users to receive awareness information in a more context-specific manner. However, the context mapping mechanisms underlying this concept is highly complex, and it lacks a formal model to specify the two types of context.

Presentation Unlike the *space-based* that relies on the users to monitor the shared space and perceive the presented awareness information, event-based systems make the awareness cues more perceivable for the users than other elements in the environment through different kinds of notification mechanisms [60]. The notification can be done in different ways. In GroupDesk [39] and POLIAwaC [95], different urgency levels are defined to determine the form of presentation of event information at the user interface. A high urgency would typically lead to a disruptive notification, such as popping up a message window, whereas a low urgency could reflect the information by a change of color of the object's icon and leave the details of information to explicit user request. In NESSIE, the presentation of awareness information is performed by configurable indicators, including simple windows for the listing of event information, different background images, or sounds. Tools are offered that allow the users to easily map awareness events to suitable indicators [72].

3.2.3.2   Support for comprehension

Comparing with space-based systems, the comprehension of events provides a more challenging task for the users, as the awareness information is typically presented in separate event-triggered displays peripheral to a person's current task-oriented concern [19]. Moreover, event-based systems often do not provide explicit representation of the field of work. As a result, in order for the user to comprehend an event, he/she has to build and maintain a mental model of the inferential framework derived from the field of work, which increases the cognitive load for the user.

To alleviate this problem, some event-based systems attempt to collocate event notifications within the context of work. For example, NESSIE provides the awareness information by the presentation of pictorial activity indicators for the members of a group as part of the shared workspace user interface [72]. In the virtual school project, Carroll et al. suggest that the deadlines and external events should be collocated with process and planning representations [19]. However, studies on supporting comprehension of events are still very limited. Questions such as what types of events should be collocated with what types of contextual representations are largely undetermined.

3.2.3.3   Support for projection

Similar to space-based systems, explicit support for projection is rarely discussed in existing event-based awareness systems as well. Most systems assume this as a cognitive task that is performed by human users.

3.2.3.4   Support for feedthrough

In existing event-based systems, the process of feedthrough can be supported by a specific type of events, i.e. activity events [39]. Activity events are generated by the system. Each time the state of an object changes due to some action of a user, a new event is generated to describe the change. Then the event is sent to the event processing server, and is distributed to the users who subscribe to it.

3.2.3.5 Support for manifestation

Rittenbruch et al. introduce and explore the notion of "intentionally enriched awareness" [77], which refers to the process of actively engaging users in the awareness process by enabling them to express intentions. They situate the "intentionally enriched awareness" between the event-based awareness systems where actors are not involved at all, and the communication tools where high level of effort from actors is required. This is very similar to our alignment of supporting the three basic awareness propagation processes, and the development of "intentionally enriched awareness" can be considered as part of the manifestation process, as it emphasizes the role of actors to make some of their internal states (intentions, reasons, etc.) along with their activities visible to others. Their AnyBiff prototypical system is one of the limited existing attempts to support *manifestation* in awareness systems, which allows users to generate, share, and use a multitude of activity indicators to reveal intentions. The AnyBiff system is mainly used for supporting social awareness in relatively loose-coupled collaborative activities, and hence the information that the users can make visible to each other is very limited. In complex collaborative environments that we are interested in this study, we believe much more aspects of individual awareness could be made visible by the users. Supporting manifestation involves not only help users express their intentions, but also make their interpretations of awareness information during comprehension/projection, or the results of their decision-making visible.

3.2.4 Summary

Table 3.2 shows a comparison of how the major design aspects of supporting awareness processes have been achieved in the *space-based* and *event-based* models.

Awareness Processes	Space-based models	Event-based models
Percepetion - Selection	1. intersections of focus and nimbus that are defined in various forms [78, 84, 8] 2. comparison of field values agaist sensitivity functions [93]	1. type-based filtering [39] 2. content-based filtering [36] 3. context-based filtering [41]

Perception - Presentation	1. in-place presentation [12] 2. awareness widgets [46, 32]	1. urgency-based notifications [39, 95] 2. configurable indicators [72]
Comprehension	1. visualization of shared spaces 2. direct linking of awaress information to the context of its origin	Collocation event notifications into the display and control of work objects [72, 19].
Projection	rarely supported	rarely supported
Feedthrough	1. video-audio links [27] 2. artifacts mediated [99]	Activity event generation and distribution [39]
Communication	separately supported	separately supported
Manifestation	1. annotations [111, 105] 2. add-on awareness [93]	intention-enriched awareness [77]

Table 3.3: Existing Studies in Awareness Support

Existing systems follow these two types of awareness models have their own pros and cons in supporting awareness processes.

Space-based models have the advantage that the awareness information is presented in the context of its origin, i.e. the associated object in the shared space, and therefore ease the comprehension process. They relies on the users to monitor the field of work and perceive the awareness information. This can be done efficiently if the local scopes of different users are largely overlapped, as the users can pick up the awareness information peripherally as they work on their own work [86]. However, this becomes much more problematic when the activities of users are distributed and each of them is working on a different set of activities, as actively monitoring the shared space requires extra attentions and efforts from the users.

On the other hand, event-based models provides a more lightweight way to present awareness information, as only the aspects of awareness information that is of relevance are presented. Furthermore, the event-based presentation is pushed to the users by making the events more perceivable to the users. In this way, the users do not need to switch their attentions to others activities until the event notification happens. However, awareness information in the form of events is usually presented

separately from the inferential framework to comprehend it, which leads to extra effort in the comprehension process.

3.3   Discussion

The major purpose of reviewing existing awareness systems in Section 3.2 is not to provide an exhaustive evaluation on existing studies. Instead, we use the review to explore design challenges in supporting awareness in complex, distributed geo-collaborative activities that will motivate our approach in the following chapters.

In general, we can identify two major design challenges based on the analysis in Section 3.2.

1. The challenge of scaling up.

Existing systems to support awareness processes are usually designed to support collaborative activities at relatively small and medium scales, it becomes a much more difficult task to support awareness in complex real time activities as we are interested in this study. As we described in Chapter 1, the collaborative activities we consider in this paper are a subset of these complex collaborative setting with two major characteristics: (1) *high level of complexity*: a large number of team workers are geographically distributed in different locations, yet engaged in interdependent activities that require effective coordination. (2) *high level of dynamics*: the actors work in dynamic settings that entail rapid changes in environment and activity, and as a result their specific awareness needs keep changing as the activities are developed.

The two awareness models (*space-based* and *event-based* models) have their own strengths and drawbacks when the collaborative activities scale up.

On the one hand, *space-based* models manage the awareness through the interaction of collaborators and rely on the users to monitor the field of work and perceive the awareness information. This provides more flexibility to handle increased level of dynamics. As the whole field of work as a shared space is explicitly visible to each user, they can pick up the awareness information relevant to them even when their own interests have been changed. However, space-based models become much more problematic when the level of complexity in the field of work increases. When the number of objects, actors and their activities is significantly

increased, the whole field of work will becomes extremely large. Representing the whole field of work and relying on the users to monitor and retrieve awareness information from it becomes a challenging task, as it requires a lot of extra attentions and efforts from the users.

On the other hand, *event-based models* provides a more lightweight way to present awareness information, as only the aspects of awareness information that is of relevance are presented. Furthermore, the event-based presentation is pushed to the users by making the events more perceivable to the users. In this way, the users do not need to switch their attentions to other's activities until the event notification happens. As a result, event-based models can handle more complex situations than space-based models. However, the effectiveness of event-based models largely depends on the quality of event subscriptions, which often requires that considerable domain knowledge be explicitly embedded. As argued by Fuchs et al. [38], event-based systems seem to work satisfactorily for situations where workflow can be clearly defined in advance or if the application is known from the beginning. When the level of dynamics increases in collaborative activities, such condition can no loner hold. The user's awareness needs are often in the flux of changes, as their activities evolve. Hence, the event-based models becomes less effective in high level of dynamics.

The above analysis clearly shows that neither space-based nor event-based models in existing studies can effectively support awareness processes in collaborative activities when both the complexity and dynamics scale up. Therefore, a new awareness model that can leverage the strengths of both space-based and event-based models becomes extremely important.

2. The challenge of integrated support.

As we conceptualize the awareness phenomena in complex, distributed collaborative activities as a continuous development of awareness knowledge through a variety of integrated cognitive and social processes, it is very important that all the individual and team processes are well supported. However, the review of existing awareness systems shows that some awareness processes have very limited support in existing studies.

At the individual level, existing awareness systems have focused on supporting perception, the higher level of awareness processes are relatively less supported. A

possible reason is that the higher-level awareness processes usually involve sophisticated cognitive and reasoning capabilities where the human can do better than the computer. As a result, most awareness systems leave the comprehension and projection to the users. However, in complex collaborative activities, as we discussed earlier, the scaling problem becomes significant, and hence it becomes much more important for the computer to provide functions to amplify and enhance human cognition, not only in the stage of perception, but also in comprehension and projection.

At the team level, existing awareness systems have focused on either supporting the explicit communication among human collaborators, or different approaches to providing 'shared spaces' representing the fields of work so that the actions of each other become visible to each other. However, less discussion has been given to support the awareness interaction via mutual displaying and monitoring, i.e. the manifestation process, which actually is an even more important aspect of awareness in many complex, real world collaborative activities [47].

As a result, the second design challenge in supporting awareness in complex collaborative activities is to consider the awareness support from a collective perspective and provide integrated support for the whole awareness development cycle.

Chapter 4

Our Approach: Overview

Our approach to supporting awareness in complex, distributed geo-collaborative activities aims to address the two major design challenges identified in Section 3.3:

1. As neither space-based nor event-based models in existing studies can effectively support awareness processes in collaborative activities when both the complexity and dynamics scale up, a new awareness model that can leverage the strengths of both space-based and event-based models becomes extremely important.

2. By conceptualizing the awareness phenomena in complex, distributed collaborative activities as a continuous development of awareness knowledge through a variety of integrated cognitive and social processes, it is very important to consider the awareness support from a collective perspective and provide integrated support for the whole awareness development cycle.

This chapter provides an overview of our approach. The general design principle of our approach is to emphasize the active role of computer system to not merely support, but rather promote awareness in complex collaborative activities. In the following, we first describe the design concept of awareness promotion, and then describe the major components of our awareness promotion framework.

4.1 From Awareness Support to Awareness Promotion

In order to address the design challenges of scaling up and integrated support, we argue that the computer system needs to play an active role to promote awareness among collaborators. Our awareness promotion approach has its basis in two basic parent disciplines: artificial intelligence (AI) and human-computer interaction (HCI).

1. From AI, we consider the computer as *an active agent* [15], situated within the collaborative environment that adaptively sense, acts, and reacts within this environment over time, to pursue its goal of promoting awareness.

2. From HCI, we act on the premise that computers and humans have fundamentally asymmetric abilities [25]; therefore, we focus on developing divisions of responsibility that exploit the strengths and overcome the weaknesses of both, and utilizing interaction techniques to facilitate effective *human-computer collaboration* [100].

4.1.1 The role of computer: from tool to actor

In existing awareness systems, the role of computer can be described in the metaphor of 'tool'. The 'tool' perspective emphasizes the human achieves his/her goal through the computer application [13]. In term of supporting awareness, it means the human actors use the computer to achieve awareness. The tool perspective emphasizes the control on the human user's side, and how the computer is used to achieve awareness depends on the human user's own knowledge. For example, in space-based systems, the system relies on the user to monitor the shared space to perceive awareness information; in event-based systems, the user needs to explicitly express his/her interests so that the computer can filter out events.

However, when the collaborative activities become more and more distributed and complicated, more is demanded of the user to control the computer. As we conceptualize the awareness phenomena as a distributed cognitive system (Section 2.4), each user usually only have partial knowledge about the whole collaborative

situation, and it is unlikely that the user will have the full knowledge of what exactly should be done in order to achieve awareness as a group. The user may have no idea what background information should be attended to in order to interpret a piece of awareness information that is out of his/her local scope of work. Or he/she cannot decide on who should be notified when his/her activities are changed. As a result, the user either uses the partial knowledge to develop awareness that is often incomplete, vague, or even incorrect; or has to extend the mental model to represent more knowledge that out of his/her local scope of work, which inevitably increases the user's cognitive load.

As a result, we believe that merely relying on the user to control the computer as a 'tool' to achieve awareness in distributed, complex collaboration is ineffective. Alternatively, a better approach is to allow the computer to take some control away from the user and actively perform some tasks to help the user to achieve awareness. Instead of relying on the user's internal, partial knowledge to control the awareness development, the computer can maintain a much more complete knowledge representation of the whole field of work at the systematic level, and make use of this knowledge to infer the user's need in the various awareness processes so as to provide awareness support. In this way, the computer can be considered as a 'mediator' actively engaged in the collaborative activities along with human actors, but with its own goal (i.e. to assist human actors to achieve awareness), its own mental model (i.e. the knowledge representation of the whole field of work), and its own behaviors (i.e. the capabilities to reason about the user's need in the various awareness processes and adapt interaction and information presentation techniques to support it).

4.1.2 Human computer collaboration

While considering the computer as an active actor to support awareness, it does not mean we can delegate all the tasks to the computer and build a completely automatic system. Existing studies in HCI and cognitive science have suggested that the relationship between human users and artificial intelligence should be treated as joint cognitive systems [25] or human-computer collaboration systems [100], emphasizing the appropriate partition of responsibility between human ac-

tors and computer systems. This perspective of human computer interaction is based on two premises.

1. It begins with the premise that computers and humans have fundamentally asymmetric abilities. The computer's strength lies in its ability to store more information than can be stored in the human's short-term memory, to perform faster data retrieval and processing, and to conduct more reliable low-level routine inferences [15]. On the other hand, the human's strength lies in the ability to integrate information from multiple sources to provide insight necessary to draw complex, higher level inferences, and the capability to handle unexpected situations with the help of long-term memory, experiences, and even intuition. As a result, the goal of computer support is actually to investigate the relationship between the cognitive characteristics of the human and the cognitive characteristics of the computer, and get the computer to complement the human [25].

2. It assumes that the human and the computer be cooperative to each other in the interaction. It means not only the computer will keep track of the human's activities and adapt its behaviors to support human performance, but also the human is willing to exposing his/her goals and activities to the computer, helping system maintain the knowledge representation, giving away control to the computer, and modifying the computer's behaviors when necessary [100]. The goal is not to shield users from the complexities of interaction, but rather to focus on the use of interaction techniques to facilitate effective human-computer collaboration.

In our approach, we also base on these two premises to propose supporting the awareness development in distributed, complex collaboration as a human computer collaborative system. While human actors still need to undergo the cognitive processes to develop individual awareness, and use it to make decision and perform activities in their own local scopes; the computer system take the responsibility to maintain a collective picture of the whole field of work, and utilize this knowledge to facilitate the various cognitive and social awareness processes among human actors.

In sum, we use the term 'awareness promotion' to define the paradigm of awareness support that follow these two design principles: (1) the computer plays the role as an 'actor' actively engaged in the collaborative activities along with human actors, maintains a collective knowledge representation of the whole field of work, and uses it to adapt behaviors to support the various awareness processes; (2) the computer provides adequate interaction techniques to allow the human actors to collaborate with it to develop awareness at both the individual and team level.

Table 4.1 illustrates the major differences between the traditional awareness support paradigm and awareness promotion in addressing the different awareness processes. From the table, we can clearly see some of the distinguishing characteristics of awareness promotion.

1. First, the awareness promotion approach emphasizes the computer's knowledge representation of the field of work, and its relationship to awareness information. One one hand, the awareness information is used to update the computer's knowledge representation so that it matches the current state of the field of work. On the other hand, the knowledge representation is used by the computer in almost every awareness process to perform a variety of reasoning tasks to support awareness.

2. Second, comparing with the traditional awareness support, the awareness promotion approach shows much more frequent interactions between the computer and the human users. Each awareness process involves both the computer's reasoning and the human's cognition, as well as the interaction to combine them together.

Awareness processes	Awareness support	Awareness promotion

Percepetion	*Space-based* models: • the *computer* shares information in a common space • the *human* monitors the shared space to perceive information *Event-based* models: • the *human* subscribes to events based on interests • the *computer* filters out events based on human subscription	• the *computer* uses the information to update its knowlege representation • the *computer* uses its knowledge representation to infer who the information is relevant to, and present it only to the relevant actors • the *user* can modify the computer's knowledge representation
Comprehension	*Space-based* and *event-based* models: • the *computer* links the awareness information to the context of its origin • the *human* infers the connection between the context of origin and his/her work context	• the *computer* connects the awareness information to the human's work context • the *human* interprets the information, and sends the result to the computer • the *computer* updates its knowledge representaton based on human interpretation

Projection	*Space-based* and *event-based* models: • the *human* performs the projection on the own	• the *computer* performs the projection based on its knowledge representation • the *computer* presents the projection results to the human • the *human* reviews and modifies the computer's reasoning • the *computer* updates its knowledge representation based on human modification
Feedthrough	*Space-based* models: • the *computer* broadcasts the effect to everyone *Event-based* models: • the *computer* notified the effect to all subscribers	• the *computer* uses the effect to update its knowlege representation • the *computer* uses its knowledge representation to decide on who should receive the effect and notify them • the *human* can control who can see the effects of his/her activities

Manifestation	*Space-based* models:
	- the *human* creates an annotation - the *computer* makes it visible to everyone *Event-based* models: - the *human* indicates his/her intention as an event - the *computer* notified the event to all subscribers	- the *human* generates awareness information indicating some aspects of his/her individual awareness - the *human* can control who can see the generated awareness information - the *computer* uses the generated awareness information to update its knowlege representation - the *computer* uses its knowledge representation to decide on who should receive the information and notify them

Table 4.1: Awareness support v.s. awareness promotion

We believe these distinguishing characteristics of awareness promotion approach provide the potential to address the two design challenges, i.e. handling the scaled up complexity and dynamics, and providing integrated awareness support. First, the awareness v approach utilizes the computational knowledge representation to model the field of work and offloads some of the representation and reasoning efforts from the human to the computer. Hence, it can handle more complex situations than existing awareness models. Meanwhile, the knowledge representation is dynamically updated to reflect the current state of the field of work, which allows it to handle increased level of dynamics. Second, as the computer is equipped with the computational knowledge representation and reasoning capabilities, it allows the computer to provide support on the higher level of awareness processes.

4.2   The Awareness Promotion Framework

To operationalize the awareness promotion approach, we propose a computational awareness promotion framework. Following our conceptualization of the awareness phenomena in Section 2.4, our approach is built on top of two major knowledge components: a computational representation of the field of work based on the SharedPlan theory, and an event-driven model of the awareness processes. Then the computer system's behaviors to promote awareness are embedded in the interaction between these two components. On one hand is how the computer constructs and develops the knowledge representation of the field of work within the event-driven processes, and on the other hand is how the knowledge representation is used to promote these event-driven awareness processes.

This section provides an overview of the awareness promotion framework and discusses the major design choices, but leave the details to the next few chapters.

4.2.1   Computational representation of the field of work

The nature of awareness phenomena in distributed, complex collaboration as we conceptualized in Section 2.4, and the goal of promoting awareness impose several requirements for the knowledge representation of the field of work:

1. The knowledge representation should provides a precise model that can capture knowledge about all the three constructs that compose the field of work, i.e. the basic elements of human activities, the local scope of work for each actor, and the various dependency relationships among the activities.

2. The knowledge representation should be dynamically updated. Since the awareness needs of users change quickly with the field of work in distributed, complex collaboration, it is essential that the knowledge representation should be kept updating so that it always reflects the current state of the changing field of work.

3. The knowledge representation needs to be formalized and can be computationally modeled so as to support computer reasoning.

Existing awareness systems have provided several approaches to representing the field of work. The AREA system [38] describes the field of work as semantic networks including relationships among objects, where objects can be human actors artifacts, or aggregations such as groups of people. The representation is primarily used for the users to specify which objects and associated events they are interested in and when they want to be informed. The Atmosphere model [75] describes the field of work as a hierarchically structured workspace that consists of a set of *spheres* and *sub-spheres*. Users classify their actions on artifacts by mapping them into different spheres. The MoMA model [93] applies a reaction-diffusion metaphor to model the field of work as a set of entities embedded in an interaction space, which behave by using diffusion and reaction capabilities. This metaphor is based on the idea that whenever two or more entities have contact, their states are modified in some way. Their states are then propagated to others through fields in the space.

Although these representations have shown their capabilities to support specific aspects of awareness in their respective application domains, none of them can meet all the three requirements for knowledge representation to enable awareness promotion. The AREA model provides a good representation of the activities and the dependencies using the semantic networks, but the local scopes of each user are not captured. The model is static (as it is pre-determined by the designers), and informal (as it is primarily used as a descriptive framework). The Atmosphere model organizes the field of work around the artifacts without explicit representation of activities. It supports the specification of each user's local scope using private 'spheres', but no dependency relationships between activities are supported. It supports the modification to the representation by human users, but the system cannot automatically adapt the representation to changing environment. The MoMA model can support all the three constructs of the field of work as the definition of entities and spaces are generic, and provide some reasoning capabilities. However, it does not support the dynamical adaptation of the model.

In this study we draw knowledge representation and reasoning techniques from existing studies in artificial intelligence to develop a computational model of the field of work that can satisfy all the three requirements. An important feature of our model is the focuses on intentions, beliefs, knowledge, and other attributes

of actors' mental states [42]. As human actors' awareness processes are directed by their mental models (Section 2.4.2.1), we believe that understanding and representing human actors' mental states allow the computer to infer each actor's local scope, and derive awareness needs from it. We will describe details of the computational representation of the field of work in Chapter **??**, and the dynamic adaptation behaviors in Chapter **??**.

4.2.2 Event driven awareness processes

In our approach, we adopt the general event-driven interaction paradigm [34] to model the awareness processes. In an event-driven mode of interaction, actors (both human and computer) communicate by generating and receiving events. Each actor receives events from environment and other actors, reacts to them, and generates new events to other actors. We believe that the event-driven paradigm is suitable for modeling awareness processes in our study for two major reasons:

1. The collaborative environments under discussion in this study are naturally centered on events. As we focus on the set of physical collaborative activities that entails rapid and frequent changes in environment and activities, human activities are usually triggered by the detection of events, and then are performed to analyze and react to these events. In the motivating scenario of emergency response, for example, the collective efforts are triggered by an incident, or a disaster, i.e. an event happening in the environment. To respond to the initial event, actors perform activities that add more events building up the situation. In addition, unexpected new events can occur in the environment, requiring continuous response [103]. It is these critical events that form the actors' awareness needs and cause the actors to adopt certain goals and perform activities to resolve possible problems.

2. The event-driven paradigm provides an effective way to represent and process awareness information. Instead of asking the actors to monitor the environment and pull awareness information from it, events are pushed to the actors as they occur. In this way, the actors do not need to split their limited attentions to monitor the environment and each other's activities. This is very important for supporting awareness, as one general design concern for

awareness system is that awareness must be achieved with minimal attention and effort so that it does not interrupt the current line of action [86].

Our event-driven model of awareness processes shares some commonality with existing event-based models, as both use events as the basic unit to organize and present awareness information. However, they also differ in several important ways:

1. In existing event-based models, the computer primarily plays the role as producers of events. It detects events from sensors in the environment or feedbacks of human actions, filters them out based on user subscriptions, and present them to the users. However, in our approach, the computer is also the consumer of events. As to maintain the knowledge representation of the field of work, the computer needs to take the events as input of its reasoning process, and use them to make inference and update its knowledge representation. In this way, the computer behaves like other human actors to consume the events to develop its own awareness.

2. The concept of events in our approach has a much richer meaning than existing event-based models. It is not only used to describe the occurrences in the environment and in human activities, but also the psychological experience of these occurrences as human actors perceive and interpret them. We make a clear distinction between real world occurrence, event, and awareness in Section 4.2.2.1.

3. Existing event-based models focus on the generation and presentation of events to the users, but our approach emphasizes the whole process of awareness development driven by events. This means we are not only concerned about how to select and present events to the users, but also how to support the interpretation of these events, and how new events are generated based upon existing ones.

In sum, we consider the event not only as a way to represent awareness information, but also an effective interaction mode to drive the awareness development. Each round of awareness development is triggered by receiving certain events, and finished with generating new events that will be consumed by other actors. In

the following, we first clarify our concept of event, then describe the event-driven awareness processes.

4.2.2.1   The concept of event

Before going further we should clarify what we mean by an event. The concept of the word 'event' has been used in the literature from different perspectives:

1. The first meaning refers to an actual occurrence (the something that has happened) in the real world. Set out by Quine [73], events in this meaning are first-class entities that can be localized in space and time, broken into sub-parts, and arranged in a taxonomic hierarchy. Research using this concept of events primarily focuses on studying the internal structures of the events. For example, in [109], complex geographical phenomena, such as wildfire or precipitation, have been modeled in a hierarchy of events, processes, and states. In [4], the event-based approach has been adopted to model all the types of occurrences in movement analysis. The focus here is to derive the different event types that may occur and identify the relationships between them.

2. The second meaning takes us into the realm of computerized event processing, where the word 'event' is used to mean a programming entity that represents this occurrence [96]. Each event in this notion is a message that describes an real world occurrence by its source, location, time, and other measurable properties. A single event occurrence can be represented by many event entities, and a given event entity might capture only some of the facets of a particular event occurrence [34].

3. In event-based awareness systems, the concept of event has a more specific meaning as representing a specific type of awareness information that might be relevant to the user's work. It is usually an application-specific concept, depending on the set of awareness information that is supported by an awareness system. For instance, the AREA system [38] describes events as actions performed on an artifact and the event classes are derived from the artifact

class hierarchy and possible operations on them. The ENI system [41] describes events from the sensors associated with actors, shared artifacts, or any other objects that generate events related to them.

In this study, we use three different words to avoid the confusion when defining events:

1. We use the word *'occurrence'* to denote the real world happenings. It can be in the environment, e.g. the occurrence of a traffic accident at a location; or associated with an object, e.g. a vehicle arrives at the accident spot; or associated with human activities, e.g. the successful performance of first aid on a victim.

2. The word *'awareness'* is used to refer to the human consciousness about the occurrences, events, and their relevance to ongoing or future human activities. It emphasizes that awareness is inherently a cognitive, interpretive, and predicative concept that reflects a state of mind.

3. We use the word *'event'* to denote a computerized entity that describes a piece of awareness information that is relevant to an actor's work. On one hand, it can be the description of a real world *occurrence* by its measurable properties, e.g. the information about a traffic accident with time, location, number of victims etc. On the other hand, it can also be the externalization of an actor's *awareness* knowledge, e.g. an actor's belief that the accident will block the traffic and cause a delay on delivery of medical team.

4.2.2.2 Awareness processes

Along with the distinction between the concepts of *'occurrence'*, *'awareness'*, and *'event'* is the notion of dynamic transformations between them (Figure 4.1). The transformations are tied to different processes in awareness development.

The direct transformation between *occurrence* and *awareness* corresponds to the individual awareness development cycle that has been described in Section 2.2.2. A real world *occurrence* is transformed into *awareness* through an actor's individual awareness processes, i.e. *perception*, *comprehension*, and *projection*.
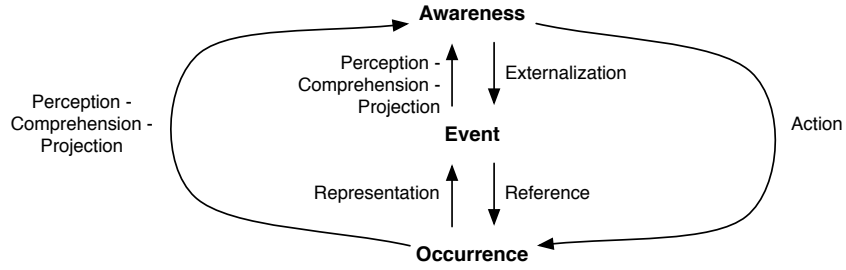
Figure 4.1. Transformations between occurrence, event, and awareness

Then, the achieved *awareness* guides the actor's *action* that may generate further *occurrences* in the real world.

The thing becomes more interesting when the computer support is involved and the transformations are driven by *events*. One one hand is a real world *occurrence* can be captured and represented as an *event* by the computer system, and presented to a human actor. Instead of perceiving the *occurrence* directly, the actor perceives the corresponding *event* in the computer interface, and develops the *awareness* upon it through the actor's individual awareness processes, i.e. *perception*, *comprehension*, and *projection*. On the other hand is some aspect of the actor's *awareness* can be externalized as a new *event*, which refers to some current *occurrence* or predicts future *occurrence* in the real world.

The event-driven transformations can also be used to describe the different awareness propagation processes across multiple actors. The process of *feedthrough* can then be described in the following steps: (1) an actor's *awareness* guides his/her *action* that generates a new *occurrence* in the real world; (2) this new *occurrence* is then captured and represented as an *event* by the computer, and presented to another actor; (3) the other actor develops the *awareness* upon receiving the new event. Similarly, the process of *manifestation* can also be described as follow: (1) an actor's *awareness* is externalized as a new *event*, which refers to some current *occurrence*; (2) this new *event* is then presented to another actor by the computer; (3)the other actor develops the *awareness* upon receiving the new event.

Figure 4.2 shows an example of how these event-driven awareness processes can be combined together to describe a complex trajectory of awareness development

that involves three actors in a group. The awareness is propagated from *Actor 1* to *Actor 2* through the process of *manifestation*, and is then propagated from *Actor 2* to *Actor 3* through the process of *feedthrough*.
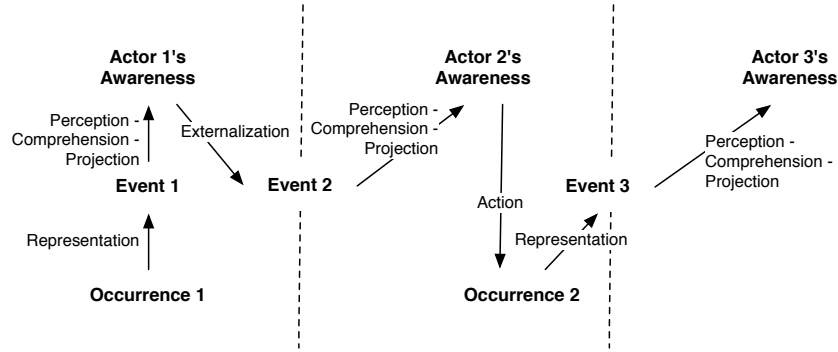
Figure 4.2. An example of event-driven awareness development trajectory

4.2.3   The framework

Built on top of the computational representation of the field of work, and the event-driven model of the awareness processes, our awareness promotion framework focuses on the interaction between these two components (Figure 4.3).
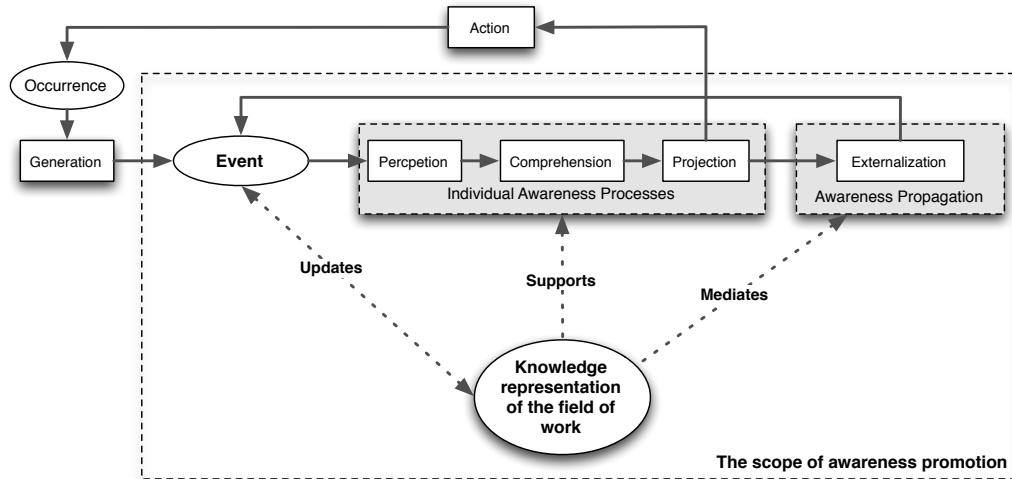
Figure 4.3. Awareness promotion framework

On one hand is how the computer constructs and develops the knowledge rep-

resentation of the field of work within the event-driven processes. As we argued in Section 4.2.1, one of the requirements for the computational representation of the field of work is that it should support dynamic adaptation so that it always reflects the current state of the changing field of work. We utilize the *events* to achieve this goal. As human actors use a subset of the events to develop their individual awareness, the system processes every event to develop its knowledge representation of the whole field of work. The events can be generated by sensing the occurrences in real world, or they can be the results of human actors' externalization of their individual awareness. All of these events will be processed by the computer system to update its knowledge representation of the field of work.

On the other hand, the knowledge representation is used by the computer system to promote the awareness processes. Following the design principle of human-computer collaboration emphasizing the appropriate partition of responsibility between human actors and computer systems (Section 4.1.2), the promotion role of the computer system is to strike a balance between the system's reasoning capabilities and providing visual and interactive support. With the computational knowledge about the field of work, the system can offload some of the reasoning effort from the human actors. Meanwhile, the system knowledge can also be visualized to help the human actors perform their part of the work or overwrite the computer's behaviors.

As shown in Figure 4.3, we exclude the event generation process, i.e. the transformation from real world occurrences to events; and the action process, i.e. the transformation from awareness to real world occurrences, from the scope of awareness promotion in this study. It does not mean that these processes are less important or they can be separated from the rest. Rather, we believe these processes are relatively standalone processes that are out of the scope of this study, and therefore we do not claim contributions to these processes.

The next two chapters provide details of this awareness promotion framework. Chapter **??** describes the two knowledge components, i.e. the computational representation of the field of work and the specification of events, and the mechanisms for updating the knowledge representation with events. Chapter 6 describes the promotion of individual awareness processes and awareness propagation.

Chapter 5

Our Approach: Knowledge Representation and Updating

In this chapter, we describe the two major knowledge components: the computational representation of the field of work, and the events. We first provides a formalization of the field of work using a first-order logic, augmented with several modal operators and meta predicates. The formalization is not intended to be directly implemented, rather it is intended to be used as a specification for designing knowledge representation and reasoning capabilities. We then discuss how the formalization can be computational represented using the PlanGraph model. Then, we discuss the specification and typology of events in our approach.

After defining the two major knowledge components, we discuss the the knowledge updating process, which covers two important issues: on one hand is updating the knowledge representation of the field of work so that it always reflects the current state of the collaborative activity by reacting to the various external and internal events through several reasoning processes. On the other hand, these reasoning processes also enrich these events with system generated knowledge that is used to promote the human actor's awareness processes discussed in next chapter.

5.1 Formalizing the Field of Work

In general, the field of work in a collaborative activity can be defined as a domain model that includes three related sub-models $FoW = \{ER, LS, DEP\}$: the model

of basic entities and their relations ER, the set of all actors' local scopes LS, and the set of dependencies among activities DEP. ER provides the basic vocabulary for defining local scopes LS and dependencies DEP. We elaborate on the three components in the rest of this section.

5.1.1 Entities and relations

5.1.1.1 Entities

Following the conceptualization of the field of work in Section 2.4.1, we define three basic entities that are the basic building blocks in a collaborative activity:

1. **Actors** are defined as human actors participated in a collaborative activity, capable of making decisions and performing actions. $AR = \{ar_1, ar_2, ..., ar_n\}$ is the set of all the actors in a collaborative activity.

2. **Resources** are anything that can be used in the transformation process of an activity, including both material resource and resources for thinking. We use $RES = \{res_1, res_2, ..., res_n\}$ to denote the set of resources used in a collaborative activity.

3. **Actions** include all the actions that have been performed, is under execution, or will be performed in a collaborative activity. $ACT = \{a_1, a_2, ..., a_n\}$ denotes the set of actions.

We presume all the basic entities fall into types. Each *action type* represents a class of actions that can be performed by executing a set of subsidiary tasks, commonly called a *recipe*. For example, an actor might rent a car by walking to the rental car center, selecting the car, paying the money, and so forth. Each *resource type* represents a class of resource objects that share the same list of properties and behaviors. For example, every car at the rental car center has a make, model, rental price, etc. Each *actor type* represents a role that abstracts the behaviors of a social actor within some specialized context or domain of endeavor [14]. Its characteristics are easily transferable to other social actors. For example, every actor at the rental car center in the role of 'receptionist' can help the customer to rent a car.

We follow Hunsberger and Ortiz [52] to use the @ constructor to make the distinction between types and instances of entities in the formalization. Each type expression is defined as a unique keyword (e.g. $AType$), then each instance is in the form of the type keyword followed by a set of arguments necessary to identify it (e.g. $AType@arg_1(val_1)@arg_2(val_2)...@arg_n(val_n)$). For example, $drive$ can be an action type and an instance could be $drive@car(car_3)@from(loc_1)@to(loc_2)$ that specifies an instance of driving car_3 from location loc_1 to loc_2. $receptionist@name(Mike)$ is an actor named $Mike$ in the role of $receptionist$. To simplify the expression, we may sometimes use a type keyword followed by a subscript to define an instance. For example, $receptionist_1$ can be used to define an instance of $receptionist$.

5.1.1.2 Relations

The relations between these entities can be defined using predicate symbols. For example, predicate symbol $Located_in$ can be used to indicate the spatial relation of one entity is inside the second entity. The fact that actor ar_1 is in the office $room_3$ can be expressed as $Located_in(ar_1, room_3)$. The list of predicate symbols and their meanings are usually domain dependent. However, here we are more interested in defining a subset of relations between these entities that reflect the common characteristics of activity structure, and are used later to define the local scopes LS and dependencies DEP.

Actor's intention towards action   We follow the SharedPlans theory [42] to define three types of intention predicates, $Pot.Int$, $Int.Th$, and $Int.To$, to represent the *intentions* towards an action that have been adopted by an actor.

1. $Pot.Int$ is used to indicate that an actor would like to adopt an intention that the action be performed, but which has not yet be committed to.

2. $Int.Th$ is used to represent an agent's intention that the action be performed.

3. $Int.To$ is used to represent an agent's intention to do some action.

The major difference between these three is the level of commitment the actor has on the action. $Pot.Int(ar_1, act_1)$ indicates that actor ar_1 is considering adopting the intention that the action be performed, but has not yet committed

to it. $Int.Th(ar_1, act_1)$ indicates that actor ar_1 has intended that the action act_1 be successfully performed, but it could be performed by other actors. The actor may commit to help the other actors to get it done, or avoid conflicts, but is not participated in the performance of the action. In the proposition $Int.To(ar_1, act_1)$, the actor ar_1 has full commitment to perform action act_1, i.e. ar_1 will participate in the performance of the action act_1.

In addition to the three intention predicates, we also define the $Perform$ predicate to indicate that the actor is in the process of performing the action.

Actor's capability of performing action   The *capability* of an actor to perform an action can be defined using three predicates:

1. *Knows* is used to indicate that the actor has the knowledge about how to perform the action, i.e. the actor has at least one recipe to perform the action.

2. *Able* indicates that the actor has the ability (e.g. expertise, skills, resources) to perform the action.

3. *Workable* indicates that the actor can actually perform the action by satisfying all the constraints and pre-conditions.

The three *capability* predicates describe different levels of capability of an actor on an action. $Knows(ar_1, act_1)$ merely says that ar_1 knows how to perform the action act_1, but may not be able to perform it. For example, a blind man may know how to drive a car, but do not have the ability to drive it. Furthermore, $Able(ar_1, act_1)$ indicates that the ar_1 has the ability to perform the action act_1, but it does not necessarily mean the actor can actually perform it. A man who is able to drive the car may be located in a different country, and hence he cannot actually drive it. $Workable(ar_1, act_1)$ defines the highest level of capability, as the ar_1 can meet all the constraints and actually perform the action act_1.

Relations between actions   An important characteristic of actions is that they can be either basic or complex. A *basic* action can be directly executed. A *complex* action, however, cannot be directly executable because it needs to be decomposed

into subsidiary actions. The relations between an action and its subsidiary actions can be represented by the predicate *Sub.Act*. $Sub.Act(act_1, act_2)$ indicates that act_1 is a subsidiary action of performing act_2 within the current plan. The *Sub.Act* predicate only indicates the relationships between these two actions under the context of current plan. If the plan is changed, the relationship may no longer hold.

Besides the *decomposition* relation between actions, the actions can also be related by the *precedence* relation. The *Precedes*) relation defines the temporal order of performing two actions at the same level. For instance, $Precedes(act_1, act_2)$ indicates that act_1 should be performed before act_2.

Relations between action and resource   We use two predicates, *Consumes* and *Produces* to define the relations between an action and a resource. $Consumes(act_1, res_1)$ refers to the situation that the performance of action act_1 requires the use of resource res_1. On the other hand, $Produces(act_1, res_1)$ indicates that action the performance of act_1 will produce the resource res_1, or make the resource ready for other actions.

add a table to summarize all the relations described in this section.

5.1.2   Local scopes

We define the local scopes of actors based on two types of relations between actors and actions, i.e. *intention* and *capability*, defined in Section 5.1.1.2.

1. The *intention* relations define the set of actions that an actor has intention or potential intention towards their performance. As we assume the actors' behaviors be goal-driven [63], this set of actions can be used to define the sub-space of the field of work that the actor is willing to work on, we call it *local scope of intention*.

2. The *capability* relations define the set of actions that the actor have certain level of capability to work on. This set of actions define the sub-space of the field of work that the actor can have impact on, we call it *local scope of capability*.

The whole local scope of work for an user is then defined as the union of these two sub-spaces defined by the *intention* and *capability* relations (Figure 5.1). The actions within the intersect of these two sub-spaces are these that the actor actively participate in. The actions within the *local scope of intention*, but outside the *local scope of capability* are these that the actor may need help from other actors. The actions outside the *local scope of intention*, but inside the *local scope of capability* are where the actor can offer help to other actors.
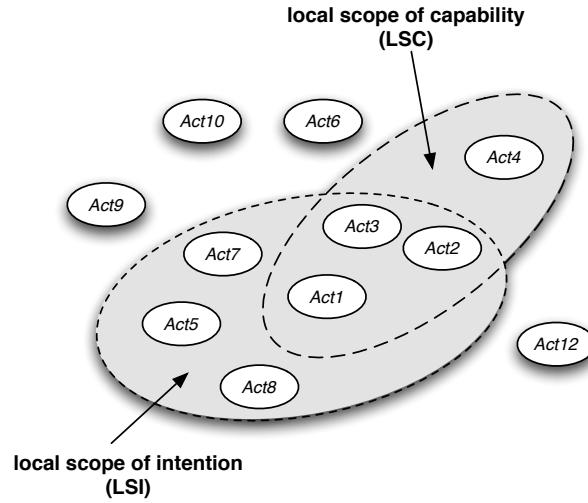
Figure 5.1. The structure of local scope of work

One thing to note is that the actions within the same sub-space can have different relations with the actor. That is, the actions in the *local scope of intention* can be either potentially intended ($Pos.Int$), intended that it be performed ($Int.Th$), or intended to be performed ($Int.To$). Similarly, the actions in the *local scope of capability* can also be related to the actor with different levels of capability (i.e. $Knows$, $Able$, $Workable$). As a result, we cannot define the local scope for each actor just as a set of actions, but also with the corresponding relations.

Following this, we define the *local scope of intention* for an actor $LSI(ar)$ as a set of tuples, and each tuple includes three elements: the actor, the action, and the intention relation between them:

$$LSI(ar) = \{< ar, act, rel(ar, act) >\}$$

where $act \in ACT$ and $rel \in \{Pos.Int, Int.Th, Int.To, Perform\}$.

Similarly, the *local scope of capability* for an actor $LSC(ar)$ is defined as:

$$LSC(ar) = \{< ar, act, rel(ar, act) >\}$$

where $act \in ACT$ and $rel \in \{Knows, Able, Workable\}$.

Thereafter, the *local scope of work* for an actor $LS(ar)$ is defined as:

$$LS(ar) = LSI(ar) \cup LSC(ar)$$

5.1.3 Dependencies

We can now focus on the modeling of dependencies between actions. In general, a dependency is defined as a meta-predicate (DEP) on two actions (act_1, act_2) and a proposition p, where act_1 depends on act_2 because of some proposition p, and is denoted by $DEP(act_1, act_2, p)$. We follow the terms used in [108] to call the depending entity act_1 the *depender*, and the entity who is depended upon act_2 the *dependee*, and the proposition representing the dependency relation *dependum*.

Based on the various types of relations between actions and resources defined in Section 5.1.1.2, we can use them to define the various types of dependencies described in our conceptual framework in Section 2.4.1.3.

Temporal dependency   The predicate $Precedes$ can be used to define the temporal dependency between actions, i.e. the performance of the action act_2 as *depender* depends on the performance of the action act_1 as *dependee*, because act_1 is a pre-requisite action that must have been completed at the time when a_2 is performed. $Precedes(act_1, act_2)$ is the *dependum* in this case.

$$DEP(act_2, act_1, Precede(act_1, act_2))$$

Resource dependencies   The predicates $Consumes$ and $Produces$ can be used to define the three types of resource dependencies.

A *fit dependency* occurs when two actions (act_1, act_2) collectively produce the same resource (res_1). In this case, the two actions are mutually dependent on each

other, i.e. both can be *depender* and *dependee* at the same time. The *dependum* is the combination of two predicates: $Produces(act_1, res_1)$ and $Produces(act_2, res_1)$.

$$DEP(act_1, act_2, Produces(act_1, res_1) \land Produces(act_2, res_1))$$
$$DEP(act_2, act_1, Produces(act_1, res_1) \land Produces(act_2, res_1))$$

A *flow dependency* arises whenever one action act_1 produces a resource res_1 that is used by another action act_2, i.e. the performance of the action act_2 as *depender* depends on the performance of the action act_1 as *dependee*, because of the two predicates: $Produces(act_1, res_1)$ and $Consumes(act_2, res_1)$.

$$DEP(act_2, act_1, Produces(act_1, res_1) \land Consumes(act_2, res_1))$$

A *sharing dependency* arises whenever multiple actions both use the same resource. Similar to a fit dependency, the two actions are mutually dependent on each other, because of the two predicates: $Consumes(act_1, res_1)$ and $Consumes(act_2, res_1)$.

$$DEP(act_1, act_2, Consumes(act_1, res_1) \land Consumes(act_2, res_1))$$
$$DEP(act_2, act_1, Consumes(act_1, res_1) \land Consumes(act_2, res_1))$$

Goal dependency The predicate $Sub.Act$ can be used to define the goal decomposition dependency between two actions. It indicates that action act_2 depends on action act_1 because act_1 is a subsidiary action that must have been completed to achieve act_2 in current collaborative plan, i.e. $Sub.Act(act_1, act_2)$

$$DEP(act_2, act_1, Sub.Act(act_1, act_2))$$

5.2 Representing the Field of Work with Plan-Graph model

In this section, we present the knowledge representation of the field of work based on PlanGraph model. By modeling an collaborative activity within a PlanGraph, the entities and relations in the field of work as we formalized in Section 5.1.1 can

be represented as different components of the PlanGraph. Then we show how the local scopes can be easily derived from the PlanGraph model. Last, we describe the construction of dependency network from the PlanGraph model.

5.2.1   The PlanGraph model

The PlanGraph model is designed to represent the dynamic knowledge in the human-computer collaboration based on the SharedPlans theory [18]. The Plan-Graph model has been used in several existing studies to model human-computer collaboration in different applications, e.g. the natural conversational interface to geospatial databases [18], collaborative dialog-based system to communicate vague spatial concepts [17], and context-aware mobile mapping [107].

In general, a PlanGraph represents the knowledge about a collaborative effort towards a common goal in a hierarchical way (Figure 5.2). The root of a PlanGraph is the overall goal of the actors in a collaborative activity, which is decomposed recursively as actions through the adoption of recipes. The whole PlanGraph therefore is a shared plan corresponding to the root action, while each sub-tree with a sub-action as the root represents the shared plan for that sub-action. In a PlanGraph, besides the action nodes, we also represent two types of nodes: a *parameter* represents an informational or physical object that is used by an action; and a *condition* represents a state of affairs in the world that the actors would like to achieve.

There are several ways that these three types of nodes can be connected in a PlanGraph:

1. An action can be decomposed into several parameters, conditions, and subsidiary actions, where the parameters indicate all the informational or physical objects that will be used by the action. All these parameters need to satisfy their own constraints before the action can be performed, and they are accessible by all the subsidiary actions at the same level. The conditions under an action list all the constraints that need to be satisfied before the action or any subsidiary actions can be performed.

2. Each parameter can be decomposed into subsidiary actions and conditions.

The children actions of a parameter are used to identify the value of the parameter, i.e. they are used to satisfy the knowledge-precondition on the parameter. The conditions attached to a parameter are the constraints that need to be satisfied before the parameter is ready to be used by upper action.

3. Each condition can only be decomposed into subsidiary actions that are used to achieve the condition.

The PlanGraph model also encodes the actors that are participated in each action or sub-action. Because the relations between actors and actions can be many-to-many, i.e. an actor can work on multiple actions, and one action can involve multiple actors, we represent knowledge of actors within their relations towards each action. Therefore, each action node in a PlanGraph includes several attributes to store the relations with participating actors as defined in Section 5.1.1.2: *Intentions* records the different intention relations of each actor towards the action. *Capabilities* indicates the level of capability of each actor to perform the action, such as whether they have knowledge to identify a recipe for the action; or whether the agents can bring it about.
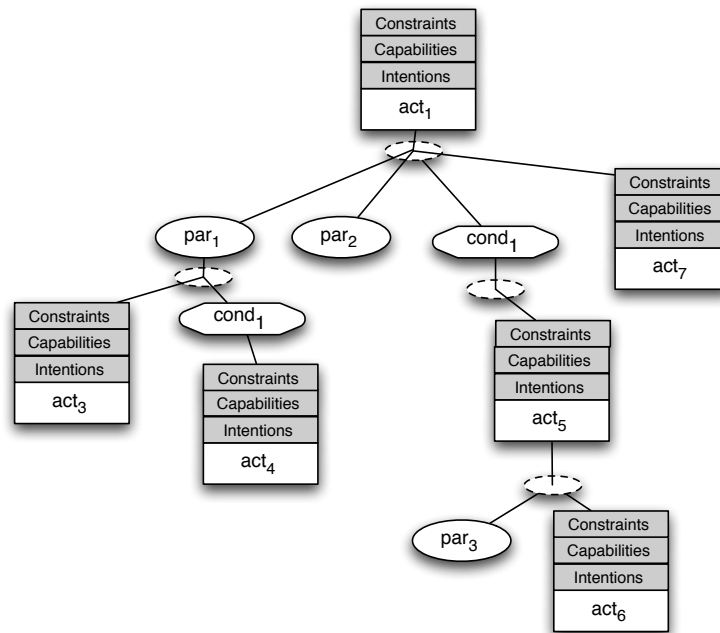
Figure 5.2. Structure of a PlanGraph

In many applications, such as the dialog-based interfaces [17, 18], the whole collaborative activity can be represented using one single PlanGraph, as the overall goal in one interaction session is unique. However, in complex collaborative activities, it is possible that multiple goals are active at the same time. As a result, the knowledge representation of field of work in these situations is often a forest that includes multiple PlanGraph trees in parallel.

5.2.2   Representing elements and relations

The PlanGraph model allows us to represent the basic elements and entities in the field of work. The actions, actors, and resources are first-class objects in the PlanGraph model, and the multiple relations among then can be represented as the structural relations in the PlanGraph model (Table **??**).

Add the class diagram or the picture similar to the one used in the social modeling book to show the different entities

Actors   Each actor is represented an object with a unique *ID* in the PlanGraph model. Each actor object in the PlanGraph maintain the current state of the corresponding actor, such as the name, the location, the role of the actor, and the expertise he/she has. Each actor object also maintains a list of actions that the actor is currently participated in.

Actions   Actions are modeled as a type of node within the hierarchical structure of the PlanGraph. The *goal* of the action is represented as a condition node, indicating the expected effect of the action when it is successfully performed. The current *execution state* of the action and the *current recipe* to perform this action are stored as attributes of each action node. The *Intentions* and *Capabilities* record the different relations between participating actor towards the action. The *Constraints* record the possible constraint relations between subsidiary actions. The *parameters*, *conditions*, and *sub-actions* point to the other nodes that form the sub-plan of this action.
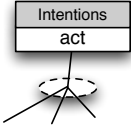
Resources   Resources are modeled as parameter nodes in a PlanGraph. Each parameter node records information about the current states of the corresponding

resource in the domain, such as the location of a rescue vehicle, or the capacity of the decontamination station. Parameter nodes can be collective or individual. A collective parameter indicates a collection of resources with the same type, for instance, a parameter representing all the victims in the same rescue operation. Each resource points to at least one sub-action that is used to identify the resource, and may points to conditions that the resource needs to satisfy.

Relations between actors and their actions   The relations between actors and their actions can be directly retrieved from the *Intentions* and *Capabilities* attributes attached to each action, recording the different relations between participating actor towards the action.

Relations between actions   The two major types of relations between actions are composition and precedence. The former is directly encoded in the hierarchical structure of a PlanGraph. Each action node has an attribute *sub actions* recording its subsidiary actions under current plan. As a result, the fact that acton act_1 is included in the list of *sub actions* of action act_2 represents the relation that action act_1 is the subsidiary action of doing act_2. The precedence relation that reflects the temporal order of doing two actions is encoded in the *constraints* slot of their parent action node.

Relations between resources and actions   The relations between resources and actions are modeled as the composition relations between action nodes and parameter nodes in a PlanGraph. Each action node has a slot *parameters* recording all the parameters under current plan, and each parameter has a slot *sub actions* recording the actions to identify the parameter, and the *conditions* that include subsidiary actions to manipulate the parameter. On the other hand, all the sub actions at the same level of the parameter are consumers of this parameter.

Relation	Structure in PlanGraph
$Pot.Int(ar, act)$	search the *Intentions* slot attached to each action node:
$Int.Th(ar, act)$	
$Int.to(ar, act)$	
$Perform(ar, act)$	
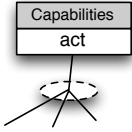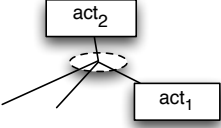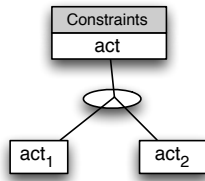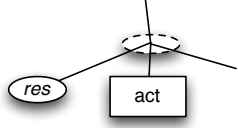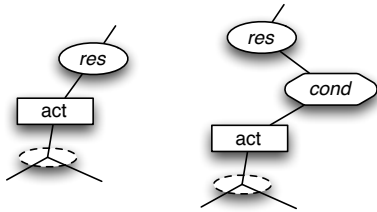
$Knows(ar, act)$	search the *Capability* slot attached to each action node:
$Able(ar, act)$	
$Workable(ar, act)$	Capabilities / act
$Sub.Act(act_1, act_2)$	act_1 is a subsidiary action node under act_2:
	act_2 / act_1
$Precedes(act_1, act_2)$	act_1 and act_2 as ordered siblings:
	Constraints / act / act_1 / act_2
$Consume(act, res)$	a parameter node representing a resource res and a action node act as siblings:
	res / act
$Produces(act, res)$	act is a subsidiary action node under the parameter node representing a resource res:
	res / act ; res / cond / act

Table 5.1: Representing basic relations in PlanGraph

5.2.3 Constructing local scopes

Based on the definition of local scopes in Section 5.1.2, the local scope of an actor can be dynamically constructed by depth-first traversing through the PlanGraph model to search for all the actions that have intention or capability relations with the actor.

We define a procedure $BUILD\text{-}LS$ that takes an actor object ar and a Plan-Graph PG to construct both the local scope of intentions LSI and local scope of capabilities LSC for the actor. We define a sub-procedure $ADD\text{-}To\text{-}LS$ that takes an actor object ar and a action node act in the PlanGraph to add the action to the local scope LSI or LSC based on whether the corresponding relation exists. The $ADD\text{-}TO\text{-}LS$ procedure then calls itself on all the children nodes to traverse to the deeper levels in the PlanGraph. In this way, the process to construct the local scopes $BUILD\text{-}LS$ is composed of the initialization of LSI or LSC, and then a recursive call of $ADD\text{-}TO\text{-}LS$, beginning on the root node of the PlanGraph.

```
 1: procedure BUILD-LS(ar, PG)
 2:     LSI ← [ ], LSC ← [ ]
 3:     root ← PG.root
 4:     ADD-TO-LS(ar, root)                    ▷ start the recursion from root
 5: end procedure
 6: procedure ADD-TO-LS(ar, act)
 7:     for all int in act.intentions do              ▷ check the current node
 8:         if int.actor == ar then
 9:             LSI.add(ar, act, int)
10:         end if
11:     end for
12:     for all cap in act.capabilities do
13:         if cap.actor == ar then
14:             LSC.add(ar, act, cap)
15:         end if
16:     end for
17:     for all par in act.parameters do              ▷ recursion on parameters
18:         for all subact in par.subacts do
19:             ADD-TO-LS(ar, subact)
20:         end for
21:     end for
```

```
22:     for all cond in act.conditions do              ▷ recursion on conditions
23:         for all subact in cond.subacts do
24:             ADD-TO-LS(ar, subact)
25:         end for
26:     end for
27:     for all subact in act.subacts do               ▷ recursion on subacts
28:         ADD-TO-LS(ar, subact)
29:     end for
30: end procedure
```

If there are multiple actively PlanGraph, the local scopes can be generated by calling $BUILD\text{-}LS$ on each PlanGraph, and then merge the local scopes of intentions LSI and local scopes of capabilities LSC separately.

Show a result of the constructed local scope in the same concrete example of plangraph in the motivating scenario

5.2.4   Constructing dependency network

From the PlanGraph model, we can also dynamically construct the corresponding dependency network to represent how the actions are dependent on each other. Formally, a dependency network $DN = (V(DN), E(DN), \psi)$ is defined as a directed graph with the following characteristics:

1. $V(DN) = \{ACT \cup PARAM \cup COND\}$ is the set of all the nodes in the dependency network, all the *dependers* and *dependees* are action nodes ACT, and the *dependums* can be any parameter RES or condition nodes $COND$.

2. The set $E(DN)$ is a set of links between the nodes. Each link can be represented as a pair of nodes in $V(DN)$, i.e. $\psi : E(DN) \rightarrow V(DN) \times V(DN)$. The link can indicate a direct dependency between two action nodes, or a incoming link that connects a *depender* and *dependum*, or an outgoing link that connects a *dependum* and *dependee*.

The construction of dependency network can be also achieved by depth-first traversing through the PlanGraph model to search for all the action-action and action-resource relations. We define a procedure $BUILD\text{-}DN$ that constructs a dependency network from a PlanGraph PG. As the set of nodes $V(DN)$ can be calculated from the set of links $E(DN)$ by removing all the duplicate nodes in

$E(DN)$. The goal of the $BUILD\text{-}DN$ is to build $E(DN)$. Similarly, we define a sub-procedure $ADD\text{-}To\text{-}DN$ that takes an action node act in the PlanGraph to add dependencies that starting from the node act, i.e. the act as the $depender$.

```
 1: procedure BUILD-DN(PG)
 2:     E ← [ ]
 3:     root ← PG.root
 4:     ADD-TO-DN(root)                          ▷ start the recursion from root
 5: end procedure
 6: procedure ADD-TO-DN(act)
 7:     for all subact in act.subacts do                    ▷ check for Sub.Act
 8:         E.add(act, subact)
 9:     end for
10:     if act.hasParent() then
11:         for all constr in act.parent.constraints do       ▷ check for Precedes
12:             if constr.next == act then
13:                 E.add(act, constr.prev)
14:             end if
15:         end for
16:         for all par in act.parent.parameters do         ▷ check for parameters
17:             for all subact in par.subacts do
18:                 E.add(act, par), E.add(par, subact)
19:             end for
20:         end for
21:         for all cond in act.parent.conditions do         ▷ check for conditions
22:             for all subact in cond.subacts do
23:                 E.add(act, cond), E.add(cond, subact)
24:             end for
25:         end for
26:     end if
27:     for all par in act.parameters do              ▷ recursion on parameters
28:         for all subact in par.subacts do
29:             ADD-TO-DN(subact)
30:         end for
31:     end for
32:     for all cond in act.conditions do             ▷ recursion on conditions
33:         for all subact in cond.subacts do
34:             ADD-TO-DN(subact)
35:         end for
```

```
36:        end for
37:        for all subact in act.subacts do              ▷ recursion on subacts
38:            ADD-TO-DN(subact)
39:        end for
40: end procedure
```

5.3 Representing Events

In Section 4.2.2.1, we discuss the difference between *'occurrence'*, *'awareness'*, and *'event'*, and define *'events'* to refer to the computerized entities that are used in an awareness system to represent knowledge about either real world *'occurrence'* or the results of *'awareness'* processes. Before delving into how events can be used by the computer system to update the knowledge representation of the field of work, or used by the human actors to develop awareness, we need to discuss how they are computationally represented and what types of events are defined in this study. In this section, we first describe the general representational structure of events, then discuss the major types of events, and how they are represented differently.

5.3.1 Structure of events

We adopt the similar approach with many existing event processing systems [61] to represent each event as a structured object consisting of a named set of attributes. Formally, an event e is a nonempty set of *attributes* $\{a_1, a_2, ..., a_n\}$, where each a_i a name/value pair (n_i, v_i) with name n_i and value v_i. It is assumed that names are unique, i.e., $i \neq j \Rightarrow n_i \neq n_j$, and that there exists a function that uniquely maps each n_i to a data type T_i that is the type of the corresponding value v_i.

The set of attributes for each event should help answer questions such as this: What occurrence or awareness aspect it refers to? When did it happen? Where did it happen? What other information is associated with its happening? The answers to these questions are usually depend on the type of event they are associated with. An *event type* is a generalization for a set of event objects that have

Show a result of the constructed dependency network in the same concrete example of plangraph in the motivating scenario should consider the state of the actions here or leave it for belief propagation? how to handle multiple PlanGraphs? shared resource, what else?

the same semantic intent and same structure [34], i.e. they share the same set of attributes, but may have different values. Each *event type* has a unique event type identifier. In this study we use simple descriptive text strings for these identifiers, for example the phrase *"LocationChanged"* identifies an event type that can describe any instance of an object's location change. Identifying the set of *event types* is an application specific task, as actors in different applications have different awareness needs and capabilities to detect events. We describe the major event types that are supported in this study in Section 5.3.2.

As arbitrary attributes can be included in each event type, we can distinguish between three kinds of attributes carried in each event, following the definitions in [34] (Figure 5.3):

1. The *header* consists of generic information about the event, such as the event type and occurrence time, etc. The name and meaning of these header attributes are not specific to a particular event type.

2. The *payload* contains a collection of attributes carrying the data that describes the actual occurrence. Unlike header attributes independent of the actual event type, the payload attributes are defined per event type.

3. An event can also contain free-format *open content* information that provides a mechanism that an event producer or the awareness system can use to enrich an event object with extra contextual information, such as human-readable explanation, multi-media content etc.

Header The header of an event contains the common attributes that are included in every event object. Unlike payload or open content that are optional, a header is required for every event representation. In general, the header of an event needs to include the following attributes:

1. *Event type.* This attributes stores the event type identifier that uniquely identifies the event type of this event.

2. *Event identifier.* This is a unique identifier for each individual event object.
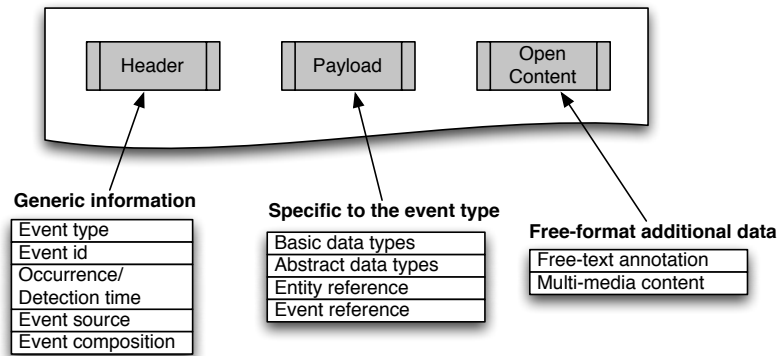
Figure 5.3. The structure of an event (adapted from [34] p.63)

3. *Occurrence/detection time.* The occurrence time attribute records the time at which the real world occurrence happens. In some cases, the event producer might not be able to determine the time when the event actually occurred. For example, if the producer examines the state of some external entity only at periodic intervals. In such cases, the detection time is used instead that records the time at which the event became known by the event producer.

4. *Event source.* This is the entity that originates this event. This can be either an external sensor, or a human actor in the collaborative system.

5. *Event composition.* This is a boolean attribute that denotes whether the specific event is a composite event or not. A composite event is one whose payload is made up of several different event instances.

Payload The attributes that make up the event payload are used to carry the data that describes the actual occurrence. The set of attributes included in each event is a variable that depends on the corresponding event type. There are several types of data that can be included as payload attributes:

1. *Basic data types.* The value in an payload attribute can simply be in the basic data types, such as string, numeric boolean, date/time etc.

2. *Abstract data types.* Attributes can also have abstract data types that are structures composed of other data types. For example, many events includes

a geographic attribute to records the whereabout of the represented real world occurrence. This attribute can be a point-based representation as a latitude/longitude pair, or more complicated as a route or a geographic area, which are in abstract data types.

3. *Entity reference.* Instead of records the information directly in an attribute, the event can also records information by pointing to entities represented in the field of work. For example, an *'ActionPerformed'* event may use the reference to the action node stored in the PlanGraph model to indicate which action is performed.

4. *Event reference.* Some events may contain references to other events. For example, a composite event may use the event references to records the primitive events that it is composed of.

Open content The open content of an event can include any attributes an event producer or the awareness system can use to provide additional contextual information about the event. For example, it is used in the awareness externalization process to allow the actors to provide the human-readable explanation of their interpretations.

5.3.2 Event types

As we argue in Section 4.2.2.1 that *events* can be used to represent both description of real world *occurrences* and externalization of human actors' internal *awareness* knowledge, a fundamental distinction should be made between these two categories of events, we call the former *external events*, and the latter internal events. The distinction between external events and internal events are important, because (1) they require different representational structures, i.e. the payloads of events have different set of attributes; (2) they are consumed differently by the system when updating the knowledge representation of the field of work; (3) and they are treated differently in human users' awareness processes.

Another distinction to make is the difference between *primitive events* and *composite events*. Composite events prevent the users from being overwhelmed by

a large number of primitive event by providing them with a higher-level abstraction [61]. Generally, a composite event is made up of several other events (either primitive or composite), according to a specification of relations between them.

In the following, we first describe the major primitive event types in both external and internal categories, and then discuss the composite events as a special event type with its own payload structure.

5.3.2.1 External events

As we conceptualize a collaborative environment as consisting of a variety of entities and their relations, the external events can be defined to indicate any kinds of changes on either these entities or their relations.

Events on entities We define each external event on an individual entity as a semantic function that changes the entity's property or state. We follow the general event ontology proposed in [55] to define the following categories of external events on individual entities:

1. *State Change.* An event is a state change event if the occurrence yields a change of states on an entity. All the possible states of an entity are usually can be expressed as a discrete state machine, and each state transition indicates a possible state change event. For example, Figure 5.4 shows the transition diagram with all the possible execution states of an action. Each valid transition in the diagram can be considered as a state change event.
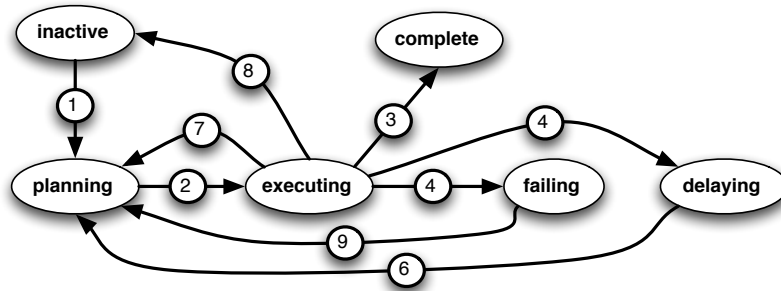
Figure 5.4. Execution state transition of an action

2. *Existential Change over Time.* An event indicates an existential change over time if its occurrence changes the existence of an entity in temporal order, e.g. an entity did not exist in the past but it exists now.

3. *Existential Change over Space.* An event indicates an existential change over space if its occurrence changes the existence of an entity depending on its movement through space, e.g. an entity existed at location A, but now exists at location B.

4. *Value Comparison.* Another common class of external events are comparison events to indicate changes on attribute values of an entity. They can used to indicate whether an attribute value is equal, unequal, greater than, or less than a fixed threshold, or the same attribute value in the past.

Events on relations   The events on relations are used to indicate whether some relations between entities hold. For example, an event with the type *'ResourceAssigned'* indicates an assignment relation between a resource and an action starts to hold, i.e. the resource is now assigned for performing the action. Therefore, the types of external events on relations depend on the possible types of relations that can be identified in the domain.

Generally, the basic relations between entities in the real world can be divided into three categories: spatial, temporal, and conceptual [102].

1. The spatial relations link the entities through their spatial positions. The basic types of spatial relations have been well studies in the literature on geographic information systems, include binary topological [28], directional [37], and distance relations [49]. Topological relations is a particular subset of geometric relations that are preserved under topological transformations such as translation, rotation, and scaling. Some examples are relations indicating whether one entity disjoints, meets, overlaps, or contains another entity. Directional relations indicate the relative direction between two entities, such as one is at north of the other. Distance relations link entities based on their proximity in the space, such as one is within a certain range of another.

2. The temporal relations link the entities based on their temporal positions. The basic types of temporal relations are considered in the literature on

temporal reasoning [3], including binary topological, ordering, and distance relations [4].

3. The conceptual relations are an umbrella term that cover all the different types of organizational, structural, or social relations between entities in a particular collaborative activity.

From the basic types of relations, more complex types of relations can be built, such as density (clustering, dispersion), arrangement (e.g. sequence in time or alignment in space) and spatial-temporal relations. The latter are composed of spatial and temporal relations and represent changes of spatial relations over time: approaching or going away, entering or exiting, following, keeping distance, concentrating or dissipating and so on.

Figure 5.5 shows an upper level typology of the external events that can be defined on entities and relations in a collaborative environment.
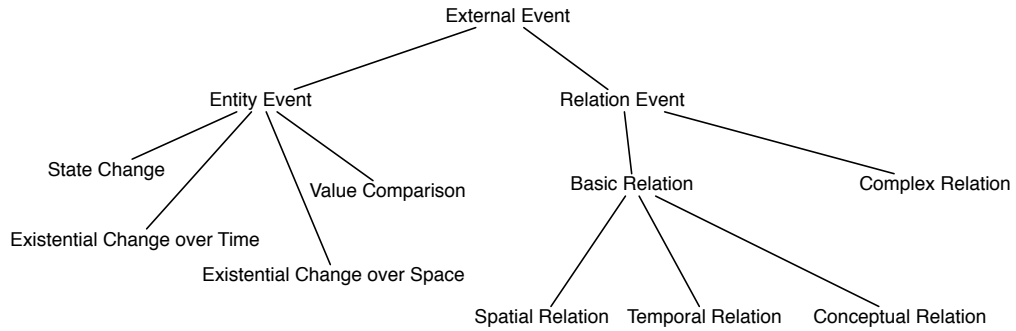
Figure 5.5. An upper level typology of external events

Relevance to the field of work   Although the number of external events that could be possibly identified is infinite, not all of them are relevant to the human actors' working context, i.e. the field of work. As a result, our goal of identifying external events focuses on finding a subset of event types that are relevant in an application domain. Existing studies to identify such a subset of event types are usually done in two ways [71]: from the 'supply' side to identify the set of event types that can be recognized by available sensors, and from the 'demand' side to define event types based on the nature of the activities supported by the event-based systems. As we are more concerned about the human actors' awareness needs

in performing their collaborative activities, we follow the 'demand'-side approach and identify the major event types based on how they could possibly contribute to the understanding of the field of work in a collaborative activity.

In general, we believe that the relevance of external events to the field of work can be analyzed based on the different functional roles they can play in updating the knowledge about the field of work. If the occurrence of an external event can imply some change in the field of work, it should be treated as relevant. Based on our conceptualization of the field of work, we can identify the following function roles for external events:

1. *Direct change to entities in the field of work.* The external events can indicate changes on individual entities that are modeled in the field of work, i.e. the property or state change on the resources, actors, or actions. For example, a state change event can be used to indicate the change of execution state for an action in the field of work. Location change events can be used to describe an actor's movement in the field of work.

2. *Direct change to relations in the field of work.* Within the different types of relation events that can be possibly identified, some of them are directly related to the various relations between resource, actors, and actions as we described in Section 5.1.1.2. For example, an external event type indicating the constitution relation between two entities can be used to describe the *Sub.Act* relation between two actions, i.e. one action is a subsidiary action to perform another one. The assignment relation event types can be used to describe relations between an actions and a resource, i.e. the resource has been assigned to the performance of the action.

3. *Action motivation.* Aside from the two cases that external events can be directly linked to the entities and relations in the field of work, the external events can also impact the field of work by motivating the actions that need to be performed. For example, an external event indicating that the fire alarm is ringing will activate my goal to escape from my office. In this case, the fire alarm is not directly linked to any entities in my current field of work, rather it motivates me to perform a new action.

4. *Indication of action performance.* In some cases, the external events can also provide some evidence implying the effects of action performance. For example, instead of a state change event directly showing the action to delivery a resource to an actor has been completed, it could be implicitly inferred from a spatial relation event that indicates the resource is now located at the actor's location. Similarly, an external event indicating the occurrence of a traffic blocking between the resource's current location and the actor's implies the delivery action is delayed.

5.3.2.2   Internal events

The internal events are used to describe the results of human actors' awareness processes. Unlike the external events can describe any entities and relations in the real world, they describe the changes of human actors' internal mental states in the field of work. The internal events are usually derived from the external events to indicate the human actors' interpretation on these external events. In this study, we identify two types of internal events: *intention* events and *belief* events.

An *intention* event indicates a human actor's adoption of certain intention towards some action in the field of work. As we described in Section 5.1.1.2, an actor's intention to an action can be of different kinds, i.e. *Pot.Int*, *Int.Th*, or *Int.To*, with different levels of commitment. Figure 5.6 shows the basic structure of an intention event. The payload of an intention event include three required attributes: an intention type indicating whether it's a potential intention, intention-that, or intention-to, a reference to the actor entity who has this intention, and a reference to the action that is intended to. An optional free-text attribute is included in the open content part of an intention event, where the human actor can provide the rationale for adopting the corresponding intention.

An *belief* event describes a belief of a human actor on some occurrence in the field of work. For example, it can be used to indicate an actor's belief that an action has been successfully performed, or his/her belief that he/she has the capability to perform an action. As belief events usually refers to some occurrence in the field of work, we represent belief events by embedding an external event representing the occurrence into its payload (Figure 5.7). However, unlike the standalone external event that indicates changes already happened, the occurrence described in a belief
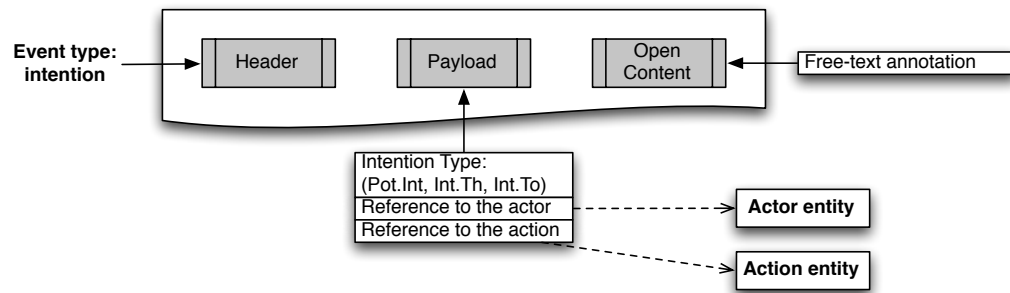
Figure 5.6. Structure of an intention event

event can be something that will happen in the future. These belief events represent the results of the human actor's projection process, i.e. what he/she expects to happen in the future. There are two attributes in a belief event's open content part: the free-text explanation of this belief, and a confidence level to describe how confident he/she is about this belief.
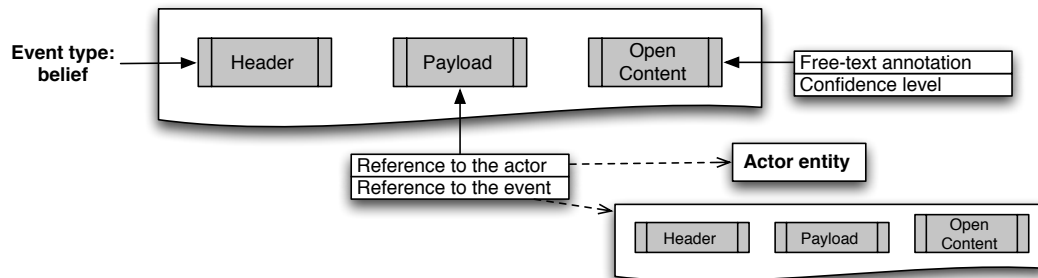
Figure 5.7. Structure of a belief event

5.3.2.3 Composite events

Composite events are a special type of events that can consist of several other events. The subsidiary events can be external or internal. Besides the list of subsidiary events, a composite event needs to also describe how these sub-events are combined together. In a simplest case, a composite event can occur only when all the sub-events occur. Moreover, a composite event can occur when some of the sub-events occur, but some do not. Or it occurs when any of the sub-events occur. A more complicated composite event language can be found in [61] that in-

cludes different relations between the sub-events, such as negation, concatenation, sequence, iteration etc. To describe the composition pattern, a specific payload attribute needs to be included in a composite event's representation (Figure 5.8).
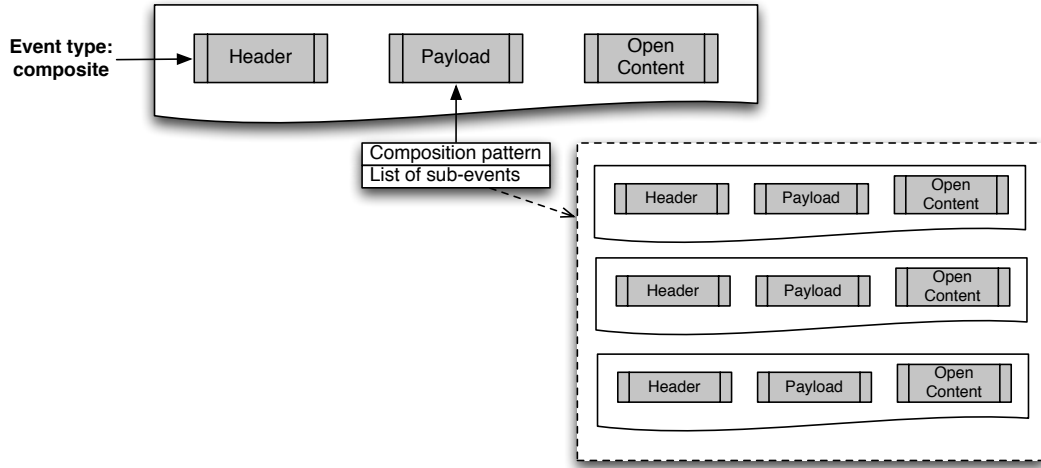
Figure 5.8. Structure of a composite event

5.4 Knowledge updating process

Each time when the system detects that there is a new event, it triggers the knowledge updating process to decide how it influences the state of the field of work and update the correspondent in the PlanGraph. In general, the knowledge updating is a four-step process of *association*, *assessment*, *elaboration*, and *propagation*.

1. *Association.* The knowledge updating starts with the association of an event with the field of work. In this step, the system searches the current Plan-Graph model for an appropriate match between the input event and the entities or relations in the field of work. If a match is found, the system uses the information stored in the event to update the PlanGraph.

2. *Assessment.* The second step is to assess how the event can contribute to new changes in the collaborative activity. It may trigger new actions that need to be added to the field of work, or change the current state of existing actions.

3. *Elaboration.* Based on the assessment of new changes, the elaboration step is to reason about the system's expectation on any new actions that need to be added to the field of work, or what actors will be potentially involved in these new actions. This is usually performed with domain specific knowledge, such as recipes of performing an action, and the specification of role responsibility.

4. *Propagation.* The propagation step focuses on evaluating how the current change can be possibly propagated other actions in the field of work because of the dependencies among them.

The four steps of knowledge updating process share some commonality with the human actor's awareness development processes. The *association* and *assessment* steps are very similar to the *comprehension* process, where the actor comprehend or understand the relevance of awareness information in relation to its tasks and goals. The *elaboration* and *propagation* can be considered as the projection of the states in the near future, as the actor forecasts likely future states in the situation. In this way, we can think of the knowledge updating process as the computer system's awareness development process. The only difference is that, as the human actor usually only needs to be aware of things in his/her local scope of work, the computer system aims to possesses the knowledge of the whole field of work through the knowledge updating.

One thing to note is that not every event will be processed through all the four steps. Some external event may not directly link to any entity in the field of work, or the system does not have the complete knowledge to assess its implication on the field of work. In such case, the event will be passed by to the human actors for interpretation. The result of human actor's interpretation as a new internal event then starts a new round of knowledge updating, during which the system's knowledge is updated.

As we emphasize in the beginning, the knowledge updating is a two-way process, i.e. while the PlanGraph model of the field of work is updated by the new event, the event itself is also developed in the system's reasoning processes. For example, in the *assessment* step, an external event *'TrafficBlocked'* causes the system to believe that the action to delivery a resource to an actor cannot be achieved, i.e. a new internal event describing the system's belief about the state change on the

delivery action can be derived. Later, in the *propagation* step, the system may generate a new internal event to indicate that because of the state change on the delivery action, the actor's action that is waiting for the resource delivery will also be impacted. In this way, the original event is derived into a chain of events as the knowledge updating process proceeds.

To recording the development of an event in the knowledge updating process, we define an *event chain* Θ as an ordered sequence of events: $\Theta = (e_0, e_1, e_2, ...)$. In the beginning of the knowledge updating process, there may be only one event e_0, i.e the original external event in the event chain Θ. As the knowledge updating proceeds, more events are added to the chain. In the assessment step, the system may generate derived events indicating the system's belief on how the field of work has been changed due to the original events. In the propagation step, the system predicates the future state changes, and attaches more anticipatory events to the chain. Figure 5.9 shows the knowledge updating process with the development of the event chain.

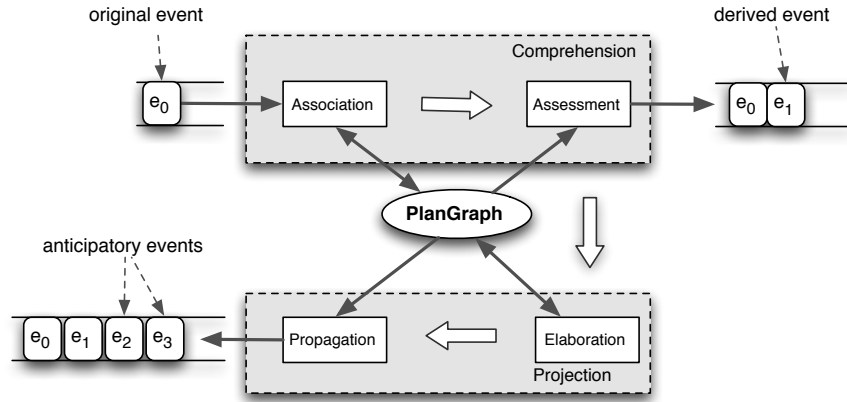

Figure 5.9. The knowledge updating process

5.4.1 Association

The knowledge updating starts with establishing the association between the input event and the current PlanGraph model representing the field of work, and use the event information to update the PlanGraph model. The association starts with the recognition of event type for each input event, as the different event types are

treated differently.

If the event is an *external event* on entities, we consider the following two cases:

1. If the event is indicating an existential change over time, the event is directly passed through to the assessment step, as the entity linked to this event did not exist in the past.

2. Otherwise, we search all the PlanGraph nodes to find any match with the entity described in the event based on their unique identifiers. If a match is found, we update the attribute information attached with the PlanGraph node based on the event type. If it is a state change event, we change the state of th node. If it is an existential change over space, we update the location information of the node. If it is a value comparison event, we update the corresponding attribute value. After updating the node, we add an entity reference in the event object, so that it can direct link to the matched PlanGraph node for later use.

If the event is an *external event* on relations, we check for the following cases:

1. If the event indicates a resource assignment relation, i.e. a resource res_1 is assigned to the performance of an action act_1. we search all the action nodes in the PlanGraph to find any match with the action entity described in the event act_1. If a match is found, we search for the parameters of act_1 in the PlanGraph to find any of them has the same resource type as res_1 and then assign the values of res_1 to the parameter. After that, we add an entity reference pointing to the parameter node in the event's payload.

2. If the event indicating a structural relation, e.g. an action act_2 is a subsidiary action of another action act_1. we search all the action nodes in the PlanGraph to find any match with the action entity act_1 described in the event. If a match is found, we create a new node with the information described in act_2, and then add an entity reference pointing to this new node into the event's payload. This can be applied to other decomposition relations, such as an action is a subsidiary action to achieve a condition, or a new parameter needs to be added to an existing action.

If the event is an *internal event*, we consider the following cases:

1. If the event is an *intention event*, we search all the action nodes in the PlanGraph to find any match with the action entity described in the event. If a match is found, we search for the *Intentions* associated with the action node to find any existing actor has the same identifier as the actor entity described in the event. If so, we update the intention type based on the intention event. Otherwise, we add the actor and the corresponding intention into the the *Intentions* associated with the action node. After that, we add an entity reference pointing to the action node in the event's payload.

2. If the event is a *belief event* about an actor's capability to perform an action, we search all the action nodes in the PlanGraph to find any match with the action entity described in the event. If a match is found, we search for the *Capabilities* associated with the action node to find any existing actor has the same identifier as the actor entity described in the event. If so, we update the capability type based on the belief event. Otherwise, we add the actor and the corresponding intention into the the *Capabilities* associated with the action node. After that, we add an entity reference pointing to the action node in the event's payload.

3. If the event is other *belief events*, we apply the association rules directly on the subsidiary event describing the content of the belief.

In sum, two results are achieved if an association between the event and the PlanGraph model is found in this step. First, the corresponding PlanGraph entity or relation is updated based on the new information carried by the event. Second, the event is enriched with a direct link to the corresponding PlanGraph node, that is the context of origin of this event is identified.

However, not all the events are directly associated with the PlanGraph model. As we discussed in Section 5.3.2.1, some external events may describe occurrence on the entities that are not currently in the PlanGraph, but implicitly impact the field of work by motivating new actions or implying the effects of action performance. These events will be passed through to the assessment step for further analysis.

5.4.2   Assessment

In the assessment step, each event is evaluated to check how it can lead to changes towards the action performance, such as the execution state change of an action, or a condition. The changes towards action performance can be explicitly expressed in the event, and is directly associated with the action node during the association step. For example, it can be an internal event from a human actor indicating the belief that the resource delivery action has been successfully performed. However, in the assessment step, the system is more interested in the events that impact the action performance implicitly, where inference becomes necessary. For example, an external event indicating that a resource is now located at an actor's position can implicitly indicate the successful performance of the resource delivery action. In this case, a new event showing the state change of the resource delivery action will be derived and added to the event chain along with the original event.

The knowledge stored in the PlanGraph allows the system to perform some routine assessment tasks that are universal across different application domains. Some of the basic inference tasks can be described as follow:

1. *Goal conditions on action nodes.* If an event describes changes on entities or relations that are included in an action's goal condition, the system can evaluate the action's goal condition. If the goal condition becomes holding because of this event, we derive a new state change event on this action, and push it into the event chain.

2. *Condition nodes.* If an event describes changes on entities or relations that are included in a condition node, the system can evaluate the condition node. If the condition becomes holding or no longer holding because of this event, we derive a new state change event on this condition, and push it into the event chain.

3. *Parameter nodes.* If an event describes changes on an resource that is assigned to a parameter node, the system can evaluate the parameter's subsidiary conditions. If a condition becomes holding or no longer holding because of this event because of this event, we derive a new state change event on this parameter, and push it into the event chain.

The second type of tasks that the system can perform during the assessment process is to check whether the event can activate new action that needs to be added to the field of work. This type of events is usually called *triggering events*, as they are often not directly associated with any current nodes in the PlanGraph, but will trigger some new action to be performed. For example, every time a new victim is found in an emergency response operation will trigger a new rescue action to be performed. In this case, the initial event about the discovery of a new victim cannot be associated with any existing actions, but asks for a new action to be performed. The assessment of action activation requires a set of pre-defined domain-specific activation rules, so that every event can be searched through the activation rules to find whether it satisfies any of the conditions. If so, a new action is added to the field of work as a new PlanGraph, and the new derived event is generated to indicate the activation of the new action.

Furthermore, the assessment step can involve more sophisticated inference techniques, such as spatio-temporal reasoning [10], pattern recognition [110], or case-based reasoning [54], to enhance the system's reasoning capabilities. However, these reasoning techniques often require a great number of domain knowledge, and lack flexibility to handle unexpected events. As we discussed in Section 4.1.2, one of the design principles of our approach is to leverage the different capabilities of computers and humans. As a result, in the assessment step, we design the system to focus on the more reliable low-level routine inferences that can be directly performed in the PlanGraph model, and leave the complex, higher level assessment to the human actors. Hence, some events may not be considered as contributing to the collaborative activity by the system in the assessment process. Rather, they are later interpreted by human actors and send back to the system as internal events, which will then be used to update the system knowledge.

5.4.3 Elaboration

The main goal in elaboration step is to advance the collaborative activity from the system side based on the change from the new event. After the assessment process, the context of the activity is changed, and the system attempts to elaborate the PlanGraph to accommodate the changes.

Based on the specification of SharedPlan theory [43], the system can elaborate the current PlanGraph in several ways.

1. *Recipe selection.* The system can contribute to the collaborative activity by retrieving a recipe for a new action from the knowledge base, i.e. by providing a default way to perform this new action.

2. *Parameter binding.* If any of the parameters is unbound to any values, the system will search the knowledge base to find any action that can be performed to identify the value for the parameter.

3. *Condition satisfaction.* If any of the pre-conditions is not holding, the system will search the knowledge base to find any action that can be performed to satisfy the condition.

4. *Actor allocation.* If any of the actions has not been committed by any actors, the system search for the actors who might be potentially intended to or capable of performing the action.

The elaboration step is not performed for every event, rather it is only triggered by a subset of events that are related to the development of the collaborative activity.

1. Events on structural relations. Whenever an event indicates that a new action is a subsidiary action to another action, a way to identify a parameter, or to achieve a condition, the new action will be added to the PlanGraph in the association step, and needs to be elaborated.

2. Events on goal activation. The elaboration needs to be performed whenever a new action has been added to the field of work because of some triggering event during the assessment step.

3. Events leading to condition violation. Whenever an event leads to the fact that some condition is no longer holding in the assessment step, the elaboration needs to be performed on the condition node to identify any action that can be performed to satisfy it.

The elaboration process is achieved with the support of two types of pre-defined knowledge: (1) the recipe knowledge about how to derive an action into a sequence of parameters, pre-conditions, and subsidiary actions, how to identify a unbound parameter, or how to satisfy a condition, etc.; (2) the knowledge about actor roles and their corresponding responsibilities and capabilities. The former allows the system to extend the field of work by adding new action, parameter, or condition nodes into the PlanGraph model, and the knowledge about actor roles allow the system to reason about who are likely to be interested in these newly added entities, or who have the capability to work on these new entities, so that the system can extend the actors' local scopes to these newly added entities.

The elaboration process is predictive as it reflects the system's prediction on how the collaborative activity will be advanced based on the new event occurrence. The system provides a default plan of performing an action, and identifies the potential actors who might be interested in it. After the elaboration process, the PlanGraph not only reflects the current state of the collaborative activity, but also shows the potential next steps based on the system's knowledge. However, the results of elaboration process never dictate how the human actors will eventually develop the action. The human actors may later generate new internal events to revise the plan generated by the system, or change their intention/capability levels towards the action in the PlanGraph.

5.4.4   Propagation

Propagation is another predictive process that the system can perform to predict future state changes in the field of work. It is triggered by the events that either directly indicate (in the association step) or imply (in the assessment step) state changes on the entities in the filed of work, and then reason about how these initial state changes can be propagated to other actions due to the multiple dependencies between them. A simple example could be be: if the execution state of an action act_1 is changed from *executing* to *failed*, then the parent action act_2 (i.e. $SubAct(act_1, act_2)$ holds) will likely to be impacted and may be also changed to *failed* if the plan is not changed.

The propagation is performed on the dependency network, which can be easily

constructed from the PlanGraph model (Section 5.2.4). The dependency network abstracts away the detailed information on each action node and focuses on the dependency relationships among them, we can adopt more efficient network-based reasoning models to perform the propagation. In this study, we employ the Bayesian network to perform the propagation [68]. Bayesian networks are directed acyclic graphs in which the nodes represent multi-valued variables, and the arcs signify direct dependencies between the linked variables and the strength of these dependencies are quantified by conditional probabilities. The purpose of the Bayesian network is to give a belief in each possible value for each node after some evidence arrives.

In the context of our model, the nodes are the basic elements in a dependency network, i.e. dependers, dependums, and dependees, which represent the entities in the field of work, i.e. actions, resources, or conditions. The evidence fed into the network includes certain state changes on these entities. Thus, the purpose of the Bayesian network is to update the systems belief in the states for every other node after some state change occurs on a node.

To operationalize the Bayesian network, we need to follow the following steps: (1) construct the Bayesian network, (2) assign the conditional probabilities for each link, (3) and perform the belief updating when some events arrive.

Construction of Bayesian networks By following the algorithm in Section 5.2.4, we can construct the dependency network from the PlanGraph model, and then the construction of Bayesian network is very straightforward. We translate each basic element in the dependency network into a node variable, and the different dependency relationships into corresponding links in the Bayesian network. The possible values for each node are determined based on the possible execution states for each type of node variables. At a given time, an action can be at one of the following states: *inactive*, *planning*, *executing*, *complete*, *failing*, and *delaying*. Each condition can be *open*, *waiting*, or *holding*. Each resource can be *unavailable*, *waiting*, or *available*. Figure 5.10 shows all the states for each type of node and the possible state transitions.

(a) state transitions of an activity

(b) state transitions of a resource

(c) state transitions of a condition
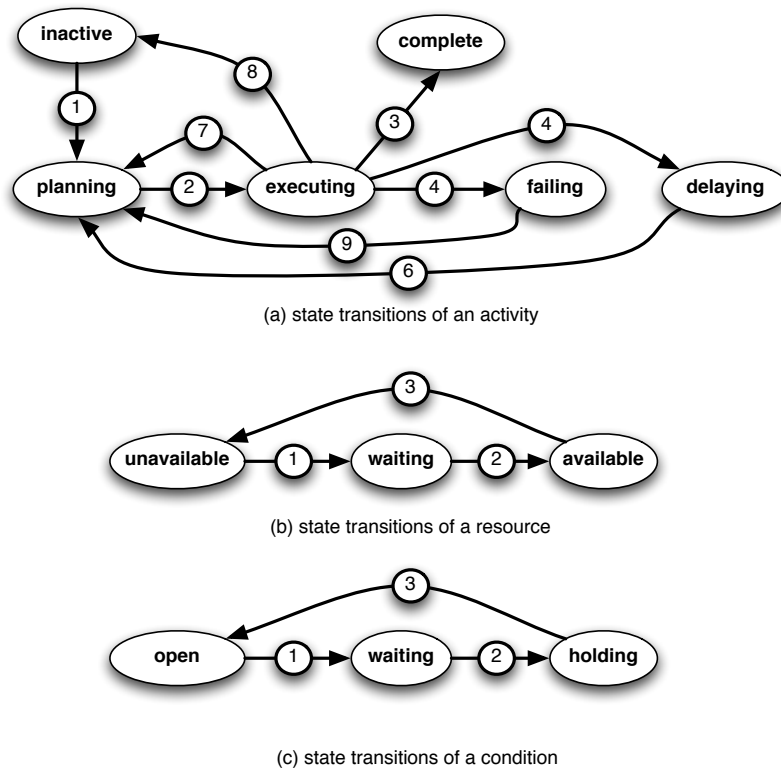
Figure 5.10. Types of node variables and their possible values in a Bayesian network

Assignment of conditional probabilities Before any propagation commences, we need to assign the conditional probabilities for each node that form the conditional probability matrices. An element of the conditional probability matrix looks like $P(x_i | u_{1j_1}, ..., u_{nj_n})$, and gives the probability of state i for node x conditioned on the states of its parent nodes. For example, the conditional probability of an action node indicates the possibility of each state for this action, given the states of all the resources, conditions, actions that it depends on.

In our study, we develop a set of heuristic rules to assign the initial conditional probabilities for each type of nodes, by evaluating the criticality of each parent node, and the opportunities for re-planning. For example, if a critical resource is needed for every possible way to perform an action, the state of the resource as *failing* will definitely lead to the *failing* of the action. However, if there are other possible plans to perform the action that do not require this resource, the

probability of the action being failing will be decreased.

The assignment of conditional probabilities provides the system with the default reasoning capability to propagate the state changes that can be later overwritten by the user's interpretation. When a user changes the belief in the state of an activity, the change will overwrite the systems initial belief through the belief updating. Because of this interactive nature, the purpose of the initial probability assignment is just to provide a good guess about the propagation from the system's perspective and does not have to be perfectly accurate.

Belief Updating   We follow Pearls belief propagation algorithm [68] to perform belief updating whenever a state change occurs. In this approach, the belief in each value of a node variable is divided into two parts: the part emerges from its ancestors and the part emerges from its descendants, and the final belief is ascribed by multiplying the two parts. As a result, the belief updating is performed as a bidirectional propagation process.

1. Starting at the node where the initial state change occurs, the system calculates how the state change will change the beliefs on each parent node. If the change on a parent node is significant, i.e. above a given threshold, the parent node becomes active, and will be further propagated to its parent. This is called *causal* propagation since the updating is from a cause to an effect to indicate how a state change of the cause will lead to the change of the effect.

2. On the other hand, the belief updating can also calculated from an active node to their children. This is called *evidential* propagation as the reasoning flows from evidence to hypothesis.

In our approach, both the causal and evidential propagation will be performed. The causal propagation is used to provide the system's predicted state changes on the actions that are dependent on the action where initial state change occurs. The evidential propagation occurs whenever a user modifies the systems prediction on a given node and the system uses it as an evidence to trace back and revise the previous beliefs on the dependees.

The result of the Bayesian network-based propagation will be used to enrich the event chain with anticipatory events. Starting from the node that the initial event is linked to, the system use the previous probability distribution before the propagation and the current values to perform the Cartesian product on each node. Each value in the new two-dimension table indicates the probability of each possible state change. If a significant state change has been detected (by comparing with pre-defined threshold), it will be added to the event chain as a new anticipatory event. For example, Figure 5.11 shows the result of a propagation process on a parameter node. Before the propagation, the parameter had a high probability to be in the state of *waiting*, and after the propagation, the probability distribution changed. By calculating the Cartesian product, we can find the most significant state change on the node is from *waiting* to *delay*, with a confidence level of 0.83808. As a result, a new anticipatory event indicating that the state of this parameter has been changed from *waiting* to *delay* will be pushed into the event chain.

previous values

unavailable	waiting	available	delay
0.01	**0.97**	0.01	0.01

current values

unavailable	waiting	available	delay
0.019	0.009	0.108	**0.864**

cartesian product

from \ to	unavailable	waiting	available	delay
unavailable	0.00019	0.00009	0.00108	0.00864
waiting	0.01843	0.00873	0.10476	**0.83808**
available	0.00019	0.00009	0.00108	0.00864
delay	0.00019	0.00009	0.00108	0.00864
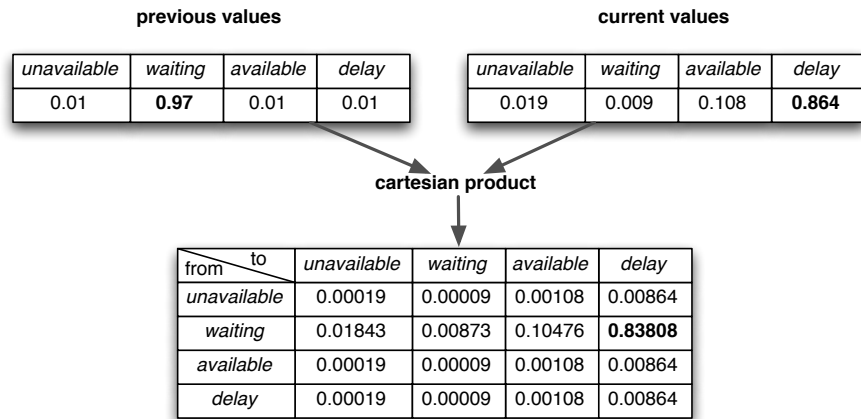
Figure 5.11. An example of calculating state-change probabilities

5.4.5   An Example

To demonstrate the knowledge updating process, we consider a simplified version of the victim rescue activity in the emergency response scenario to see how the PlanGraph model of the field of work is updated through several consecutive events, and meanwhile the events are enriched into event chains. The task involved in this example is quite simple. Whenever a victim is found, it needs to be rescued by

sending to a medical station for treatment. We assume there is only one station and one medical professional (*med*) working at it. There might be several drivers ($dr_1, dr_2, ...$) with rescue vehicles that can deliver the victim to the station.

Event 1: *'NewVictim'* The first event sent to the system is an external event reporting that a new victim has been found. The event has an event type *'NewVictim'* and has several key attributes (e.g. id, occurrence time, location, required delivery time). Figure 5.12 shows the knowledge updating process performed on this event.

1. When the event is first sent to the system, the system attempts to associate with any existing entities in the PlanGraph model. Because this is the start of the activity, no association can be found.

2. Then the event is further processed in the assessment step, where the system checks whether the event can lead to changes towards the action performance or trigger new action. By checking the association rules stored in the knowledge base, the system finds that every *'NewVictim'* will activate the goal to rescue it. Following this activation rule, the initial PlanGraph is generated with just one action node (*'rescue'*) representing this new action to rescue the victim.

3. After the assessment, the system finds that a new action has been added to the field of work, which triggers the elaboration process. The system searches the knowledge base to find a recipe for the *'rescue'* action, and extend the PlanGraph model with the parameters, conditions, and sub-actions. In this example, there is one parameter that is the *'victim'* who needs to be rescued, one condition (*'victimAtStation'*) that is the victim needs to be located at the medication station, and then the sub-action (*'medicalTreat'*) to perform the medical treatment on the victim. As these subsidiary entities are also new to the PlanGraph, the elaboration continues on each of them. During the elaboration of the parameter, the system assigns it with the value from the input event. The condition is elaborated with a new action (*'transport'*) to achieve it, which is further elaborated into subsidiary parameter and actions (*'vehicle'*, *'pickup'*, *'deliver'*). During the elaboration of action *'medicalTreat'*

and action *'transport'*, the system also looks for the potential actors who might be interested in these actions. Based on the actor *med*'s role as a medical professional, the system believes that *med* has the potential intention (*pot.int*) and is able (*able*) to perform the *'medicalTreat'* action. Similarly, the system believes all the drivers $dr_1, dr_2, ...$ have the potential intention (*pot.int*) and are able (*able*) to perform the *'transport'* action.

4. Because the event does not indicate any state change on current actions, the propagation is skipped.

As we can see in Figure 5.12, after the knowledge updating process, the Plan-Graph is updated to reflect the system's prediction on how the activity will be advanced. At the same time, the initial event is augmented with a new attribute pointing directly to the parameter node *'victim'* in the PlanGraph.
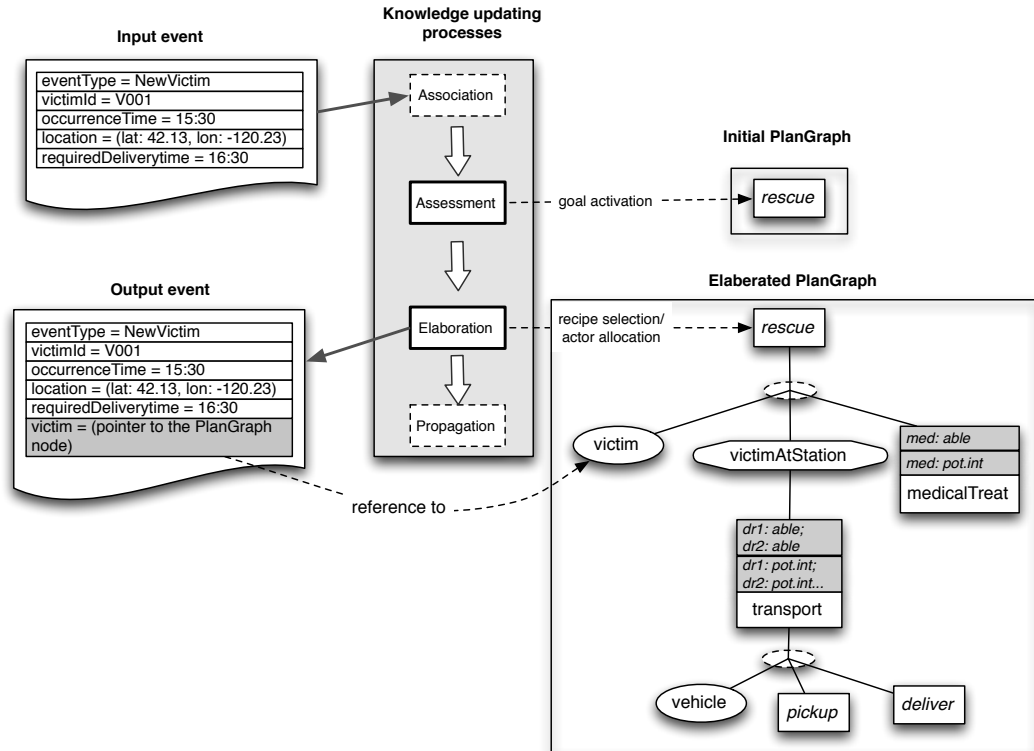

Figure 5.12. The knowledge updating example (*Event 1*)

Event 2: *'IntentionChange'* The second event is an internal event that is generated by the driver dr_1. After he interprets the first *'NewVictim'* event, he decides that he has the intention to perform the *'transport'* action to deliver this new victim to the medical station. As a result, he generates this *'Intention'* event to update his intention on the *'transport'* action from *pot.int* to *int.to*. This *'IntentionChange'* event has several key attributes, including the actor's id (dr_1), the action he intends to perform (*'transport'*), and the intention type (*int.to*).

As the system receives this event, the system first attempts to associate it with any existing entities in the PlanGraph model. Because this is an intention event, the system traverses through the PlanGraph to search for any match between the action (*'transport'*) mentioned in the event and the action nodes in the PlanGraph. When the system is able to find such a match, the system uses the information in the event to update dr_1's intention level on action *'transport'*. Meanwhile, the corresponding attributes of the actor and the action in the event are updated with direct pointers to the nodes in the PlanGraph.

The knowledge updating process then proceeds to the following steps, but none of them leads to further changes in both the PlanGraph and the event itself. Figure 5.13 shows the whole process performed on the second event.
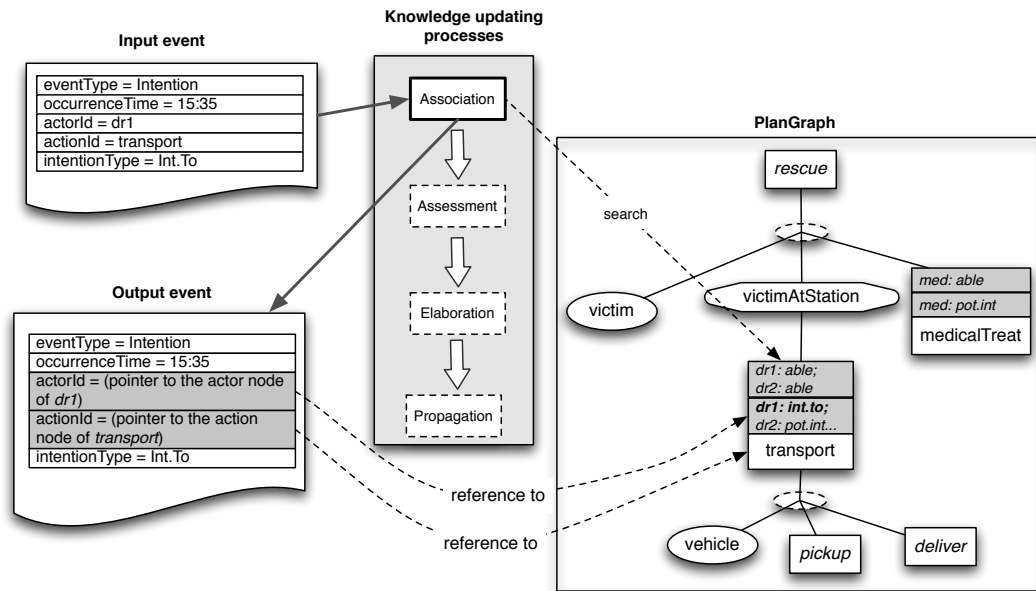

Figure 5.13. The knowledge updating example (*Event 2*)

Event 3: 'FuelLevelLow'   The third event happens after driver dr_1 starts the action *'transport'* and is on the way to pick up the victim. It is an external event indicating that the fuel level on driver dr_1's vehicle is running extremely low. The event has an event type *'FuelLevelLow'* and carries information about the driver, the vehicle, and the current fuel level. Figure 5.14 shows the knowledge updating process performed on this event.

1. When the event is first sent to the system, the system attempts to associate with any existing entities in the PlanGraph model. Because this is an external event indicating an attribute change on an entity, the system traverses through the PlanGraph to search for any match between the entity (*'vehicle'*) mentioned in the event and the nodes in the PlanGraph. When the system is able to find such a match, the system uses the information in the event to update the value attached to parameter (*'vehicle'*).

2. Then the event is further processed in the assessment step, where the system checks whether the event can lead to changes towards the action performance. By checking the conditions attached with the parameter (*'vehicle'*), the system finds that because of the low fuel level on the vehicle, the parameter becomes unavailable to perform the *'transport'* action. As a result, a new state change event *'ExecStatChange'* on the execution state of the parameter is derived, and added to the output event chain.

3. As no new action nodes are added to the PlanGraph, the elaboration process is skipped.

4. Because a state change event has been derived in the assessment step, the system attempts to predict future state changes in the field of work in the propagation step. A Bayesian network is constructed based on the current PlanGraph model and used to reason how likely the other entities in the PlanGraph will be impacted by the initial state change. After the Bayesian network-based reasoning, the system finds that the dr_1's action to pick up the victim (*'pickup'*) is likely to change its state from *executing* to *delaying*, and a new anticipatory event describing this state change on the execution state of the *'pickup'* action is generated, and added to the output event chain.

As we can see in Figure 5.14, after the knowledge updating process, the initial event is augmented into an event chain with three events: the original external event *'FuelLevelLow'*, the execution state change event on the parameter *'vehicle'* derived in the assessment step, and the anticipatory event predicting the execution state change event on the action *'pickup'* in the propagation step. This example shows the idea that not only the knowledge representation of the field of work, i.e. PlanGraph model is updated because of the event, but also the PlanGraph model can enrich the original event with system generated knowledge that is used to support the human actor's awareness processes in following chapter.

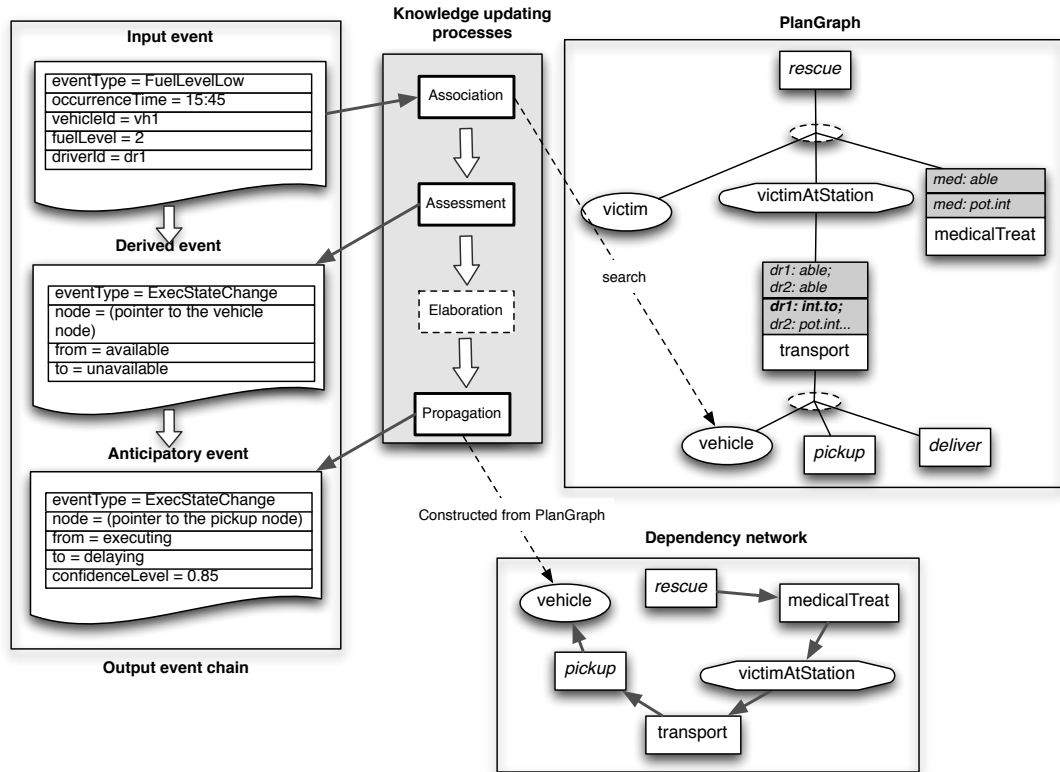**Figure 5.14.** The knowledge updating example (*Event 3*)

5.5   Discussion

In this chapter, we first focus on the knowledge representation of the field of work. We employ the PlanGraph model to represent the basic entities and relations in the

field of work, and then use this model to derive the knowledge about local scopes of work and dependencies. The PlanGraph theory exhibits some important characteristics that make it suitable to satisfy the three requirements for representing the field of work described in Section 4.2.1.

1. The PlanGraph models collaborative activities as hierarchically structured subsidiary actions. The basic components of actions, parameters, conditions, and contraints can be used to capture all the basic activity entities and the relations between them can be used to derive the various dependency relationships.

2. The PlanGraph model also encodes the mental attitudes requirements of human actors in the actions, including different types of intentions and beliefs on their capabilities. This knowledge provides the basis to identify local scopes of participants.

3. A critical point made in the PlanGraph model is the emphasis on the development of collaborative activities. With the development of the activity, the PlanGraph model needs to be updated accordingly, so that it always models the current state of the field of work.

4. The PlanGraph model provides a set of reasoning capabilities that can be operationalized to support computer reasoning.

Following that, we discuss the representation of events and identify the major events supported in this study. The central idea of our approach is the interaction between these two knowledge components, which has been described in the knowledge updating process. On one hand, the various events are consumed by the computer system to update its PlanGraph-based representation of the field of work. On the other hand, the PlanGraph model enrich the events with more meaningful contextual information, which is then used in next chapter to support the event-driven awareness processes.

Chapter 6

Our Approach: Promoting Event-Driven Awareness

The knowledge representation is used by the computer system to promote the awareness processes. Following the design principle of human-computer collaboration emphasizing the appropriate partition of responsibility between human actors and computer systems (Section 4.1.2), the promotion role of the computer system is to strike a balance between the system's reasoning capabilities and providing visual and interactive support.

6.1   Supporting Event Perception

6.2   Supporting Event Comprehension

6.3   Supporting Event Projection

6.4   Mediating Event Propagation

1. the ability to externalize awareness information

   2. the ability to control visibility

   3. the ability to detect conflicts

   4. the ability to seek/offer help

supporting perception

support comprehension

events are enriched, more context information, visual support

support projection

6.5 The Basic Interaction Scheme: Publish/Subscribe

6.6 Existing Notification Approaches

6.6.1 Topic-Based Approach

6.6.2 Type-Based Approach

6.6.3 Content-Based Approach

6.7 Activity-Based Approach

6.7.1 Types of Events

6.7.1.1 External and internal events

To understand events in geo-collaboration, an important distinction has to be made between *external* events (in the environment) and internal events (in the activities).

External events are changes in the physical environment that have impact on the performance of collaborative activities. For instance, the occurrence of a traffic accident may block the traffic flow, which makes an actors activity of delivering equipment to a medical station impossible to finish on time. External events in the environment can be characterized in three major categories: changes of identity (e.g. the occurrence of a traffic accident), attribute changes of an object (e.g. the contamination level of the chemical plan is increased), or spatial changes (e.g. the impacted area is enlarged due to the wind condition).

Internal events are state changes on the basic elements (i.e. resources, goals, activities) of geo-collaboration. Resource events indicate changes on the state of

any resources that are used in the activities. Resource events can impact response activities in different ways: (1) the creation of a new resource can trigger new activities (for example, a new victim may lead to the activity to perform decontamination and medical treatment); (2) the state change of a resource can impact the activity that requires the use of the resource (e.g. the mechanical breakdown of a vehicle makes the delivery activity unable to complete); (3) the state change of a resource can enable/disable the activity that depends on it (e.g. a victims location change leads to the satisfaction of a precondition that the victim must be located at the medical station, which further enables the activity to perform medical treatment on the victim). Condition events indicate changes on a certain relationship between multiple objects in a collaborative activity. For example, a condition may express a spatial relationship that the victim must be located at the decontamination station. The condition will change the state from open to holding when a driver has picked up the victim and transpored him/her to the station. Activity events are described as change of activity states, such as the initiation of a new activity, the completion or delay of an ongoing activity.

The distinction between external and internal events is very important to identify the set of events that should be captured and modeled in the awareness mechanism. Not all the external events in the environment are important for the actors to perform their activities. Rather, only a subset of the external events that can lead to changes within the activities (i.e. the internal events) is meaningful. The derivation of internal events from external events is an active cognitive process that requires interpreting the meaning of external events within the context of the collaborative activities.

6.7.1.2 Local and remote events

The distinction between external and internal events is based on the boundary between the surrounding environment and the overall collaborative activities as a whole. However, in order to understand the awareness information needed for each individual, another important distinction between local and remote events has to be made.

The local and remote events are defined with aspect to the local scope of work for each individual actor. Local events reflect the state changes of basic elements

(i.e. resources, goals, activities) within an actors local scope of work. For instance, the report of a new victim that needs to be decontaminated, the exceeding capacity of a decontamination station are local events within the decontamination managers local scope of work. Remote events are the changes outside an actors local scope of work. For instance, a victim that doesnt need to be decontaminated is transported to a shelter is a remote event for the decontamination manager. The distinction of local and remote events is relative to each individual actor. A local event of one actor may be a remote event for another actor due to their different local scopes of work. In addition, a remote event can be internal, reflecting the state changes of another actors activities, or it can be external, reflecting changes in the environment.

The distinction between local and remote events is important to identify the relevance of events that should be notified to each actor. Local events reflect changes within an actors local scope of work and therefore are directly relevant to the actor. On the other hand, due to the existence of the web of dependencies among activities, some of the remote events can (or potentially can) lead to changes in the actors local scope of work, i.e. the impact of a remote event may be propagated to the local scope of the actor as derived local events, and therefore they also become relevant. For instance, the traffic jam on the road is a remote event for the decontamination manager. However, because the traffic jam happens on the way of a victim to be delivered to a decontamination station, it can potentially delay the decontamination operation on the victim, which becomes a relevant local event for the decontamination manager. As a result, the goal of an awareness system is to notify an actor not only all the relevant local events, but also the subset of remote events that can be propagated to the actors local scope through the web of dependencies.

6.7.2   Event Subscription

6.7.2.1   Specifying Local Scopes

6.7.2.2   Defining Event Patterns

6.7.3   Event Matching

6.8   A Simulation Experiment

We conduct a simulation experiment to compare the content-based event notification approach and the proposed activity-based approach. The basic hypothesis is that, in dynamic environment where the user's activity changes frequently, the proposed activity-based approach provides a more accurate way for the user to express their awareness interests than the content-based approach. We use the emergency response scenario as described in the Introduction Chapter to perform the experiment, from the perspective of the decontamination manager.

6.8.1   Variables of Interests

We are interested in the capability of these two notification approaches in handling different types of activity dynamics. Particularly, we are interested in three types of dynamics that may occur in the scenario:

1. **Parameter Change**: the value of a parameter is changed. In the scenario, it can be the case when the decontamination manager re-assign a new station where a victim will be decontaminated.

2. **Plan Development**: the plan of the actor is changed. In the scenario, it can be the case when the decontamination manager starts to concern about certain equipment delivery after the stock is running low.

3. **Local Scope Change**: the actor's local scope of work is changed. In the scenario, it can be the case when another manager joins the activity, and take a subset of tasks away from the modeling manager.

To perform the experiment, we define four scenes to reflect the three types of activity dynamics:

1. Initial scene (S_0): indicates the current state of the decontamination manager's activities

2. Parameter change scene (S_1): indicates one of the parameters of the decontamination manager (station) has been assigned with a new value, comparing with S_0

3. Plan development scene (S_2): indicates one of the conditions has been elaborated into a more concrete plan, comparing with S_1

4. local scope change (S_3): indicates the another actor joins the activity, and the local scope of the modeled user has been reduced to a smaller set, comparing with S_2

6.8.2   Simulating Event Generation

A set of events are randomly generated based on the whole group activities. Given the current PlanGraph model, the events can happen on any of the activity, resource, or condition node, and indicate any types of possible changes that are defined in Chapter 5.

6.8.3   Creating Subscriptions

Two sets of content-based subscriptions will be generated.

1. The first set of subscriptions are defined merely based on the current interest of the manager in the initial scene (S_0). This set only includes the events that are relevant to the manager's current activities. Therefore, it can be considered as the minimal subscription set of events, using the content-based notification approach (CON_{min}).

2. The second set of subscriptions are defined based on considering all the three possible activities dynamics defined in scenes (S_1, S_2, S_3). It includes all the possible events that might be relevant to the manager in all the scenes. Because we attempt to include the maximum set of events that could be relevant for the manager across all the scenes, we define this content-based subscription as (CON_{max}).

In addition, the activity-based subscription will also be generated, following the steps described in previous section. The subscription includes the specification of local scope of the decontamination manager, his/her intentions and detailed event patterns on each node in the local scope. We define this activity-based subscription as ACT.

6.8.4 Procedures

The experiment includes four runs.

1. The first run R_0 is the human judgment on the relevance of the simulated event set to form the baseline result. The human analysts run through the list of simulated event set to decide on whether each event is relevant to the modeled decontamination manager, under the four scenes respectively. To reduce the subjective errors of human judgment, two analysts will perform the judgment separately and then discuss on the differences to form the final sets of events.

2. In the second run R_1, the simulated event set will be fed into the matching program following content-based notification approach, using the set of subscriptions defined as CON_{min}.

3. In the third run R_2, the simulated event set will be fed into the matching program following content-based notification approach, using the set of subscriptions defined as CON_{max}.

4. In the fourth run R_3, the the simuated event set will be fed into the matching program following activity-based notification approach, using the set of subscriptions defined as ACT.

6.8.5 Measures

Two measures will be used to compare the results of different notification approaches:

1. **Miss rate** is defined as the ratio between number of events that are considered as relevant in R_0 but missing in the result of R_i ($i \in \{1, 2, 3\}$), and the total number of events considered as relevant in R_0.

2. **False alarm rate** is defined as the ratio between number of events that are considered as relevant in R_i ($i \in \{1, 2, 3\}$), but missing in the result of R_0, and the total number of events considered as relevant in R_i ($i \in \{1, 2, 3\}$).

6.8.6 Results

Some results we expect from the experiment can be:

1. Comparing with the context-based notification using subscriptions defined as CON_{min}, the activity-based notification has lower miss rate in the scenes S_1, S_2, S_3. The difference between miss rates of the two methods increases as the scene evolves.

2. Comparing with the context-based notification using subscriptions defined as CON_{max}, the activity-based notification has lower false alarm rate in the scenes S_1, S_2, S_3. The difference between false alarm rates of the two methods decreases as the scene evolves.

6.8.7 Discussion

6.9 The Cognitive Basic

The cognitive principles that guide the design of visual displays for event interpretation.

1. The schemata theory (Bartlett et al.): when human respond to incoming stimuli, they make use of their existing knowledge as a frame of reference to make sense of the incoming stimuli and produce behavior. The active organization of existing knowledge is defined as 'schema'. Schemata can be merely mental templates in human mind, or retrieved dynamically from the external visual artifacts.

2. The relevance principle (Kosslyn 2006): Visual displays should present no more or no less information than is needed by the user. Presenting all of the relevant information in the display relieves the user of the need to maintain a detailed representation of this information in working memory, where presenting too much information in the display leads to visual clutter or distraction by irrelevant information (Rosenholtz et al. 2007; Wickens & Carswell, 1995).

3. The task specificity principle (Hegarty et al. 2009). Visual displays are used for many different tasks, and there is no such thing as a 'best' visual display, independent of the task to be carried out with this display.

What these cognitive principles can inform us are:

1. When the user interprets an awareness event, he/she needs to activate and connect the event to his/her contextual knowledge of the situation. The visual display can enhance the interpretation by externalizing part of the knowledge in the visual representation and freeing up working memory resources.

2. However, the effectiveness of the visual display depends on how much relevant information is represented. We want to provide the users with just enough contextual information without causing visual clutter or distraction.

3. The relevance of contextual information to be displayed is dependent on the current task to be carried out, or the decision that need to be made by the user.

6.10   Activity-Aware Event Interpretation

The problem we want to address here is that: how the computational model of activities and local scopes can inform the system to decide on what contextual information should be displayed when the users interpret awareness events.

The decisions can be made based on two factors: the incoming event and the activities/local scope the user is working on. By maintaining the PlanGraph model,

the system can infer how the event will impact the user's activities, which can be used to infer what decisions the user needs to make next, or what tasks the user will focus on, which can be used to decide on what should be displayed on the user interface.

Generate the set of rules based on SharedPlan Theory.

Some examples:

1. If the event initiates a new goal of the user, by elaborating on the plan, the system can infer that the user will work on identifying the set of parameters first.

2. If the event indicates the failure of a parameter, and there is no plan to fix the failure, the system can infer that the user will re-assign the values of this parameter.

3. if the event indicates the failure of a parameter, and there is plan to fix the failure, the system can infer that the user will work on the plan to fix it.

6.11   Experimental Study

6.11.1   Hypotheses

The general assumption is that maps with contextual information that is adapted to the user's current activities are more effective to support awareness event interpretation, than maps with fixed contextual information.

6.11.2   Experimental Design

6.11.2.1   Participants

Twelve undergraduate or graduate students will be recruited as participants in this study. All participants need to use computers on a daily basis. Prior experience with emergency response planning or operations is welcomed, but not required.

6.11.2.2   Tasks

The participants are asked to perform simulation tasks in an emergency response scenario, from the perspective of the decontamination manager. The tasks that the participants need to perform are driven by the events they receive.

Some example events and triggered tasks include:

1. E1: A new victim that needs to be decontaminated is reported by the victim manager. The task that the participant needs to do is to assign a decon station for the victim.

2. E2: The impacted area is enlarged and a decon station becomes inside of the impacted area. The task that the participants need to do is to re-assign the victims that are assigned to the station to other stations.

3. E3: A type of resource in a particular decon station is running low in stock. The participant need to request for a resource delivery from one of the suppliers.

4. E4: The delay on a victim's arrival at assigned station. The participant needs to check whether it will conflict with the victim's deadline on decontamination. If so, he/she needs to plan for sending the victim to other station.

6.11.2.3   Design settings

Participants are randomly assigned to one of the two design settings in one experiment session to perform their tasks. The three design settings are described as follow:

- The first design setting (D_1) shows a map showing locations of all the stations, victims, and resource suppliers as separate layers. The user can turn on/off each layer, or filter the objects based on spatial distances.

- In the second design setting (D_2), the features displayed in the map are adapted based on the event and the tasks that the user needs to perform.

6.11.2.4   Procedure

In the beginning of each experiment session, the participant is randomly assigned to one of the three design settings and given a couple of minutes to learn the functions of the assigned design setting. In addition to the training of the simulation system, the participant is also provided with the general picture of the collaborative activity. Information about other agents, related resources in the activity,

and the current goals and tasks of the group are presented to the participant, so he/she has a clear understanding of the task in the context of the whole activity and provides the basis for interpreting the awareness events. After the training session, the participant is asked to perform the simulation task. During the performance of the task, the participant is notified with a list of awareness events, and asked to finish the assessment form to answer relevant questions after each event notification.

Each experiment session is composed as a series of interaction episodes between the participant and the system. Each interaction episode defines how an agent (role-played by the participant) responds to a particular event in the collaborative scenario. Each interaction episode starts with the provision of an awareness event in the interface, and then the participant is perform corresponding tasks based on their understanding of the event.

6.11.2.5 Measurement

To measure the overall performance during the experiment session, we record the completion time of each experiment session and the participant's responses to each awareness event. In the design phase of the awareness events, each event is associated with a set of optimal responses. The participant's responses are then compared with the optimal set to indicate the quality level of task completion.

6.11.3 Results

In general, we expect that

1. the participants in design setting D_2 will generate more valid responses than the participants in design setting D_1.

2. the participants in design setting D_2 should spend less time to finish the tasks than the participants in design settings D_1 in average.

6.11.4 Discussion

Chapter 7

Our Approach: Architecture and Implementation

Case Studies

Possible variables:

1. complexity of interdependencies (e.g. the number of dependencies)

2. task-specific asymmetries (e.g. to what extent the local scopes are overlapped)

3. dynamics (e.g. chances of re-planning)

4. concurrent multiple activities

Chapter 9

Conclusion and Future Work

Bibliography

[1] Marilyn Jager Adams, Yvette J. Tenney, and Richard W. Pew. Situation Awareness and the Cognitive Management of Complex Systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):85–104, March 1995.

[2] Rosa Alarcón and David Fuller. Intelligent Awareness in Support of Collaborative Virtual Work Groups. In Jörg Haake and José Pino, editors, *Groupware: Design, Implementation, and Use*, volume 2440 of *Lecture Notes in Computer Science*, pages 369–384. Springer Berlin / Heidelberg, 2002.

[3] J F Allen and G Ferguson. Actions and events in interval temporal logic. *Journal of logic and computation*, 4(5):531, 1994.

[4] Gennady Andrienko, Natalia Andrienko, and Marco Heurich. An event-based conceptual model for context-aware movement analysis. *International Journal of Geographical Information Science*, 25(9):1347–1370, September 2011.

[5] P Antunes, C Sapateiro, J Pino, V Herskovic, and S Ochoa. Awareness Checklist: Reviewing the Quality of Awareness Support in Collaborative Applications. *Collaboration and Technology*, pages 202–217, 2010.

[6] H Artman and C Garbis. Situation awareness as distributed cognition. In *Proceedings of ECCE*, volume 98, 1998.

[7] Gregory Bedny and David Meister. Theory of Activity and Situation Awareness. *International Journal of Cognitive Ergonomics*, 3(1):63–72, January 1999.

[8] Steve Benford and Lennart Fahlén. A spatial model of interaction in large virtual environments. pages 109–124, September 1993.

[9] Steve Benford, Chris Greenhalgh, Tom Rodden, and James Pycock. Collaborative virtual environments. *Communications of the ACM*, 44(7):79–85, July 2001.

[10] Brandon Bennett, Anthony G Cohn, Frank Wolter, and Michael Zakharyaschev. Multi-Dimensional Modal Logic as a Framework for Spatio-Temporal Reasoning. *Applied Intelligence*, 17(3):239–251.

[11] R Bentley and T Horstmann. Supporting collaborative information sharing with the World Wide Web: The BSCW shared workspace system. *Proceedings of the 4th Int. WWW Conference*, 1995.

[12] Thmoas Berlage and Markus Sohlenkamp. Visualizing Common Artefacts to Support Awareness in Computer-Mediated Cooperation. *Computer Supported Cooperative Work (CSCW)*, 8(3):207–238, 1999.

[13] Susanne Bodker. Computers in Mediated Human Activity. *Mind, Culture, and Activity*, 4(3):149–158, July 1997.

[14] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, May 2004.

[15] Scott M Brown, Eugene Santos, and Jr. Active User Interfaces. Technical report, Intelligent Distributed Information Systems Laboratory, University of Connecticut, 1999.

[16] F Cabitza, M P Locatelli, and C Simone. Promoting Process-Based Collaboration Awareness to Integrate Care Teams. In *Business Process Management Workshops*, pages 385–396. Springer, 2009.

[17] Guoray Cai, Hongmei Wang, and Alan MacEachren. Communicating Vague Spatial Concepts in Human-GIS Interactions: A Collaborative Dialogue Approach. volume 2825 of *Lecture Notes in Computer Science*, pages 287–300. Springer Berlin / Heidelberg, 2003.

[18] Guoray Cai, Hongmei Wang, Alan M. MacEachren, and Sven Fuhrmann. Natural Conversational Interfaces to Geospatial Databases. *Transactions in GIS*, 9(2):199–221, March 2005.

[19] J M Carroll, D C Neale, P L Isenhour, M B Rosson, and D S McCrickard. Notification and awareness: synchronizing task-oriented collaborative activity. *International Journal of Human-Computer Studies*, 58(5):605–632, 2003.

[20] John M Carroll, Mary Beth Rosson, Gregorio Convertino, and Craig H Ganoe. Awareness and teamwork in computer-supported collaborations. *Interacting with Computers*, 18(1):21–46, 2006.

[21] John M. Carroll, Mary Beth Rosson, Umer Farooq, and Lu Xiao. Beyond being aware. *Information and Organization*, 19(3):162–185, July 2009.

[22] P Carstensen, T Tuikka, and C Sø rensen. Are We Done Now? Towards Requirements for Computer Supported Cooperative Software Testing. In *17th IRIS Seminar, Sy{\\"o}te, Finland*, pages 6–9, 1994.

[23] Marcelo Cataldo, Patrick A Wagstrom, James D Herbsleb, and Kathleen M Carley. Identification of coordination requirements: implications for the Design of collaboration and awareness tools. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 353–362, New York, NY, USA, 2006. ACM.

[24] K Crowston. A taxonomy of organizational dependencies and coordination mechanisms. *MIT Center for Coordination Science Working Paper*, 1994.

[25] Nikunj P. Dalal and George M. Kasper. The design of joint cognitive systems: the effect of cognitive coupling on performance. *International Journal of Human-Computer Studies*, 40(4):677–702, April 1994.

[26] P Dourish and V Bellotti. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pages 107–114. ACM New York, NY, USA, 1992.

[27] Paul Dourish and Sara Bly. Portholes: supporting awareness in a distributed work group. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '92*, pages 541–547, New York, New York, USA, June 1992. ACM Press.

[28] M J Egenhofer. Deriving the composition of binary topological relations. *Journal of Visual Languages and Computing*, 5(2):133–149, 1994.

[29] M R Endsley. Measurement of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):65–84, 1995.

[30] M R Endsley, W B Jones, K Schneider, M McNeese, and M Endsley. A model of inter-and intrateam situational awareness: implications for design, training, and measurement. *New Trends in Cooperative Activities Understanding System Dynamics in Complex Environments*, pages 46–67, 2001.

[31] Mica R. Endsley. Toward a Theory of Situation Awareness in Dynamic Systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):32–64, March 1995.

[32] Thomas Erickson, David N. Smith, Wendy A. Kellogg, Mark Laff, John T. Richards, and Erin Bradner. Socially translucent systems: social proxies, persistent conversation, and the design of babble. In *Proceedings of the SIGCHI conference on Human factors in computing systems the CHI is the limit - CHI '99*, pages 72–79, New York, New York, USA, May 1999. ACM Press.

[33] A Espinosa, F J Lerch, R E Kraut, E Salas, and S M Fiore. Explicit vs. implicit coordination mechanisms and task dependencies: One size does not fit all. *Team cognition: Understanding the factors that drive process and performance*, pages 107–129, 2004.

[34] O. Etzion and P. Niblett. *Event Processing in Action*. Manning Publications Co., 2010.

[35] Robert S. Fish, Robert E. Kraut, Robert W. Root, and Ronald E. Rice. Evaluating video as a technology for informal communication. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '92*, pages 37–48, New York, New York, USA, June 1992. ACM Press.

[36] Geraldine Fitzpatrick, Simon Kaplan, Tim Mansfield, David Arnold, and Bill Segall. Supporting Public Availability and Accessibility with Elvin: Experiences and Reflections. *Computer Supported Cooperative Work (CSCW)*, 11(3):447–474, 2002.

[37] A U Frank. Qualitative Spatial Reasoning about Cardinal Directions'. *Auto Carto 10: Technical Papers*, page 148, 1991.

[38] Ludwin Fuchs. AREA: A Cross-Application Notification Service for Groupware. *Proceedings of the Sixth European Conference on Computer-Supported Cooperative Work*, pages 61–80, 1999.

[39] Ludwin Fuchs, Uta Pankoke-Babatz, and Wolfgang Prinz. Supporting cooperative awareness with local event mechanisms: the groupdesk system. pages 247–262, September 1995.

[40] S R Fussell, R E Kraut, F J Lerch, W L Scherlis, M M McNally, and J J Cadiz. Coordination, overload and team performance: effects of team communication strategies. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 275–284. ACM, 1998.

[41] Tom Gross and Wolfgang Prinz. Modelling Shared Contexts in Cooperative Environments: Concept, Implementation, and Evaluation. *Computer Supported Cooperative Work (CSCW)*, 13(3):283–303, 2004.

[42] B J Grosz and S Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

[43] Barbara J. Grosz and Luke Hunsberger. The dynamics of intention in collaborative activity. *Cognitive Systems Research*, 7(2-3):259–272, June 2006.

[44] Jonathan Grudin. Groupware and social dynamics: eight challenges for developers. *Communications of the ACM*, 37(1):92–105, January 1994.

[45] Carl Gutwin and Saul Greenberg. A Descriptive Framework of Workspace Awareness for Real-Time Groupware. *Computer Supported Cooperative Work (CSCW)*, 11(3):411–446, 2002.

[46] Carl Gutwin, Mark Roseman, and Saul Greenberg. A usability study of awareness widgets in a shared workspace groupware system. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work - CSCW '96*, pages 258–267, New York, New York, USA, November 1996. ACM Press.

[47] Christian Heath, Marcus Sanchez Svensson, Jon Hindmarsh, Paul Luff, and Dirk vom Lehn. Configuring Awareness. *Computer Supported Cooperative Work (CSCW)*, 11(3):317–347, 2002.

[48] Mary Hegarty. The Cognitive Science of Visual-Spatial Displays: Implications for Design. *Topics in Cognitive Science*, 3(3):446–474, July 2011.

[49] D Hernandez, E Clementini, and P Di Felice. Qualitative distances. *Spatial Information Theory A Theoretical Basis for GIS*, pages 45–57, 1995.

[50] Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design Science in Information Systems Research. *MIS Quarterly*, 28(1):75–105, March 2004.

[51] Scott E Hudson and Ian Smith. Techniques for addressing fundamental privacy and disruption tradeoffs in awareness support systems. In *CSCW '96: Proceedings of the 1996 ACM conference on Computer supported cooperative work*, pages 248–257, New York, NY, USA, 1996. ACM.

[52] Luke Hunsberger and Charles Ortiz. Dynamic intention structures I: a theory of intention representation. *Autonomous Agents and Multi-Agent Systems*, 16(3):298–326.

[53] E Hutchins and G Lintern. *Cognition in the Wild*, volume 262082314. MIT press Cambridge, MA, 1995.

[54] G Jakobson, J Buford, and L Lewis. Towards an architecture for reasoning about complex event-based dynamic situations. In *Third International Workshop on Distributed Event-Based Systems*, page 62. Citeseer, 2004.

[55] Ken Kaneiwa, Michiaki Iwazume, and Ken Fukuda. An Upper Ontology for Event Classifications and Relations. In Mehmet Orgun and John Thornton, editors, *AI 2007: Advances in Artificial Intelligence*, volume 4830 of *Lecture Notes in Computer Science*, pages 394–403. Springer Berlin / Heidelberg, 2007.

[56] A Kittur, B Lee, and R E Kraut. Coordination in collective intelligence: the role of team structure and task interdependence. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 1495–1504. ACM, 2009.

[57] Scaife M. and Rogers Y. External cognition: how do graphical representations work? *International Journal of Human-Computer Studies*, 45(2):185–213, 1996.

[58] T W Malone and K Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys (CSUR)*, 26(1):87–119, 1994.

[59] P Markopoulos. A design framework for awareness systems. *Awareness Systems*, pages 49–72, 2009.

[60] D. Scott McCrickard, C. M. Chewar, Jacob P. Somervell, and Ali Ndiwalana. A model for notification systems evaluation—assessing user goals for multitasking activity. *ACM Transactions on Computer-Human Interaction*, 10(4):312–338, December 2003.

[61] Gero Mhl, Ludger Fiege, and Peter Pietzuch. Distributed Event-Based Systems. November 2010.

[62] Susan Mohammed and Brad C. Dumville. Team mental models in a team knowledge framework: expanding theory and measurement across disciplinary boundaries. *Journal of Organizational Behavior*, 22(2):89–106, March 2001.

[63] B A Nardi. *Context and consciousness: activity theory and human-computer interaction*. The MIT Press, 1996.

[64] U Neisser. *Cognition and reality: Principles and implications of cognitive psychology*. WH Freeman/Times Books/Henry Holt & Co, 1976.

[65] A A Nofi. Defining and measuring shared situational awareness. Technical report, DTIC Document, 2000.

[66] Donald A Norman. Design principles for cognitive artifacts. *Research in Engineering Design*, 4(1):43–50, 1992.

[67] Antti Oulasvirta, Renaud Petit, Mika Raento, and Sauli Tiitta. Interpreting and Acting on Mobile Awareness Cues. *Human–Computer Interaction*, 22(1):97–135, 2007.

[68] J Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann, 1988.

[69] Elin Rø nby Pedersen and Tomas Sokoler. AROMA: abstract representation of presence supporting mutual awareness. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97*, pages 51–58, New York, New York, USA, March 1997. ACM Press.

[70] P P Perla, M Markowitz, A A Nofi, C Weuve, and J Loughran. Gaming and shared situation awareness. Technical report, DTIC Document, 2000.

[71] Jens Pottebaum, Alexander Artikis, Robin Marterer, and Rainer Koch. Event Definition for the Application of Event Processing to Intelligent Resource Management. In *Proceedings of the 8th International ISCRAM Conference*, number May, Lisbon, Portugal, 2011.

[72] Wolfgang Prinz. NESSIE: An Awareness Environment for Cooperative Settings. *Proceedings of the sixth conference on European Conference on Computer Supported Cooperative Work*, pages 391–410, 1999.

[73] W O Quine. Events and reification. *Events*, pages 107–116, 1985.

[74] A B Raposo, L P Magalhães, I L M Ricarte, and H Fuks. Coordination of collaborative activities: A framework for the definition of tasks interdependencies. In *Proceedings of Seventh International Workshop on Groupware*, pages 170–179. IEEE, 2002.

[75] Markus Rittenbruch. Atmosphere: A Framework for Contextual Awareness. *International Journal of Human-Computer Interaction*, 14(2):159–180, June 2002.

[76] Markus Rittenbruch and Gregor McEwan. An Historical Reflection of Awareness in Collaboration. *Awareness Systems*, pages 3–48, 2009.

[77] Markus Rittenbruch, Stephen Viller, and Tim Mansfield. Announcing Activity: Design and Evaluation of an Intentionally Enriched Awareness Service. *HumanComputer Interaction*, 22(1-2):137–171, 2007.

[78] Tom Rodden. Populating the application: a model of awareness for cooperative applications. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work - CSCW '96*, pages 87–96, New York, New York, USA, November 1996. ACM Press.

[79] Mark Roseman and Saul Greenberg. Building real-time groupware with GroupKit, a groupware toolkit. *ACM Transactions on Computer-Human Interaction*, 3(1):66–106, March 1996.

[80] E Salas, C Prince, D P Baker, and L Shrestha. Situation awareness in team performance: Implications for measurement and training. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):123–136, 1995.

[81] P M Salmon, N A Stanton, G H Walker, D Jenkins, C Baber, and R McMaster. Representing situation awareness in collaborative systems: a case study in the energy distribution domain. *Ergonomics*, 51(3):367–84, March 2008.

[82] Paul M. Salmon, Neville A. Stanton, Guy H. Walker, Chris Baber, Daniel P. Jenkins, Richard McMaster, and Mark S. Young. What really is going on? Review of situation awareness models for individuals and teams. *Theoretical Issues in Ergonomics Science*, 9(4):297–323, July 2008.

[83] Paul M. Salmon, Neville A. Stanton, Guy H. Walker, Daniel P. Jenkins, and Laura Rafferty. Is it really better to share? Distributed situation awareness and its implications for collaborative system design. *Theoretical Issues in Ergonomics Science*, 11(1-2):58–83, January 2010.

[84] Ovidiu Sandor, Cristian Bogdan, and John Bowers. Aether: an awareness engine for CSCW. pages 221–236, September 1997.

[85] K Schmidt and L Bannon. Taking CSCW seriously. *Computer Supported Cooperative Work (CSCW)*, 1(1):7–40, 1992.

[86] Kjeld Schmidt. The Problem with 'Awareness': Introductory Remarks on 'Awareness in CSCW'. *Computer Supported Cooperative Work (CSCW)*, 11(3):285–298, 2002.

[87] Kjeld Schmidt and Carla Simonee. Coordination mechanisms: Towards a conceptual foundation of CSCW systems design. *Computer Supported Cooperative Work (CSCW)*, 5(2):155–200, 1996.

[88] S Y Shen and M J Shaw. Managing coordination in emergency response systems with information technologies. *Proceedings of the Tenth Americas Conferences on Information Systems*, pages 2110–2120, 2004.

[89] Y Shu and K Furuta. An inference method of team situation awareness based on mutual awareness. *Cognition, Technology & Work*, 7(4):272–287, 2005.

[90] J S Sichman and R Conte. Multi-agent dependence by dependence graphs. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 483–490. ACM, 2002.

[91] Riyaz Sikora and Michael J Shaw. A Multi-Agent Framework for the Coordination and Integration of Information Systems. *Management Science*, 44(11):pp. S65–S78, 1998.

[92] C Simone, M Divitini, and K Schmidt. A notation for malleable and interoperable coordination mechanisms for CSCW systems. In *Proceedings of conference on Organizational computing systems*, pages 44–54. ACM, 1995.

[93] Carla Simone and Stefania Bandini. Integrating Awareness in Cooperative Applications through the Reaction-Diffusion Metaphor. *Computer Supported Cooperative Work (CSCW)*, 11(3):495–530, 2002.

[94] Kip Smith and P. A. Hancock. Situation Awareness Is Adaptive, Externally Directed Consciousness. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):137–148, March 1995.

[95] M Sohlenkamp, W Prinz, and L Fuchs. PoliawaC: design and evaluation of an awareness-enhanced groupware client. *AI & Society*, 14(1):31–47, 2000.

[96] MD Spiteri. *An architecture for the notification, storage and retrieval of events.* PhD thesis, University of Cambridge, 2000.

[97] N A Stanton, R Stewart, D Harris, R J Houghton, C Baber, R McMaster, P Salmon, G Hoyle, G Walker, M S Young, M Linsell, R Dymott, and D Green. Distributed situation awareness in dynamic systems: theoretical development and application of an ergonomics methodology. *Ergonomics*, 49(12-13):1288–311, January 2006.

[98] Neville A. Stanton, Paul M. Salmon, Guy H. Walker, and Daniel Jenkins. Genotype and phenotype schemata and their role in distributed situation awareness in collaborative systems. *Theoretical Issues in Ergonomics Science*, 10(1):43–68, January 2009.

[99] Kimberly Tee, Saul Greenberg, and Carl Gutwin. Artifact awareness through screen sharing for distributed groups. *International Journal of Human-Computer Studies*, 67(9):677–702, September 2009.

[100] Loren G. Terveen. Overview of human-computer collaboration. *Knowledge-Based Systems*, 8(2-3):67–81, April 1995.

[101] J.J. Thomas and K.A. Cook. A visual analytics agenda. *IEEE Computer Graphics and Applications*, 26(1):10–13, January 2006.

[102] Brian Tomaszewski and Alan M. MacEachren. Geo-historical context support for information foraging and sensemaking: Conceptual model, implementation, and assessment. In *2010 IEEE Symposium on Visual Analytics Science and Technology*, pages 139–146. IEEE, October 2010.

[103] M. Turoff, M. Chumer, B.V. de Walle, and X. Yao. The design of a dynamic emergency response management information system (DERMIS). *Journal of Information Technology Theory and Application (JITTA)*, 5(4), 2004.

[104] J Uhlarik and D A Comerford. A review of situation awareness literature relevant to pilot surveillance functions. Technical report, DTIC Document, 2002.

[105] Chunhua Weng and John H. Gennari. Asynchronous collaborative writing through annotations. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work - CSCW '04*, page 578, New York, New York, USA, November 2004. ACM Press.

[106] C. D. Wickens. Situation Awareness: Review of Mica Endsley's 1995 Articles on Situation Awareness Theory and Measurement. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 50(3):397–403, June 2008.

[107] Bo Yu and Guoray Cai. Using Intentions and Plans of Mobile Activities to Guide Geospatial Web Service Composition. In *Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific*, pages 141–148. IEEE, 2010.

[108] E S K Yu and J Mylopoulos. An actor dependency model of organizational work: with application to business process reengineering. In *Proceedings of the conference on Organizational computing systems*, pages 258–268. ACM, 1993.

[109] May Yuan. Representing Complex Geographic Phenomena in GIS. *Cartography and Geographic Information Science*, 28:83–96(14), 2001.

[110] L Zelnik-Manor and M Irani. Event-based analysis of video. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II—-123. IEEE, 2001.

[111] Qixing Zheng, Kellogg Booth, and Joanna McGrenere. Co-authoring with structured annotations. In *Proceedings of the SIGCHI conference on Human Factors in computing systems - CHI '06*, page 131, New York, New York, USA, April 2006. ACM Press.

Vita

Bo Yu

Here goes the vita. To be added.