Comput Math Organiz Theor (2007) 13:113–146
DOI 10.1007/s10588-006-9002-7

Complexity theory and collaboration: An agent-based simulator for a space mission design team

Narjès Bellamine-Ben Saoud · Gloria Mark

Published online: 13 September 2006

Abstract In this paper, we investigate how complexity theory can benefit collaboration by applying an agent-based computer simulation approach to a new form of synchronous real-time collaborative engineering design. Fieldwork was conducted with a space mission design team during their actual design sessions, to collect data on their group conversations, team interdependencies, and error monitoring and recovery practices. Based on the fieldwork analysis, an agent-based simulator was constructed. The simulation shows how error recovery and monitoring is affected by the number of small group, or sidebar, conversations, and consequent noise in the room environment. This simulation shows that it is possible to create a virtual environment with cooperating agents interacting in a dynamic environment. This simulation approach is useful for identifying the best scenarios and eliminating potential catastrophic combinations of parameters and values, where error recovery and workload in collaborative engineering design could be significantly impacted. This approach is also useful for defining strategies for integrating solutions into organizations.

Keywords Extreme collaboration · Collaborative design · Complexity theory · Agent-based modeling and simulation · Map of interdependencies · Errors · Sidebar conversations

1 Introduction

In this paper, we investigate the new emerging theory of complexity (Axelrod and Cohen, 2000) and how it can be used to benefit collaboration. We apply a computer

N. Bellamine-Ben Saoud (✉)
RIADI-GDL Laboratory, University of Tunis, Tunisia
e-mail: Narjes_bellamine@fulbrightweb.org

G. Mark
Department of Informatics, University of California, Irvine
e-mail: gmark@ics.uci.edu

simulation approach to a new form of synchronous collaboration used in engineering. By doing so our goal is to provide quantitative and empirical results to help plan cooperative processes.

Complexity theory is actually not a theory but rather is a paradigm, set of procedures and techniques, and an approach to complex systems (Carley, 2001). It involves the study of many actors and their interactions (Axelrod, 1997). Much of the formal work in complexity theory is conducted in the fields of physics, biology, and chemistry; however, complex processes also occur in organizations (Carley, 2001). An emerging research area is the application of complexity theory to study organizations (e.g., McKelvey, 1999; Lissack, 1999; Goldstein, 1999; Dent, 1999; Ballot and Weisbuch, 2000; Bar-Yam, 2000)

It is recognized that computer simulation, especially agent-based simulation, is valuable for studying complex systems. Computer simulation in general, and agent-based simulation in particular, is a primary research tool of complexity theory (Axelrod and Cohen, 1997). Several agent-based simulation applications have been developed in many domains (e.g. Kreft et al., 1998; Strader et al., 1998; Terano, 2000). Simulation studies have also been done to understand and model various configurations of people and collaborative technologies (c.f., Dugdale et al., 2000; Pavard and Dugdale, 2000).

In this paper, we provide a brief overview of fieldwork conducted with a collaborative design team. From these results, aspects of the collaborative process were identified and incorporated into a simulation model. Our main objectives in this paper are to design and develop an agent-based simulator in order to (1) have a virtual collaboration environment where we can evaluate different cooperation scenarios that apply to the engineering practices we have observed from a case study, and (2) to study selected group activities involved in monitoring an information source and recovering errors.

Our computer simulation approach can assist design choices involving technologies, actors, and environments. The approach could identify the best scenarios and eliminate, for example, "catastrophic" combinations of parameters and values. It would also be useful for defining strategies for integrating solutions into organizations by providing ideas on what to change first and the probable consequences of such a change. Our findings would contribute to the design of solutions for computer-supported cooperative work by providing a "manageable" evaluation of great number of different configurations.

This paper is organized as follows. In Section 2, we describe results from a case study of collaboration in the domain of space mission design. In Section 3, we describe how complexity theory relates to this type of collaboration. In Section 4, we explain the main collaboration aspects we modeled and motivations. In Section 5, we detail the simulation model and validation. In Section 6, we present results of the model. In Section 7, we relate significant points of complexity theory to our case study and results. In Section 8, we present our conclusions and description of future work.

2 The case study: extreme collaboration in a warroom

In this paper we simulate a unique face-to-face collaborative setting, where a collocated team uses computer technologies, an innovative design process, and an electronic room environment to streamline communication and information flow. Though the effect of collocating teams has been investigated (Teasley et al., 2000) there still remains a number of questions on the effects of this type of work, such as stress, scalability of people, and the ability to detect errors.

We base our computer simulation on data collected from three months of fieldwork observations of an actual space mission design team in a large aerospace organization. For a more detailed description of the collaboration process and fieldwork see (Mark, 2002). The team designs proposals which define all aspects of a space mission, e.g. the telecommunications devices, power and propulsion, etc. Team M[1] is composed of a leader and sixteen members who are engineers with expertise in a particular subsystem for space mission design, such as power, thermal, or telecommunications systems. The whole team works together in a large technology-mediated room, which we call a warroom. Each team member has a workstation at one of four tables in the room, but they move often around the room to converse with others. Collaboration is supported by various technologies in the warroom: three public displays which can be networked to anyone's workstation, databases of past missions, an orbit visualization program, and a simple publish-subscribe system of networked spreadsheets to exchange information.

2.1 Methodology

Forty-two hours of observation were made, coding the group activity in real time, and focusing on small group (*sidebar*) conversations. In addition, in-depth interviews with the team members were conducted. Two sessions were videotaped, but permission to videotape other sessions was not allowed by the customers.

2.2 Overview of the team collaboration

The fieldwork observations revealed that the group members solve design problems by changing continually between individual, small group work, and entire team work. In sidebars, engineers form a small group in which they discuss an aspect of the design in parallel to the rest of the team working. They may negotiate starting values for calculations, may question the assumptions behind a parameter, can challenge another's result, or discuss an alternative plan for a part of the mission. Sidebars are characterized by being very dynamic (team members join and leave quickly), and semi-public. The team members are aware that their sidebar discussions are being overheard by teammates, and that their content may be relevant for others' work.

2.3 Interdependencies among the team members

A number of approaches have targeted shared understandings in teamwork and their role in collaboration: shared meaning (Weick, 1979), shared mental models (Levesque et al., 2001; Carley, 1997), and transactive memory (Liang et al., 1995; Hollingshead, 2000). One of the important notions that has emerged from this body of work

[1] A pseudonym.

is that *what* information is shared by the team is critical in guiding its collaboration (Cannon-Bowers and Sales, 2001; Klimoski and Mohammed, 1994). Transactive memory theorists are interested in the interdependencies among a group of people which leads to a more complex body of knowledge than what each individual possesses. By sharing task-specific knowledge, tacit understandings can develop over time which is used by team members to coordinate their actions (Cannon-Bowers et al., 1993). Team M members share the understanding of most, if not all, of the interdependencies that exist among all the team members. Over time, this knowledge of the complex interdependent relationships in team M has become tacit; the engineers do not explicitly speak about their interdependencies but through their actions they use this knowledge to share information. The engineers can determine whether a sidebar might likely be relevant for them or whether they should "broadcast" information so that a sidebar could be overheard by a team member for who it is relevant. For example, the Telecom Systems engineer needs information from the Control Data Systems, Attitude Control Systems, and Ground Systems engineers to calculate his results. Figure 1 shows a "map" of the team interdependencies for information. This map was built iteratively by asking each individual team member about their connections with others. Consistency was checked among the team members.

2.4 Monitoring the environment

Monitoring the environment has helped the team recover from calculation errors, when they hear a discussion about a value that is inconsistent with what appears on their own spreadsheet. Overhearing others has also led to the recovery from software errors because the information discussed publicly led team members to question

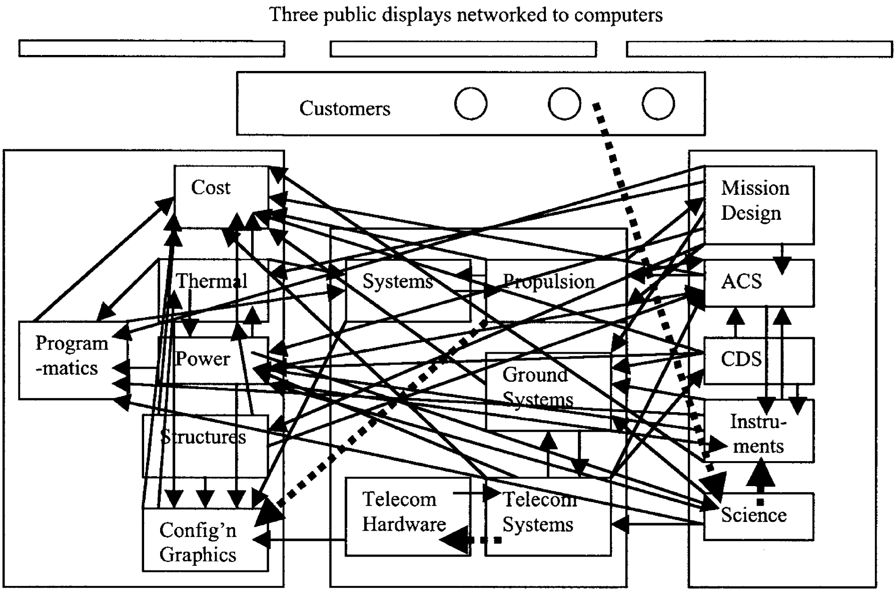


Fig. 1 An overview of the team interdependencies for information

their software program's performance. Frequently a team member will shout out to the room when an error is discovered, e.g. a value of a mission parameter from the public display. Generally public discussion ensues, and the team then adjusts their calculations.

3 Extreme collaboration is a complex system

In early discussions of complex systems, Simon (1969, p. 86) stated that complex systems are composed of "a large number of parts that interact in a nonsimple way", and that in complex systems, "the whole is more than the sum of the parts". In complex systems we do not have pure superposition of phenomena and processes (Keyser, 2000).

Mark (2002) refers to the collocated space mission design team process as an example of extreme collaboration. Extreme collaboration is a very complex process where *heterogeneous agents* (actors: engineers, a leader, customers), who are *dynamically organized* into subgroups, *adapt* their behavior (working individually, within small groups, moving around the room), *monitor* information selectively and give *feedback* continuously and in an *unpredictable* way. They perform these actions to solve problems, improve the mission design, and ultimately refine costs associated with the mission.

Pavard and Dugdale (2000, 2001) identified four specific properties of complex systems, which fit with our case study of extreme collaboration: non-determinism, limited functional decomposability, the distributed nature of information and representation, and emergence and self-organization. We will explain each of these properties as it relates to our case study.

Non-determinism refers to the fact that it is impossible to anticipate precisely the behavior of the system even if we completely know the functions of its constituents (Pavard and Dugdale, 2001). Two important examples of non-deterministic and non-traceable mechanisms in our case are: (1) sidebar formation, and (2) broadcasting information within the working room as a result of engineers talking together in a sidebar. At each moment of the meeting we cannot predict which subsystems will call for a sidebar and with which partner the engineer will discuss or ask for explanations. We also cannot predict how a sidebar will evolve (i.e. for which engineers it will be relevant and which engineers are ready or even able to join). It is extremely difficult to trace the flow of information associated with the broadcasting of sidebar discussions: neither the actors, nor the observer have the means or the cognitive resources to know who heard the message and even less to know how it was interpreted (Pavard and Dugdale, 2001).

Limited functional decomposability. Since our system is dynamic and is in permanent interaction with its environment, it is therefore difficult to study its properties by decomposing it into functional stable parts. In the current study, it has been observed that physical collocation provides multi-sensory awareness of teammate activities: overhearing conversations, the team leader's comments, watching the public displays, seeing who is speaking with whom, and seeing published results on various displays. Further, the engineers dynamically monitor activity by focusing on those with whom they are interdependent for information.

Distributed nature of information and representation. In Team M, the common object of cooperation, i.e. the mission proposal, is naturally distributed between subsystems and managed by their representatives. The same information (e.g. values or calculations related to the spacecraft) are often integrated and represented in a specific form by each subsystem. (The same data can be converted in different calculation systems having different physical units; a temperature may be expressed in Celsius or Kelvin scale.) Also, this team shares resources (e.g. public displays), which enable knowledge distribution and information sharing among the actors. Also, the broadcast mechanism (in verbal sidebar communication, or public displays) provides information diffusion, which becomes “distributed” in the group. Due to the environmental factors, and to the actors’ availability (in terms of their workload which then affects the attention resources they can allocate to monitor activities in the room), we cannot be sure who is capturing the “diffused” information, and how it is interpreted and represented by the actors. In fact, the information distribution is opportunistic and cannot in any way be predicted. Not only do the team members monitor information specific to their team role, but also for information that informs them of the team progress, as the team is under time pressure.

Emergence and self-organization: Emergence is one of the most important properties of complex systems. Intuitively, a property is emergent when it cannot be anticipated from knowing how the components of the system function. By its multiple local interactions, the system can behave along some global features (emergent), which allow it to evolve towards more effective modes of organization (self-organization) without calling upon exterior or interior structuring operations (Pavard and Dugdale, 2001). For our case, teamwork shifts between individual, subgroup, and entire group work. It is often unpredictable in real-time collaboration as to how a team should divide itself to solve an emerging problem. Thus Team M is capable of self-organization. Sidebar conversations emerge and actors’ behaviors evolve over time so that they group and regroup to solve the problem-at-hand.

4 Main concepts being modeled

The main objectives of this computational modeling approach are to understand, answer, and solve different questions and problems concerning extreme collaboration. The first and immediate questions aim at understanding sidebars and modeling error detections: How do sidebars enable engineers to catch errors? How does the level of noise and the spatial location of the engineers affect detection of errors? More broadly, we are interested in how an agent-based simulator can help design new collaboration technology solutions. We are building the simulator with the purpose of simulating two “cooperation patterns” important in extreme collaboration, and in cooperation in general: *sidebar monitoring* and *error recovery*. These emergent cooperation patterns are considered to affect the high quality and rapid results of the Team M performance.

4.1 Sidebar conversation: Unpredictable and dynamic process

A *sidebar* is a dynamic subgroup. It is a set of two or more participants joining together to solve an emerging problem. These participants are not necessarily in the

Table 1 Errors: Sources and catching process

Who detects an error	How errors are detected	Why errors occur
Engineer	monitoring verbal conversations	making wrong assumptions
Leader	discussing in a sidebar viewing a public display using software systems	software use (e.g. not updating values)
		not talking to the right person stress and fatigue

same place in the room so their conversation can be heard by everyone in the room. The cocktail party phenomenon[2] can occur in the noisy environment. Anyone in the design process can initiate a sidebar and new participants can always join. Consequently, a sidebar can be characterized by its location in the room, its topic of conversation, its duration, its size, and its members. During our simulation, studying the influence of such parameters and others (actors, activities, interdependencies) can give us an in-depth understanding of: (1) sidebar formation based on monitoring information content, (2) the influence of room location, on sidebar formation, and duration, and (3) the sidebar as a pattern of communication (dynamic sub-network) in the group.

4.2 Error recovery

From the field observations, different sources of errors were identified. First, errors can be discovered from the public displays (e.g. showing a spreadsheet of various parameter values, the orbit visualization, or the spaceship configuration). One or more team members may detect a wrong value or a wrong configuration. For example, when the spacecraft orbit goes through an eclipse, solar cells cannot be used. Second, any engineer can discover an error, e.g. a wrong assumption or result, and announce it to the entire team. Third, the leader can detect errors, as he maintains an overview of the design process. For example, he can discover an error through noting inconsistencies of results on people's monitors. Table 1 summarizes who detects errors and how, and their sources.

Errors are characterized by their type, the time they occur, and the engineering subsystem source. A simulation using the seating configuration and known group interdependencies enables us to compute several measures of the error recovery process. We can compute *when and how errors are caught*: by whom, and from which information source (e.g. a sidebar). As a first step, we simulated sidebar conversations and error catching in extreme collaboration by focusing on the activities of the engineers, the sidebars, and errors. We applied a bottom-up approach to model our teamwork by extracting real collaboration scenarios from field observations to simulate sidebars.

[2] The cocktail party phenomenon refers to when people hear and respond to their names from a conversation in a noisy environment, such as a cocktail party (Cherry, 1953). We use the term cocktail party to refer to when the engineers would respond to keywords associated with their subsystem design in the noisy warroom (see Mark, 2002).

5 The agent-based simulator

We have designed a computer simulator, which is a virtual environment, with the following properties: (1) it can take into account the actors, their activities and interdependencies as well as the environment in which they are collaborating, (2) it will be able to mimic the observed team collaborating, (3) it can support experimentation with new, "hypothetical" scenarios, and (4) as a generic tool, it can offer reusable modules, which may be re-used by other organizations and collaborative situations.

In this section we show the model overview, and then we explain modeling details, simulator processes and parameters. We then describe the developed simulator, and finally discuss how we validated this model-simulator to get it ready for virtual experiments detailed in the following section.

5.1 Model overview

We are using an Object-Oriented approach to design and develop our model and simulator in an iterative way. The main components of our model are the environment, sidebar, error and team, where the team is composed of its members. The main classes of our object class model are member, environment, sidebar, and error. Figure 2 shows an overview of our model.

5.2 Modeling details

We describe here in detail the main assumptions we made and the relevant features considered for modeling each of the above classes. We detail then the environment and its evolving level of noise, the team, the members and their concurrent behaviors, the sidebars as well as the errors and their related evolving processes.

5.2.1 Modeling environment

The *environment* class includes mainly the level of noise in the room. It is intentionally separated to support an independent analysis of the environment factors, a description of the communication artifacts used by the group (e.g. location of the public display

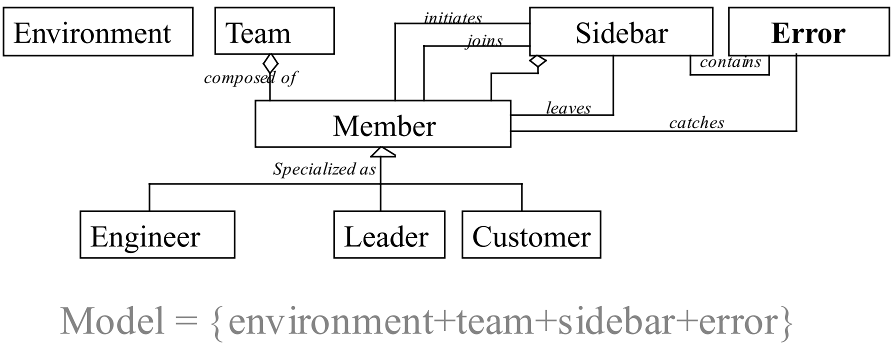


Fig. 2 An overview of the model

and its content) and to enable future additions of new factors (e.g. new collaboration artifacts).

From the data analysis of the field observations, we found an average of 100 sidebars to occur during each three-hour design session, ranging from 20 seconds to 20 minutes of participation for each member. In this warroom, it has been observed that in a normal conversational tone, a conversation can be monitored by all the team members. Also, it had been observed that the room can become quite noisy; engineers reported that this hinders them from effective monitoring.

Consequently, in our environment class design, we consider two constant attributes (room dimensions: width, height) and a variable attribute (level of noise). The dimension attributes represent the physical characteristics of the collaboration space (room(s)). They are set at the beginning of the simulation and are maintained until it ends: they are constant for a simulation and they are used as the reference for describing the relative room arrangement and to describe the actors and sidebar location within the room at each step. For our collocated synchronous cooperative work, this environment class is instantiated once reporting the single-room characteristics. In the case of physically distributed collaboration, different instances are needed where each would describe a different room.

5.2.2 Modeling the level of noise

Based on the fieldwork, the number of simultaneous sidebar conversations during a session varies from zero to five. Figure 3 shows the number of simultaneous sidebars and their relative duration per session, obtained from three observed sessions (which show consistent data). In all cases, we notice that we have a decreasing function: as soon as the number of simultaneous sidebars increases, the corresponding percentage of time spent in sidebars decreases. The averaged ratios are computed and represented in the pie chart in Fig. 4.

From the chart in Fig. 4, we notice that during about 30% of the time there are no sidebar conversations, and during this time, each engineer is working at his/her

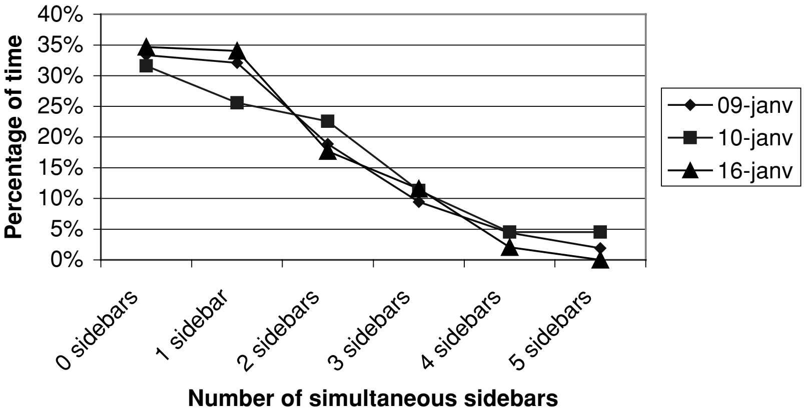
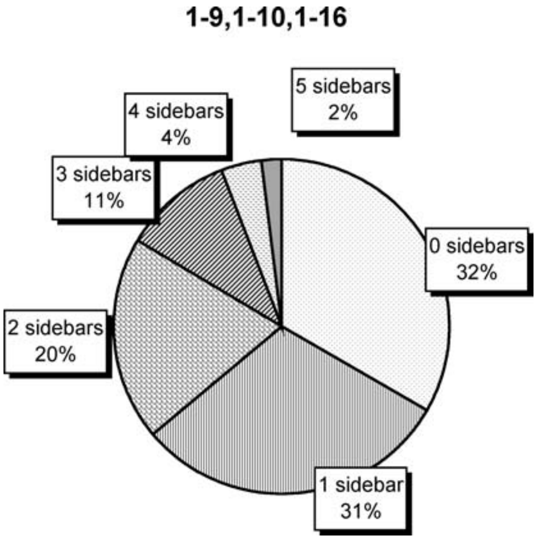


Fig. 3 The number of simultaneous sidebars and their relative duration per session

Fig. 4 Average proportion of simultaneous sidebars during the entire session



own workstation performing calculations or seeking information. Also, about 30% of the time there is only one sidebar. This unique sidebar may sometimes be the entire group in a conversation, or more often, it is a subset of team members trying to solve an emerging problem together or provide an explanation of results that have been exchanged. For our simulator design, we consider that a unique sidebar can be easily monitored by team members outside of the sidebar. We assume in this case that the *level of noise is low*.

During an average of 20% of the session time, two simultaneous sidebars take place and only 11% of the time, three simultaneous sidebars occur. In both of these cases, members who are not involved in sidebars can still monitor this environment, and we assume it can be monitored from everywhere in the room. We assume in this case that the *level of noise is medium*.

Only 6% of the time the group is split into four or five sidebars. In this context, the environment becomes noisy. Monitoring all the conversations becomes difficult for an engineer at his workstation who is not in a sidebar. We assume his ability to overhear is reduced such that he can primarily only hear messages from neighbors at the next workstation.

We modeled the level of noise in the environment as a function of the number of simultaneous sidebars in this room. The level of noise is updated at each simulation step (an *updateLevelOfNoise* method is provided in the *environment* class, and a repeating schedule calls this method at each step).

5.2.3 Modeling the sidebar conversation

As shown in Fig. 4, about 100 sidebars occur during a session and there is at least one sidebar during 63% of a session. We model sidebars as a source of information and a source of noise. Each sidebar is characterized by its *initiator*: the team member who needs to solve a problem, or needs an explanation or data to provide input for his subsystem. The initiator would approach the appropriate person in the warroom and initiate this sidebar. It is also characterized by its *beginning and end time*. Each sidebar is considered as an event occurring from *beginTime* to *endTime* attributes. Last, it has a *location* in the room. For our model we assumed that all sidebar members would physically move to the same location in the room to be close to other members. At *beginTime*, the sidebar location is the partner's location. As all engineers are continuously monitoring the environment, according to the content of what they are overhearing, they may decide to join the sidebar or not. Consequently, for each sidebar, we built the list of all its members and the time when they joined it. At each moment, the sidebar size is equal to this sidebar member list size. When the sidebar grows in size to more than five people, the sidebar moves to a larger empty place in the warroom. From observations, this space is either at the front or back of the meeting room.

5.2.4 Modeling the team and members

The *team* class refers to the entire Team M group. It includes its size and the list of members. In our model, the *member* class refers to any human actor in the group. It is characterized by the *name*, the *role* (e.g. leader, power engineer, thermal engineer), and *seating position* (workstation location).

A team member's behavior may be summarized by the diagram of Fig. 5. By default, all engineers are computing and at the same time monitoring their environment (default activity, status = 0). At the begin time of a sidebar, the initiator's status changes from *computing* to *inSidebar* (status = 1). For each actor, according to his ability to hear and the relevance of the information, he would decide to join the sidebar or not. An engineer may join a sidebar when he detects an error. All sidebar joiners remain in it until end time, when they leave it (status = 2), they fall into a state of resuming (status = 3) before working individually again (status = 0). We allow 2 steps for the engineers to resume their workstation calculations after leaving a sidebar.

Behavior of agents while involved in sidebars: For purposes of simplification, our agents are kept in each sidebar until it ends. We assume that an *"inSidebar"* agent would catch and process only the messages of the sidebar to which he belongs. This implies that all his attentional resources are fully involved in the sidebar and that he has greater probability of finding errors in a sidebar, when an error exists within the sidebar.

5.2.5 Modeling communication and messages

From the field observations and their analyses, there are two sources of information: (1) verbal communication is when actors interact or respond upon hearing relevant keywords, and (2) communication of data is mediated by the public displays and public spreadsheets.

The representation of the verbal communication message is an abstraction of the content. Messages are rather abstractions of packets of information that are generated and sent by one actor and received and processed by another. When broadcast in the

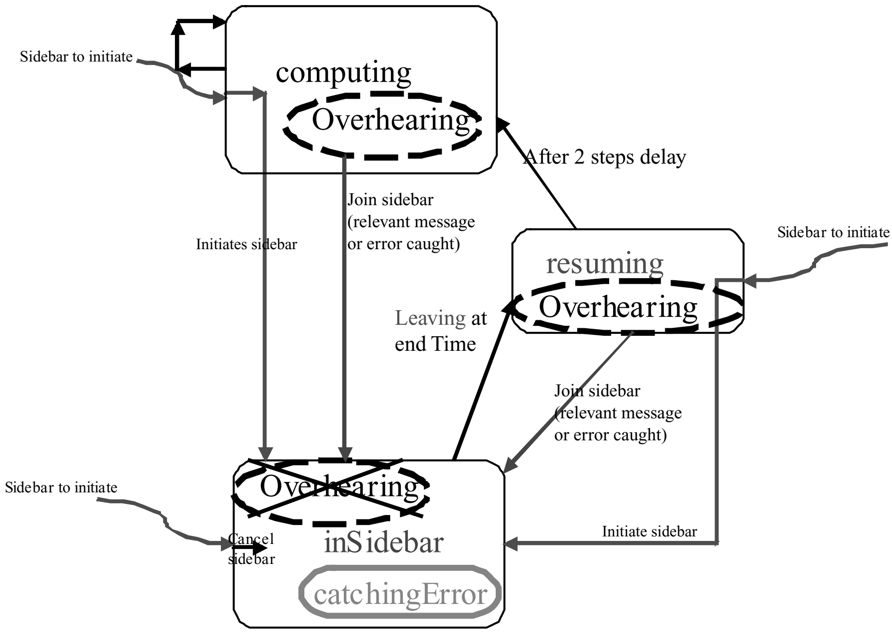


Fig. 5 Diagram showing a typical team member's behavior during a session

warroom, they may contain errors, and may be considered as relevant or not for those hearing/receiving such messages.

5.2.6 Modeling the ability to hear

According to the distance between actors and the environment factors (mainly the level of noise) some messages can be lost. We model the hearing ability and assume that on sending a message from an actor A to an actor B, the sent message will be added to B's *heardList* only if B can hear; where this ability to hear function is related to distance and the level of noise as follows: if the level of noise is low or medium, messages can be heard by all team members; if the level of noise is high, only neighbors can hear. A and B are considered neighbors if they are at neighboring workstations.

5.2.7 Modeling sidebar relevance

On overhearing one or many sidebars (stored in his heardList), an engineer processes this data and decides whether to join a sidebar or not. The same sidebar output may be relevant for some members but not others. Many strategies may be applied for deciding about the relevance of what is heard. In our model we experimented with two "information relevance and sidebar-joining strategies":

– Strategy one is when one receiver considers a heard message relevant for him if it comes from a sidebar where at least *the initiator or his sidebar partner* are included in his interdependency map. In other words, this assumes that the sidebar topic concerns an engineering subsystem that the actor is interdependent with.

– Strategy two is when one receiver considers a heard message relevant for him if it comes from a sidebar where at least *one of the sidebar members* is interdependent with him in terms of exchanging information. In other words, one of the engineers in this sidebar is interdependent with the listener.

For both of these strategies, we can predict that a member with more work interdependencies than others will have higher chances to join a sidebar. This has been confirmed by simulation. We predict the second strategy would generate bigger sidebars (in terms of size) than the first one. These two strategies have been chosen since we think that they are closest to those in the real design work, since the field study emphasized the importance of interdependencies and the engineer's selective behavior.

5.2.8 Modeling errors

In our simulator, we will generate errors randomly while sending messages. We model also an *error* class as a separate object class. This allows us to keep track of each error separately and to easily associate group members with errors. Each error is characterized by: its *beginning time* (when it occurs), *sidebarIdErr* (within which sidebar the error occurs), a boolean variable *caught* (*set to true when the error is detected*), *end time* (when the error is caught), and *errorDetectorsList* (storing the name and status of agents detecting this error).

5.2.9 Modeling error-catching

Since errors are modeled as being a sidebar attribute which is set to false by default and set to true if the sidebar outgoing message is erroneous, error catching may be performed by all those members who are in the sidebar (source of this message) and those who are not involved in any other sidebar. For each of these categories of potential detectors, an appropriate catching error probability is assigned. Members involved in sidebars have high probabilities of catching errors, as they are focusing on the sidebar. This probability is referred to as *catchErrorProbability*. Members outside sidebars catch errors with lower probability as their attentional resources are distributed in monitoring the whole environment (i.e. in multiple sidebars) and they are focusing on their own tasks (mainly computing). In our model we consider a *catchErrorProbability* and an *attentionResourcefactor* (e.g. 0.2 or 0.4), and compute the *inSidebarCatch ErrorProb* as the product of the two previous ones. *InSidebarCatchErrorProb* is thus diminished by reduced attention resources.

5.3 Simulator processes: Sidebars and errors

The occurrence of sidebar events, encapsulating potential errors, will generate dynamic group interaction. As soon as a sidebar is initiated, one or many members in the group will react by hearing, answering, or moving. As the simulation progresses, communication between actors and movement across the room are shown.

The sidebar processes are: (1) *Initiating a sidebar*: The initiator calls for a sidebar at begin time if an agent I is the initiator, and if he is in his seating position/location, this agent moves to one partner P, (2) *Finding a partner*: A partner is chosen randomly from the initiator's interdependency map among the connected members who are not already involved in other sidebars. After moving to the partner, they together begin a sidebar which will be broadcast to the warroom, (3) *Reminding of a sidebar*: During the whole sidebar duration a new sidebar message is broadcast: (*sidebarFrequencyBroadcast,* e.g. 3 or 6 steps), and (4) *Ending/stopping a sidebar*: At end time, all the sidebar members are set to the state "leaving" and set back to their seating places and the sidebar is over.

To study errors, we consider errors as being generated during sidebar conversation broadcasting. We consider the probability of error occurrence. When a sidebar message is broadcast by one sidebar member at a certain time, this message may be erroneous or not. This information is "propagated" and on hearing a message, any member may detect an error and join the sidebar where this error has been broadcast from, according to the error-catching probability (defined in the previous section).

5.4 Simulator parameters: Input and output

Simulator input. At the beginning of a simulation run, three input files are setup:

1. A group properties file including, for each member of the group: name, *role*, *seatXPos*, *seatYPos*. According to the seating position of every member, the neighbor of everyone is set. When seating position values are out of the range of the room size, they refer to a spatially distributed team.
2. A sidebar file including a series of "events" (sidebars occurrences) characterized each by: *beginning time*, *end time*, *initiator.* According to the BeginTime, one or more sidebars may be run on the same time. This enables us to test several scenarios with one or more sidebars.
3. A map of interdependencies, where for each subsystem representative S, are listed all the subsystems T with whom S has a functional interdependency, i.e. all those from which an input may be needed for S to perform its calculations. This interdependency map is based on Fig. 1. Another representation of this same interdependencies map as a matrix has been used for analysis work (see Appendix 1).

Environment attributes (room size and public display positions and contents) are set to default values. Size of the group is set to 18 as default value (which is the real size of Team M). In all cases, both of the input files (sidebar and members' descriptions) have to be filled out with the adequate values corresponding to the scenario to be simulated.

Simulator output. The simulator provides an overview plan visualizing continuous communication, movement, and physical settings. Graphs trace over time the number of sidebars, the level of noise, and the activities of each actor. For each sidebar, its duration and size over time are computed. Additional representations of events and results may be added. Simulation parameters and their values are saved into files for further statistical analyses and comparisons.

At each simulation run, generated errors and their properties are stored in output files: error number, its occurring time, the sidebar number within which it is generated, a true or false value corresponding to caught or not, the detection time and the detector agent with its status (mainly whether it is inSidebar or computing). These output parameters, correlated to input ones, would be subject for further analysis to examine how environment and group properties affect error recovery.

5.5 The simulator

Our simulator is developed using the SWARM[3] platform (developed by the Santa Fe Institute) and is written in the java language. According to our main objectives, we have developed a first model and implemented an agent-based simulator focusing on the sidebar conversations and error recovery, as detailed in previous paragraphs. Figure 6 shows our simulator interface.

In fact, the simulator, as shown in Fig. 6, provides various output representations. A control panel enables the user to run a simulation step by step or continuously, and to stop it. A display is given of the working room, showing the spatial arrangements of subsystems as well as the location and the status of cooperating agents. Magenta rectangles with the subsystem names show the assigned positions of agents where they have their terminals; the ovals represent the engineers *computing* (white color), *initiating* a sidebar (pink), *involved* in a sidebar (cyan if newcomer, blue otherwise), *detecting* an error (yellow), *leaving* a sidebar (red), *resuming* individual work (green); and broadcast sidebar communications are represented by links (green by default or red when a message is erroneous) connecting message sender to receiver. Both a cumulative and instantaneous session view are displayed. Graphics show the level of noise and number of simultaneous sidebars over time. Also, each engineer's status on each step is plotted. Finally, after the end of the simulator, a synthetic representation of performed sidebars and generated/detected errors are shown. A textual trace of execution shows where various comments are written to report on the occurring simulation events and evolution. Added to these real-time on-simulation displayed output results, several output files are generated for further analysis by statistical tools (such as SPSS).

5.6 Validation of the model

Validation is a complex issue (Carley, 1996). General discussions of validity for computational models point to one or more of the following six types of validation: conceptual, internal, external, cross-model, data and security (Knepell and Arangno, 1993; Carley, 1996). Conceptual or theoretical validity refers to the adequacy of the underlying conceptual or theoretical model in characterizing the real world. Internal validity refers to whether the computer code is correct and the program is free of coding errors. External or operational validity is concerned with the adequacy and accuracy of the computational model in matching real world data. Carley states that validation is often a multi-year multi-person endeavor; relative complexity of computational models, statistical training and analysis expertise, massive amounts of simulated data, and then the amount of time needed to examine it, are the main arguments of why this is such a long and labor-intensive endeavor.

[3] www.swarm.org

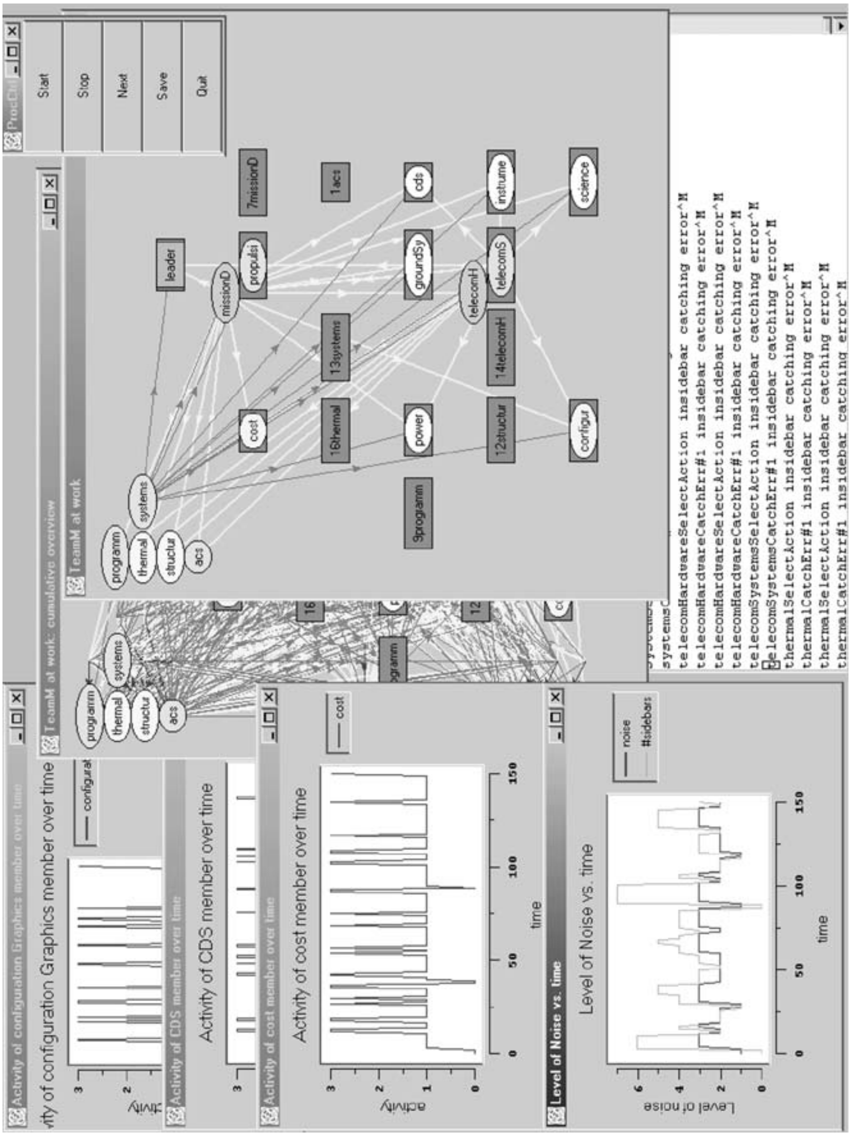


Fig. 6 The simulator interface

Table 2 Simulator parameters

Simulator input	Tested values
Input sidebar events files	Sidebar109, Sidear110, Sidebar116
Probability of error generation	0.0, 1.0, 0.8, 0.5, 0.2, 0.15, 0.1, 0.05, 0.01,
Error catching probability (in sidebar)	0.0, 1.0, 0.9, 0.8, 0.7, 0.5, 0.2, 0.1,
Attention resources	0.0, 0.7, 0.5, 0.4, 0.3,0.2
Sidebar broadcast frequency	3 steps or 6 steps

In our case, we focused on the internal and external validation of our simulator. Therefore, in order to check that the programs are free of coding errors, different combinations of parameter values have been given as input to the simulator. Real scenarios, corresponding to the series of sidebar events, extracted from fieldwork, have been used too. Series of simulations are executed with different combinations of parameters for each input file and for both of the joining strategies. Extreme values of parameters have been voluntarily tested to study the simulator behavior. Table 2 summarizes the set of simulator input as well as their values affected on running the simulation:

A first set of simulations has been conducted in order to focus on sidebars only; thus the triplet (*ProbError*, *ProbCatching*, *AttentionResources*) affected to (0, 0, 0) means that no errors are generated in the model and that the engineers join sidebars according to their relevance only. By varying the frequency of broadcasting (3 or 6) we verify that the evolution of the size of sidebars depends on applying a joining strategy. For the first strategy, where relevance is based on linkage of the hearer to the initiator or its partner, the size of sidebars evolves more rapidly with frequency 3 than 6. But in both cases, this size is bounded and does not reach 16 (the whole group).

For the second strategy, where relevance is considered according to all sidebar members and their inclusion in the hearer's interdependency map, we notice that the size of the sidebar evolved gradually after each broadcast message and as soon as the sidebar grows, the number of newcomers increases. So, for a frequency of 3, we noticed that the rate of growth is more rapid than in the case of 6. In the first case we get bigger sidebars for long duration, and with "isolated" sidebars (i.e. no other sidebars in the same time) we rapidly get the whole group involved. Consequently, if meanwhile there is a new sidebar event, it is cancelled since its initiator remains in the sidebar he lately joined until its end time. That is why in this case we notice that the number of effectively performed sidebars was less than the number of input events. Also, this number of sidebars is often bigger in the case of frequency 6 than 3.

A second series of simulations has been conducted in order to focus on error generation only (not catching) by our model; thus (*ProbCatching*, *AttentionResources*) affected to (0, 0) means that no errors are detected. By varying the *ProbError* (1, 0.5) and the frequency of broadcasting (3 or 6) we verify that the number of generated errors is related to the number of all broadcast messages (which is the sum of all reminding sidebar times, i.e. the sum of duration of all performed sidebars divided by the broadcasting frequency).

A third set of simulations is focused on detecting errors within the sidebars only (i.e $AttentionResources = 0$):

- (*ProbError*, *ProbCatching) = (1, 1)* shows that all generated errors are detected in the following step which is predicted since in the sidebar there is at least the initiator who would have full chances to find the error.
- (*ProbError*, *ProbCatching) = (1, 0.7)* shows a slight increase in delay detection (2 steps instead of 1 for some errors) and where all generated errors are detected.
- (*ProbError*, *ProbCatching) = (1, 0.5)* shows again an increase in detection delays for more errors (2 step-delay for 15/235 errors, 3 step-delay and 5 step-delay for one error in each case). With the strategy 2 and 3 steps frequency, one error out of 235 remains undetected.
- On decreasing the probability of catching (0.25, 0.1) we check that similar simulator behaviors are observed. Increases in the number of non-detected errors and in the detection delay occur.
- On changing the number of generated errors by decreasing the *ProbError*, and by varying *ProbCatching*, we find similar results to the above. Thus, error catching within sidebars is a process which is independent of the number of generated errors.

A fourth series of simulation runs focused on error detection both within sidebars and outside sidebars. In this case we focused more on varying the *AttentionResources* parameter. For high attention resource values (0.7), all errors are detected. Also the ratio of number of detected errors outside sidebars over the number of errors detected within sidebars (for the second strategy and for a frequency equal to 3) is close to 1, i.e. about an equal number of errors are caught by within and outside sidebar team members. On decreasing resource values (0.3), the previous ratio becomes 2, i.e. there are double the errors caught by within-sidebars than outside sidebars.

This error catching process corresponds adequately to what we expected. All these observations confirm that our simulator process gives adequate behavioral representation. In conclusion, from field observations, as the sidebar size is rarely more than 7, we have to choose parameters and select a strategy that gives reduced sidebar sizes. Strategy number 1 and a broadcasting frequency of 6 are retained. So we need to assign low values for error generation.

According to previous observations, with an error-catching probability of more than 0.5, in all simulated cases, about all errors are detected by within-sidebar members. Therefore, for experimental purposes and as we would like to assess the influence of group configuration (in terms of simultaneous sidebar number) on error detection, we need to assign values less than or equal to 0.5 to give chances to outside members to catch errors. Finally, for attentional resources, they should be close to 0.3.

6 Virtual experiments

Having a validated simulator, where conceptual, internal and external verifications have been checked, we then used it to run new virtual experiments and to discuss "what if" scenarios. We present here the performed experiments and the main results obtained by running these virtual experiments, which answer the following research questions: (1) how do sidebars grow and how do they relate to interdependency maps, and (2) how do sidebar conversations affect error detection and consequently group performance (in terms of errors detection)? For each issue, we explain first the design of the experiments and the scenarios we've built to investigate it. Then we give the results and our interpretation.

6.1 Relationship between sidebar evolution and interdependency maps

In order to answer the questions: how do sidebars grow and how do they relate to the interdependency map, we experiment here with sidebar growth and information diffusion as a function of "interdependency maps" (network structure). We run hypothetical scenarios looking for this.

From observations, according to the sidebar initiator and duration, the sidebar size increases more or less rapidly. In order to better understand the question of how sidebar conversations both grow and diffuse information to the whole working team, several experiments have been performed to study the evolution of sidebar size as a function of their initiators.

6.1.1 Virtual experiments design

For this experiment and for each engineer, a series of 150 sidebar independent events, of a twelve-step-duration each, has been given as an input file. These input events were independent in the following sense. At each moment, only one sidebar occurred so that all potential members can join it, i.e. no one is busy, involved in a different sidebar, or unreachable, since the noise level is low. Simulations have been performed for each of these sixteen input files.

6.1.2 Results and interpretations

The mean size of sidebars have been computed for each initiator and is displayed in Fig. 7. The joining sidebar strategy 1 (where joining a sidebar is based only on knowing the sidebar initiator or his partner, computed randomly at each initiation among the initiator's interdependency map) has been used for Fig. 7.

From these series of experiments, we note that there is no direct or linear relation between the number of incoming connections and the resulting sidebar size. Also, on displaying the same results and grouping them according to the outgoing connections of each engineer (see Fig. 8), we discovered again a disparity between values, and no direct relation between the number of connections and sidebar growth. It is rather a more complicated process, where the number of connections is less important than to which subsystem an agent is connected.

In fact, in order to investigate the importance of which subsystem an agent is connected to, we studied the sidebar compositions for each initiator by focusing on the members and the time when they join the sidebars. For the set of initiators {cds, missionDesign, science, structure, TelecomHardware} having one incoming link each, we notice that the sidebar sizes vary according to with who an initiator is connected and it appears to be related to who the partner is. Similarly, by analysing all other sets of initiators having equal incoming links, we find the same conclusion: the sidebar size varies according with who an initiator is connected and who the partner is.

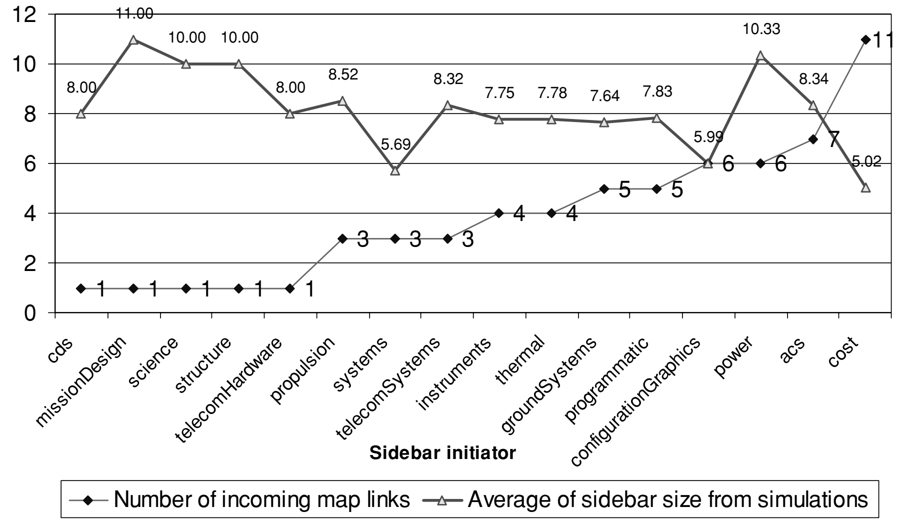


Fig. 7 Mean size of sidebars and number of incoming links for each initiator

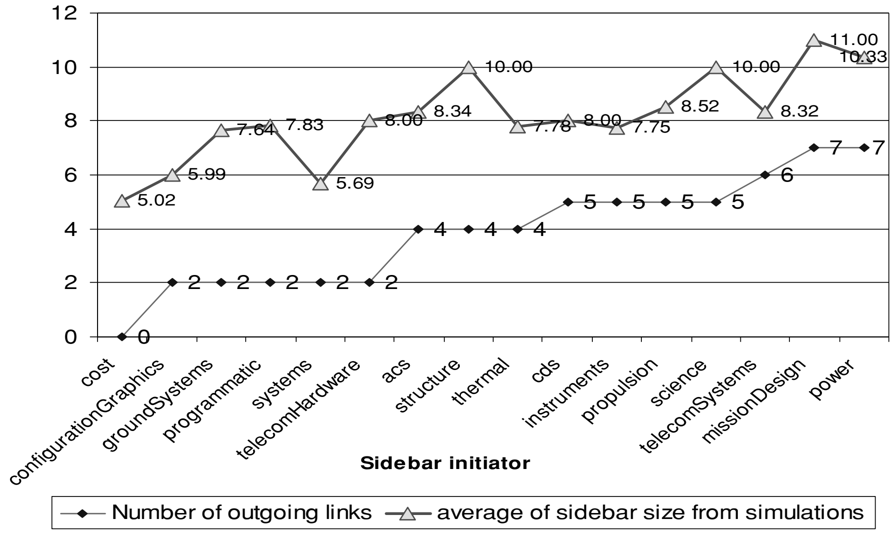


Fig. 8 Mean size of sidebars and number of outgoing connections of each initiator

For the first set of initiators, having only one potential partner, we find that a part of sidebar joiners are all those who have the initiator in their own maps and the remaining part are those who have the partner (the first agent reached by the initiator, with whom the sidebar begins) in their own maps. We notice also, Fig. 9, that the number of sidebars is directly related to the number of the incoming links to the set (initiator, partner).

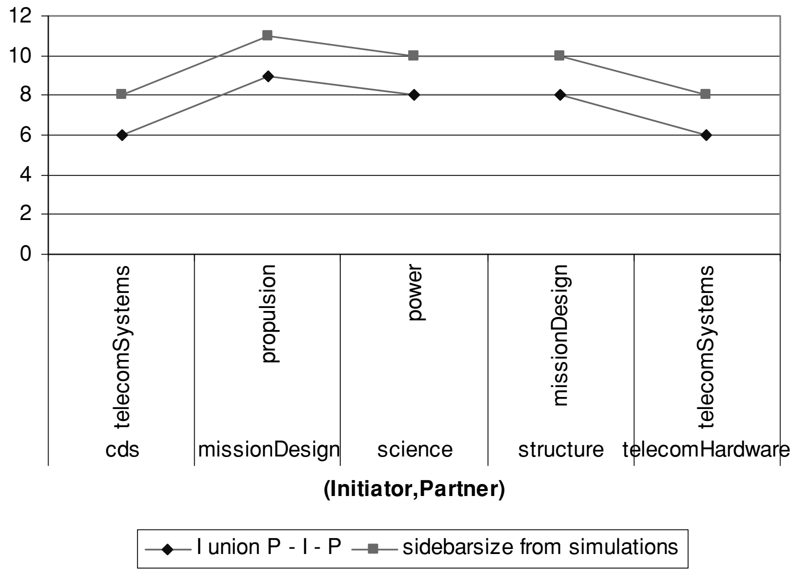


Fig. 9 Sidebar size of initiators having one partner and number of the incoming links to (initiator, partner)

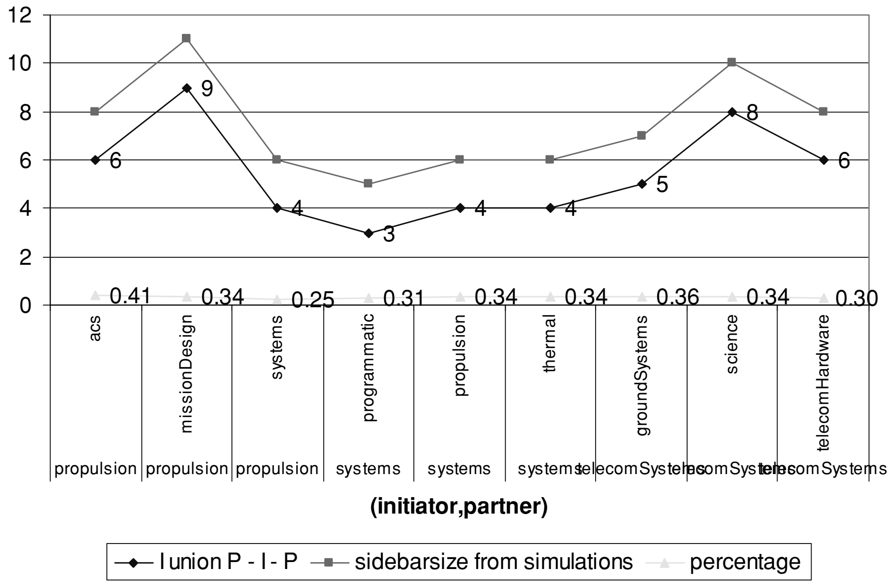


Fig. 10 Second example showing relation of the size of sidebars to the number of the incoming links to the set (initiator, partner)

In another example of a set of initiators {propulsion, systems, Telecomsystems}, having each three potential partners, we find also that the number of sidebars is directly related to the number of the incoming links to the set (initiator, partner) (see Fig. 10).

For all other couples, similar analyses have been made and the same conclusion was found. Consequently, we iterate our analysis by considering not only the initiator,

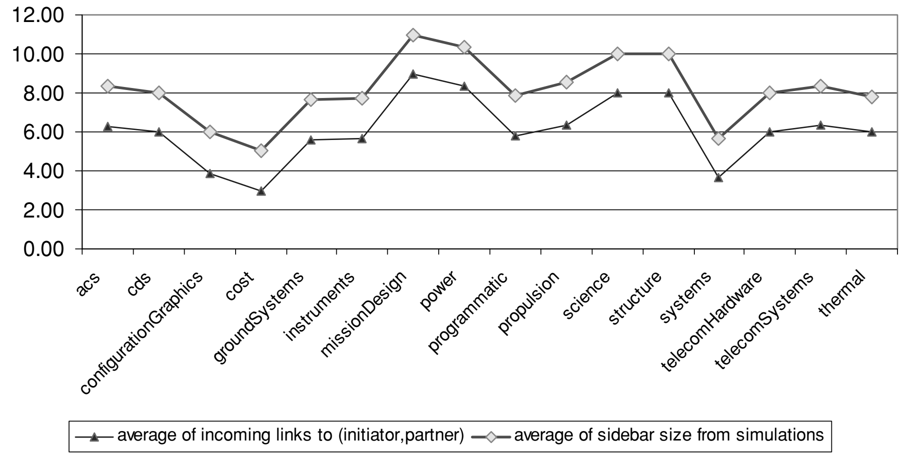


Fig. 11 Correlation between computed average of incoming links and the simulated sidebar sizes for each agent

but also the couple (initiator, partner). For each sidebar type, characterized now by (initiator, partner), we computed from the individual maps the set of all potential joiners Sjoin: Sjoin = Sinit UNION Spart, where

– Sinit = the set of all the agents, other than the partner, having the initiator in their map
– Spart = the set of all the agents, other than the initiator, having the partner in their map
– We use the UNION set operator to count agents only once.

After computing these Sjoin sets, for the 62 cases (for all possible couples initiator, partner deduced from crossing the interdependency maps), we calculate for each initiator the average size of its relative sidebars joiners. As shown on Fig. 11, we found a good correlation between this computed average and the simulated sidebar sizes for each agent.

In Fig. 12, we show the same data, but after sorting them according to this averaged joiners set size.

Finally, we conclude that the sidebar size and growth is related to the first initiating couple (initiator, joiner), and for the initiator, what is more important than the map degree is the agent with whom one is connected.

As a last conclusion in this section, we notice that even though the Cost engineer needs input from a great number of subsystems (11), he is involved very often in many sidebars, but cost sidebars are relatively small ones, since the number of joiners is reduced. So in order to get more engineers participating in sidebars involving cost, he needs an additional joining process such as a calling method by the leader who would invite/call explicitly needed members to join.

In conclusion, from these experiments and by crossing results with interdependency maps analysis we find the following. First, there is no direct relation between the number of (incoming/outgoing) connections of initiator and sidebar growth. Second,

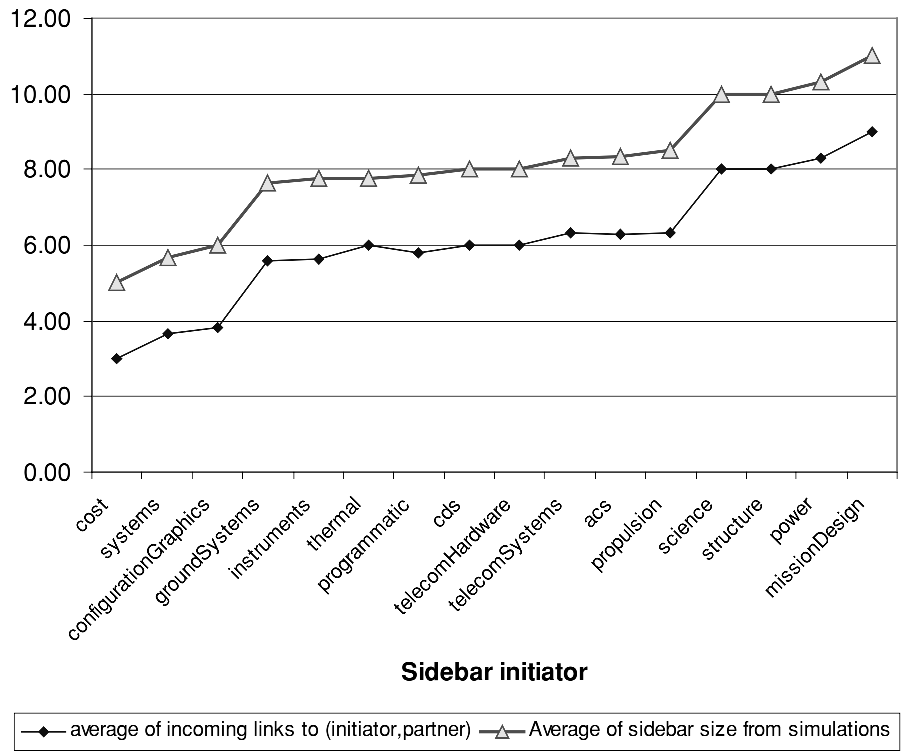


Fig. 12 Correlation between computed average of incoming links and the simulated sidebar sizes for each agent, sorted according to averaged joiners set size

which subsystem an agent is connected to is more important than the number. It reveals that it is the key for sidebar size growth since we found out a direct relation between the number of links coming to the initiating couple (initiator, partner) and the sidebar size. Last, some subsystems need an additional joining process (e.g. for joining cost sidebars).

6.2 Number of simultaneous sidebars and error detection

The main issue investigated in this section is the relationship between sidebars and error detection. In fact, in our model sidebars have two concurrent and opposite effects on errors. Sidebars are a source of information for the whole team enabling one or many members to contribute in detecting errors. At the same time, sidebars may produce noise which may hinder error detection. The question then becomes, how can sidebars affect error detection?

6.2.1 Designing simulation experiments

We had to build hypothetical input files in order to simulate noisy sessions. Since our collaboration process is dynamic and non-deterministic, building such files was not as simple as we thought. It was not enough to generate a random series of events with overlapping durations (i.e. when the two sidebars overlap). Sidebar growth and evolution cannot be predicted a priori. Even by scheduling overlapping sidebars, we cannot predict their actual concurrent occurrence since we cannot ensure that the next sidebar's initiator is not already involved in a previous one (where the new planned sidebar would never occur). Consequently, the construction of hypothetical input files was not systematic.

Faced with such an experimental design, we generated hypothetical sidebar input files, ran simulations with them, and then made an a posteriori evaluation of the number of simultaneous sidebars over time and analyzed the effective level of noise for such files. The selection of input files for experiments was then done according to the percentages of levels of noise within a session duration. (The duration of the experiments is fixed to the same duration as real sessions.)

Figure 13 shows a sample of six experiments we selected; where each line corresponds to an experiment: the X-axis corresponds to the level of noise 1 (i.e. there is at most one sidebar), 2 (i.e. the number of simultaneous sidebars is 2 or 3), or 3 (i.e. there are at least 4 sidebars simultaneously). The Y-axis corresponds to the percentage of the corresponding level of noise.

The same data is displayed in Fig. 14 focusing on the level of noise. Figure 14 shows clearly that the experiments E5 and E6 are quiet ones (level one during most session duration and never level three). E1, E2 and E3 are examples of noisy sessions. We observed a number of simultaneous sidebars achieving 8: 25 to 50% of the session duration has a high level of noise, and only 20% of the time the level of noise is low.

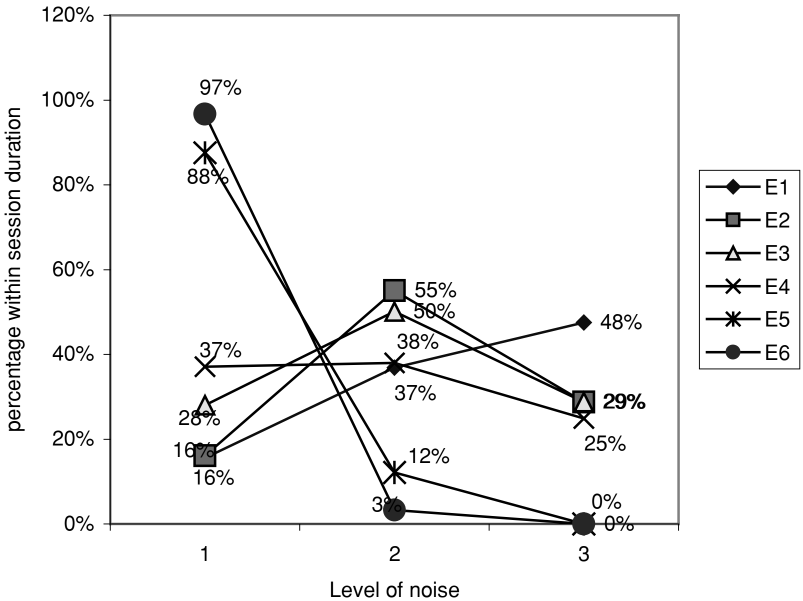


Fig. 13 A sample of six experiments; X-axis shows the level of noise and the Y-axis shows the percentage of the corresponding level of noise

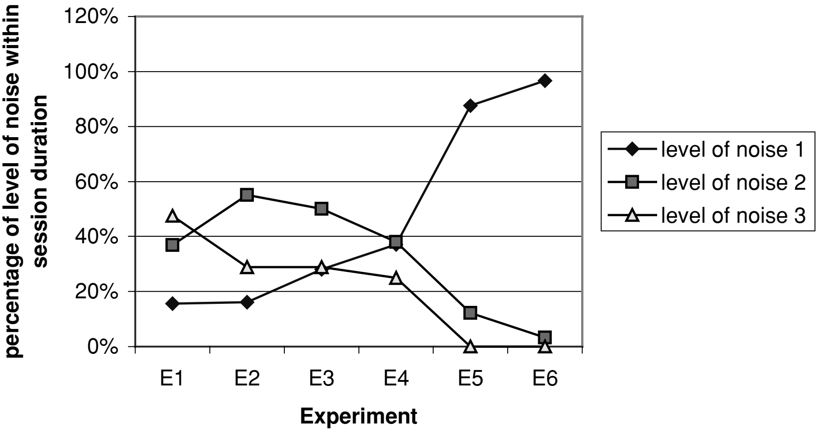


Fig. 14 Percentage of each level of noise per experiment

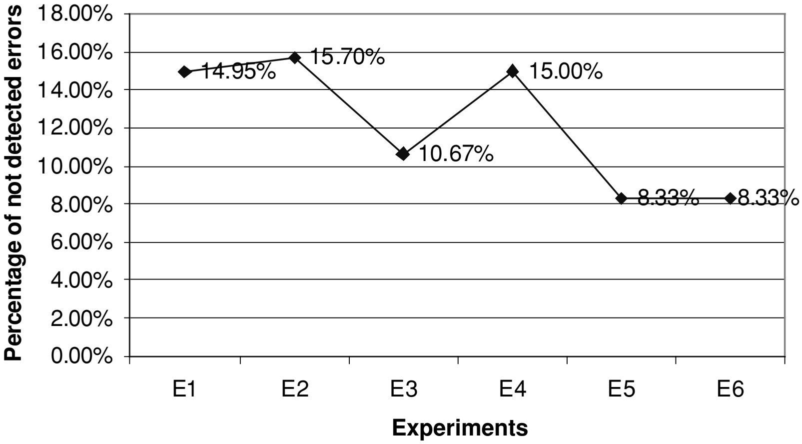


Fig. 15 Simulator results showing that errors remain in the system

E4 is an example of a medium level session: both Figs. 13 and 14 show that for about a third of the duration, it is quiet (37%), a third has a medium level of noise (37%), and only 25% of the duration has a high level of noise.

6.2.2 Results and interpretation

On running the experiments, we noticed that some errors remain undetected during the simulated session. As shown in Fig. 15, even during "quiet" sessions, a few errors may remain in the system. This is quite true when we remember that from the field observations, error detection also occurred by watching the public-display, or results from the publish-subscribe system.

Our experiments show also that during noisy sessions, more errors remain undetected. The obtained values shown in Fig. 15 show that noise decreases the group performance in terms of error detection. At the same time, the value of 15% is not high. So why in our model, do errors remain undetected? Also, why are most of the errors caught at all?

A simulation run of the "noisy" cases shows the number of sidebars varying from 4 to 8 (corresponding to level of noise 3). Therefore, as our team size is 16, we have varying sidebar sizes from 2 to 4, which is relatively small. During these time lapses, about all members are involved in sidebars. So, they cannot join new sidebars before the end time of sidebars which they currently are involved in. Consequently, error detection is potentially done by within-sidebar members only. We have then a reduction in both chances of detecting: exterior members are not available, and sidebars are small ones. Furthermore, the members who are not involved in any sidebar can handle only their closest neighbor's messages and monitor a limited area in the room since their ability to hear is then reduced when the environment is noisy.

6.3 Summary of experimental results

To summarize, our first simulations show several things. First, there is no direct or linear relation between the number of incoming/outgoing connections and sidebar growth. It is rather a more complicated process, where not only the number of connections is important but rather "to whom/to which subsystem" an agent is connected to is more important. Second, the joining sidebar strategy influences the sidebar growth, composition, delay and duration. Third, according to subsystems, additional joining processes (e.g. explicit calling by the leader) are needed to get closer to real sidebars. Fourth, building hypothetical input files in order to simulate quiet/medium/noisy sessions was not as simple as we thought since sidebar growth and evolution, mainly in the case of having concurrent ones, cannot be predicted a priori. We also cannot ensure that the next sidebar's initiator is not already involved in a previous one where the new planned sidebar would be canceled. Fifth, even during "quiet" sessions, a few errors may remain in the system. Last, during noisy sessions, more errors remain undetected. Noise decreases the group performance of error detection.

7 Discussion

In this study, we applied complexity theory to understand a form of synchronous collaboration that we call extreme collaboration. To the untrained observer, extreme collaboration seems chaotic. Yet through this process, a team of sixteen engineers can produce highly complex space mission designs. We simulated the process to better understand how sidebar communication relates to the level of noise and error detection. We now discuss how this model-simulator can be used in the CSCW field, how it relates to complexity theory, the limitations of our model-simulator and data, and outline our perspectives.

7.1 Helping CSCW design choices

One of the goals of our simulation was to construct scenarios of collaboration that could assist collaborative technology designers. Our experiments showed that what is important is not the number of connections among people, but rather to which subsystem an agent is connected. Our experiments showed clearly that as the number of sidebars increases in the room, we can expect the noise level to increase such that it becomes more difficult to monitor other sidebars. The noise in the room competes with one's limited information processing. This result suggests technology support that can aid people in such a war room environment to identify sidebars that have a high probability of likely relevance for their own subsystem work. One example is a monitoring device that can inform engineers when a grouping of relevant people occurs.

Further, our experiments showed that in both quiet and very noisy sessions, errors can remain undetected. In the first case, it may be because information is not verbalized; in the latter case, it is because the noise in the room interferes with detection. This suggests that the facilitator of the team could benefit the process by controlling the group configuration. The facilitator should encourage that information be verbalized by inviting engineers to join a sidebar when appropriate. When the number of sidebars grows too big, then the facilitator should work to maintain an optimal number by using strategies such as combining sidebars or helping to negotiate decisions that can lead to quick closure.

7.2 Applying complexity theory to extreme collaboration

Yaneer Bar-Yam compiled a list of thirteen "significant points" in the study of complex systems (Bar-Yam, 2000). By studying extreme collaboration, modeling and simulating its main features we tackled the major significant points as follows:

(1) *Multi-scale description:* on observing Team M and modeling its face-to-face mission design process we focused both on getting information concerning groups (the whole team and emerging sidebars) and individuals (engineers, mission subsystems and their specific functional dependencies and fine behaviors).
(2) *Fine-scale influences large-scale behavior:* in our model we simulated and observed how individual agents interacting together produce sidebar dynamics. According to his/her status, internal interdependency maps and the environment level of noise, an agent may join a sidebar or not. It has been observed that sidebar formation and composition is far from being a linear or predictable process.
(3) *Pattern formation:* in this study, our main patterns are sidebar conversations.
(4) *Complexity:* a great deal of data has been collected from the field site; several intentional and realistic assumptions have been made to build a first simple model. In fact, reducing the complexity of the developed simulator enables us to identify the necessary, minimal and bounded set of information describing our system (e.g. the complex interdependency map shown on fig. 1).
(5) *Behavioral response complexity:* in our model, we managed this complexity by focusing on the most important factor, modeling the varying level of noise. Also, we introduced a response function which is common for all engineers related to their relative locations within the room. Therefore, from our experience we conclude that these simplifying assumptions are fundamental to understand and assess agents' actions and behaviors within this environment.

(6) *Emergence:* This is related to the dependence of the whole on parts, the interdependence of parts, and the specialization of parts (Bar-Yam, 2000). In our case study, we have considered these main key concepts: e.g. we have considered the specialization of each engineer, his/her interdependence with the remaining team members and the dependence of the whole team work in a warroom on each subsystem and on the proposal production process in general. Through simulation we could experiment with "what if" scenarios and study how changes in one part affect the others.

(7) *Evolution* is related to the dynamic of change of the collection of complex systems that are present as a part of a larger system. According to the field observations, the team M members actively monitor the environment and filter their information by focusing on sidebars that might be most relevant for them by understanding their interdependencies in the team. In their own words, they claim that they have "maps" of their own interdependencies. Consequently, it would be worthwhile for future models of similar realtime collaborative teams to consider the evolution and internalization process of relevant and bounded interdependencies among cooperating actors. Also, norm emergence, i.e. the implicit working rules and commitments between roles, within such groups would be of great interest to understand the benefits of extreme collaboration.

(8) *The relationship of descriptions and systems:* on modeling our system, assumed to be the design process of a collocated collaborating team, we have based our descriptions on field observations and made assumptions adequate to the real system. The developed simulator enables the recognition of the initial system as well as the relevant real system properties encoded in the model (e.g. sidebar phenomena, agent's behaviors, spatial distribution within the environment, etc.).

(9) *Composites:* "to form a new complex system take parts of other complex systems and recombine them. For this work, parts must be partially independent" (Bar-Yam, 2000). In our case, we have modeled almost independently the agent's behaviors, the sidebar process and the error process, and then observed the evolution of the whole complex system.

(10) *Agent-based modeling and simulation* has been our central bottom-up approach for this case study. It is an efficient way to focus on the environment, actors, specific characteristics and interactions. The simulation platforms, such as SWARM, helped in building the discrete-event simulator and provided probes and analysis tools of micro as well as macro behavior of this developed artificial society of agents.

For this study, we are not concerned with the remaining three points, since:

(11) *Multiple (meta) stable states:* in our case and within our study's frame we have not identified stable states.

(12) *Selection is information (a la Shannon theory):* we have not applied any selection—replication—competition mechanisms

(13) *Control hierarchy:* in this first model, we still have not included the leader who would be the "central controller and single component controlling the collective behavior".

7.3 Limitations of the data, of the model and of the experiments

We have two basic types of limitations in our study, associated with the data and the model.

7.3.1 Data limitations

In this study we were not allowed to record the team in action. In general, fieldwork that is used for developing simulation models should be as precise and complete as possible. It should include recorded data that enables the timing of behavior as well as detailed descriptions of the team members. For this purpose, video recording is the most complete and useful data for collecting precise quantitative results. However, other data such as observations, interviews and field notes are also valuable, as in our case. Video records should therefore be complementary to observations and written records. Yet video records may sometimes be insufficient to get all the needed and detailed data. A video record involves a tradeoff of breadth (the whole view of the room which do not show the details of individuals in sidebars) and depth (by zooming in on one sidebar or one team member, the rest of the room is left out). Consequently, the observer should be involved in the process of modeling and simulating in order to help make the best decisions about what data is necessary, to make assumptions while building the model, and to validate the developed simulator.

Having the exact timing of all steps in the team process is very useful for developing the model. What is especially valuable is a complete description of sidebars including the total number of all sidebars, where they occur, their exact duration, details on each sidebar formation and evolution: who joined and when, who left and when and even notes on content discussed. Such data would help us define, for example, leaving strategies from sidebars rather than keeping all joiners until the end. We would also be able to measure or better define the level of noise and refine its function by considering the number of members involved in sidebars and their noise volume.

Having access to the complete description of all dialogues and communication would make it easier to collect and extract a long and rich set of keywords and expressions that are selectively monitored. All the sidebar topics could also be collected.

Finally, video recording can be very helpful to trace the error process. In fact, when an error is detected, it is usually a result of one or many previous activities. By getting an "historically" accurate view of the system, we should be able to evaluate the time spent before detecting an error and its propagation depth (in terms of the number of subsystems having received the erroneous data without detecting this error). Last, having a complete visual recording of the collaboration is useful for studying and comparing the monitoring of different sources of information.

7.3.2 Model limitations

Based on our initial objectives of having a very realistic simulator identical to real design sessions, we realized that it was too ambitious to get a "complete" model as a first step and would be not only too complicated to implement but also too difficult to check and validate and even to experiment and use virtually. Thus, we decided to adhere to the KISS[4] principle on one side, and to focus on understanding the fundamental process of sidebar conversations. Consequently, we have voluntarily implemented a "simplified", "bounded" and realistic model, as a first stage in our research in this field, trying to understand sidebar phenomena and their related impacts on error detection.

The following lists several limitations of the model related to each of the model components:

- *Environment*: (1) technologies mediating collaboration in this warroom (e.g. three public displays, publish-subscribe system of networked spreadsheets, or any new communication artifact to be studied virtually) have not been modeled and only verbal information exchange between team members has been considered; (2) the level of noise in the environment is modeled as a factor of the number of simultaneous sidebars and the number of participants in such sidebars is not taken into account.
- *Team*: (1) only the sixteen engineers have been represented. The leader has not been included; (2) Communication is modeled as verbal sidebar discussion only where engineers are monitoring verbal sidebar conversations; (3) other concurrent actor' behaviors and information sources (e.g. leader, neighbor's screen, public display, or any collaboration artifact) should be added in future models. Mediated and direct communication should also be considered.
- *Sidebars*: (1) only two joining sidebar strategies have been modeled and tested separately. Other joining strategies can be added and even a mix of some of them may have different consequences. In other concurrent engineering environments, there may be other joining strategies with different consequences; (2) No leaving strategy has been applied; (3) sidebars are modeled as the only source of error, other sources of errors should be added for future work.
- *Errors*: (1) sidebar discussion is the only source of error detection modeled; error detection by way of public information systems was not modeled, (2) the error source is generic and the specific kinds of errors as shown in Table 1 were not tested.

7.3.3 Experiment limitations

We still can make new virtual experiments with this simulator version in order to test several factors. First, we can study the impact of the actual spatial configuration of the room (e.g. workstation location, positioning of public displays, room dimensions). Second, we can analyze groups and their relationship to the spatial configuration of the room to investigate whether there is an "optimal" group configuration for better error recovery especially during noisy sessions. Third, we can focus on error propagation and the time spent before their detection.

7.4 Future research

Our future work in this application field can be continued as two kinds of activities. First, immediate research activities would use this simulator in-depth by designing and running further hypothetical experiments in order to assess "modified extreme

[4] KISS stands for "Keep It Simple, Stupid" (Axelrod, 1997).

collaboration" and to do a 'what if' analysis asking how things are likely to change under other conditions, such as: (1) changing the room configuration, spatial arrangements and group distribution, (2) changing the group configuration (e.g. getting two or more engineers representing each subsystem), or (3) "splitting" Team M into two or more remote rooms (e.g. by testing different separated configurations and comparing the flow of information between two sites). Then, we would study the influence of space, time, and communication artifacts in the environment on error monitoring by the group.

Second, we plan to incrementally build more complete and realistic model versions including additional sources of information, and enrich agents' behavior by making them able to monitor, selectively access, and filter the team leader, public displays, and neighbor's screen. Consequently, additional fieldwork would be needed to calibrate future models. Our ultimate goal is to get such simulations to support and design collaboration technology for distributed organizations by focusing on distance, dependencies, and delay in a global collaboration.

After extreme collaboration design sessions, all Team M members reported stress after such intensive cooperative work. Modeling stress and exhaustion could be useful to help in finding new combinations of contexts and scenarios to alleviate stress and workload. For example, it may be valuable to find the impacts of spatial organization, room arrangements, room size, redundancy, and diversity in communication tools in a collaborative setting.

8 Conclusions

In this paper, we presented in detail our simulator, which is an agent-based system where autonomous agents behave according to their own information needs (with respect to their interdependencies with others), taking into account the current characteristics of the environment. We have modeled a set of actors, which are conceptually similar (same abstract classes, rules and methods for each agent) but concretely different (each instance has its own properties and social network). We have therefore developed a simulator with sixteen kinds of engineers with contextual micro-behaviors and we observed the emergence of the entire team reorganization in sidebar conversations.

By this modeling/simulation activity we show that we can create a virtual environment with cooperating agents interacting in a dynamic environment. Within modeling activities we can begin to understand in depth the cooperation processes, and we can find out the parameters, variables, and strategies hidden in this work. By running various scenarios, we can understand early the complex processes of the system. Consequently, this simulation approach would be of great benefit for concurrent engineering mainly if it is considered early in the development life cycle: after the analysis and specification phases and before the detailed design steps. Such a simulator can be a design support tool for choosing better solutions for design. Our ultimate aim in applying computer simulation to study complex systems remains to understand and model various configurations of computer supported cooperative work.

Appendix 1: Interdependency matrix

	acs	*cds*	*configuration Graphics*	*cost*	*ground Systems*	*instruments*	*mission Design*	*power*	*programmatic*	*propulsion*	*science*	*structure*	*systems*	*telecom Hardware*	*telecom Systems*	*thermal thermal*	**Number of incoming connections**
acs	–	1	–	–	–	1	1	1	–	1	–	1	–	–	1	–	**7**
cds	–	–	–	–	–	–	–	–	–	–	–	–	–	–	1	–	**1**
configuration Graphics	–	–	–	–	–	–	–	1	–	1	–	1	1	1		1	**6**
cost	1	1	1	–	1	1	–	1	1	1	–	1	–	–	1	1	**11**
groundSystems	–	1	–	–	–	1	1	–	–	–	1	–	–	–	1	–	**5**
instruments	1	1	–	–	–	–	–	1	–	–	1	–	–	–	–	–	**4**
missionDesign	–	–	–	–	–	–	–	–	–	1	–	–	–	–	–	–	**1**
power	1	1	–	–	–	1	1	–	–	–	1	–	–	–	1	–	**6**
programmatic	–	–	–	–	–	1	1	1	–	–	1	–	–	–	–	1	**5**
propulsion	1	–	–	–	–	–	1	–	–	–	–	–	1	–	–	–	**3**
science	–	–	–	–	–	–	–	1	–	–	–	–	–	–	–	–	**1**
structure	–	–	–	–	–	–	1	–	–	–	–	–	–	–	–	–	**1**
systems	–	–	–	–	–	–	–	–	1	1	–	–	–	–	–	1	**3**
telecom Hardware	–	–	–	–	–	–	–	–	–	–	–	–	–	–	1	–	**1**
telecom Systems	–	–	–	–	1	–	–	–	–	–	1	–	–	1	–	–	**3**
thermal	–	–	1	–	–	–	1	1	–	–	–	1	–	–	–	–	**4**
Number of outgoing connections	**4**	**5**	**2**	**0**	**2**	**5**	**7**	**7**	**2**	**5**	**5**	**4**	**2**	**2**	**6**	**4**	

Acknowledgments This material is based upon work supported by the National Science Foundation under Grant No. 0093496. We are grateful to Wan Sze Ng for her help, Team M members, and the Fulbright organization whose grant benefited the collaboration between the authors. We wish to also thank the reviewers whose helpful comments and cogent remarks helped us improve the paper.

References

Axelrod, R, Cohen MD (2000) Harnessing complexity: Organizational implications of a scientific frontier. Free Press

Axelrod R (1997) The complexity of cooperation: Agent based models of competition and collaboration. Princeton, Princeton University Press, New Jersey

Ballot G, Weisbuch G (2000) Editors of applications of simulation to social sciences. In: Proceedings of the second international conference on computer simulations and the social sciences, Paris, September 18–20, Hermes Science Publications, Paris

Bar-Yam Y (2000) Editor-in-chief of unifying themes in complex systems. In: Proceedings from the International conference on complex systems. Advanced Book Program, Perseus Books, Cambridge, Massachusetts

Cannon-Bowers JA, Salas E (2001) Reflections on shared cognition. J Organ Behav 22:195–202

Cannon-Bowers JA, Salas E, Converse S (1993) Shared mental models in expert team decision-making. In: Castellan J (ed) Individual and group decision-making: Current issues: 221. LEA Publishers, pp 221–246

Carley K (2001) Intra-organizationl complexity and computation. In: Baum JC (ed) Companion to organizations, Blackwell, Oxford UK

Carley K (1997) Extracting team mental models through textual analysis. J Organ Behav 18:533–558

Carley K (1996) Validating computational models (working paper). Available at http://www.heinz.cmu.edu/wpapers/author.jsp?id=carley

Cherry EC (1953) Some experiments upon the recognition of speech, with one and with two ears. J Acoust Soc Amer 25:975–279

Dent EB (1999) Complexity science: A worldview shift. Emergence 1(4)

Dugdale J, Pavard B, Soubie JL (2000) A pragmatic development of a computer simulation of an emergency call centre. In: Dieng R, Giboin A, Karsenty L, De Michelis G (eds). Designing cooperative systems: The use of theories and models. IOS Press

Goldstein J (1999) Emergence as a construct: History and Issues. Emergence 1(1)

Hollingshead AB (2000) Perceptions of expertise and transactive memory in work relationships. Group Processes & Intergroup Relations 3(3):257–267

Keyser D (2000) Emergent properties and behavior of the atmosphere. In: Bar-Yam Yaneer (ed) Proceedings from the international conference on complex systems. Advanced Book Program, Perseus Books, Cambridge, MA, pp 33–41

Klimoski R, Mohammed S (1994) Team mental model: Construct or metaphor? J Manag 20(2):403–437

Kreft JU, Booth G, Wimpenny JWT (1998) BacSim, a simulator for individual-based modeling of bacterial colony growth. Biology 144

Levesque LL, Wilson JM, Wholey DR (2001) Cognitive divergence and shared mental models in software development project teams. J Organ Behav 22:135–144

Liang DW, Moreland R, Argote L (1995) Group versus individual training and group performance: The mediating role of transactive memory. Pers Soc Psychol Bull 21(4):384–393

Lissack MR (1999) Complexity: The science, its vocabulary and its relation to Organizations. Emergence 1(1)

Mark G (2002) Extreme collaboration. Commun ACM 45(6):89–93

McKelvey B (1999) Complexity theory in organization science: Seizing the promise or becoming a fad? Emergence 1(1)

Olson JS, Covi L, Rocco E, Miller, JW, Allie PA (1998) Room of your own: What would it take to help remote groups work as well as collocated groups?. In: Proceedings of CHI' 98

Pavard B, Dugdale J (2000) The contribution of complexity theory to the study of socio-technical cooperative systems. InterJournal: New England Complex Institute Electronic journal

Pavard B, Dugdale J (2001) An introduction to complexity in social science. http://www.irit. fr/COSI/training/complexity-tutorial/complexity-tutorial.tml

Simon HA (1969) The sciences of the artificial. MIT Press, Cambridge, MA

Strader TJ, Lin FR, Shaw MJ (1998) Simulation of order fulfillment in divergent assembly supply chains. J Art Soc Social Simul 1(2)

Teasley S, Covi L, Krishnan MS, Olson JS (2000) How does radical collocation help a team succeed?. In: Proceedings of CSCW2000, ACM Press, New York

Terano T (2000) Analyzing social interaction in electronic communities using an artificial world approach. Technol Forecast Soc Change 64

Weick KE (1979) The social psychology of organizing. McGraw-Hill, New York

Narjès Bellamine-Ben Saoud is an Associate Professor at the University of Tunis and Researcher at RIADI-GDL Laboratory, Tunisia. After Computer Science engineering diploma (1993) of the ENSEEIHT of Toulouse, France, she received her PhD (1996), on groupware design applied to the study of cooperation within a space project, from the University of Toulouse I, France. Her main research interests concern studying complex systems particularly by modeling and simulating collaborative and socio-technical systems; developing Computer Supported Collaborative Learning in tunisian primary schools; and Software Engineering. Her current reserach projects include modeling and simulation of emergency rescue activities for large-scale accidents, modeling of epidemics and study of malaria, simulation of collabration artifacts.

Gloria Mark is an Associate Professor in the Department of Informatics, University of California, Irvine. Dr. Mark received her Ph.D. in Psychology from Columbia University in 1991. Prior to UCI, she was a Research Scientist at the GMD, in Bonn, Germany, a visiting research scientist at the Boeing Company, and a research scientist at the Electronic Data Systems Center for Advanced Research. Dr. Mark's research focuses on the design and evaluation of collaborative systems. Her current projects include studying worklife in the network-centric organization, multi-tasking of information workers, nomad workers, and a work in a large-scale medical collaboratory. Dr. Mark is widely published in the fields of CSCW and HCI, is currently the program co-chair for the ACM CSCW'06 conference and is on the editorial board of Computer Supported Cooperative Work: The Journal of Collaborative Computing, and e-Service Qu@rterly.