Project number: 21743
Project acronym: Escritoire2
Project full name: Distributed Crisis Management using Remote Collaboration Technologies

Marie Curie Outgoing International Fellowship (OIF)

Deliverable D2

Implementation of Asymmetric Distributed Collaboration

Period number: 2

Period covered: from Oct 2007 to Mar 2008

Start date of project: 1 October 2006

Project coordinator name: Chris Firth
Project coordinator organization name: Thales Research and Technology (UK)

Organization name of lead contractor for this deliverable: Thales Research and Technology (UK)

Due date of deliverable: 31 March 2008

Date of preparation: March 2008
Date of SESAM submission: -

Duration: 3 years

Revision: 1

Author: Mark Ashdown

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)
Dissemination in level PU (public)

Contents

1 Introduction

This document describes our implementation of an asymmetric distributed collaboration system for supporting coordination of urban search and rescue. Deliverable D1 (Ashdown, 2008), the first document on this project, described the domain, applicability of technology, and a task analysis. This one, deliverable D2, describes our design of a collaborative system to support the tasks analyzed in D1.

We are investigating technology to support a team that is distributed over a geographical area by providing synchronous communication between its members. In particular, we are using an urban search and rescue scenario, where a tactical coordinator is in a command centre, and several operational personnel are positioned in the field. The tactical actor can exploit the benefits of a large display device, so we will give him a tabletop display, while the field personnel are constrained to using small handheld devices because they must be highly mobile. The actors and computing devices are depicted in Figure 1. The large difference in display sizes, together with various other differences between the work and environments of the tactical and operational actors, leads to a very asymmetric form of collaboration. For instance, a *what-you-see-is-what-I-see* (WYSIWIS) system is not feasible because the handheld user can only see a very small proportion of the total information that is displayed on the tabletop.

The cognitive task analysis (CTA) described in deliverable D1 gives us a set of top-down goals for the interfaces we design, that is, a list of information and functions that should be available. We also have a choice of implementation methods driven by the bottom-up technical factors, such as hardware characteristics, networking, and availability of application programming interfaces (APIs).

Section 2 of this document describes our design for the interfaces to support urban search and rescue on the linked tabletop and handheld displays. Section 3 describes our implementation of the system in terms of hardware and software components. This is a fairly general design that could be used for various collaborative systems. Section 4 gives more detail on several specific parts of our simulation of urban search and rescue. In particular, the virtual world in which the search and rescue operation takes place, and the mapping system that allows the geospatial data to be displayed. Finally, in Section 5, we summarize our findings and give some pointers to areas that could be addressed with further work on the technical factors.

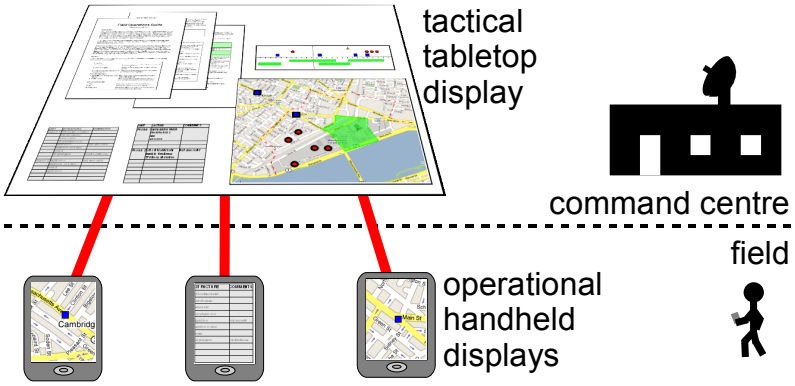

Figure 1: A tabletop is used in the command centre, and is linked to handheld displays in the field.

2 Interface Design

The results of our cognitive task analysis for urban search and rescue coordination are given in the previous deliverable D1 (Ashdown, 2008) in Section 3.3.5. The purpose of the CTA is to generate a set of requirements for the interface we are creating, and in this case our requirements can be summarized as display and function requirements for three types of information: maps, schedules, and forms. We have decided to present these three types of information in shared workspaces, so

collaborators can share the visual information and interact with it synchronously while they talk with each other over the audio channel.

Because of the asymmetry in the collaboration, strict WYSIWIS is not possible, but we would still like to keep the views of the collaborators similar so they can be used to ground conversation. We have also provided workspace awareness mechanisms to help the collaborators remain aware of each other's actions and to enhance communication. The following sections describe how we have used shared workspaces and workspace awareness, then explain the design for the shared maps, schedules and forms.

2.1 Shared workspaces

We are providing collaborators with a shared visual workspace, that is, a visual channel to complement the audio channel that is already provided by two-way radios. Some things are better said orally, whereas others are better shown visually, so we expect that the ability to share visual information will enhance communication.

Video conferencing and groupware systems usually assume roughly symmetric software, hardware, and environment at each end of a collaborative link. In our scenario there is closely-coupled synchronous collaboration between participants, so a what-you-see-is-what-I-see (WYSIWIS) would help, but as described in Section 1, our collaboration is very asymmetric which makes WYSIWIS infeasible. The handheld is much smaller than the tabletop, and there are various other asymmetric factors, so we need to design for asymmetry.

For the shared workspace, the model-view-controller paradigm is applicable. All users in the collaborative system share the same model, but may have different views of that model depending on the information that is pertinent to each user, and how closely they must work together. For symmetric synchronous collaboration, WYSIWIS is suitable. For asynchronous collaboration very different views may be given to different users since common ground is not so important. For instance one user may view the roads in an urban area, while another is viewing the sewage system: they can work separately, then combine their results at the end. For the asymmetric synchronous case that we are studying, we would like to keep the views as similar as possible while acknowledging that various factors, including hardware differences, preclude having identical views on different devices.

Based on the cognitive task analysis described in deliverable D1, we have three types of information to present for our USAR scenario—maps, schedules, and forms. We have decided to implement these as 2-dimensional visual workspaces. The tactical actor on the tabletop can see the whole 2D space, while the operational actor on the handheld can see just a part of it (Figure 2). This paradigm is an obvious choice for the maps, but we also use it for the schedules and forms, to ensure a consistent method for presenting and filtering information.

Choosing to display all the information types in these 2D workspaces does constrain our interface designs somewhat, but we think it is worth it to gain a clear and consistent method for viewing subsets of the information and allowing multiple participants to share the same underlying data while providing a basis for conversation and collaboration.

The transformation from complete workspace to partial view consists of a translation and a scaling. In the case of maps a uniform scaling is preferable, but for the schedules we relax this to allow a non-uniform scaling (for example, the bottom part of Figure 2 shows a wide part of the workspace mapped to a handheld display with a different aspect ratio) because in that case the axes—time on the horizontal and task on the vertical—are orthogonal in a semantic sense, so their scale factors may be chosen independently.

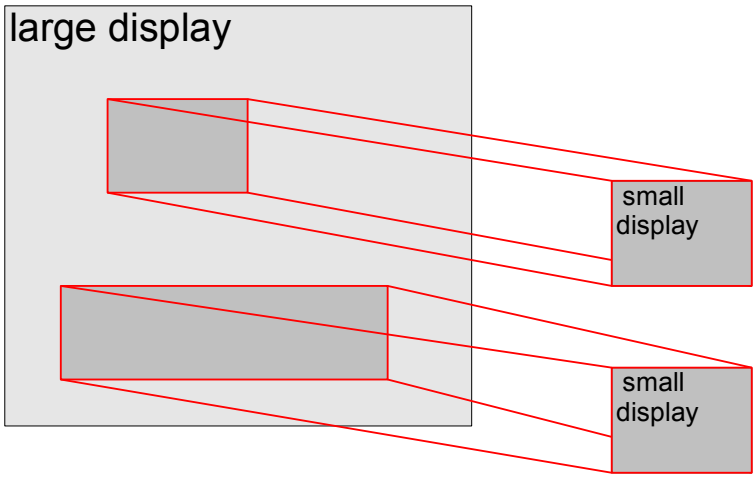

Figure 2: The tactical actor is given a full view of a large 2D workspace on the tabletop display. The operational actors may view part of that space on their handheld displays. Visibility regions are provided on the tabletop.

2.2 Workspace awareness

If we have a 2D visual workspace shared between collaborators as described above, where information may be viewed and modified, we should support each collaborator's awareness of the interactions of others with the workspace. Direct-touch interfaces like handheld displays and tabletops, with touch or pen input, tend to be expressive (Reeves, Benford, OMalley, & Fraser, 2005): the inputs and outputs are visible allowing spectators to observe the experience. However, with remote collaboration the cues that are so expressive in collocated collaboration are lost, so we must support awareness explicitly through the use of workspace awareness mechanisms.

Gutwin and Greenberg define workspace awareness as "the up-to-the-moment understanding of another person's interaction with the shared workspace" (Gutwin & Greenberg, 2002). These interactions are the things that are happening within the temporal and physical bounds of the task that the group is carrying out over the visual workspace. In face-to-face collaboration, people communicate to each other intentionally using speech and gesture, and also communication occurs consequentially because collaborators are able to observe each other. For instance, people's actions and gaze directions provide useful cues for onlookers even though they are not primarily intended as communication. We would like to provide this type of *intentional communication* and *consequential communication* in our remote collaboration system, thus we need to explicitly design the network protocol and visualizations to do so.

To provide a combination of intentional and consequential communication we have implemented telepointers, traces, annotation, visibility regions, and feedthrough of actions like selection. Telepointers are copies of one person's pointer location on all displays. Traces show a history of the last few seconds of the pointers' locations, allowing collaborators to quickly trace line and shapes. The annotation feature allows collaborators to draw on the map, or other workspace, to provide extra information. Visibility regions are like Gutwin and Greenberg's radar views (Gutwin & Greenberg, 2002): they provide awareness of a collaborator's perspective by showing what another person is seeing. Because the handheld displays have a restricted view, the regions they are viewing are shown on the tabletop so the tactical actor can avoid making confusing references to items that the operational actors cannot see. Feedthrough for actions like selecting an item on the map allows remote collaborators to see when one is actually interacting with an item, rather than just moving the pointer over it. Figure 3 shows examples of the five workspace awareness mechanisms for a map display.

There are both intentional and consequential aspects to these mechanisms. For instance, the traces are like the telepointer information integrated over time, and the annotations are like a permanent

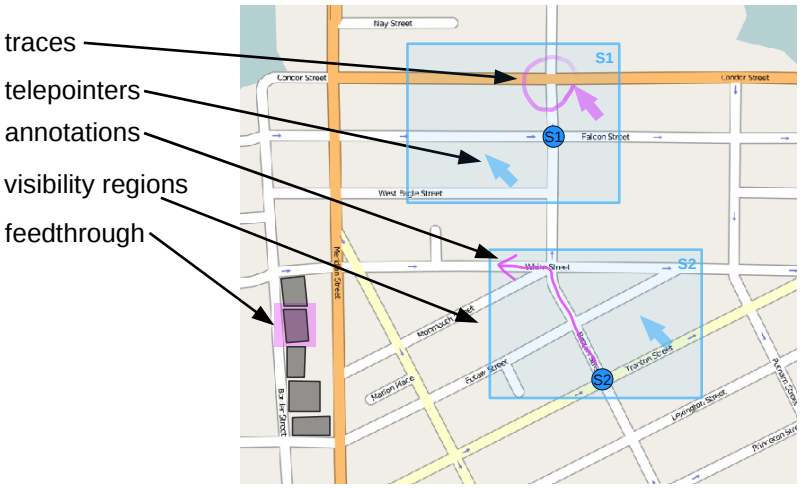

Figure 3: Five workspace awareness mechanisms applied to a shared map.

version of the more ephemeral traces. We have described the visibility regions and feedthrough as consequential communication, but the feedthrough, in particular, could use used intentionally to draw a fellow collaborator's attention to some item in the workspace.

As mentioned, Figure 3 shows the workspace awareness mechanisms applied to a map display, but we have applied these uniformly to the other types of data too. Our choice of using 2D visual workspace for each type of data allows this reuse of the mechanisms, and that in turn provides consistency across the various information types.

2.3 Maps

Our assumptions about the USAR scenario mean that some spatial data is immediately available. All units have GPS (satellite tracking) so we have geospatial data being updated in real time. Also we assume some data on the structures (buildings) in the urban environment exists. This is basic information such as the location and function of each building, and could be entered in the initial *structure triage* phase of the USAR process, if it is not available automatically. The CTA has given us a list of information that must be shown on the map, and a list of corresponding map layers we have implemented is given in Table 2 (Appendix A, page 18).

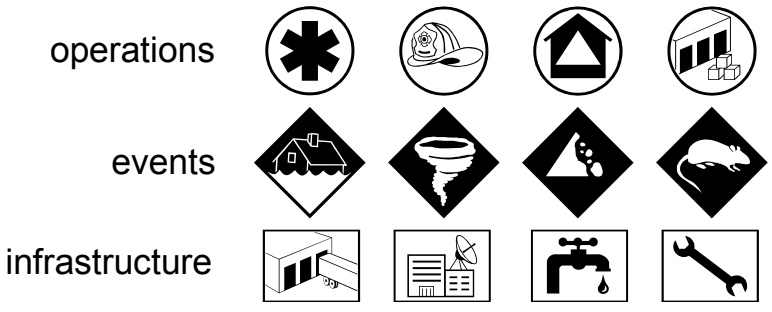

Figure 4: Map symbology. Circles denote operations, diamonds events, and rectangles infrastructure.

Essentially there are five types of data that can go on the map: a raster layer, markers (points), vectors (lines), polygons (areas), and text. These are standard types that will be supported by any mapping software.

For the base raster layer various online options, such as Google Maps, are available for many urban areas around the world, and these are reviewed in Section 4.2. For the markers, we have based

our symbology for the maps on symbol standards for emergency management and first responders[1] (Figure 4). Circles are used for operational items such as search and rescue units, diamonds are used for events such as fires or victims found, and rectangles are used for infrastructure, such as buildings that must be searched. Vectors and polygons are added to display spatial features, such as building outlines, and text can be added to add further data to the spatial items on the map.

Map data is never perfectly accurate, and in large emergency incidents this is particularly an issue because there will be changes to the landscape, such as blocked roads and damaged buildings, that are not shown on the map. We have implemented free-form annotation for the map so that information can be added by any of the collaborators to indicate errors in the existing map, or extra information that may be useful. This is *geometric annotation*, as defined by Heer et al. (Heer, Viegas, & Wattenberg, 2007), who say it is like drawing shapes on an 'acetate layer' above the map.

Each of the map layers listed in Table 2 has three boolean properties, which are explained below:

- Dynamic: the data changes over time, such as the GPS locations of search and rescue units.
- Selectable: the items can be selected with the pen or finger. This selection should feed through to other collaborators, and may affect displays other than the map.
- Editable: the data can be changed by user. An example is the annotations, which are purely user-created.

As described earlier, this is a remote collaboration system, so workspace awareness mechanisms are desirable and we have chosen a set of them (Figure 3). This are implemented via explicit messages being transmitted between devices, and a list of the messages required, and the visualizations by which they affect the map display, are listed in Table 3. The purpose of differentiating between user (U) and global (G) data in that table is that some data is specific to a user, such as the viewport for generating a visibility region, whereas other data is added to the shared model and should be viewable by everyone, such as the annotations.

2.4 Schedules

In addition to the spatial information on the map, we must display temporal information to allow the team to plan their search and rescue operation and possibly adjust the plan if circumstances change. We would like to show the planned future operations, and also how closely the actual performance matched the plan.

For temporal plans (schedules) graphical timelines are better than cognitively difficult textual representations, but care must be taken to highlight the right information (Cummings & Mitchell, 2005). As with the map, we present the schedule in a shared 2D workspace, where the tactical actor views the whole space, and the operational views just part of it. The tactical actor can see all team members' schedules, plus other temporal events, on one large timeline. The operational actor can see the part of the larger schedule that corresponds to his own work.

Figure 5 gives an overview of our design for the schedule timeline. The bold vertical line indicates the present time. Tasks to the right of that are the scheduled future tasks, where the location and length of the bars is determined by the scheduled start times and durations. Information to the left of the present time shows actual task times versus the scheduled ones, so that plans can be compared to performance.

This schedule representation is like a Gantt chart, but rather than having the full staircase shape with one task per row, it has only two rows for each operational unit. This is a space-saving design, and is possible because tasks for a single unit do not over lap. However, some overlap still occurs when comparing predicted to actual performance, so there is a tradeoff between the compactness of the display and the completeness of the temporal information.

Figure 6 shows some detail on the behaviour of the schedules. The red and green highlighted part of the bars shows late and early starts to tasks. If an operational unit is very late to a task, thus

[1] http://www.fgdc.gov/HSWG/index.html

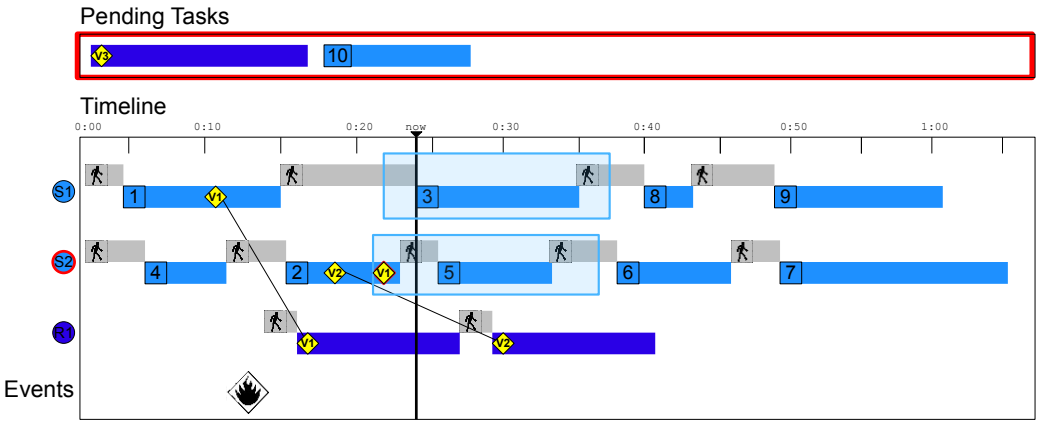

Figure 5: Design for the schedule.

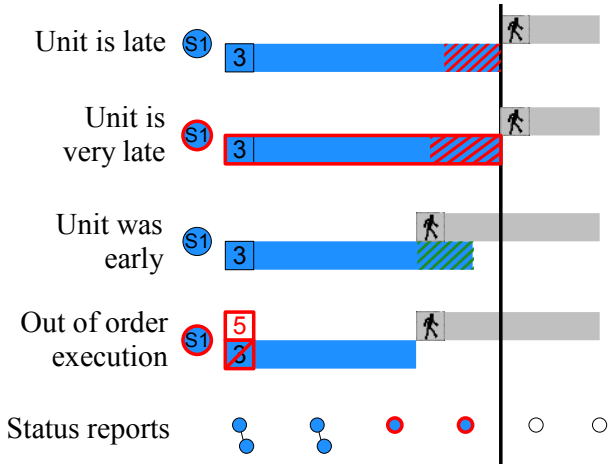

Figure 6: Visual representations of various scheduling possibilities.

exceeding a threshold, the task is flagged with a red outline. It is possible that search units may disregard the schedule and perform searches out of order, so in this case out-of-order execution is signalled, then the tactical actor can update the schedule to match what actually happened.

An issue that was uncovered in our discussion with USAR experts is that units are often required to report their status at regular intervals. The bottom of Figure 6 shows our facility in the schedule for doing that. The circles represent scheduled status reports, consisting of a simple acknowledgement from the unit if all is well, and the responses are shown as extra circles below. The red circles in the figure show missing reports.

Just as with the map, the schedule is arranged in a 2D workspace, and the tactical actor sees all of it while the operational actor gets a restricted view. We use the same workspace awareness mechanisms here as for the map (Figure 3 and Table 3), which provides consistency.

The schedule has various interactive functions, and Table 4 on page 20 lists them. In that table the *C* column indicates which user can control the change, which may be the system, the tactical actor, or an operational actor. The *V* column indicates the users that can view the change, which always includes the tactical actor, and optionally the operational.

2.5 Forms

Filling in forms in an important requirement in USAR operations, to disseminate information to the rest of the organization, for accountability, and for legal reasons. Figure 7 shows a paper form developed by Hampshire Fire and Rescue Service[2] for recording the results of a search. In our particular scenario the search of a structure, and finding of a victim, require a report to be made. We have implemented the necessary forms using standard graphical user interface components, such as drop-down choices and text fields.

We take the same approach to the forms as the other shared workspaces: the tactical actor sees the whole 2D workspace on the tabletop, while the operational actor can see a portion of it on the handheld. Awareness information allows the tactical actor to remain aware of what the operational actor is seeing, and allows the actions of both to be apparent.

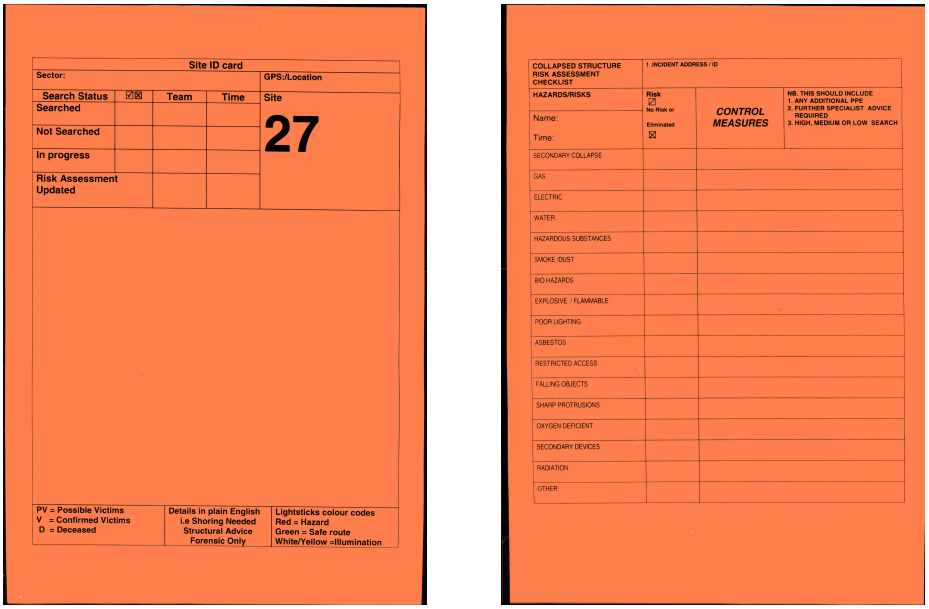

Figure 7: Scans of USAR form from developed by Hampshire Fire and Rescue Service.

3 Collaboration Infrastructure

Our concept for asymmetric synchronous collaboration is based on predicted advances in tabletop displays, handheld devices, and ad-hoc wireless networking, but because the market has yet to produce off-the-shelf products in these areas we have assembled a custom system to test our designs. The sections below describe the hardware combination and software infrastructure we have used.

3.1 Hardware

Much development effort is being expended by companies on improving mobile devices, particularly to exploit the large global market for mobile phones, and the power and flexibility of these devices is increasing rapidly. Similarly, tabletop displays are now a hot topic. It is still awkward and costly to assemble a tabletop display, but over the next few years, large, high-resolution direct-touch displays to become commercially available. For instance, Microsoft will be installing Surface[3] computers in shops, hotels, and casinos in 2009, and Smart Technologies [4] will be releasing a similar product in 2009.

Rather than work on the hardware, we are using currently available hardware and concentrating on the interfaces that can be used on existing and future tabletop and handheld devices. Thus, when the new devices become available, we will have software to run on them.

[2] http://www.hantsfire.gov.uk/
[3] http://www.microsoft.com/surface/
[4] http://smarttech.com/

3.2 Tabletop hardware

In our tabletop system, output is via a tiled front-projected display. Figure 8 shows a two-projector display, where one projector creates a high-resolution area in front of the user, while the other fills the outer region with lower resolution, to create a focus-plus-context effect. The main input device is a large-format digitizer with a 4×3-foot active area, and a cordless stylus. A small wireless keyboard is available for text entry. The desk is run from a single high-end PC. PCI-express graphics cards drive the projectors, and perform the warping to align the graphics.

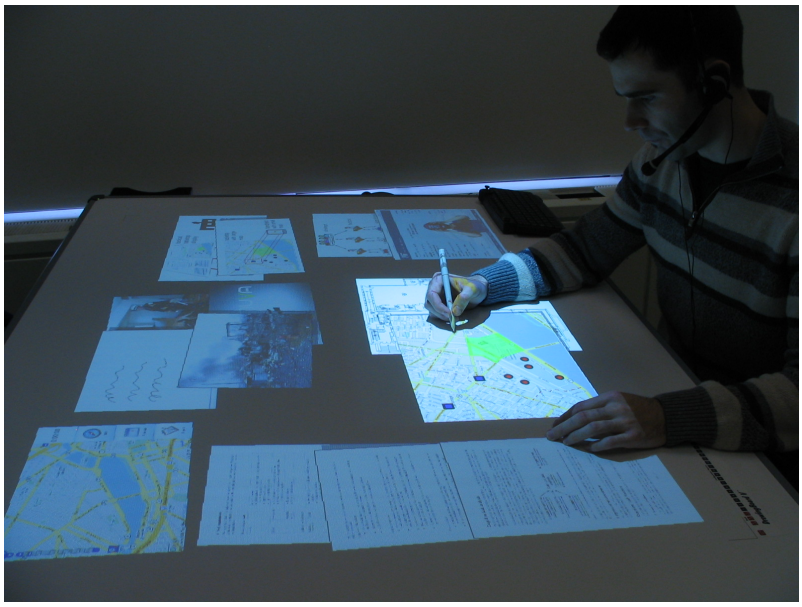

Figure 8: Tabletop display for the command centre.

The hardware is basically the same as the Escritoire (Ashdown & Robinson, 2005), but the software to drive the display is T3, a tabletop toolkit written in Java by Phil Tuddenham (Tuddenham & Robinson, 2007), which is available as open source.[5] In our system the tabletop display and input devices are driven by a modified version of T3.

3.3 Handheld hardware

The handheld device we are using is a Sony Vaio UX, which is a full laptop computer in a small form factor. It runs a standard desktop operating system (Windows Vista) so it is convenient for development.

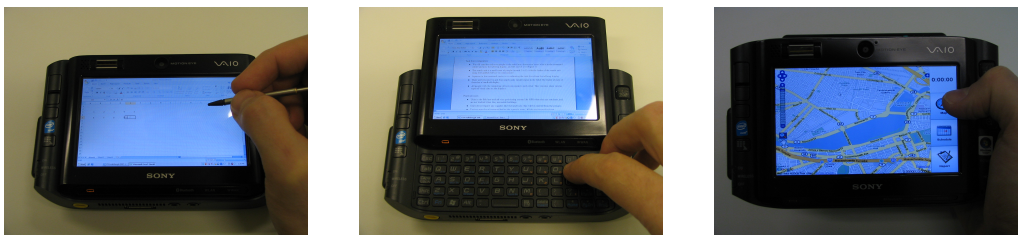

Figure 9: Sony Vaio UX handheld PC. Stylus (left) and keyboard input (middle). Our touch interface (right).

Figure 9 shows the device in use. It has a stylus, a keyboard, and a high-resolution screen, but using standard programs on it like a word processor or spreadsheet is difficult because it is very fiddly. We

[5] http://www.cl.cam.ac.uk/research/rainbow/t3/

have eschewed the stylus and keyboard, and just use the touch screen with the fingers, to create an interface more like the iPhone.

We are using this device for our experiments, although it is a bit bulky so in practice we would want to have something that fits easily into a pocket, or can be strapped to an arm. The Android Platform[6] offers an ideal basis for running our programs on mobile phones. Android is currently only available to run as an emulator on a desktop computer, but hardware should become available, at least to developers, around the end of 2008.

3.4 Hardware asymmetry

The handheld and tabletop devices we are using are obviously very different in size, but apart from that we have tried to keep them similar. They are both rectangular direct-touch surfaces displaying 2D views of a shared workspace. Despite this, the input hardware has some differences.

The tabletop uses a large-format digitizer for input. It has a cordless stylus with three buttons (two on the shaft and one in the nib), it senses the stylus at up to an inch from the surface, and it has an active area of 4×3 feet. We provide a cordless keyboard in case text entry is necessary. The handheld has touch input from a finger and can only sense the finger when it is actually touching the surface. The active area is 2.25×3 inches.

A single-button mouse has two states: *tracking* when the button is up, and *dragging* when it is down (Buxton, Hill, & Rowley, 1985). Our touch screen is also a two-state device, but they are a different two states: *out-of-range* when the finger is off the surface, and *dragging* when it is on. The digitizer has three states, which are the union of those for the mouse and the touch screen. Because the touch screen of the handheld and the digitizer of the tabletop have different states, care must be taken when making an interface that must operate on both. For instance, the handheld display needs separate tools for gesturing (traces) and annotating, because it does not have the tracking state that the mouse and pen have. Also, although writing by making annotations is fine with the pen, it is difficult with the finger, so device-dependent interface tweaks are necessary.

3.5 Software components

Figure 10 shows the software components of our system, with existing software coloured grey, and custom components we have developed coloured green.

We provide audio communications for each actor via a headset and Voice Over Internet Protocol (VOIP) software. We use *all-informed* audio, where all actors can hear all others at all times. This is the standard setup used by USAR teams with their current technology. We are using Mumble[7], which is an open-source cross-platform VOIP system. Possible alternatives are Teamspeak[8] and Ventrilo[9].

The tactical actor uses a single device—the tabletop. The operational actor uses a desktop computer to simulate the physical search and rescue task, and also the handheld display to facilitate communication with the tactical actor. There are various software components that allow these parts to be linked together for remote collaboration, and they are shown in Figure 10. The tactical and operational actors each have a T3 program and a VOIP client providing the visual and auditory channels over which they will communicate. The operational actor also has programs representing the search and navigation tasks.

The figure shows four server programs that allow the collaborators to interact in different ways. The VOIP server, as previously mentioned, allows them to talk to each other. The web server makes the shared maps available. The 'RemoteTable' component stores all of the shared state for the collaborative interfaces, and forwards events. Finally, the 'RISK simulator' runs a simulation of the urban environment in which the operational actors can move around. The part of the system on the

[6] http://code.google.com/android/
[7] http://mumble.sourceforge.net/
[8] http://www.teamspeak.com/
[9] http://www.ventrilo.com/

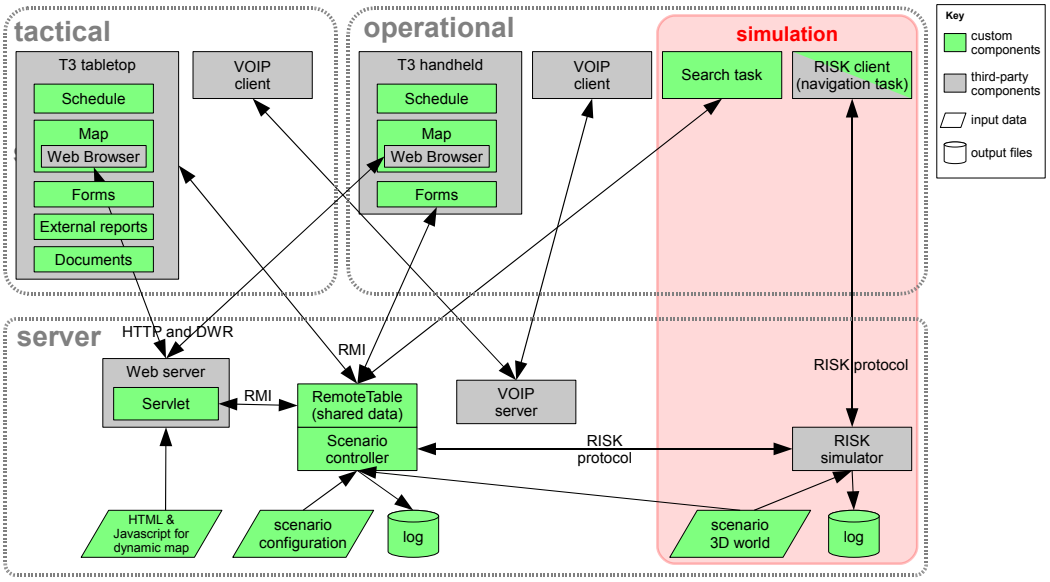

Figure 10: The system

right, shaded in red, simulates the physical search and rescue work of the operational personnel, and is only used in experiments. In a completed system, the real world would take the place of those components.

The T3 toolkit runs separately on the tabletop and each of the handhelds. Each handheld displays a single T3 'tile', whereas the tabletop displays multiple tiles for the map, schedule, forms, and other information. The handheld is restricted to showing just one of these workspaces at a time. The handheld display could be implemented as a standard Java program, without T3, but we use it on all devices to be as consistent as possible.

3.6 Distributed system

Because our synchronous collaboration system must be divided over several computing devices, it is a distributed computing system and the communication between components must be optimized for performance while avoiding problems like race conditions.

There are three parts to the system: tactical, operational, and server. The RemoteTable held in the server is the central data store. After some searching we did not find a good shared data API for real-time applications written in Java, so we created this one. It handles two types of data (Table 1): shared data and event streams.

For shared data the most recent version of the data is held in the server, and is available for any client that subscribes to receive updates. For listeners that slowly process the updates, multiple pending updates are combined into a single update. For shared data only one producer, that is, one client that modifies the data, should be connected. This avoids concurrency problems.

Data like the pointer events that are used to generate telepointers are sent through the RemoteTable via event streams. In this case there can be multiple producers and multiple consumers. Concurrency problems are avoided because each event is an immutable object that is broadcast to all subscribers, then discarded.

Table 1 lists the properties of the two types of data held in the RemoteTable. Having all data go through a central server is useful for administrative and experimental purposes, because all configuration files can be stored in one place which avoids inconsistencies, and the progress of the experimental scenario can be monitored and logged from a single location.

Type	Producers	Consumers	Sequencing
Shared data	One	Many	Only most recent data matters
Event stream	Many	Many	All events are processed

Table 1: Two types of shared data are held in the 'RemoteTable' server.

4 Crisis Simulation

Below are descriptions of two noteworthy parts of our simulation of an urban search and rescue scenario: RISK, which we use to simulate navigation in an urban environment, and the dynamic map, which we have based on existing online mapping tools.

4.1 RISK

To simulate an urban environment in which operational actors, working as search team leaders, can walk around, we have used RISK[10]. It is comprised of an editor in which urban environments can be constructed, a discrete event simulator that handles agents moving around the world and performing actions, and an interface that allows a person to view a 3D rendering of the world and control a character in it. The system is designed for testing crisis management technologies.

Figure 11 shows the RISK world editor, which is used to assemble a virtual world. We have filled our world with buildings from Google 3D Warehouse[11].

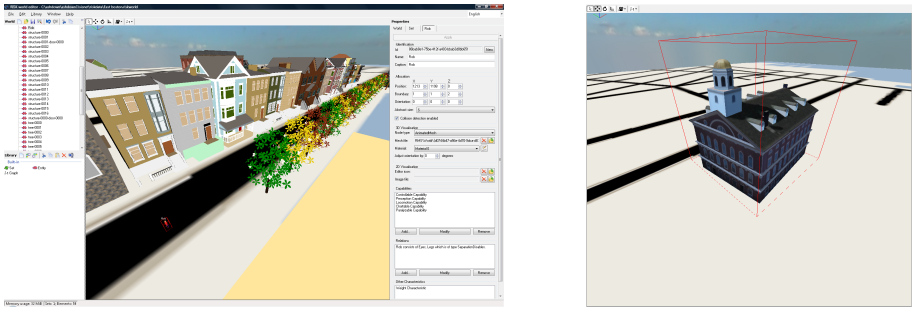

Figure 11: RISK world editor (left). Adding a new building to the virtual world (right).

Figure 12 shows the first-person perspective of a person walking around in the virtual world. In the right image, the blue boxes represent the entrances to the buildings. The user can select one to start a search of the building. This is the view that the operational actor will have in our experiments.

We have written a custom client for RISK that connects to the RISK server and continuously receives the locations of users as they move around. This takes the place of GPS receivers, and the locations are passed to the shared map to be displayed as markers.

Initiating a search of a structure causes the view of the user to switch from RISK to a simple task we inserted to take the place of the search task. This is a low-fidelity replacement for the work of actually searching a structure. Our aim here is not to support the actual searching task—that challenge is being addressed by other research, such as that on USAR robots. Rather, the search task here is just something to occupy the operational actor, and provide some uncertainty about the search duration.

[10] http://forge.decis.nl/projects/risk
[11] http://sketchup.google.com/3dwarehouse/

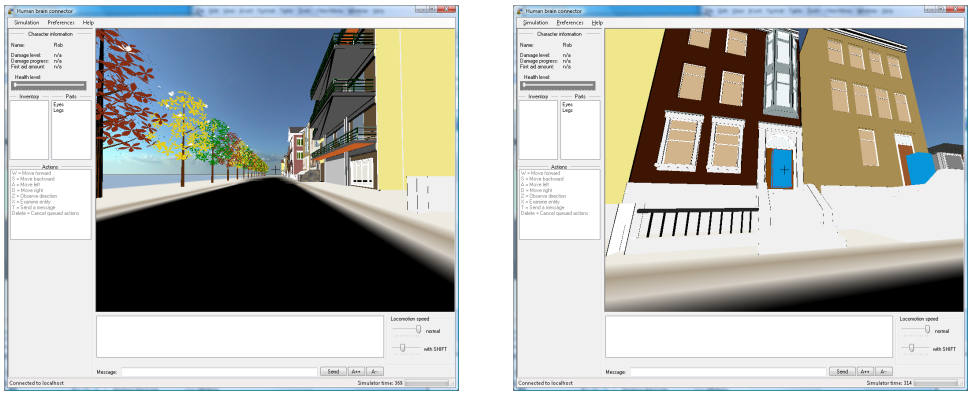

Figure 12: RISK user view. This is the view the user sees while walking down a street.

4.2 Mapping

Online mapping via the web has become popular over the last few years, and it has brought geographic information systems to the masses. Improvements in the services offered over the web continue to be made. We have assumed that it is possible to access a street map of the urban area of the incident, and we augment that map with dynamically generated data.

4.2.1 Mapping tools

We run a web server in addition to the RemoteTable (Figure 10) so that we can make use of online mapping tools, and automatically get the benefits of any improvements to those tools. Making the map web-based also means that an observer of the experiment can connect via a standard web browser to see what the participants are doing.

Various web-based mapping services have become popular in recent years:

- Google Maps[12]
- Yahoo Maps[13]
- Microsoft Virtual Earth[14]
- OpenLayers[15]

We are using OpenLayers, which is a client-side application that is written in Javascript and runs in a web browser, and is free and open source software. There are then various options for the back end that provides the data, such as MapServer[16] and TileCache[17]. We are using Open Street Map[18], which is a free editable street map that covers many places in the world. Map data is added from public sources, and is also uploaded by many volunteers from their GPS devices, or created manually using free mapping programs.

We have assumed that this type of mapping data is available for any urban environment that will be encountered, and we have also assumed that the locations of buildings to be searched are known. Nowadays building plans must be submitted to many governments electronically, so building locations should be known for many cities, but if they are not, they could fairly quickly be added manually. At the beginning of a USAR operation, in a phase known as structure triage, a quick survey is made of the structures in the area to produce the initial search schedule which then guides the detailed searches that we are simulating in our scenario. A handheld interface for the structure triage could allow structure locations to be added by simply touching points on the map.

[12] http://maps.google.com
[13] http://maps.yahoo.com
[14] http://www.microsoft.com/virtualearth
[15] http://www.openlayers.org
[16] http://mapserver.gis.umn.edu/
[17] http://www.tilecache.org/
[18] http://www.openstreetmap.org/

Accuracy of the mapping data is an issue. For instance, a map may not include roads that have been added recently or ones that have been deemed too small, or it may simply contain erroneous data. This is especially true after a large disaster like an earthquake, where the area may have been damaged, with buildings collapsing and routes being blocked. In this case, the operational actor on the ground should notice the discrepancies as he encounters them. He can then speak to the tactical actor about any problem, and he or the tactical actor can annotate the map to indicate to themselves and other team members the new information that has been discovered.

4.2.2 Dynamic data

Web-based mapping systems normally provide only static maps, but for our USAR application we need a dynamic map that updates in real time. Examples of the dynamic data that must be visualized are the positions of team members, annotations to the map by team members, and external reports from the wider emergency response organization.

In our system, the web pages containing the maps are provided by a standard web server. A Java servlet on the server keeps its data synchronized with the shared data in the RemoteTable (Figure 10) which is the central location for data. The web server runs Apache Tomcat [19] and the servlet is deployed on this as a web application. Extra Javascript in the browser, in addition to the OpenLayers API which is used to draw the map, handles the map layers, and interaction from the mouse. This Javascript renders the various map layers and the workspace awareness mechanisms such as telepointers and visibility regions.

In addition to client-pull, where data transfers are initiated by the web browser, to have the map change dynamically we need server-push, where transfers are initiated by the server. We have implemented this using DWR[20], a Java/Javascript library that supports asynchronous function calls between browser and servlet in both directions. Data sent this way is converted to the JSON format[21] which defines string representations for combinations of simple data types and objects.

Our map, in addition to being viewable on the tabletop and handheld displays, can be viewed from any standard web browser, and updates in real time. This is an advantage for doing experiments and demos, and could also be very useful for an emergency response organization: anyone can connect and follow the progress of the team. Security of these systems is an issue, and although we are not addressing this here, map layers could be encrypted so only authorized personnel can access them.

5 Further Work

During development we have uncovered several technological issues that would benefit from further work, which are described in Section 5.1. We have also formed a plan for testing the interfaces we have created.

5.1 Technology

We have identified some areas for further work on the underlying technology that would benefit applications of synchronous communication in a visual workspace.

[19] http://tomcat.apache.org/
[20] http://getahead.org/dwr/
[21] http://www.json.org/

5.1.1 Networking

One can imagine four levels of networking that could be implemented in our distributed USAR scenario: no communication, where devices are not linked at all; broadcast, where data is sent one-way from a central location to all units; asynchronous network, where peer-to-peer messages can be sent in an asynchronous manner, like email and SMS; and synchronous network, where the real-time collaboration that we have described here is possible. The last of these is the most demanding for the network technology, because it places strict demands on latency and packet loss. We hope to show that the benefits of the synchronous visual collaboration that it allows will be demonstrated by this work, thus justifying the cost of the more advanced network.

Network traffic is often divided into three streams: *voice*, *video*, and *data*. Data is the catch-all term used for anything that is not voice or video, but for real-time applications it becomes necessary to differentiate more finely between streams of data. This issue has been considered extensively in online games where high performance and a smooth user experience are vital (Dyck, Gutwin, & Pinelle, 2006).

A networking library tailored to synchronous collaboration would be very useful, and variable quality of service at the network level would be very powerful. Features that could be included in such a software library include optional aggregation of transmitted data, so for instance, Nagel's algorithm which is used for buffering data in TCP could be switched on or off for each stream. Also, the ability to set priorities for different streams within the same connection would allow some streams, such as telepointer events which are small and time sensitive, to be sent before less urgent data such as new bitmap data for the map.

In our implementation there are two types of items in the central server: shared data and event streams. Updates to the shared data use *reliable sequenced* transmission, as defined by Dyck et al. (Dyck et al., 2006), while events use *reliable ordered* transmission. Various other combinations are possible, and a supporting API and protocol stack would make it much easier to exploit these in synchronous collaborative applications.

5.1.2 Collaboration GUIs

Section 2.2 describes our implementation of workspace awareness in this application. The MAUI toolkit (Hill & Gutwin, 2005) adds workspace awareness and networking to standard Java widgets, but only supports standard widgets like buttons and menus. We have presented a system based on a 2D workspace, and multiple views onto that, which we have tried to keep close to being WYSIWIS while acknowledging that some differences in views are necessitated by the role-based asymmetry. It would be helpful to have a software library for building these types of shared workspace systems that require workspace awareness.

Our shared workspace model is also inherently multiuser. Conventional human-computer interfaces tacitly assume a single user, so simply sharing a conventional interface among multiple people is often not appropriate. Recently the single user assumption has been challenged by tabletop displays, which, in contrast, are often assumed to allow multiple collocated users from the start. Libraries supporting multifinger and multiuser input are starting to appear, and these features are being incorporated into operating systems. MPX[22] is an example of such generalized input being incorporated into the X Windows System. Development of features like these should be promoted so developers are freed to produce powerful multiuser and distributed applications.

[22] http://wearables.unisa.edu.au/mpx/

5.2 Experimentation

During the final phase of this project we will be evaluating the effectiveness of the collaborative medium we've designed based on shared workspaces for supporting cooperation of an urban search and rescue team. This will provide us with recommendations for improving our current design, and more general insights for synchronous asymmetric distributed collaboration.

Questions:

1. Does the system have the right features necessary for the team to co-ordinate their task.
2. Does the system have the right workspace awareness features to allow synchronous collaboration over the shared data.
3. What affect does the level of networking have on performance of the team?
4. How do collaborators handle the asymmetry?

Question 1 addresses whether our task analysis, interface design, and software development have resulted in a computer system that satisfies the actual information and function requirements of a USAR task force. The subset of USAR work we are currently addressing is defined by the scenario we have created. Question 2 addresses whether our system also allows synchronous collaboration in the shared 2D workspaces we have provided. The hope is that this synchronous collaboration will be more affective than voice communication alone.

For question 3 we will test the system with three levels of networking: no network, asynchronous network, and synchronous network. These options are explained below. We hypothesize that having a better network will result in better performance, which will go some way to justifying a more advanced network that can support synchronous collaboration. Number 4 is a qualitative question to uncover the strategies people use to work around or exploit the differences in the interfaces they are using. The handheld display is much smaller, and it shows a very restricted subset of the information when compared to the tabletop. It will be useful to discover how people deal with this.

In our scenario there is one tactical actor at the tabletop, and two operational actors, representing search units, that have handheld devices. These three people will be placed in separate rooms, connected by the shared displays on the devices and 'all-informed' audio that allows every person to hear everything else that the others say.

Three levels of networking will be used:

- no network
- asynchronous
- synchronous

This will allow us to measure performance gains that occur when synchronous collaboration is possible, which will be a justification for implementing this type of collaborative system in a real-world setting. Another possible independent variable is the accuracy of the information initially provided in the map and schedule. Real emergency incidents exhibit a lot of uncertainty, and we hypothesize that increased uncertainty will lead to greater interaction between the team members and thus greater demand for the synchronous remote collaboration features of our system.

More detail about the experimental design and dependent variables we will use will be given in deliverable D3, the next report from this project.

APPENDIX

A Tables

This appendix contains some tables detailing the features of the interactive map and schedule described Section 2.

Name	Type	Dynamic	Selectable	Editable	Simulation source	Real world source
Base layer						
Image layer	image	N	N	N	Online mapping	Mapping service
Structures						
Structure outlines	polygon	N	Y	N	From RISK file	Mapping service
Doors	marker	N	N	N	From RISK file	Mapping service
Areas						
Outline of incident zone	polygon	N	Y	N	Drawn by hand	Same as simulation
No-go areas	marker	N	N	N	Drawn by hand	Same as simulation
Units						
Search units	marker	Y	Y	N	From RISK simulator	GPS
Search unit trails	line	Y	N	N	From RISK simulator	GPS
Rescue units	marker	Y	Y	N	From RISK simulator	GPS
You-are-here marker	marker	Y	N	N	From RISK simulator	GPS
Schedule						
Ordered list of structures	line & marker	Y	Y	N	Predefined scenario & scheduler	Structure triage & scheduler
Annotations						
Pen strokes	line	Y	N	Y	User annotation with pen	Same as simulation
Routes	line & marker	Y	N	Y	User annotation with pen	Same as simulation

Table 2: Layers in the dynamic map.

Data	Owner	Format	Visualization	Notes
Viewport	U	Lat,long coords for position and size of viewport. Zoom level.	Visibility region. Indication of search unit (S1,S2) should be given	Tactical sees visibility regions for operational units.
Feedthrough	U	For selection, Id of object that is selected.	Region around selected item is highlighted	
Gesturing	U	Cursor locations in lat,long coords. Timing information.	Telepointers and traces	Timing information is important because these are very sensitive to delay.
Edits	G	Id and new location of any marker that is moved.	Map data is updated on server when it is changed	
Annotation	G	Strokes drawn by the pen (in lat,long coords).	Pen strokes	Annotations can also be erased.

Table 3: Data transmitted to allow synchronous collaboration and awareness, and corresponding visualizations. Owner of the data can be U (data is specific to one user) or G (data is shared globally).

Function	C	V	Details	Visualization
Automatic updates: system changes some shared data				
Movement over time	S	T,O	The display updates on each clock tick	'Now' line moves to the right. Operational view pans automatically if the line goes off screen
Move back late events	S	T,O	If we pass the scheduled start time for a task, delay it	Task is held at the 'now' line
Move forward early events	S	T,O	If an event starts earlier than scheduled, move it to the left	Task is moved to the 'now' line
Scheduled time for task expires	S	T,O	Triggered if actual time is significantly over scheduled time	Red outline around task
Add pending task	S	T	A new task (search or rescue) is added to the system	Add task to pending list. Pending list is highlighted
Out of order execution	S	T,O	Detect out of order execution	Display red marking, and difference between scheduled, and actual tasks
Status displays	S	T,O	Any red highlights in timeline also cause unit icon to be highlighted	Red outline around unit icon on left of timeline
Late status reports	S	T,O	If report is not received in predefined time, show alert	Status report circle(s) turn red
Editing: a user makes a change				
Drag task from 'pending' to timeline	T	T,O	Task lengths are automatically calculated. Time for moving between tasks is automatically calculated	Transparent block shows dragging location
Drag tasks to new positions in timeline	T	T,O	Time for moving between tasks is recalculated	Transparent block shows dragging location
Gesturing	T	T,O	Using pen	Traces and telepointer
Gesturing	O	T,O	Using finger	Traces and telepointer
Highlighting	T,O	T,O	Each displayed item has an ID. Moving pen over item highlights it and associated items on other displays.	Items change appearance upon highlighting
Related functions: actions on other displays that affect the schedule				
Search started	O	T,O	Search unit indicates start of search when entering a building	Start of task moves past 'now' line
Search completed	O	T,O	Search unit indicates end of search when exiting a building	End of task moves past 'now' line
Victim found	O	T,O	Search unit fills in form on victim	Rescue task appears in pending box
Provide status report	O	T,O	Search unit has simple status report button	Late status reports highlighted in red
Possible Additions: these could be added in future versions				
Change to structure details	S	T,O	Predicted time may change due to external information	?
Enter revised search time	O	T,O	Search unit changes scheduled search time	?

Table 4: Interactive schedule functions. C column shows which actor controls a function, V column shows which actors view the result. S=system, T=tactical, O=operational.

References

Ashdown, M. (2008). *Asymmetric Distributed Collaboration in Emergency Response.* (Deliverable D1 for European Commission FP6 Project 21743 (Escritoire2))

Ashdown, M., & Robinson, P. (2005). Escritoire: A Personal Projected Display. *IEEE Multimedia Magazine*, *12*(1), 34–42.

Buxton, W., Hill, R., & Rowley, P. (1985). Issues and Techniques in Touch-Sensitive Tablet Input. In *Proceedings of SIGGRAPH 85* (pp. 215–224).

Cummings, M. L., & Mitchell, P. J. (2005). Managing Multiple UAVs through a Timeline Display. In *Proceedings of AIAA 2005 FIXME.*

Dyck, J., Gutwin, C., & Pinelle, D. (2006). Beyond the LAN: Techniques from Networked Games for Improving Groupware Performance. In *Proceedings of 2007 International ACM Conference on Supporting Group Work* (p. 291300).

Gutwin, C., & Greenberg, S. (2002). A Descriptive Framework of Workspace Awareness for Real-Time Groupware. *Journal of Computer-Supported Co-operative Work*, *11*(3–4), 411-446.

Heer, J., Viegas, F. B., & Wattenberg, M. (2007). Voyagers and Voyeurs: Supporting Asynchronous Collaborative Information Visualization. In *Proceedings of CHI 2007.*

Hill, J., & Gutwin, C. (2005). The MAUI Toolkit:Groupware Widgets for Group Awareness. *Computer Supported Cooperative Work*, *13*(5–6), 539–571.

Reeves, S., Benford, S., OMalley, C., & Fraser, M. (2005). Designing the Spectator Experience. In *Proceedings of CHI 2005* (pp. 741–750).

Tuddenham, P., & Robinson, P. (2007). T3: Rapid Prototyping of High-Resolution and Mixed-Presence Tabletop Applications. In *2nd IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP'07).*