Activity-Based Collaboration

“The network is opening up some amazing possibilities for us to reinvent content, reinvent collaboration.”
- Tim O’Reilly

By Morten Esbensen and Søren Nielsen
(080484-mortenq,180486-snielsen)

Supervised by Jakob E. Bardram
May 2010

IT University of Copenhagen
Rued Langgaards Vej 7, DK-2300 Copenhagen S
Abstract

Activity-based computing (ABC) is a new paradigm, in which the basic computational unit is expressed as an activity of a user rather than a file or an application. A basic tenet within ABC is that users can share any activity, resource or service they may need. This may span a single building (like a hospital) or the globe (like from Denmark to India). The ABC framework version 5.0 implements activity-based computing, but does however not support collaboration.

This thesis focuses on investigating activity-based collaboration, such as how to facilitate synchronisation of activities and communication between users. A proof-of-concept solution to these issues are developed upon the ABC framework version 5.0. This is evaluated and discussed and relevant findings will be presented as well as future work is suggested.
Acknowledgements

First of all, thanks to our supervisor Professor Jakob E. Bardram, for the great help and counseling throughout the entire process. Also thanks to ABC research team member and Post-Doc Jonathan Bunde-Pedersen, who has been very helpful along the way. We also owe a ‘thank you’ to the people normally in the pITLab, for being so tolerant and helpful.

Finally we want to thank our test users Marco, Christian, Jeppe, Tobias, Kasper and Tim for their engagement in the testing phase. We really appreciate your help.
# Contents

<table>
<thead>
<tr>
<th>1 Introduction</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Context and motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Goals</td>
<td>2</td>
</tr>
<tr>
<td>1.4 Methods</td>
<td>2</td>
</tr>
<tr>
<td>1.5 Results</td>
<td>2</td>
</tr>
<tr>
<td>1.6 Overview</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2 Activity-based computing</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Background</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Activity sharing and collaboration</td>
<td>5</td>
</tr>
<tr>
<td>2.3 ABC framework version</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3 Activity-based collaboration</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Computer Supported Co-operative Work</td>
<td>10</td>
</tr>
<tr>
<td>3.2 Collaboration in a real world scenario</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4 Implementation</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Synchronisation</td>
<td>16</td>
</tr>
<tr>
<td>4.2 User awareness list</td>
<td>18</td>
</tr>
<tr>
<td>4.3 Participant list</td>
<td>19</td>
</tr>
<tr>
<td>4.4 Real time video and audio</td>
<td>20</td>
</tr>
<tr>
<td>4.5 Action log</td>
<td>22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5 Testing and evaluation</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Evaluation of collaborative properties</td>
<td>25</td>
</tr>
<tr>
<td>5.2 Functionality test</td>
<td>26</td>
</tr>
<tr>
<td>5.3 Usability test</td>
<td>28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6 Conclusions</th>
<th>33</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Future work</td>
<td>34</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A Thesis Focus Change</th>
<th>37</th>
</tr>
</thead>
</table>

| B Questionnaire with results       | 38 |
1 Introduction

In this chapter, we introduce the concepts of collaboration in activity-based computing (ABC). We will describe some background on the subject and explain why this bachelor's thesis is relevant. Furthermore we will present our problem, describe our goals, our methods, and briefly summarise our results. Finally we will give an overview of the remainder of the thesis.

1.1 Context and motivation

In a non-office work situation, for example at a hospital, work is characterised by being collaborative, often interrupted, mobile and with the need of communication [1]. The treatment of a patient may involve several employees, like a nurse and a doctor. It might span over several different locations, like the operation room and the ward, and over longer time periods. The characteristics of this kind of work calls for computer systems to be pervasive in its environment. Activity-based computing is a concept that has been developed upon these considerations, from studies of how real-life work is being carried out.

Today computer systems are used widely in work situations and while they provide good support for office tasks, like writing reports or manipulating data, they do not support higher-level activities very well. ABC is a new paradigm, which aggregates the computational units, like a file or an application, into an activity of a user. Rather than searching the harddrive for specific files to open in specific applications, activity-based computing offers the possibility to just resume an activity, in which the files are known as resources, and the applications are embedded within the user interface.

According to Bardram et. al. "... a core aspect of everyday activities – especially in a hospital – is their collaborative nature." [2, p.212]. Given the importance of collaboration in everyday human activities, we wish to investigate some collaborative aspects within ABC. We will be looking at communication and synchronisation and how to support these types of collaboration in the ABC framework.

1.2 Background

Since 2003 Jakob E. Bardram has been leading the ABC research, which is financially funded by The Danish Council for Strategic Research [3]. The research team has developed four versions of the ABC framework, which implements activity-based computing, and is currently developing a version 5. The previous versions have all contributed with relevant work on collaboration, which will be described in section 2.2. The framework has been developed and managed by people from the Computer Science Department the University of Aarhus, the IT University of Copenhagen and Medical Insight A/S.

The version under development, in The Pervasive Interaction Technology Lab at The IT University of Copenhagen, has a Java-based peer-to-peer infrastructure based on the AEXO infrastructure, developed by Jonathan Bunde-Pedersen [4]. In this version a user interface is being implemented...
Chapter 1. Introduction

in Adobe’s open source framework, Flex. Both the ABC framework and the user interface will be described in more detail in chapter 2.

A basic tenet within ABC is the principle that users can share any activity, resource or service they may need. This may span a single building (like a hospital) or the globe (like from Denmark to India). The ABC user interface however does not support collaboration and therefore needs to be extended.

1.3 Goals

In order to support collaboration the ABC framework needs to facilitate synchronisation of activity states and communication between users. One problem is to determine what level of synchronisation is needed and how to obtain that. Another is to ensure efficient communication through different communication levels. Within these two main problems we define three goals.

- Activity based collaboration
- Unified collaboration model
- UI widgets

Through investigations we will define activity controlled collaboration and using these findings we wish to come up with a unified collaboration model to support different kinds of collaboration. The unified collaboration model will then be implemented as several UI widgets in the existing ABC framework.

These goals can be used as a guideline for the remainder of the thesis and through them we will argue that collaboration is an important subject within activity-based computing.

1.4 Methods

The aforementioned goals will be achieved using the following methods.

1. Investigation
   The we will investigate collaboration and discuss how we can apply it to an activity-based environment. This will be achieved by analysing and discussing real-world collaboration and project it upon the ABC framework.

2. Proof-of-concept
   Based on the analysis and discussion of important elements in activity-based collaboration we will develop a proof-of-concept solution implementing state synchronisation and communication. The solution will be developed upon the existing ABC client.

3. Evaluation
   Lastly we will evaluate the proof-of-concept implementation. We will define a set of collaborative scenarios and have test persons perform these using our implementation. We will then have the test persons fill out a questionnaire and interview them based on their answers. Then we will add our own observations and based on this evaluate our implementation and discuss if it supports collaboration satisfactory.

1.5 Results

We have investigated activity-based collaboration and built a proof-of-concept. We defined two features needed to support collaboration synchronisation and communication and implemented them.
Chapter 1. Introduction

1. Synchronisation
We have implemented state synchronisation of operations. The state of any operation (position or size) is saved at runtime, and users collaborating on actions will share operation states.

2. Communication
We have implemented four components supporting communication; user-awareness list, participant list, real time video and audio and an action-log. Users are able to see whom of the other users are present based on location, they are able to call each others or leave notes in the action-log. Also, when users collaborate on an action, video communication is automatically established.

We have tested our implementation using collaborative scenarios and shown that the features does support collaboration – however there are room for improvements. Real time video and audio was very popular while synchronisation was widely seen as annoying. Based on our evaluation we have proposed a list of future improvements to the system. Our proof-of-concept solution will be used in ABC client to support the collaborative aspects of activity-based computing.

1.6 Overview
This section will shortly summarise each chapter and underline why they are relevant for the thesis and how they try to cover our goals.

Chapter 2 will describe the concepts of activity-based computing and the ABC framework version 5. First activity-based computing is described by explaining the principles around the concepts. We then discuss the principle of activity sharing in detail as this highly relates to our problem. Last the ABC framework version 5 is explained using screenshots to help the reader understand how the framework supports activity-based computing. Chapter 2 serves to explain the concepts upon which our project is build.

Chapter 3 will discuss collaboration. First summarize some of the relevant work done in the field of computer supported co-operative work (CSCW). We then analyse some concepts of collaboration in a real-world scenario. This analysis will help us to decide which elements to include in our proof-of-concept to support collaboration. We divide collaboration into different types based on time and location and argue that different approaches should be made to support these different types. Finally we discuss our problem by analysing activity-based collaboration. In this last section we discuss different kinds of activity-based collaboration and propose different solutions to support them. Chapter 3 highly relates to goal one and two, as we investigate activity based collaboration and propose a unified collaboration model.

Chapter 4 describes our proof-of-concept implementation. The chapter is divided into sections, where each section describes one functionality we have implemented. The sections will contain screenshots and UML-diagrams to increase the understanding of how the collaborative components work. Chapter 4 highly relates to goal 3, as it describes and documents our the UI widgets we have used in our proof-of-concept.

Chapter 5 will evaluate and discuss our implementation. We start off by describing how we wish to test our implementation, and then we define a series of tests. We then present the test results, that together with the problem analysis in chapter 3 forms the basis of a discussion and evaluation of our implementation.

Chapter 6 concludes the thesis and summarises the key contributions of the presented work. Suggestions to areas for future work within activity-based collaboration will be listed as well.
2 Activity-based computing

In this chapter we will present the basic concepts within ABC and the ABC framework. We will explain why collaboration is important and we will introduce some technologies and frameworks, that will help us obtain this collaboration.

2.1 Background

As described in the introduction activity-based computing is centered around human activities, e.g. the writing of this Bachelor’s Thesis. Figure 2.1 illustrates the activity aggregation of services and resources used within the work of the activity “Bachelor’s Thesis”. The activity uses three different services which each operate on a number of resources. As opposed to the traditional application-file model of computers, the center of focus for the user is not the applications or the files but the activity. Opening this activity will cause the relevant services and resources to be available to the user.

![Diagram of activity aggregation](image)

**Figure 2.1:** The "Bachelor's Thesis" activity, illustrating how an activity embeds applications and resources.

Activity-based computing is defined around six principles, being; activity-centered, activity suspension and resumption, activity roaming, activity adaptation, activity sharing and activity discovery \[3\]. These principles form the basis of activity-based computing and we will therefore give a concise introduction to them.
Chapter 2. Activity-based computing

1. **Activity-centered**
   A range of resources are collected in a coherent set as a computational activity. The activity acts as an abstraction of the application- and document-centered model layer to support the user's work.

2. **Activity suspension and resumption**
   Any activity can be resumed and suspended by its participants. When resuming an activity, all resources within the activity is brought back as they were left. This gives the user an opportunity to switch between activities and therefore handle interruptions.

3. **Activity roaming**
   An activity is stored and modelled in an infrastructure from where it can be distributed to machines that can resume, suspend and manipulate it. Activities can be suspended on one device and moved to another where it is resumed.

4. **Activity adaptation**
   An activity is able to adapt to its environment, being the resources offered on the device on which the activity is resumed. These resources, such as the CPU or display, will all together decide how to render the activity. An activity can therefore be rendered differently when moving it from a wall display to a PDA.

5. **Activity sharing**
   An activity can have multiple participants. A participant can either resume an activity and continue another's work, or if other participants are resuming simultaneously on other devices, their states will be synchronised. In addition a desktop conference will be initialised to help users collaborate.

6. **Activity discovery**
   Activities are context-aware and are able to adapt and adjust themselves according to their context. This could be to suggest present resources or adapt the user interface to the current work situation.

Implementing these concepts brings us a step closer to support human activities in an ubiquitous environment and thus move away from the regular application- and document-centered model. In this thesis we are concentrating on the collaborative aspect of activity-based computing and we will therefore go in depth with the fifth principle, activity sharing.

2.2 **Activity sharing and collaboration**

Without activity sharing, there is no collaboration. An activity with multiple participants allows for the participants to collaborate both asynchronously and synchronously. Asynchronous collaboration is obtained when a participant resumes the work of another, and synchronous collaboration is obtained when multiple participants are engaged in the same activity at the same time. While asynchronous collaboration calls for sessions to be stored while the activity is suspended, synchronous collaboration adds even more requirements.

When collaborating synchronously, users will expect some kind of awareness as when collaborating in a normal office environment. Synchronisation and communication is a part of a normal office environment and to achieve a user feeling of a convenient and effective workspace in ABC, this have to implemented in some way.

Synchronisation has previously been studied within the ABC framework and one of the recurring problems is to find a balanced level of state synchronisation [5]. It is simply not feasible to synchronise every action (such as moving a scrollbar) performed by the users. The distribution
of events will most likely give a performance overhead, and collaborative systems will therefore
require some kind of synchronisation policy. This has been implemented in the ABC framework
version 3.1 using optimistic state synchronisation, meaning a shared state replaces the old one [1].
This will potentially cause concurrency conflicts, but studies have shown that the users adapt their
interaction towards each other and thus minimise the number of conflicts [6].

Version 3.1 also implements a participant list, chat, voice-link and telepointers. These collabora-
tion widgets was primarily developed for remote collaboration, but later studies in hospitals have
shown that clinicians use some of these features when co-located [7]. This is an interesting finding
when discussing collaboration in ABC. We know that these widgets can improve remote collabora-
tion, but now we also know that the widgets can improve understanding and collaboration when
co-located, and this knowledge is used when discussing collaboration in chapter 3.

Videoconferencing has been discussed before, but has always been a subject to problems. In
the early development phase of version 4 MS NetMeeting was used for videoconferencing, but the
explicit dial-up introduced several usability problems [7] and in version 3.1 the video-link was
replaced with an audio-only link [1].

2.3 ABC framework version 5

The ABC framework version 5 is the current under-development implementation of ABC. The ABC
framework provides a runtime infrastructure handling the complexities in the management of
distributed and collaborative activities.

2.3.1 Architecture

The concepts of ABC is implemented as a Java-based peer-to-peer infrastructure based on the
AEXO infrastructure. AEXO is a hierarchical map (HMap) and event-based subscription system
[8], which allows the ABC clients to subscribe to any subtree of the map and thus receive updates
whenever something is altered.

![Figure 2.2: The main components in the ABC infrastructure](image)

The ABC infrastructure, as illustrated in fig. 2.2 consists of the following main components.
• The activity manager is an AEXO component responsible for managing activities including related data such as services, resources, users, clients and other activity managers. The activity manager also implements the activity controller, service registry and activity manager registry which are all accessible from the HMap and therefore available for any client.

• The activity controller serves as a RESTful webservice, responsible for communication between the distributed HMap and the clients. The activity controller handles the management of data, activities, clients, users and states. When the activity controller is called from a client and e.g. changes an activity’s state, the activity manager is notified.

• The service registry is responsible for managing available ABC services. Once an ABC service is discovered by the activity manager, the service is enlisted in the service registry.

• The activity manager registry is responsible for managing other relevant activity managers. The activity managers are enlisted in the activity manager registry, making them available for the clients.

• The context manager is responsible for managing context information.

• ABC services are running separately with no connection to the activity manager and announces themselves using the Bonjour type _abc-service._tcp.local..

• The activity model is a subtree of the HMap and is responsible for arranging activities. Clients can subscribe to any activity, action, operation or resource within the activity model. Figure 2.3 illustrates how the activity model is structured.

![Activity Model Diagram]

Figure 2.3: The general activity model implemented in the ABC framework version 5.

2.3.2 Interaction design

The ABC client is written in Adobe Flex and implements the activity model. In this section we will describe the interaction design and present screenshots of the user interface. The user interface consist of two general views; activity universe and action view. The activity universe provides an overview of the relevant activities, where the action view provides a specific action workspace for the user.
Activity universe

The ABC client starts off by registering itself on an activity manager using the `RegisterClient()` method on its activity controller. When registered, the client is listed in the map and thus have access to information such as "relevant activities". These activities are then loaded into the activity universe and the client subscribes to each activity.

The activity universe, as shown in fig. 2.4, consist of a menu bar (1) and the relevant activities in the current context. Each activity (2) has a list of actions (3) available. The action list can be shown and hidden by pressing the arrow button on the activity (4). Each action has an icon, a title and a status indicator, being red if suspended and green if resumed. The arrows (5) shown beside the actions can be pressed to shuffle through the list of actions.

![Activity Universe Diagram](image)

**Figure 2.4:** The activities "Bachelor's Thesis", "Data Mining Project" and "Holiday planning" floating around in a snapshot of the activity universe.

The tab in the menu bar, "AM", can be unfolded and show the available activity managers. This is, as of this phase of the development, used to connect to an activity manager and logging in. This has to be done before being able to resume an action. The cross in the menu bar simply closes the client. The background allows for the user to drag around, and by doing that, panning through the "universe". Activities as well can be dragged around.

When pressing an action, `resumeEnactment()` is called on the activity controller and the action is resumed if possible and the user is taken to the action view.

**Action view**

The action view, as shown in fig. 2.5, consist of an extended menu bar (1) and some operations (2). The operations are arranged in a grid and cannot overlap one another. Each operation embeds its
resource using a render defined by a specific MIME-type. In this case some source code, a website and two PDF documents.

The menu bar adds the tabs "Resources", "Services", "Info" and "Suspend" compared to the activity universe. "Resources" holds a list of available resources within the activity, while "Services" holds a list of available services in the current context. "Info" contains information about the current activity and does also contain a list of the activity's actions, so the user is able to shift between actions without going back to the activity universe. "Suspend" obviously suspends the action by calling the \texttt{suspendEnactment()} on the activity controller and returns to the activity universe.
3 Activity-based collaboration

In this chapter we will describe and discuss activity-based collaboration, as designed in this project. First we will briefly describe previous work in CSCW. We will then look at key concepts of collaboration in a real-world work environment and use scenarios from work situations to identify needs for collaborative support. Place and time will be used to distinguish between the different types of collaboration.

Finally we will then discuss how these concepts can be applied to an activity-based environment and redefine the scenarios to reflect situations using a collaborative supporting activity-based system.

3.1 Computer Supported Co-operative Work

Synchronous work is a common topic within CSCW and even though single-user widgets can be redesigned to support multiple users, groupware widgets such as participant status, telepointers and workspace awareness has shown to be of much importance [6]. These widgets has been developed over the years and combined with audio and video communication [9], they form the basis of collaboration in the ABC client.

The challenges met when designing a user interface within CSCW is quite different from the traditional user interface. In collaborative systems the direct connection between the user’s and the system’s activity is broken, and we will therefore be forced to re-think the elements and functionality [9].

Several studies suggest using a session management approach [6, 10], which has been used in previous implementations of the ABC framework [1]. Other applications such as Corona [11] suggest using a publish-subscribe and group approach.

Our discussion of collaboration is inspired by previous work within CSCW however we will argue later that our implementation differs from the session-based approach as well as the Corona approach.

3.2 Collaboration in a real world scenario

To effectively support collaboration in the ABC framework, we need to investigate how collaboration works in a real world scenario. It is important for the ABC framework to support collaboration in a familiar way. Instead of redefining it we introduce collaboration based on time and place.

3.2.1 Synchronous collaboration

In this section we will look at synchronous collaboration. We will later look at asynchronous collaboration and show that these types require different computational support. Using location as a parameter, collaboration can be further divided into two general types: remote collaboration and co-located collaboration. We investigate these two types separately as they show different characteristics:
Remote collaboration

Remote collaboration refers to the situation where two or more users are located in different locations. This distance between users creates a need for collaborative software to be able to establish communication and coordination between the users. The following example illustrates a remote collaboration scenario. The scenario is a meeting within a research project group and takes place at the IT-university of Copenhagen (ITU). Each Monday all participants of the project group get together for a status meeting. Since the participants are located in different locations; some at the IT-university of Copenhagen and others at Aarhus University (AU), a computer with a skype connection is being used to transmit sound.

Scenario A:

At ITU five people are sitting around a table discussing an issue with a program being developed. They have a computer with a skype connection, and in the other end of that connection, at AU, two more people are participating. As they discuss how to solve the problem, the people at ITU refers to a figure on their computer. But as the people at AU do not have the figure, they request it being sent. After the discussion they look at the code in the program where they suspect the error to occur. The people at ITU explain to the people at AU exactly where in the code they are looking. They examine the code and discuss the problem and eventually come up with a few possible solutions they wish to test. After the discussion they wrap up the meeting, and the people at AU requests the notes from the meeting in an email.

When we investigate this example, several important aspects of remote collaboration emerge. The first thing we notice is the importance of an audio-link between the remotely located participants. This communication channel is crucial for the collaboration and is used constantly to exchange information. The second thing to notice is the lack of computational support for sharing resources. The figure that the people at ITU used to examine the problem had to be sent to AU for them understand and participate in the discussion. Furthermore, all notes that were taken during the meeting were afterwards compiled in an email and sent to all participants. This scenario shows the need for a communication channel, sharing of view and sharing of resources.

Co-located collaboration

Co-located communication is the situation where persons collaborating an the same location.

Following is a scenario illustrating co-located collaboration. The scenario is a project writing session at the IT-University of Copenhagen.

Scenario B:

Three students (here named A, B and C) sit around a table at the university solving a coding assignment. Some work has already been done and they discuss how to proceed. Student A shows the others his proposal for a solution by bringing forth a document on his computer and explains it to the others. While student A explains, student B takes notes in his notebook. After student A has finished explaining they all discuss further, and Student C adds a few notes in student B's notebook. They agree upon a solution, delegate the tasks and start working.

This example show some interesting facets of co-located collaboration. The first thing to notice is the situation where student A explains the others about his thoughts by showing them his solution proposal on the computer. Using artifacts such as a computer screen, a document, a graph etc. is vital to the collaboration to emphasise a point or simply to let others know what is being talked about. This aspect is important to model in a collaborative system, to let users collaborate in a way they are used to. Not supporting this kind of collaboration puts extra work on the users, as they would have to explicit their explanations not having artifacts to rely on. The second thing
to notice from the example is the use of the notebook to gather information. Unlike working on different copies of a document in a computer, whenever someone grabs the notebook, writes in it or somehow changes it, that change is reflected in the group. To fully support co-located collaboration, this aspect as well should be supported by the program. This scenario shows the need for sharing of resources and sharing of views.

### 3.2.2 Asynchronous collaboration

The previous examples of collaboration are all carried out synchronously. That is, participants collaborate in real-time by discussing or coordinating their work. However, in many cases collaboration will happen asynchronously. Asynchronous collaboration refers to the situations where several persons are working on the same task however not at the same time. This kind of collaboration will often be coordinated so the users know what their roles are, however there is also a need to coordinate work as it progresses. This coordination can be done in real-time collaboration like arranging a meeting, a conference or similar or it can be done by leaving messages to each other.

The following is a scenario displaying asynchronous collaboration. The scenario is a work situation at the pITLab at the IT University of Copenhagen.

**Scenario C:**

Two people have been developing a system handling advanced calculations using several computers on a network. One of the people starts the system to stress test it over night, and when leaving he puts a post-it note on the computer for the other person to notice, when he comes in the next morning.

The example shows the need to be able to leave notes in the system to notify others of changes. These messages can be a formal e-mail or a wiki-entry to all participants involved in the collaboration, or it can be an informal post-it note on a desk or a whiteboard as in the scenario. For the ABC framework to support asynchronous collaboration, we want to make sure it is possible for users to leave messages to each other.

Co-located and remote collaboration in an asynchronously situation is different in the way that asynchronous remote collaboration might require more coordinating and communicative support. The difference between co-located and remote asynchronous collaboration however is not significant, so we consider asynchronous collaboration without considering location.

### 3.2.3 Summary

The previous examples provides us with some interesting knowledge about collaboration. In both synchronous collaboration examples there is a need for the possibility of sharing different resources and views. In the remote situation there is also a need for audio communication. Furthermore in an asynchronous collaboration scenario there is a need for leaving notes to each other. In the next section we will discuss how to support these needs in the ABC framework.

### 3.3 Activity-based collaboration

In this section we will discuss how to support collaboration in the ABC framework. The discussion on real world collaboration showed that there is a need to support communication and sharing of resources. Based on these findings we define two general areas of collaborative supporting features; communication and synchronisation. We will end up proposing a unified collaboration model for supporting activity based collaboration.
3.3.1 Scenarios

We rewrite the work scenarios from the previous section to see how they would play out using an activity-based computing system supporting collaboration.

ABC scenario A:
At ITU five people are sitting around a table discussing an issue with a program being developed. They have a computer with the ABC user interface running. Toggling the participant list of the "19-04-2010" action in the "Monday meetings" activity they see that the people at AU are online and working on the action. They enter the action, and a real-time video and audio channel is opened to the people at AU. As they discuss how to solve the problem, the people at ITU add a figure to the action. The system immediately updates and the figure pops up at the computer at AU. After the discussion they look at the code in the program where they suspect the error to occur. They take turns looking at the code in the shared and synchronised code-editor. Eventually they come up with a few possible solutions they wish to test and they make notes in a log with their findings.

Looking at scenario A played out with ABC support we see that the awareness of other participants, automatic video-conferencing, sharing of resources, shared views and an action log provide the participating meeting members with efficient tools for collaboration. The awareness list starts the meeting at ITU by showing them that the people at AU are online. The real-time video and audio communication lets the users communicate, and as resources are shared a lot of explanation can be cut away and the time instead be used on discussing the important matters. They then end the meeting by leaving information in the action log making it available to all participants of the activity.

ABC scenario B:
Three students (here named A, B and C) sit around a table at the university solving a coding assignment. Some work has already been done and they discuss how to proceed. They have all entered the "Coding" action of the "Data Mining Assignment" activity. Student A shows the others his proposal for a solution by writing his proposal in the shared and synchronised code-editor while explaining it to the others. While student A explains, student B takes notes in the text-editor resource. After student A has finished explaining they all discuss further, and student C adds a few notes in the text-editor. They agree upon a solution, delegate the tasks and start working.

Scenario B played out with ABC support also show the usefulness of the collaborative features. The sharing of resources and views lets the users use their resources in the discussion without having to turn around their screens so the whole table can watch. Also the synchronised text-editor lets the group collect and edit their notes in an easy way.

ABC scenario C:
Two people have been developing a system handles advanced calculations using several computers on a network. One of the people selects the "Test" action of the "Developing" activity and starts the program to stress test it over night. When he leaves he leaves a note in the action log for the other person to notice when he enters the action the next morning.

Looking at scenario C we see that being able to leave notes removes the need for post-its. These log entries are also stored so they can be accessed later in the process.

3.3.2 Communication

The discussion of collaboration and the ABC scenarios showed that there was a great need for the collaborating participants to communicate. Some level of communication should always be
maintained when a user logs onto the ABC framework.

In a real life scenario, working at a hospital, there is always someone around you and we strive to make the communication in the ABC framework the same way. From seeing who is in a room to actually work and talk with them. We want to make the communication levels as buoyant as possible, allowing the users to act as they would do in an everyday situation. To obtain this we wish to implement a communication form with different levels that adjust according to the users work. This can be achieved by making communication more specific as the users work gets more specific. The levels of communication are (in order from most general to most specific):

- **Awareness of other users and their activities**
  When browsing the activity universe the user will be presented with a list of relevant users. The user will have the opportunity to make calls to the relevant users presented, see their location, their status and what they are working on.

- **Awareness of participants on a specific activity**
  When browsing through activities, we want to show who is also participants on a given activity, where they are located psychically and if they are working on the activity at the moment. This can be achieved by expanding the activities in the activity universe to also show a list of participants with relevant information attached.

- **Chat/log by activity**
  When a user selects an action, it can be compared to walking into an operation room or walking a ward round. Therefore we want the user to enter a "room" where he can communicate with the people around him. The room will present an action log, illustrating previous messages, as post-its could do in real life. With multiple users working on the action, the log will function like a chat.

- **Video/audio link**
  If a user selects an action and others are present, real time video and audio will let him talk to the other participants to further emphasise the room-feeling. This real time video and audio however should only begin if the online participants are located remote.

### 3.3.3 Synchronisation

We wish to implement synchronisation to give users the feeling that they are working in the same workspace (as would happen at a meeting around a table) and to give users the opportunity to highlight, move, resize or in other ways use resources to illustrate points when collaborating.

Synchronisation should be done automatically without involving the user. As ABC version 5 already supports synchronisation of activities we focus on implementing state synchronisation. State synchronisation will synchronise the states (location, zoom-factor, scroll bar position etc.) of all operations of an activity. We wish to implement synchronisation to occur automatically when a user works on an activity.

### 3.3.4 Summary

Analysing real world collaboration we have identified important collaborative features. We have applied these to ABC and proposed a list of concrete supporting implementations. These supporting implementation form our unified collaboration model. Figure 3.1 visualises this model and show how these implementations can support the different types of collaboration.
Chapter 3. Activity-based collaboration

Figure 3.1: A CSCW matrix illustrating features to support collaboration.

The features shown have been implemented in the ABC client as we will describe in the next chapter.
4 Implementation

In this chapter we will describe in details how collaboration has been implemented in the ABC framework version 5. This features both synchronisation and each of the four levels of communication described in section 3.3.2.

4.1 Synchronisation

Synchronisation of resources is already present in the ABC framework. Whenever a resource changes (like when adding a new line of code) this change is propagated through the framework and reflected on every client. We have implemented operation state synchronisation, which differs from the aforementioned by only synchronising states, such as position and size, and not the data shown within the operation.

4.1.1 Interaction design

State synchronisation works as illustrated in fig. 4.1. (1) Client A resizes an operation from the upper left corner and thus changes both the operation’s position and size. (2) shows client B’s screen before the resizing, whereas (3) and (4) shows client A and B’s screens after the states has been synchronised.

Figure 4.1: State synchronisation on operations.
Chapter 4. Implementation

Note that it is only the state of the operations which are synchronised. The users can pan around on the screen and thus choose which resources to focus on.

### 4.1.2 Technical design

In previous versions state synchronisation has been handled with session management as described in section 2.3, but in this version synchronisation is handled within the operation itself. As illustrated in fig. 4.2, an operation embeds its own state and the clients will, when subscribing to the activity or action, receive updates whenever the state changes.

![Diagram](https://via.placeholder.com/150)

**Figure 4.2:** The interaction between the activity manager, activity controller and clients when synchronising states.

The activity controller offers the method, `setOperationState`, which takes the operation id and a key value pair as arguments. This method is called from the ABC client as shown in fig. 4.3.

```java
public class OperationState extends AexoObject {
    ... 
    private function setKeyValue(key:String, value:String):void {
        var httpService:HTTPService = new HTTPService();
        httpService.method = "GET";
        httpService.url = host + "/ActivityController/SetOperationState(" + operationId + "," + key + "," + value + ");
        httpService.resultFormat = "xml";
        httpService.send();
    }
}
```

**Figure 4.3:** Setting operation state on the activity manager.

Each operation is visualised through a `ResourceContainer` component and listens, as previously described, for changes in the operation state. In this implementation the operation state only contains information on position and size, but the implementation allows for easy implementation of further state attributes. The actual synchronisation of operations is shown in fig. 4.4.
private function onOperationStateUpdated(e:AexoObjectEvent):void {
    var state:OperationState = e.target as OperationState;
    actionViewLayout.move(this,state.x,state.y);
    actionViewLayout.resize(this,state.width,state.height);
}

Figure 4.4: Actionscript code for state synchronisation of operations.

The implementation uses optimistic state synchronisation, just as the previous versions.

4.2 User awareness list

We have implemented a user awareness list to increase the awareness of other users and to add
the option to call users directly from the activity universe or the action view.

4.2.1 Interaction Design

To give users the opportunity to see other relevant users who are online, we have added a menu
item to the menu, that when toggled displays such a list. The user items shown in the list, contains
information about the user's connection status, location and current workspace. Figure 4.5 shows
how a green or a red circle (1) indicate if the user is online or offline and what action a user is
working on (2).

Figure 4.5: The user awareness list shows the relevant people in the context. It indicates their connection
status, where they are and what action they are working on.

As shown in fig. 4.5, both Morten Esbensen, Søren Nielsen and Jakob E. Bardram are located in
the pITLab, while only Morten and Søren are online and only Morten is working on an action,
being A0.0. This overview gives a collaborative feeling and can be used to initiate communication to other online users. This is done by entering a pie menu connected to each user item in the list.

### 4.2.2 Technical Design

The user awareness list subscribes to a location in the HMap on which it receives updates whenever users are logging in and out, entering and leaving or resuming and suspending actions. This information is then parsed and reflected in the list.

### 4.3 Participant List

As described by the six ABC principles in section 2.1, an activity can be shared among multiple participants. This causes a need for attaching a participant list to each activity giving the user a collaborative overview.

#### 4.3.1 Interaction Design

In the ABC user interface all activities can be selected to show the actions associated with them. We have added another option to show a participant list with all participants, their connection status and which action they are working on if any. Also the participant list is available in the action view.

![Participant List](image)

**Figure 4.6:** The participant list shows the participants on the given activity, including their status, location and what action they are working on.

As illustrated in fig. 4.6, the participant list features information on the participants in the given activity. In this case we can see that both Morten Esbensen and Søren Nielsen are online, although located in pITLab and ITU respectively (1). Furthermore Jakob E. Bardram is offline and located in his office in 4D at the IT University. The list also displays which actions the users are working on, and in this case Morten is working with action A0.0 (2).
4.3.2 Technical Design

The participant list subscribes to an activity in the HMap on which it receives updates whenever the participants are logging in and out, changing location or resuming and suspending actions. This information is then parsed and reflected in the list.

4.4 Real time video and audio

We have implemented a single component to handle real time video and audio communication. The component displays all video-streams, and can be accessed both in the activity universe and the action view.

4.4.1 Interaction Design

The real time video and audio component has a layout that enables it to be placed on top of operations in the action view. It is semi-transparent to make sure it does not hide other resources, and it is moveable and resizeable so users can adjust it to the layout they prefer. Figure 4.7 shows two screenshots of the real time video and audio communication component.

Figure 4.7: Displaying real time video and audio communication between two clients.

Real time video and audio illustrating how two users on different clients can use different resources within the same activity. (1) User A is reading a PDF document, while (2) user B is working with some code in the code editor.

4.4.2 Technical Design

Video and audio communication has been implemented using RTMFP (Real-Time Media Flow Protocol). RTMFP is a protocol that supports peer-to-peer communication between Flash Player and AIR application endpoints [12]. The protocol is built upon UDP [13], which ensures less overhead as well as more time-precise delivery of data. This makes it useful for video and audio communication where delivery time is important.

In order to use the protocol, applications must connect and register to an RTMFP server – we use Adobe’s Stratus 2.0 server. The RTMFP server enables the connection between endpoints by assigning unique ID’s to them. These ID’s are then translated by the server into network addresses, which are used to start the peer-to-peer communication, as shown in fig. 4.8.
Figure 4.8: Peer-to-peer communication controlled by the RTMFP server. Client A, B and C communicate with each other while keeping a connection to the RTMFP server.

The ID's however, cannot be retrieved from the server and the clients must therefore facilitate a method to exchange ID's. This is done by adding the ID as a parameter, when clients register on the activity controller.

Placing and receiving calls

The process of calling and receiving is illustrated in fig. 4.9 where client A calls client B.

When the CommunicationComponent's are created they each publish a listening stream (1-2). These streams will listen for incoming calls. Client B then calls client A (5) by creating a new stream (5.1), that connects to client A's listening stream (6). Both clients then create and publish outgoing streams for sending video and audio (7-12). Then each of them create incoming media streams and connect to each other, to start receiving data (13-16). Each client now has an outgoing...
stream sending and an incoming stream receiving data.

As seen in fig. 4.9 NetStreams are used for transferring data. At least three NetStreams are created by a client – four for the calling client. Published NetStreams send data and are created as shown in Figure 4.10.

```java
outgoingMediaStream = new NetStream(connection, NetStream.DIRECT_CONNECTIONS);
...
outgoingMediaStream.publish("rtvideo");
```

Figure 4.10: A published NetStream for sending data.

Played streams are used to receive data and are created as shown in fig. 4.11.

```java
incomingMediaStream = new NetStream(connection, farID);
...
incomingStream.play("rtvideo");
```

Figure 4.11: A played NetStream for receiving data.

The farID being the RTMFP-assigned ID of the client to be called. This publish-play method provides a simple way to send and receive data using NetStreams.

### Automatic dial-up

When a user enters an action the program will automatically call any other users working on the action if they are not co-located. Calls are placed to all participants on an action that have resumed the action and are online.

### Hanging up

Currently the only option to hang up is to close the video-component. When the component is closed, all outgoing and incoming media streams are closed, all videos are removed and the component itself is hidden. The the listening stream however is kept open so it still is possible to place and accept calls.

## 4.5 Action log

The action log works as a log as well as a chat. All posted messages are saved on a server so they can be reloaded, however they are also sent out to all users connected to the chat in real-time. This way it can act as a simple way to leave notes to each other and as a chat as we know it from other applications.

### 4.5.1 Interaction Design

We have designed action log to look as simple as possible. Figure 4.12 shows our action log. It consists of a text area displaying log entries (1), a text input field and a send button (2) and information about the number of users online on the action (3). Users can enter text in the text field and send it after which the text will be sent to other users as well as saved on the server.
4.5.2 Technical Design

The action log has also been implemented using RTMFP, using its peer-to-peer group communication. The peer-to-peer group communication is used by connecting and registering to the server and creating or joining a group. When connected to a group several methods for sending data directly to other connected clients are available. Using groups also eliminates the need for exchanging ID’s, as the RTMFP server will handle connections and routing when joining a group.

A chat component is created whenever a user selects an action. When the chat component is initialised, it retrieves previous log entries using the IChatLog interface.

```typescript
package dk.itu.activities.ui.components.chat {
    public interface IChatLog {
        function post(action:String, message:String):void;
        function get(action:String):void;
        function get messages():String;
    }
}
```

The interface contains shown in fig. 4.13 three methods that implementing classes need to implement. `post(action:String, message:String)` posts a message to a server. `get(action:String)` downloads previous posts for an action. Once the posts have been downloaded, they can be retrieved using the `get messages()`.

We have constructed a small php-server to handle storing and retrieving log-entries, and we have created a class `HTTPChatPost` implementing the `IChatLog` interface handling the communication. Saved log-entries are saved on the server in .txt files named after their associated action.

To implement sending posts to other users we again use the Adobe RTMFP and Stratus 2 to handle connections. After remote stored log entries are loaded, the connection is established using
the `NetGroup` class which will also handle sending and receiving of posts. The process is illustrated in the sequence diagram below.

```
1: group = new NetGroup(connection, groupSpec)
2: addEventListener(NetStatusEvent.NET_STATUS, onGroupStatusEvent);
3: post(message);
5: onGroupStatusEvent(event);
```

![Sequence Diagram](image)

**Figure 4.14:** Action log sequence diagram

When instantiated the ChatComponent creates a new NetGroup with the net connection obtained from the RTMFPConnection and a group specifier created with the name of the action (1). The ChatComponent registers an event listener with the NetGroup listening for events with logposts (2). The log is now ready for use and can either send or receive posts. On sending the `send()` method of the NetGroup is invoked to send the post other users (3) and the `send()` method of HTTPChatPost is invoked to store the post on a server (4). When receiving an event, the ChatComponent will get the text and add it to its textbox (5).

Each second the log polls the NetGroup for the number of connected users in that group, and updates the display accordingly.
# 5 Testing and evaluation

In this chapter we will test and evaluate our proof-of-concept implementation. It will be tested and evaluated with regards to the goals stated in chapter 1 and the analysis and discussion in chapter 3. We will first evaluate our implementation based on different properties that collaborative systems should demonstrate. To test our collaborative features, we first need some functionality tests to verify that the scenarios defined in chapter 3 are supported. Second we need some usability tests, that gives an indication on whether the design is useful and how it can be improved. Based on the tests and the goals we then discuss to what extend our solution supports collaboration.

## 5.1 Evaluation of collaborative properties

Hall et. al. argue that communication features should address four requirements: reliability, awareness, failure-notification and scalability [11]. In the following we will evaluate to what extend our solution meet these requirements.

### 5.1.1 Reliability

The need for reliability is important as collaborative features are build upon network where loss of packages, dropping connections and routing problems can be expected to occur. Our real time video and audio implementation is reliable if it is able to deliver data in time. This is achieved by using RTMFP which is built on UDP to deliver data. We therefore believe the video communication is reliable in its work. Synchronisation and exchange of ID’s to communicate have been implemented using the HMap based on AEKO, for which the reliability is a key feature [4]. In all we believe that our features supports the requirement of reliability.

### 5.1.2 Awareness

Sharing resources puts a need on the system to be able to provide awareness to users to let them know who they are working with. In the ABC environment users will resume and suspend actions and other users should always be aware of these changes. We have implemented a user awareness list and an participant list to support this. Furthermore video-communication will always be activated when users are working on the same action. These features we believe provide the necessary support for awareness.

### 5.1.3 Failure notification

As things can go wrong when communicating over networks it is important for users to be notified if things do go wrong. This property mostly relates to changes in resources where users should know if any change is not correctly propagated to others. Our collaborative features are more failure tolerant as they do not affect important information. The loss of video or audio due to a temporary connection drop is more acceptable than a loss of a resource change. The log entries however, are more important to save as these may contain important information that needs to be
stored. We have not implemented failure notification for the log, but we agree that this should be done.

5.1.4 Scalability

Collaborative systems should be able to scale to the number of users expected to use the system. In our case a possible bottleneck would be the network speed when many people are videoconferencing which would cause many media streams to be handled by the program. We do not believe that the system would be used in massive video conferences, and thus requiring a lot of media streams. And therefore we believe that scalability will not be a problem within the reasonable.

5.1.5 Results

We evaluate our solution to meet the requirements mentioned. The action-log should however implement failure notification.

5.2 Functionality test

First we need to make a series of functionality tests. We do this to verify, that our implementation works as intended. If our tests is all successful we can continue with our usability tests. If they are not, we have to correct the bugs in order gain the best possible feedback from our usability tests. The features found relevant in chapter 3 are tested on its functionality. This means we have to test the functionality of people context list, participant list, state synchronisation, action log and video and audio chat. These features are tested in different ways, but all with both one and multiple clients to ensure that they scale.

5.2.1 Tests

In this section we will explain exactly what we want to test and how to do it. Each test will contain a number refering to the results listed in fig. 5.1.

User awareness list requires two tests. (1) We need to test if the user awareness list works when people are entering or leaving a location. This is done by starting up the framework and then tag some people with RFID tags. Then we will walk in and out of the location and see if this is reflected correctly in the user interface. (2) We need to test if the list is updated, when a person is logging in or out of the system and resuming/suspending an action. This is done by letting a person log in and out on one device, and have another device watch the status change in the user interface. This method is also valid for testing when a person resumes an action.

Participant list requires two set of tests. (3) We need to test if the participant list is reflecting the online/offline status of the participants. (4) We also need to test if the participant list is reflecting which actions the participants currently are working on. Both of these can be done by letting one user login, resume an action, suspend it again and finally logout, while another user keeps an eye on the activity.

State synchronisation requires two tests. (5) We need to test if a state is saved and loaded when suspending and resuming an action, respectively. This is done by resuming an action, moving the operations around, suspending it, and then resume it again. (6) We need to test if a state is synchronised when two users are working on the same action. This is done by moving an operation on one device and see if the other device moves it accordingly.
Action log requires three tests. (7) We need to test if an action log is loaded correctly when resuming an action. This is done by resuming any action and see if the previous log is loaded. (8) We need to test if an action log is updated correctly when a new message is entered. This is done by entering a message and see if the message is shown in the action log. (9) We need to test if an action log is synchronised when two users are working on the same action. This is done by resuming the same action on two different devices and let the first device write a message. If the message appear on the second device it works.

Real time video and audio requires five tests. (10) We need to test if the real time video and audio appears when two users are working on the same action. This is done by resuming the same action on two different devices at two different locations with webcams. If real time video and audio is initiated automatically it works. (11) We need to test if the video and audio disappears when one of the two users are leaving an action. In continuation of test 6 we then try to let one of the devices suspend the action. This should, if everything works, make the video and audio disappear. (12) We need to test if a user can manually disable the video chat. This is done by closing the video and audio component. (13) We need to test if an audio chat appears, when no camera is found. This is done by letting two devices – one with and one without webcam – resume the same action. The video and audio component should appear, but saying that "No video is available" and with working audio. (14) We need to test if video and audio can handle more than two users and thus work as a conference. This is done by letting three or more users resume the same action.

5.2.2 Results

In fig. 5.1 we have listed the test results. The five components have been tested in different ways using one or more users.

<table>
<thead>
<tr>
<th>Component</th>
<th>Test</th>
<th>Users</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) People context list</td>
<td>Entering/leaving</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>(2) People context list</td>
<td>Online/offline</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>(3) Participant list</td>
<td>Login/logout</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>(4) Participant list</td>
<td>Resume/suspend</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>(5) State synchronisation</td>
<td>Save &amp; load</td>
<td>1</td>
<td>✓</td>
</tr>
<tr>
<td>(6) State synchronisation</td>
<td>Synchronise</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>(7) Action log</td>
<td>Load</td>
<td>1</td>
<td>✓</td>
</tr>
<tr>
<td>(8) Action log</td>
<td>Update</td>
<td>1</td>
<td>✓</td>
</tr>
<tr>
<td>(9) Action log</td>
<td>Synchronise</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>(10) Video chat</td>
<td>Video appears</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>(11) Video chat</td>
<td>Video disappears</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>(12) Video chat</td>
<td>Disable chat</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>(13) Video chat (no camera)</td>
<td>Audio appears</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>(14) Video chat</td>
<td>Conference chat</td>
<td>3+</td>
<td>✓</td>
</tr>
</tbody>
</table>

Figure 5.1: Results of the functionality tests

As the results show, every functionality but one is working as intended. Automatically updating location using RFID and bluetooth is not possible with the current version of ABC, as it fails to notify listeners. We will however work around this problem as we describe later. This allows us to continue our testing with usability tests.
5.3 Usability test

With the successful functionality tests, we can now proceed with the usability testing. The usability tests will consist of a few collaborative scenarios that we will have test persons play. According to Cem Kaner, a good scenario test is motivating, credible, complex and easy to evaluate [14]. Therefore we target our scenarios at co-students from the IT University of Copenhagen and design our tests around activities they are familiar with. The scenarios though, will still include some complex tasks, that forces the test persons to collaborate.

The scenarios will try to investigate the improvements and regression achieved by the activity-based collaboration. What helps improving the collaboration, what is unnecessary and what can improve the collaboration even more?

We gather results from the usability tests in two ways: questionnaires and interviews. After each test the test persons are asked to fill out a questionnaire about the experience. The questionnaire consists of a series of statements in which the test persons can answer to what extend he or she agrees. The statements, as shown in appendix B, are divided into categories, however only questions relevant to the test person are presented – and they are presented without category labels and in random order. After the questionnaire we ask the test persons to evaluate the setup and describe what they find to be good and what could be improved.

To complete the following tests the users needs some knowledge about the general concepts of activity-based computing and activity-based collaboration as well as some knowledge about the functionality supporting this. As opposed to a traditional file-application based understanding of computers, ABC may not be intuitive for the test persons. This we believe also will be the case in a future deployment of an activity-based computing system, where some teaching and training of users also will be needed. Before we conducted the tests, the users were explained about the concept of activity-based computing, and they were told about the functionalities supporting collaboration.

5.3.1 Test setup

The tests were carried out at the pITLab at the IT-university of Copenhagen. The hardware consisted of: a desktop computer with a large touch display for one user, a laptop for the other user and the infrastructure was hosted on another laptop. The participants were in the same room however faced back to back and with space between them. As RFID location changing did not work we wrote a small program which we used to manually change locations of users during the tests. The proof-of-concept implementation was tested with three "teams" of two persons.
5.3.2 Tests

We have designed the scenarios to reflect a group project about developing a website. The activity will be called "Website" and it will contain different actions such as "Frontpage design", "Contact form" and "Briefing". The scenarios will test various aspects of collaboration, which we emphasise by adding resources that multiple users can work with at the same time. We have designed six scenarios that will test co-located collaboration, remote collaboration, remote to co-located collaboration, asynchronous collaboration, unaware collaboration and manually initiated collaboration.

We have written an introduction to each test person in the scenarios, so they now exactly who they are, what their objective is and how they can accomplish it.

Co-located collaboration will be testing how activity-based collaboration can be used when two or more persons are co-located as described in section 3.2.1 on page 11. This scenario in specific deals with two test persons, where one of them is a brand new project member, that needs an introduction to the project. The other test person will act as the experienced project member and will try to introduce the project for the new member using his laptop and a multitouch wall.

User A objective: You are a new member of a project group, that works on a web development project. Today you are attending a briefing, where another member will introduce the project and give you an idea of how he want the website to work. Your objective is to understand the project.

User B objective: You have been working on a project for a while and have to brief a new project member. You are a participant on the activity "Web development" and in the action "Briefing" you have arranged a set of resources, to use in your briefing. There are some diagrams and some design images that shows how you want the website to work and look. You have started the ABC client on your laptop and on the multitouch wall. The action is resumed on both devices and from your laptop you can control the resources on the multitouch wall as well.

Remote collaboration will test if and how activity-based collaboration is useful when users are
remote from each other as described in section 3.2.1 on page 11. This scenario deals with two test persons, who are supposed to collaborate in order to meet their objectives.

User A objective: You are a participant on the activity "Website". Your objective is to build an html-form in the action "Contact form". You will have to collaborate with user B and agree on how to do this. It should all be done in the same file. When done leave a note in the action-log, that states the job has been done.

User B objective: You are a participant on the activity "Website". Your objective is to build a php-script in the action "Contact form", which can save inputs from an html-form. You will have to collaborate with user A and agree on how to do this. When done leave a note in the action-log, that states the job has been done.

Remote to co-located collaboration will test collaboration where users initially are remote, but end up being co-located as described in section 3.3.2 on page 13. This scenario deals with two users who have arranged a meeting. One of the users are late, but have a mobile device. He talks to the other user, and when they are co-located their video communication is disconnected.

User A objective: You have an arranged meeting with user B, but are late. While you walk towards the meeting room you resume the action "Meeting 14/05-2010" in the activity "Website". You start the meeting while still walking towards the meeting room and when you arrive the video communication is disconnected.

User B objective: You have arranged a meeting with user A and have prepared the action "Meeting 14/05-2010" in the activity "Website". User A is late, but while he walks towards the meeting room, he resumes the activity, so you can start the meeting to avoid wasting time. When he arrives the video communication is disconnected.

Asynchronous collaboration will test the collaboration between two or more users resuming and suspending the same action over time, but never at the same time. This scenario has been described in section 3.2.2 on page 12. In this scenario we are only testing with one test person as the scenario depends on previous work. The scenario in this case is a Java class that throws an exception. Our test persons task is to fix the bug.

User objective: You are a participant on the activity "Java development". You have been assigned to fix a bug in a Java class, situated in the action "Bug fixes". Another participant has left a note in the action log describing the problem and your objective is to fix it and explain the problem in the action log when the problem is fixed.

Unaware collaboration serves the purpose of testing how activity-based collaboration can avoid duplication of work. In this scenario two users are given the same task, but they do not know. As video communication is initiated they can either chose to collaborate or just agree on who is doing it. In this scenario both test persons will have the same introduction.

User A and B objective: You are a participant on the activity "Website maintenance" and has been told to update the text on the frontpage, which is situated in the "Frontpage" action. When done, leave a note in the action log.

Manually initiated collaboration will be testing the support for direct communication between two test persons. The test persons are respectively online and working on an action, but they are both participants on the action. The second user need help to understand a paragraph in the documentation and therefore calls the first user, who were the one writing it.

User A objective: You are working on the activity "Java development", and in the action "References" you see that a reference is missing. In the action log you can see who wrote the documentation and call to ask what is missing.

User B objective: You wrote a paragraph in a documentation and a user calls you to tell that a reference is missing. You will ask him which activity and action it is concerning and help him find out what reference is missing.
5.3.3 Results

In this section we will present the results of our tests, and compare them with our expectations. If some results differ from the expected we will try to give an explanation of why that happened. The results will be divided into three categories; the questionnaire answers, the interview comments and the observations made during the tests. As we present the results we will also discuss our solution.

Questionnaire answers

When looking through the questionnaire results (see appendix B), the general feeling is that the test persons were well-disposed towards activity-based computing and its collaborative possibilities. In fig. 5.3 we have translated the questionnaire answers into a positive-negative schema, illustrating the general feeling of each subject.

<table>
<thead>
<tr>
<th></th>
<th>Positive</th>
<th>Neutral</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usefulness</td>
<td>28+12</td>
<td>8</td>
<td>0+0</td>
</tr>
<tr>
<td>Ease of use</td>
<td>18+12</td>
<td>2</td>
<td>0+1</td>
</tr>
<tr>
<td>Real time video and audio</td>
<td>18+2</td>
<td>0</td>
<td>0+0</td>
</tr>
<tr>
<td>Action log</td>
<td>4+6</td>
<td>0</td>
<td>0+3</td>
</tr>
<tr>
<td>Synchronisation</td>
<td>2+2</td>
<td>0</td>
<td>10+3</td>
</tr>
<tr>
<td>People awareness list</td>
<td>6+4</td>
<td>3</td>
<td>0+1</td>
</tr>
<tr>
<td>Participant list</td>
<td>8+3</td>
<td>2</td>
<td>2+0</td>
</tr>
</tbody>
</table>

Figure 5.3: A schema showing the test persons general happiness with the test subjects. Totally agree answers has been given 2 positive points, partially agree 1 positive point, neutral 1 neutral point, partially disagree 1 negative point and totally disagree 2 neutral points.

The schema gives a clear indication that the test persons found the collaborative features really useful and easy to use, with respectively 83% and 91% of the points being positive. The real time video and audio scores a full 100% positive points and especially the automatic dial-up was a success with only totally agree answers in the questionnaire.

The action log, people awareness and participant list gets 77%, 71% and 73% positive points respectively and even though some negative answers has been given, the feedback must be considered as mainly positive.

With 76% negative points, synchronisation really seems to be a negative experience, but the result is somewhat expected. As the test was performed on different devices, the test persons had to arrange their resources within the limits of the screen resolution. Although some of the test persons tried to agree on a placement of the resources, others found it fun to drag the resources around and thus ruin it for the other person. Not exactly a collaborative spirit.

Interview comments

The test persons in general gave positive feedback and found the concepts of ABC and the collaboration within it interesting. The real time video and audio was without a doubt the most successful feature based on the interviews. All the test persons was impressed with the idea of automatically initiating communication with video and audio, when entering the same virtual workspace. But the general impression was also that it in some occasions might be a little intrusive. Most of the test persons suggested that the user should be presented with an opportunity to answer or ignore a call or to disable video and only communicate with audio only.

Synchronisation was subject to a lot of comments and one of the recurring points, was that it did not scale well enough to support collaboration on different devices. The main agreement made on this subject though, was that for synchronisation to work, the participants would have to
work towards each other. This is an important factor, but nonetheless a common user behaviour according to Greenberg and Roseman [6]. The test persons suggested a zoom feature, giving an overview of the action's resources, to make synchronisation work.

The test users found the awareness list and the participant list to be somewhat useful however awareness was not clear enough. All lists had to be selected for the users to see them, and the focus of their attention was the synchronisation and real time video and audio communication. The awareness of other users should be made more clear in the user interface.

The action log was not a definitive success with the test persons. It was definitely unclear to the users if it was a chat or a log. The general attitude was to split the action log into two components and not mix chat and log. Furthermore should the chat be coupled with the real time video and audio and thus function as a way to share information, links, etc. without being stored in a log file. Another proposal regarding the action log was to ask the user when suspending an action, whether any comments should be added to the log. This functionality was compared to committing data to a revision control system.

Another interesting finding was the need for pinning communication. When two users are collaborating on an action and switches to another in order to continue their work, the real time video and audio will briefly disappear. And if only one of them are switching action, the real time video and audio will disappear until they again are collaborating within the same action. This is not appropriate behaviour and the system would definitely benefit from a “communication pinning” feature.

Observations

Throughout the tests we observed and recorded interesting issues and unintended use of the system. One of the first things that jumped out at us was the need for telepointers. Some of the users were expecting some kind of knowledge of what their counterpart were doing. They had a hard time finding out, that only the operations was synchronised as they expected the whole screen to be synchronised. This, of course, with no regards to the screens being in totally different resolutions.

Again the real time video and audio worked as intended. Communication between the test persons was good and it certainly seemed to help them in their collaborative tasks. It was not clear if the real time video and audio actually helped or if audio would suffice. But either way it seemed like the test persons found it very useful, as our interviews also substantiates. One of our test scenarios forced the test persons to guide the other participant into a specific action. This proved to be very intuitive, but some test persons wondered if there was a possibility to share the action directly through the communication channels. This could prove to be a time-saving feature, that would allow the participant to go directly to the action view without browsing the activity universe for the right activity and action.

Another scenario asked the two test persons to complete the exactly same task and thus force them to communicate and collaborate. Although not all of the test persons figured out they actually were given the same task, some managed to find out and they again praised the automatically dial-up for this. But the scenario really expanded the need for roles within the activity. If the test persons had been given specific roles they would have had a much better chance of knowing what the other participant were working with. If they have had the same roles, they might even have guessed that their tasks similar.

The participant list and the people awareness list was not used as much as expected. The placement in the menu bar seems to be hampering the use of these. The awareness aspect in ABC really should be in the user's awareness, but being placed in the menu bar seems to make the user unaware of its existence. A solution to this could be to use notifications indicating that someone or something entered or left a location, changed connection status, switched workspace, etc. It is important to have these awareness components, keeping the user aware of them, otherwise they will not be awareness components.
6 Conclusions

In this thesis we have discussed activity-based collaboration, constructed a uniform collaboration model, developed a proof-of-concept and tested and evaluated this. This chapter will summarise any interesting findings and conclude the thesis. Finally some future work within activity-based computing will be presented.

When defining our thesis goals we wrote that:

“In order to support collaboration the ABC framework needs to facilitate synchronisation of activity states and communication between users.”

We discussed collaboration in a real world situation where we used location and time as parameters to define different types of collaboration. Using time we distinguished between synchronous collaboration and asynchronous collaboration, and using location we distinguished between remote collaboration and co-located collaboration. We then argued that these types showed different characteristics and therefore should be addressed differently. We generalised the features needed to support collaboration into two types; communication and synchronisation and we then discussed how to support these in the activity-based computing. Based on the discussion we proposed a series of implementations to support collaboration in the ABC framework. These implementations form our unified collaboration model.

Secondly we build a proof-of-concept upon the ABC framework. The implementation consisted of four UI widgets: user-awareness list, participant list, real time video and audio and an action log as well as synchronisation of operations. We documented this proof-of-concepts by explaining all the features, their interaction design and their technical design.

We then tested and evaluated our solution. We evaluated our solution based on reliability, awareness, failure notification and scalability and found that our solution meets these requirements, with the exception of the action log not meeting the failure notification requirement. We tested functionality by using all the implemented components and verified that they worked as we expected. We then tested usability by designing test scenarios and had test persons play them out. The test scenarios were designed to be collaborative and the overall concept was programming assignments – a subject our test persons were very familiar with. We evaluated our solution based on these tests, through questionnaires, interviews and observations. The results from the usability tests showed a general satisfaction with our solution and the tests persons were specifically positive about the automatically initiated video-communication. The solution was found very useful and easy to use, by the test persons, who also seemed positive about the user awareness and participant list. Synchronisation however were not well received, but the test persons lack of willingness to collaborate must be seen as a part of the reason for the bad results.

We then evaluated our solution based on the test results. We concluded that the collaborative features we had implemented worked and that they were well received by test users. Synchronisation was the feature with the lowest satisfactory score in the usability test and we agreed that in order to support collaboration well this feature needs additional supporting features.
Both synchronisation and communication is supported by our proof-of-concept solution and even though there are room for improvement, the main goals within the implementation has been achieved. The concept was well received and many ideas and improvements was suggested by the test persons. This gives reason to suggest future work within activity-based collaboration in the ABC framework.

6.1 Future work

There are a lot of future work to be done on this subject and we have introduced quite a few ideas in section 5.3.3. The following list summarise the ideas for improvement of the ABC framework version 5.

- Disabling synchronisation
- Zoom functionality
- Pinning communication
- Divide action log into chat and log
- Integrate text chat with video and audio chat
- Make the user decide whether to receive or ignore a call
- Disabling video, and use audio only chat
- Make log comments like revision control system comments
- Implement telepointers
- Make the user more aware of awareness lists
Bibliography


Appendices
Our initial thesis name was "Activity-Based Remote Collaboration" however as we investigated the subject we found that it was important not only to explore remote collaboration but also co-located collaboration. We have changed the name and problem to reflect this change of focus.

Our initial project problem which also is the one listed in the project base was:

Activity-based computing (ABC) is a new paradigm, in which the basic computational unit is expressed as an activity of a user rather than a file or an application. The ABC Framework version 5.0, which implements activity-based computing, is currently being developed by Jakob E. Bardram and Jonathan Bunde-Pedersen. A basic tenet within ABC is that users can share any activity, resource or service they may need. This may span a single building (like a hospital) or the globe (like from Denmark to India). The framework however does not support remote collaboration and therefore needs to be extended.

In order to support remote collaboration the ABC Framework needs to facilitate synchronisation of activities and communication between users. One problem is to determine what level of synchronisation is needed and how to obtain that. Another is to ensure efficient communication through different communication levels. This project is aimed at investigating these concepts and implement a proof-of-concept using the ABC Framework version 5.0.

Our new problem is as follows:

Activity-based computing (ABC) is a new paradigm, in which the basic computational unit is expressed as an activity of a user rather than a file or an application. The ABC Framework version 5.0, which implements activity-based computing, is currently being developed by Jakob E. Bardram and Jonathan Bunde-Pedersen. A basic tenet within ABC is that users can share any activity, resource or service they may need. This may span a single building (like a hospital) or the globe (like from Denmark to India). The framework however does not support collaboration and therefore needs to be extended.

In order to support collaboration the ABC Framework needs to facilitate synchronisation of activities and communication between users. One problem is to determine what level of synchronisation is needed and how to obtain that. Another is to ensure efficient communication through different communication levels. This project is aimed at investigating these concepts and implement a proof-of-concept using the ABC Framework version 5.0.
# B Questionnaire with results

<table>
<thead>
<tr>
<th>1. Usefulness</th>
<th>TA</th>
<th>PA</th>
<th>N</th>
<th>PD</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The system will improve my collaboration with others</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The system will make me want to do more collaborative work</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The system will make me more efficient in my work</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The collaboration in the system was too much</td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>The system has the features needed to collaborate</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The system provides me with good alternatives compared to my normal tools</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. Ease of use</th>
<th>TA</th>
<th>PA</th>
<th>N</th>
<th>PD</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The system is easy to learn</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The system is easy to use</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The features are easy to understand</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>It was easy to initialize communication with others</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3. Real time video and audio</th>
<th>TA</th>
<th>PA</th>
<th>N</th>
<th>PD</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The real time video was a good way to communicate</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The automatic dial-up was useful</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. Action log</th>
<th>TA</th>
<th>PA</th>
<th>N</th>
<th>PD</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The action-log was a good way to leave notes</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The text chat in the action log was a good way to communicate</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5. Synchronisation</th>
<th>TA</th>
<th>PA</th>
<th>N</th>
<th>PD</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared state improves collaboration</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>An option for disabling synchronisation should be available</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6. People awareness list</th>
<th>TA</th>
<th>PA</th>
<th>N</th>
<th>PD</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The people awareness list was useful when browsing activities</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>It was easy calling people from the people awareness list</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7. Participant list</th>
<th>TA</th>
<th>PA</th>
<th>N</th>
<th>PD</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The participant list increases the collaborative feeling</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The participant list was useful for finding people to ask questions</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure B.1:* The abbreviations in the questionnaire refers to the following; TA = Totally agree, PA = Partially Agree, N = Neutral, PD = Partially Disagree, TD = Totally Disagree. Note: Not all questions was answered by all test persons.