Mobile Support in CSCW Applications and Groupware Development Frameworks

http://dx.doi.org/10.3991/ijim.v7i2.2469

David Johnson
University of Oxford, United Kingdom

***Abstract*—Computer Supported Cooperative Work (CSCW) is an established subset of the field of Human Computer Interaction that deals with the how people use computing technology to enhance group interaction and collaboration. Mobile CSCW has emerged as a result of the progression from personal desktop computing to the mobile device platforms that are ubiquitous today. CSCW aims to not only connect people and facilitate communication through using computers; it aims to provide conceptual models coupled with technology to manage, mediate, and assist collaborative processes. Mobile CSCW research looks to fulfil these aims through the adoption of mobile technology and consideration for the mobile user. Facilitating collaboration using mobile devices brings new challenges. Some of these challenges are inherent to the nature of the device hardware, while others focus on the understanding of how to engineer software to maximize effectiveness for the end-users. This paper reviews seminal and state-of-the-art cooperative software applications and development frameworks, and their support for mobile devices.**

***Index Terms*—CSCW, MCSCW, groupware, software development frameworks, mobile applications**

I. INTRODUCTION

The term *groupware* (a fusion of the words ‘group’ and ‘software’) was first coined in the early 1980s by Peter and Trudy Johnson-Lenz, and the then emerging Computer Supported Cooperative Work (CSCW) community adopted this rubric to try and describe computer applications to support collaboration[1]. There was a broad consensus on what the aim of CSCW research was: to explore how technology can enhance the way people work together. But Jonathan Grudin reviewed the different approaches to the research field in his 1994 IEEE Computer magazine article, ‘Computer Supported Cooperative Work: History and Focus’, and made the following observation of the challenges of the day:

“If we think of CSCW as an emerging field or common enterprise, we may be frustrated by this mosaic of different pieces, the frequent misunderstandings, and the lack of intellectual coherence. But when understood and respected, the differences form the core of richer, shared understandings.” [1]

What Grudin highlighted was that despite the consensus on what CSCW research broadly entails, the approaches to the research vary between academia and business; between researchers rooted in different disciplines, and even on a cultural level as European, North American, and Asian research each have their distinct modus operandi. Since Grudin’s review the topic areas and approaches within CSCW research have not become any better defined. In fact, an ecological analysis in [2] of the research field itself had shown that there has been a trend showing a high churn rate of and slight decline in number of authors contributing to CSCW research literature since the end of the 1980s. They further conclude that CSCW’s independence from the field of Human Computer Interaction (HCI) may rely heavily on a small subset of established researchers. From their analysis they propose that perhaps the high churn exhibited may be due to a lack of consensus on core questions in CSCW and limited room for new research directions.

As technology evolves, new opportunities for research arise. Roy Want and Trevor Pering have identified some of the opportunities that ubiquitous and pervasive computing systems present, afforded by increasing storage capacities, network bandwidths, and environmental sensing available to mobile devices [3] [4]. Although still a far cry from Mark Weiser’s seminal vision of computers becoming, *“an integral, invisible part of the way people live their lives”* [5], mobile computing goes some way to fulfilling Weiser’s prophecy. Mobile devices can be so closely coupled with their owners some might consider everyday life without them unthinkable. Mobile devices *are* an integral part of people’s lives, if not (yet) an invisible part.

The rest of this paper goes on to discuss and define Mobile CSCW, review a number of groupware applications and development frameworks, and finally discuss motivations for further research in the development of interactive, mobile, cooperative software.

II. MOBILE COMPUTER SUPPORTED COOPERATIVE WORK

First coined in 1984 by Greif and Cashman, the term ‘computer supported cooperative work’ (CSCW) was used to describe a new branch of research into how technology could be used to benefit the work environment [1]. However there has been much debate on a unified definition of CSCW [6]. Grief defines CSCW as,

“...an identifiable research field focused on the role of the computer in group work”. [7]

This definition places emphasis on identifying how computing technology fits into the processes and organization of groups. Bannon and Schmidt propose an alternative definition that focuses on understanding how people cooperate and designing technology accordingly:

“CSCW should be conceived as an endeavor to understand the nature and characteristics of cooperative

[1] *CSCW* defines the research area; *groupware* defines the technology itself.

work with the objective of designing adequate computer-based technologies". [8]

There have been numerous attempts at defining what is CSCW and what the primary aims are, but the two aforementioned definitions highlight the essential distinct elements: *supporting* group work and *designing* computer-based technologies. Schmidt discusses how CSCW should aim to fit into the cooperative work process, and not define new processes or techniques. He also identifies that CSCW is a design-oriented research area. As such, he proposed a more stringent definition.

"...an endeavor to understand the nature and requirements of cooperative work with the objective of designing computer-based technologies for cooperative work arrangements". [6]

Schmidt purports that CSCW should be focused on designing technology to support group work. To this end, CSCW research investigates determining the nature of cooperation, work, and group dynamics, with a significant proportion of researchers contributing software engineering approaches to developing groupware. Designing the technology to support cooperative work is not a trivial task, hence the establishment of CSCW as a research field in its own right.

Grudin discussed how group support was approached from the seminal years of mainstream computing [1]. He illustrated a trend in US R&D CSCW and groupware contexts that follows the technological advancements of computing hardware. From the 1960s mainframe computers provided support in the workplace via batch data processing and management information systems. The 1970s saw the prevalence of interactive minicomputers and networked systems, where office automation emerged as a solution to supporting large groups and projects. The advent of HCI research in the 1980s was largely driven by personal computing. In the 1990s this progressed to the popularization of CSCW with the Internet becoming more accessible and connecting end-users at work and in the home. The mobile device has characterized the most recent decade. Widespread wireless access to the Internet and the convergence of digital technologies make modern mobile phones as functional as (or some might argue more functional than) PCs. Each paradigm shift was driven by popularization of the technology of the day. By extending the trend Grudin identified one can infer that *Mobile* CSCW[2] is the next natural step. What can be expected beyond the age of the mobile device? The task of finding applications of sensor networks is currently fuelling ubiquitous computing research. How Mobile CSCW develops beyond today's ubiquitous devices is an open question for future research.

III. GROUPWARE APPLICATIONS

The field of CSCW and the concept of groupware cannot be traced to a single seminal article or product. Although the term CSCW was first coined by Grief and Cashman, as already discussed, it was just simply label that could be applied to existing research. Since then, there have been numerous research projects and commercial products that are considered groupware. This section reviews a number of seminal and state-of-the-art groupware applications.

A. *Basic Support for Cooperative Work*

Basic Support for Cooperative Work (BSCW) [10] is a Web-based groupware application originally developed by the Fraunhofer Society[3]. Today it is maintained and marketed by a spinoff company, OrbiTeam Software GmbH. The original motivation of the BSCW project was to develop cross-platform groupware where the authors identified the Web as a means to providing the functionality independent of the main end-user operating systems. This section discusses the original BSCW research before it was developed into a commercial product.

BSCW was designed around the concept of *shared virtual workspaces*. A workspace can be considered as an object store for collaborative work that provides an awareness agent that allows users to monitor activity within the workspace. These workspaces are accessed through a user's Web browser. Shared workspaces host a variety of multimedia including documents, pictures, videos, Web bookmarks, and group discussions. The contents of a workspace are organised and rendered in the user interface (UI) as a hierarchy of files and folders. Events are broadcast to all users whenever a single user performs an action within the workspace. This awareness framework allows users to keep track of the activity of the group with respect to workspace objects. Examples of events might include uploading or downloading documents, updating shared bookmarks, or contributing to a discussion topic. Event histories are personal to each user.

BSCW was built on a standard Web server to provide the functionality for the shared workspace through standard HTML Web pages. The Web server also acts as a server for Java clients that provided a rich UI to the shared workspaces. This allows end-users who had Java capabilities to use the BSCW system with a rich UI instead of relying on the Web interface. The BSCW server communicates with Java clients using XML. A tool to broadcast real-time awareness information between end-users' Web browsers is provided that augments the workspace UI: a Java-based application called a *monitor applet*. A secondary server called the *event server* connects with end-users' running monitor applets using a custom protocol.

Apart from the shared workspace and event broadcasting, BSCW provides a number of other features including user authentication, document versioning, access rights, workspace search, document format conversion, document annotation, integration with synchronous tools such as audio/video conferencing and shared whiteboards, and calendaring. BSCW can be considered as the de facto seminal groupware system due to its widespread adoption by academia and subsequent commercialization.

B. *Access Grid*

Initially developed by Argonne National Laboratory, Access Grid [11] (AG) is a set of technologies and supporting infrastructure to enable multimedia

[2] *Mobile CSCW* is considered as a subset within CSCW and not its own new field; the term *mobile groupware* is used to describe Mobile CSCW software applications.

[3] The Fraunhofer Society is a German research organization that focuses on applied science [9].

collaboration between large groups of geographically dispersed end-users. AG itself is not a single piece of software, but rather a collection of both hardware and software resources that include large-format visual displays, immersive virtual environments, and means to access shared Grid computing resources. AG is also a Grid itself (i.e. as in Grid computing [12]) in which participating sites share their network and hardware resources. Widely used in the international scientific community, as of 2009 AG consists of over 300 registered nodes across 30 countries. The software itself is free and open source.

Like BSCW, AG focuses on the shared virtual workspace concept. However, unlike BSCW, it is based around synchronous workspaces in which end-users interact in real-time. The primary aim is to create computer-augmented environments to facilitate natural audio and video group communications. AG differs from the traditional approach to CSCW systems in two ways. Firstly, AG aims to create environments for small groups of users rather than single users at any one endpoint (i.e. a group of users at one location communicates with groups of users at other locations, as opposed to many single users communicating with each other). To this end, part of the research and development behind AG focuses on creating room-based media conferencing facilities with large displays to encompass multiple users at a single site [13]. Secondly, AG uses the concept of *persistent virtual venues*. A virtual venue can be thought of as a location on the Internet that AG users can find and access. Each virtual venue hosts a number of *virtual rooms* that AG users can join and through which they can interact with others.

At a particular site, the site manager hosts an AG node. An AG node hosts all of the audio-visual services and connects the site to the wider network. Nodes are dedicated room-based conferencing facilities, immersive virtual environments, or 'personal' nodes for single users using a desktop computer. Room-based nodes may consist of multiple computers to host the audio/video services. An AG node runs several applications including the AG Venue Client, ViC (Video Conferencing) [14] and RAT (Robust Audio Tool) [15]. The Venue Client connects to a Venue Server, and it is the server which provides services to persist objects (data, documents etc.), authorize and authenticate users, and assign multicast addresses to connect AG nodes running ViC and RAT. Other synchronous collaborative tools have been developed and integrated with AG including a shared presentation tool, shared Web browser, and a shared whiteboard (TigerboardAG, [16]).

Since AG is designed to utilize high-performance networks and large-format multimedia displays, it provides no support for mobile collaboration.

C. SOGo

SOGo (formerly known as Scalable OpenGroupware.org) [17] is an open-source standards-based groupware server. It was originally developed by Skyrix Software AG and is based on source code released by the company in 2003. The main concept of SOGo is to provide a groupware solution that integrates with existing tools and processes in business collaborations. By developing support for industry standard network protocols and data exchange formats, SOGo provides groupware functionality independent of end-user client software. SOGo provides services to share a range of organizational data including calendars (including events and task lists), personal contacts and email boxes.

Information can be organized and shared with groups of users or kept private. Calendaring information (calendars, events, and tasks) is stored using the iCalendar standard, while contact information is stored in the vCard format. Clients exchange data with the SOGo server using Web-based Distributed Authoring and Versioning (WebDAV) based protocols. WebDAV is an extension of HTTP that allows clients software to manipulate files stored on a server. SOGo uses Calendaring Extensions to WebDAV (CalDAV) (for exchanging iCalendar data and vCard Extensions to WebDAV (CardDAV[4]) for vCard data. SOGo also supports Storage of Groupware Objects in WebDAV (GroupDAV[5]) which aims to be a groupware-specific data exchange protocol. GroupDAV encompasses both iCalendar and vCard formats and provides a richer command list than CalDAV and CardDAV. IMAP is used for providing email services as it is widely supported by a range of email clients.

SOGo provides mobile support in the form of a service to synchronize PIM[6] data between the SOGo server and an end-user's mobile device. SOGo connects to mobile devices using the open source Funambol mobile synchronization middleware [18]. Funambol communicates with mobile devices using the platform independent synchronization protocol, SyncML. By utilizing platform independent standards, SOGo provides groupware services to a wide range of desktop and mobile platforms. Apart from integrating with existing client software, the server provides a Web-based interface that emulates rich UIs provided by native clients. Focused on integration with end-users' legacy collaborative processes, SOGo's groupware functionality is based on asynchronous communication (via email), sharing data, and centralized coordination tools.

D. Microsoft Office Groove

Microsoft Sharepoint Workspace [19] is a groupware application that is integrated with Microsoft's Office suite, and previously popularly known as Microsoft Office Groove [20]. Originally developed by Groove Networks Inc., Microsoft acquired Groove Networks and its products in 2005. In this section we discuss Office Groove, which is now integrated into the Sharepoint Workspace product. Office Groove aimed to be a groupware system that facilitates dynamic collaborations and to support teamwork when in unpredicted offline states. When a network connection with a team member becomes available, data synchronization is carried out to ensure that the group has the most consistent view possible of the resources shared within the team. The Office Groove platform consisted of two parts. Firstly, Office Groove itself was a decentralized client software for end-users. Secondly, Office Groove Server was provided to integrate shared data with other server-based

[4] CardDAV is an Internet draft being developed by the Internet Engineering Task Force (IETF).

[5] GroupDAV is a draft being developed by the open-source community.

[6] Personal Information Manager, a type of application for storing data such as address books, task lists, calendars, reminders, email archives etc.

systems and to act as a relay between Groove clients where direct network connections are not possible. Even when two communicating clients are not online at the same time, the relay would act as a store-and-forward server to deliver messages as and when possible. Groove clients communicated using a proprietary protocol, the Simple Symmetric Transmission Protocol (SSTP) [21].

Like BSCW and Access Grid, Office Groove also used the concept of a shared workspace. In Office Groove a workspace is a container for shared information where, in order to account for clients sometimes being disconnected from the rest of the workgroup, Office Groove allowed views on a workspace to diverge. When the clients were reconnected, the workspaces would synchronize. This applied to all types of data contained in a workspace including shared documents, discussion threads, and structured data. For example, if a team is collaboratively editing a single document and one of the participants loses their network connection, the disconnected user's view will not update the document with the other users' edits until a connection is reestablished. Likewise, any changes made by the single disconnected user were not visible to the rest of the team until the synchronization process can occur. Presence and communication is built into workspaces including broadcasting of presence status', instant messaging, and event and activity notification. The Office Groove client ran on a desktop PC or laptop, and its ability to allow teams to work offline and synchronize later on means that mobile users could be supported, albeit on laptop computers. This characteristic made Office Groove extremely useful for teams of mobile workers in situations where network availability is a scarce or unpredictable, such as in emergency situations [7]. Office Groove however was not supported on any mobile device platform.

E. *Zimbra Collaboration Suite*

The Zimbra Collaboration Suite (ZCS) [23] is a set of corporate groupware applications provided primarily through a rich Web UI developed with Ajax [24]. Originally developed by Zimbra Inc., since 2010 all Zimbra products are owned by VMWare, Inc. The core ZCS functionality is free and open source, however commercial alternatives are offered that include closed source proprietary components. Like SOGo, ZCS is centered on providing collaboration using traditional groupware tools such as email, calendaring and contact management. Apart from providing a standard set of collaboration tools, ZCS exposes all of its functionality via a Simple Object Access Protocol (SOAP) API allowing third-party developers to design their own front-end UIs. Web mashups can also be developed using ZCS's extension mechanism called *zimlets*. With email at its core, ZCS also serves email via both IMAP and POP3 protocols.

Significant effort was put into designing the ZCS Ajax interface in order to provide a highly accessible groupware system that requires minimal setup and configuration for the end-user. ZCS provides a rich Web UI that emulates the functionality of desktop applications such as natural keyboard mapping, drag-and-drop, and mouse actions beyond one-click hyperlinking. To simplify the user experience further, the UI automatically provides hover-over contextual information. For example, if within an email the sender has written a date, ZCS will highlight it the date when it is being read. By hovering over the highlighted text, a popup message will display any calendaring information linked to that date within the ZCS system. This kind of context awareness aims to simplify collaboration and organization.

Mobile device support using ZCS is provided two-fold. Firstly, a mobile Web interface provides access to email and calendaring functionality, whilst also providing read access to shared content. Any device with a mobile Web browser can access this interface. Secondly, ZCS provides contact management, calendaring, and email functionality natively on devices through vendor specific synchronization (iOS, Android, Blackberry), mobile Web and push email.

F. *Apache Wave*

Google Wave [25] [26] was a hosted groupware service and platform developed by Google Inc. In 2010 Google announced it would ceased providing the Google Wave product as a service, and has subsequently released Wave code development open-source to the responsibility of the Apache Software Foundation [27]. Google Wave proposed a model of collaboration based on the concept of *waves*. A 'wave' is basically a shared document hosted on a server that supports concurrent modifications and near real-time updates. Each wave is comprised of *wavelets* that are elements within a wave that contain specific content elements and a list of participants. Content may include text messages sent by a user called *blips*. Blips form the basis for discussion threads (termed *conversations*) within a wavelet. Other kinds of content might include hyperlinks, video, and maps. Every user has a personalized view of a wave according to how a user has previously interacted with a wave. The effect is that a single 'wave' can be treated as synonymous with a workspace that stores discussion threads augmented with different kinds of content inserted from other sources.

Like ZCS, Wave provides a rich Web UI and also a Web services API. Apart from third-party developers being able to develop Web mashups and custom client software to interact with Google Wave using the Google Wave API, the underlying design of Wave was being developed as a set of open architecture and specification documents. Their vision was to allow the hosted Google Wave service to interoperate with other systems hosted on other Internet domains and implementing the published protocol specifications. There are two published draft protocol specifications:

(1) Google Wave Federation Protocol - an extension to XMPP (Extensible Messaging and Presence Protocol) that enables near real-time updating of waves between wave service providers and,

(2) Google Wave Conversation Model - a description of how to implement the structure of conversation elements within a wave, specifically XML descriptions and schemas for representing blips and conversations.

Despite providing an open API and published protocol specifications, Wave did not provide explicit mobile support, however since the UI was Web-based, it was possible to access the Google Wave service through a

[7] Office Groove was integrated by a team at Louisiana State University into an Emergency Operations Center (along with a number of other Microsoft products) in the aftermath of hurricane Katrina in 2006 to coordinate evacuees and volunteers [22].

smartphone's Web browser [28]. The possibility remains however to build custom mobile clients to communicate using the Wave API.

G. Comparison of Groupware Applications

A comparison of the groupware applications discussed is shown in TABLE I. Throughout the examples described, there are some common thematic and recurring approaches. BSCW, SOGo, ZCS and Wave all provide Web based interfaces. This is because the Web has moved away from a purely content-driven medium to a universal medium for serving applications to end-users [29]. SOGo and ZCS are based on integration with legacy tools, in particular making the assumption that email is the communication medium of choice in corporate scenarios where end-users are already setup and familiar with email software. The learning curve for such systems is therefore less steep. Another recurring theme when comparing groupware applications is that apart from SOGo and ZCS, the other systems are all based around shared virtual workspaces. In Access Grid their concept of a virtual venue is functionally synonymous with a workspace, as is Apache Wave's 'wave'.

TABLE I.
COMPARISON OF GROUPWARE APPLICATIONS.

	Client platform	**Concept**	**Async**	**Sync**	**Mobile**
BSCW	Web	Workspace	Yes	No	No
AG	Win, Mac, Linux	Venue	Yes	Yes	No
SOGo	Web, email	Legacy tools	Yes	No	Yes
Office Groove	Windows	Workspace	Yes	Yes	No
ZCS	Web, email	Legacy tools	Yes	No	Yes
Wave	Web	Wave	Yes	Yes	Yes

Where there is less coherence is in whether or not to provide synchronous services. AG is based primarily on providing live audio and video feeds to all group participants, while Wave allows near real-time editing of waves. Office Groove also provides real-time communication, albeit in a limited form, through instant messaging. Asynchronous collaboration is however provided by all groupware products. Persistent workspaces provide a medium in which to leave messages for other users whether or not they are currently online. Using email as a primary communication method, as in SOGo and ZCS, means that messages can be delivered as and when possible. Finally, it is clear that mobile devices are not particularly well supported by these groupware systems. Where there is mobile device support, it relies on provision of functionality through the mobile Web in a similar way to desktop browsers [30]. SOGo and ZCS additionally provide mobile device synchronization of email, contacts and calendaring natively. None provide dedicated groupware clients for mobile devices.

IV. FRAMEWORKS FOR DEVELOPING GROUPWARE

Several of the groupware applications described in the previous section can be extended to build new UIs and tools. AG and SOGo are completely open source, as is ZCS in part. Office Groove could be customized and combined with other Microsoft products to create domain specific collaboration systems, as illustrated by the Katrina Emergency Operations Center developed by Louisiana State University [22]. ZCS and Wave provide open Web service APIs for developers to develop new software clients to access groupware functionality. However none of these groupware applications provides or builds on a dedicated development framework for engineering groupware. The extension mechanisms for each product are based on interoperation with the respective existing systems.

There have been a number of attempts to develop generic collaboration software frameworks. Researchers have attempted to formalize abstractions of collaborative work and provide tools for rapid development of groupware. This section reviews a selection of the state-of-the-art frameworks for developing groupware.

A. GroupKit

GroupKit, first described in [31], is an open-source software framework for building real-time group conferencing applications. Developed at the University of Calgary, Canada, the authors built on their experiences of engineering groupware applications. They identified that groupware systems had commonality that could be exploited in a reusable programming toolkit. Roseman and Greenberg formally state the motivation for GroupKit:

"A developer using a well-designed toolkit should find it only slightly harder to program usable groupware systems when compared to the effort required to program an equivalent single-user system". [32]

By aiming to create tools that make groupware as easy to design and program as single-user applications, the authors set themselves an optimistic target of not only solving HCI issues in engineering groupware, but also in engineering systems for multiple, distributed, and unpredictable end-users. In [31] they specify two sets of requirements for GroupKit in two broad categories: Human-centered design requirements and programmer-centered design requirements. Examples of human-centered requirements include supporting different group processes and integrating with traditional ways of doing work, such as using single-user applications or non-computer based methods. These requirements center around the notion of creating a framework that is flexible enough to *support* group work without imposing any new ways of working. The programmer-centered requirements focus on the need for solving common challenges that arise in multi-user computer systems such as in groupware, including supporting multiple distributed processes, shared data, and a shared graphics model.

With these requirements in mind, GroupKit provides the following generic components:

(1) A *runtime infrastructure* - to create distributed processes and manage inter-process communication between end-user workstations.

(2) *Groupware programming abstractions* - a set of programming primitives to hide the complex functionality for supporting the groupware system.

(3) Session managers - to enable end-users to find each other and create or join conference application sessions.

The GroupKit runtime infrastructure is formed generally of three components: A registrar, session managers (on each end-user workstation), and conference applications. The registrar acts as a central process that GroupKit applications connect to that manages connections between session managers and conference application processes. It usually resides at a specific network address to serve a single community of end-users. When session manager processes are started, they connect to a registrar to discover available conferences created by other users. The session manager then deals with creating conference application processes to join or register conference sessions. Conference processes can then communicate directly with each other, as managed by the session manager.

From a groupware engineering perspective, the programming abstractions and widgets are the key building blocks for creating conferencing applications. GroupKit provides three main programming abstractions: multicast remote procedure calls (RPC), events, and environments. *Multicast RPC* enables the broadcast of function calls to all members of a group conference, thus allowing shared actions to be executed on replicated data. The authors identified this as a method to easily turn single-user applications into shared multi-user ones. *Events* are a mechanism whereby conference applications are notified when things happen within the session and within other instances of conference applications. GroupKit's infrastructure provides standard session events such as when users join and leave conference sessions. Events can be sent to the whole group or to single users, for example, when a user joins a session as a latecomer and needs their entire conference application state updated. *Environments* are simple shared data structures provided on a per-conference basis. Data is stored in an environment through key-value pairs. A conference environment is replicated between all conference participants, and changes to the environment can trigger callback functions bound to allow notification of data being added, removed, or modified.

The groupware widgets that GroupKit provides enables programmers to use reusable UI components that are common and useful in many groupware applications. The widget functionality includes *participant status* to allow users to visualize other users joining and leaving conferences, *telepointers* to enabled user gesturing through an overlaid cursor, and *location awareness* to enable multiple users to gauge what another user is viewing on a shared window or data object.

B. Activity Based Computing

Activity Based Computing (ABC) [33] [34] is a framework for supporting computer-based collaboration through explicit modelling of activities. Developed by the Centre for Pervasive Computing at the University of Aarhus, Denmark, ABC is based on collaboration and activity theory also developed by one of the original authors [35] [36] [37]. The concepts of ABC are a result of studying and modelling medical environments such as in hospitals. The authors argue that such environments are significantly different from office work. In office scenarios end-users are stationed at a single location (e.g. usually their desk), and communicate with other workers through their desktop computer (e.g. by email), even if they are in the same office or building. In the medical environment, activities are typified by extreme mobility, spontaneous collaboration, frequent interruptions, and a higher degree of communication.

In ABC, three levels of abstraction are modelled. At the most general level of abstraction, the *activity level* consists of human-centered activities. At the next level down, ABC models the computer-based services and applications required to support the higher-level activity. This level is called the *application level*. Finally, the lowest abstraction is at the *data level* which models data, files, or other material manipulated by the application level.

The ABC framework itself, like GroupKit, provides both a runtime infrastructure and programming tools to create ABC applications. ABC provides a set of server-side components that communicate with a set of standard client-side processes that in turn interface with an ABC application. On the server-side, an *activity server* manages activities, sessions, and has a persistent activity store. On the client-side, an *activity controller* maintains communication with the activity server. A *state manager* and *service registry* maintain the set of ABC applications that can handle activity services. Finally, a *session manager* maintains real-time collaborative sessions managed on the server-side by a *collaboration manager*.

ABC's activity server provides facilities for stateful applications by enabling ABC applications to persist their application states. Through this mechanism, ABC supports mobility by allowing applications to suspend and resume from the persisted states. The authors make it clear however that the intention is not to support mobile device, but rather to support end-users that might move from device to device. This is in accordance with their original studies of user behaviours in the hospital environment where medical staff may move throughout the hospital and interact on various different devices. Stateful applications in ABC enable users to suspend their activity sessions and resume them on another device.

C. Agilo

Agilo [38] [39] is an open-source groupware framework based on Java to simplify the development of groupware applications. Like BSCW, it is developed by the Fraunhofer Society, and aims to be flexible enough to develop a diverse range of groupware applications. The framework design is based on an analysis of the different variation points of groupware application features and how the variations are realized as functional components of groupware. The authors identified that many groupware frameworks are inflexible and that groupware variations are based around five key characteristics: distribution model, communication infrastructure, sharing model, concurrency model, and synchronization model. This then lead to the creation of a taxonomy centered on the variations of each point as expressed in specific system implementation.

On each variation point, the authors of Agilo attempted to design for flexibility in the software framework. Each point is described as follows:

(1) *Distribution model* - The first variation in groupware systems the authors of Agilo identified is how the nodes of the groupware system are distributed. They identified that typically distribution models either follow client-server or peer-to-peer (P2P) topologies. In practice, many P2P systems are in fact hybrid topologies incorporating some element of centralization. For example, GroupKit can be thought of as a hybrid P2P system as it uses a central registrar server, but is designed to allow communication directly between group workstations in a P2P manner. To allow both client-server

and P2P distribution models, the Agilo framework provides components for building client and server parts separately, and for P2P models, both a client and server can be used on the same node.

(2) *Communication infrastructure* - The authors secondly identify that the underlying communication infrastructure and available transport protocols significantly affect the design of groupware applications. For example, software running in a LAN environment may have access to protocols such as TCP and UDP. However over WANs such as the Internet HTTP or connectivity through firewalls is required. To implement flexibility in transport protocols, the Agilo framework at a low-level implements multiple transport mechanisms for commonly used protocols including TCP and HTTP. For application-level protocols, Agilo provides a higher-level entry point for developers to build their own marshalling mechanism to interpret protocols over the low-level transports.

(3) *Sharing model* - Groupware systems commonly provide functionality for data sharing. The authors of Agilo identify that the way in which data objects are shared between distributed nodes as a key variation point. For example, data might be centralized, replicated in whole on each node, or support a loose consistency policy that can periodically resynchronize. In the Agilo framework, the data object distribution scheme is decoupled from the objects themselves. Shared objects implement a specific interface for the Agilo Object Manager to handle.

(4) *Concurrency model* - In distributed systems, such as in groupware, concurrency is an important issue. The way in which groupware systems deal with concurrent processes that may be distributed across nodes can vary, processes could be treated with different degrees of synchronicity depending on the quality of service required by the function of those processes. Agilo uses a variety of methods to process network messages on each node, and within a node can interweave synchronous and asynchronous message thread handling.

(5) *Synchronization model* - Finally, the authors identify a number of different methods to ensure consistency across shared resources in Agilo. Consistency can be preserved through two general approaches: *avoid* conflicts by using data locks such as semaphores, or *detect and resolve* conflicts by allowing modifications and fixing conflicting reads and writes after the event. As Agilo is built on Java, the underlying Java platform already provides some object synchronization mechanisms. Besides these, Agilo also implements semaphores and mutexes, as well as atomic transactions.

The key software design concepts in the Agilo framework are encapsulated as *modules*, *messages*, and *connections*. These are very generic concepts that developers can use to build groupware; modules being client or server-side software components that communicate with each other; messages being application-specific data chucks transferred between client and server modules to implement groupware functionality; connections being an abstraction above the raw transport layer for transferring messages. Agilo is designed as to allow the development of message handlers independent of underlying network transports to allow support for standardized protocols such as SOAP and XMPP, or for developers to implement custom application specific protocols (see point 2 on Communication infrastructure above).

D. *The Coco Collaborative Computing Platform*

The Coco Collaborative Computing project developed a P2P platform for ad-hoc group formation and collaboration and described in [40], [41], [42] and [43]. Coco was developed at the University of Reading, UK, as a P2P desktop platform for software developers to build Java SE groupware applications. In addition to the core platform, Coco provided of a suite of standard groupware applications that included instant-messaging, shared whiteboards, shared Web-browsing, and content management with collaborative metadata annotation capabilities. The development framework is built on a services-based architecture with collaborative community services hosted by JXTA[8] peers and peer groups. Community data (such as user profiles and content) is represented using Resource Description Framework (RDF[9]) data structures and exchanged by peers using the Coco metadata content service [41]. Synchronous interactions are supported by the Coco Messaging Service which enables groups of participants to engage in real-time video, audio, and IM interactions and the Interaction Service, which supports real-time presence notification within group interactions. The objective in providing distinct services is to enable collaborative community applications to access either all or a subset of Coco services hosted by participating peers.

MicroCoco aimed to interoperate with the full desktop version to provide collaboration services across the device domain with the following goals: Be interoperable with full Coco peers; Provide a subset of Coco's services for collaboration; Provide useful disconnected services for when network access is limited; Be small enough to operate on hardware constrained devices; Provide a suitable user interface depending on the type of device [44]. The success of MicroCoco was somewhat limited. Due to limitations of the underlying platform support for JXTA for mobile devices at the time, pervasive connectivity between Coco and MicroCoco was possible but very inefficient. There was also a significant added complexity to bridging between Coco and MicroCoco as many of the collaboration services could not be built on the same underlying JXTA service set. The design approach of MicroCoco was to make mobile peers interoperate with desktop peers, without consideration for what makes a suitable mobile groupware application. To this end, MicroCoco could only go so far in terms of providing satisfactory groupware applications for mobile end-users.

E. *Comparison of Groupware Frameworks*

A comparison of the groupware frameworks discussed is shown in TABLE II. These groupware frameworks have a number of similarities. Each of the groupware frameworks aims to provide two key pieces of collaboration functionality: group conferencing and content/data sharing. They also provide different abstractions and metaphors that attempt to relate software objects to real world objects - a technique commonly used

[8] JXTA is an open-source generic platform independent P2P protocol with a Java reference implementation.

[9] RDF is an XML-based generic metadata description framework.

in object-oriented design (OOD) [45]. GroupKit centers on multicast RPC, events, and environments; ABC's framework is based on breaking activities down into their component computer-based applications and corresponding data. Agilo simply uses modules and messages to encompass all collaborative software modules and their application-level communication protocols. Coco takes a service-oriented approach and defines messaging, content, and interaction services. Finally, all frameworks provide a runtime infrastructure as well programming tools to support groupware systems.

TABLE II. COMPARISON OF GROUPWARE FRAMEWORKS.

	Key theme	**Conferencing**	**Content Sharing**	**Mobile**
GroupKit	Synchronous tools	Yes	Yes	No
ABC	Stateful applications	Yes	Yes	Yes
Agilo	Flexible framework	Yes	Yes	No
Coco	Heterogeneity	Yes	Yes	Yes

Despite aiming to support conferencing and sharing, each framework approaches developing groupware in their own unique way. GroupKit's main theme is to develop synchronous conferencing applications, and supports creating conferences by using a centralized registrar to aid in creating P2P connections between applications. ABC, in contrast, aims to support collaboration by providing an infrastructure that supports user mobility between devices where activity states can be saved and reloaded. Agilo looks to provide flexibility in its framework by supporting multiple and extendable configurations of groupware system. Finally, Coco, like Agilo, addresses diversity. However Coco builds on platform independent technologies to provide a set of collaboration services that can work on multiple OS platforms and unpredictable network configurations.

In terms of mobile collaboration, GroupKit and Agilo do not provide support for mobile devices at all. ABC is designed to be device independent, and the authors recognize that ABC applications may run on different kinds of devices, from desktops to PDAs. Coco provides MicroCoco for mobile support. However as already discussed, its development as an afterthought to interoperate with full Coco services means MicroCoco applications cannot provide the full experience of desktop Coco groupware applications.

V. CONCLUSION

By examining what the various features that groupware applications and frameworks aim to offer, one can conclude that there are a number of common themes that run throughout designing groupware. The comparison of groupware applications highlights the recurring concept of the *shared workspace*. This implies that a shared workspace is an essential component of any groupware system and should be supported. The analysis highlights that state-of-the-art groupware systems also aim to integrate with existing single-user software. To support this, a groupware framework should ideally have mechanisms for integrating with existing applications, as well as having an effective approach to developing or extending those applications for group work.

Groupware has begun to include support for mobile users however much of this functionality is developed as an afterthought in efforts to interoperate with legacy applications. This top-down approach to groupware development leads to mobile end-users being less well supported in group collaborations. Mobile CSCW is by no means immune from the fragmentation that Grudin previously described. In fact, developing for the mobile device domain fuels this fragmentation with diversity in hardware and software platforms that is unprecedented in modern personal desktop computing. Significant research has been carried out in Mobile CSCW, but many of the mobile groupware systems that have been developed are application specific, singleton instances of experimental software, often developed for a uniform class of mobile devices. Dealing with fragmentation in opinion of what functionality forms effective groupware, coupled with addressing diversity of mobile device hardware and software, are key motivations for new research.

One of the key motivations for research, such as that described in [46], is to enable groupware engineers to build mobile groupware from the bottom-up; to give groupware engineers the means to create mobile-centric collaboration software. Mobile CSCW is by no means immune from the fragmentation that Grudin described. In fact, developing for the mobile device domain fuels this fragmentation with diversity in hardware and software platforms that is unprecedented in modern personal desktop computing. Significant research has been carried out in Mobile CSCW, but many of the mobile groupware systems that have been developed are application specific, singleton instances of experimental software, often developed for a uniform class of mobile devices. There is a clear need to provide tools to support rapid engineering of mobile groupware. Dealing with fragmentation in opinion of what functionality forms effective groupware, coupled with addressing diversity of mobile device hardware and software, are key motivations for future research in the mobile technology, software engineering and CSCW domains.

REFERENCES

[1] J. Grudin, "Computer-supported cooperative work: history and focus", *Computer*, vol. 27, no. 5, pp. 19-26, May 1994. http://dx.doi.org/10.1109/2.291294

[2] D.B. Horn, T.A. Finholt, J.P. Birnholtz, D. Motwani and S. Jayaraman, "Six degrees of Jonathan Grudin: a social network analysis of the evolution and impact of CSCW research", in *Proc. 2004 ACM Conf. Computer Supported Cooperative Work*, Chicago, IL, 2004, pp. 582-591. http://dx.doi.org/10.1145/1031607.1031707

[3] R. Want and T. Pering, "New horizons for mobile computing", in *Proc. 1st IEEE International Conference on Pervasive Computing and Communications*, Dallas-Fort Worth, TX, 2003, p. 3-8.

[4] R. Want and T. Pering, "System challenges for ubiquitous and pervasive computing", in *Proc. 27th International Conference on Software Engineering*, St. Louis, MO, 2005, pp. 9-14.

[5] M. Weiser, "The computer for the 21st century", *Scientific American*, pp. 94-104, September 1991. http://dx.doi.org/10.1038/scientificamerican0991-94

[6] K. Schmidt and L.J. Bannon, "Taking CSCW seriously: supporting articulation work", *Computer Supported Cooperative Work*, vol. 1, no. 1-2, pp. 7-40, March 1992. http://dx.doi.org/10.1007/BF00752449

[7] I. Grief, *Computer-supported cooperative work: a book of readings*. San Francisco, CA: Morgan Kaufman, 1988.

[8] L.J. Bannon and K. Schmidt, "CSCW: Four characters in search of a context", in *Studies in computer supported cooperative work: theory, practice and design*, pp. 3-16, North-Holland Publishing, 1991.

[9] M. Thum and C. Schraivogel, "Profile of the Fraunhofer-Gesellschaft", 2005.

[10] W. Appelt, "WWW based collaboration with the BSCW system", *SOFSEM'99: Theory and Practice of Informatics*, LNCS 1725, pp. 66-78, 1999. http://dx.doi.org/10.1007/3-540-47849-3_4

[11] R. Stevens, M.E. Papka and T. Disz, "Prototyping the workspaces of the future". *IEEE Internet Computing*, vol. 7, no. 4, pp. 51-58, July-August 2003. http://dx.doi.org/10.1109/MIC.2003.1215660

[12] I.T. Foster, "The anatomy of the Grid: enabling scalable virtual organizations". *Euro-Par 2001: Parallel Processing*, LNCS 2150, pp. 1-4, 2001.

[13] L. Childers, T. Disz, R. Olson, M.E. Papka, R. Stevens and T. Udeshi, "Access Grid: immersive group-to-group collaborative visualization", in *Proc. IPT 2000: Immersive Projection Technology Workshop*, Ames, IA, 2000.

[14] S. McCanne and Van Jacobson, "vic: a flexible framework for packet video", in Proc. 3rd ACM international conference on Multimedia, San Francisco, CA, 1995, pp. 511-522.

[15] O. Hodson,, S. Varakliotis, and V. Hardman, "A software platform for multiway audio distribution over the internet", in *Proc. IEE Colloquium on Audio and Music Technology: the Challenge of Creative DSP*, London, UK, 1998, pp. 114-116.

[16] D.L Nguyen, "TigerboardAG", in *Proc. 2006 ACM/IEEE conference on Supercomputing*, Tampa, FL, 2006, p. 314. http://dx.doi.org/10.1145/1188455.1188785

[17] F. Lachapelle and L. Marcotte, "Scalable OpenGroupware.org". *Linux Journal*, vol. 2008, no. 168, April 2008.

[18] M. Gagné, "Cooking with Linux: Linux, Thunderbird and the Blackberry - a love story". *Linux Journal*, vol. 2009, no. 183, 2009.

[19] J. McCoy, "Makeover for Groove: SharePoint Workspace 2010!" MSDN Blogs. [Online] Available: http://tinyurl.com/cvjp5lz (Last retrieved 29 December 2012)

[20] Y. Chou, "Get into the Groove: solutions for secure and dynamic collaboration". *TechNet Magazine*, October 2006. [Online] Available: http://tinyurl.com/cn5epf5 (Last retrieved 29 December 2012)

[21] [MS-GRVSSTP] - v20091106: Simple Symmetric Transport Protocol (SSTP) Specification, Microsoft Corporation, 2009.

[22] J. Morello, "Building an Emergency Operations Center on Groove and SharePoint". *TechNet Magazine*, October 2006 [Online] Available: http://tinyurl.com/alhvle (Last retrieved 29 December 2012)

[23] M. Gagné, "Zimbra collaboration suite, Version 4.5". *Linux Journal*, vol. 2007, no. 157, 2007.

[24] J.J. Garrett, "Ajax: a new approach to web applications". Adaptive Path, 2005. [Online] Available: http://tinyurl.com/29nlsm (Last retrieved 29 December 2012)

[25] D. Peterson, "Hello world, meet Google Wave". *The official Google Code Blog*, Google Inc., May 2009. [Online] Available: http://tinyurl.com/chvrxft (Last retrieved 29 December 2012)

[26] L. Rasmussen, "Went walkabout. Brought back Google Wave". *Official Google Blog*, Google Inc., May 2009. [Online] Available: http://tinyurl.com/cptfumu (Last retrieved 29 December 2012)

[27] A. North, "Introducing Apache Wave". *Google Wave Developer Blog*, Google Inc., December 2010. [Online] Available: http://tinyurl.com/cwrulw9 (Last retrieved 29 December 2012)

[28] E. Schonfeld, "Google Wave's little secret: it already works on the iPhone". Tech Crunch, October 2009. [Online] Available: http://tinyurl.com/29elvwj (Last retrieved 29 December 2012)

[29] T.V. Raman, "Toward 2^W, beyond Web 2.0". *Communications of the ACM*, vol. 52, no. 2, pp. 52-59, February 2009. http://dx.doi.org/10.1145/1461928.1461945

[30] M. Halvey, M.T. Keane and B. Smyth, "Mobile web surfing is the same as web surfing". *Communications of the ACM*, vol. 49, no. 3, pp. 76-81, March 2006. http://dx.doi.org/10.1145/1118178.1118179

[31] M. Roseman and S. Greenberg, "GROUPKIT: a groupware toolkit for building real-time conferencing applications", in *Proc. 1992 ACM conference on computer-supported cooperative work*, Toronto, ON, Canada, 1992, pp. 43-50. http://dx.doi.org/10.1145/143457.143460

[32] M. Roseman and S. Greenberg, "Building real-time groupware with GroupKit, a groupware toolkit". *ACM Transactions on Computer Human Interaction*, vol. 3, no. 1, pp. 66-106, March 1996. http://dx.doi.org/10.1145/226159.226162

[33] H.B. Christensen and J.E. Bardram, "Supporting human activities - exploring activity-centered computing". *UbiComp 2002: Ubiquitous Computing*, LNCS 2498, pp. 107-116, 2002.

[34] J.E. Bardram, "Activity-based computing: support for mobility and collaboration in ubiquitous computing". *Personal and Ubiquitous Computing*, vol. 9, no. 5, pp. 312-322, 2005. http://dx.doi.org/10.1007/s00779-004-0335-2

[35] J.E. Bardram, "Plans as situated action: an activity theory approach to workflow systems", in *Proc. 5th European conference on conference on computer-supported cooperative work*, Lancaster, UK, 1997, pp. 17-32. http://dx.doi.org/10.1007/978-94-015-7372-6_2

[36] J.E. Bardram, "Collaboration coordination, and computer support - an activity theoretical approach to the design of computer supported cooperative work". Doctoral dissertation, Institute of Computer Science, University of Aarhus, Denmark, 1998.

[37] J.E. Bardram, "Designing for the dynamics of cooperative work activities", in *Proc. 1998 ACM conference on computer supported cooperative work*, Seattle, WA, 1998, pp. 89-98. http://dx.doi.org/10.1145/289444.289483

[38] A. Guicking, P. Tandler and P. Avgeriou, "Agilo: a highly flexible groupware framework". *Groupware: Design, Implementation, and Use*, LNCS 3706, pp. 49-56, 2005.

[39] A. Guicking and T. Grasse, "A framework designed for synchronous groupware applications in heterogeneous environments". *Groupware: Design, Implementation, and Use*, LNCS 4154, pp. 203-218, 2006.

[40] I.M. Bhana, D. Johnson and N.S. Alexandrov, "Supporting ad-hoc collaborations in peer-to-peer networks", in *Proc. ISCA 17th Conf. Parallel and Distributed Computing Systems*, San Francisco, CA, 2004, pp. 491-496.

[41] I.M. Bhana and D. Johnson, "A peer-to-peer approach to content dissemination and search in collaborative networks". *Computational Science - ICCS 2005*, LNCS 3516, pp. 391-398, 2005.

[42] I.M. Bhana and D. Johnson, "Developing Collaborative Social Software". *Computational Science – ICCS 2006*, LNCS 3992, pp. 581-586, 2006.

[43] I.M. Bhana, "Coco: a common platform for collaborative computing in heterogeneous peer-to-peer networks". Doctoral dissertation, School of Systems Engineering, University of Reading, UK, 1998.

[44] D. Johnson and I.M. Bhana, "Pervading Collaborative Learning with Mobile Devices." Book chapter in *Technological Advances in Interactive Collaborative Learning*, Chapman and Hall/CRC Press, 2012.

[45] J. Noble, R. Biddie and E. Tempero, "Metaphor and metonymy in object-oriented design patterns". *Australian Computer Science Communications*, vol. 24, no. 1, pp. 187-195, January-February 2002.

[46] D. Johnson, "A Platform for Supporting Micro-Collaborations in a Diverse Device Environment". *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 3, no. 4, pp. 8-16, December 2009.

AUTHOR

David Johnson is with the Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford OX1 3QD, United Kingdom (e-mail: david.johnson@cs.ox.ac.uk).

The contributions in this paper were produced as part of the author's doctoral thesis work supported by a PhD scholarship from 2003-2007 from the School of Systems Engineering, University of Reading, UK. Received 02 January 2013.. Published as resubmitted by the author 20 March 2013.