MUSE - Platform For Mobile Computer Supported Collaborative Learning

Peter Byrne, M.Sc, B.A (Mod)

A thesis submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

April 2011

Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work. I agree that Trinity College Library may lend or copy this thesis upon request.

Peter Byrne, M.Sc, B.A (Mod)

Dated: 4th April 2011

Acknowledgements

I would like to take this opportunity to thank my supervisor Brendan Tangney for his advice during writing this thesis. The focus of the thesis on the domain of digital video production owes a lot of inspiration to Inmaculada Arnedillo Sánchez and her work on the MobileDNA. Finally I would like to thank Centre for Research in IT and Education (CRITE) and the Bridge 2 College for providing a setting for the educational and usability evaluations in this thesis.

Peter Byrne, M.Sc, B.A (Mod)
University of Dublin, Trinity College
April 2011

Summary

Traditionally much research on computer supported collaborative learning, CSCL, focused on either learning through distant collaboration or face-to-face collaboration sharing a computer. A new and emergent area within CSCL is mobile learning. From a pedagogical perspective, classifications of mobile-CSCL (MCSCL) argue that leveraging off affordances of mobile devices allows the development of learning activities that significantly differ from desktop-based CSCL applications. In parallel there is emerging agreement in the mobile learning community that applications informed by the principles of constructionism, contextualisation, and collaboration are of particular value for exploiting the ready-at-hand nature of such devices. In the area of collaboration and creativity, there is agreement that digital video production supports collaborative learning, encourages creativity, self-expression and deeper thinking, and draws on students' out-of-school interest. However, to date little use has been made of ICT to support the collaborative, as distinct from the video production aspect, of the process. The mobile digital narrative approach, mobileDNA, is a pedagogical methodology to scaffold digital narrative activities and support collaboration with ICT.

Yet creating sophisticated applications, encompassing both PCs and mobile devices, raises several technical considerations including: exchanging data between mobile and fixed devices; developing appropriate user interfaces for mobile devices; heterogeneity of hardware, software and networking technology. Currently there is a dearth of middleware providing generic support for common functions, particularly spanning both fixed and mobile contexts. As a result of this lack of middleware support for mobile collaboration, application developers must build both the application and a basic middleware from scratch.

The first question asked in this thesis is: what are the functional requirements for a platform to support MCSCL applications on both mobile and PCs? To answer this

question this thesis reviews the state of the art in mobile middleware and describes the design and development of MUSE, an MCSCL platform for multimedia applications which addresses the technical and pedagogical considerations above. MUSE uses a service orientated architectural model to facilitate the reuse of loosely-coupled components, thereby enabling the development of flexible and reconfigurable applications suitable to this heterogeneous environment. The architecture provides a distributed Model-View-Controller paradigm for interacting with shared data. Networking support adapts as users move from reliable LANs to less reliable networking technologies for mobile devices. In this way, MUSE provides a transparent network layer to create a reliable communication link over varying networking technologies for example: HTTP, TCP/IP, 3G, and MMS. This thesis focuses upon digital narrative production, and inspired by mobileDNA, MUSE provides a toolkit of generic services for application developers to use in this space. Furthermore, two MCSCL applications for creating digital narratives have been developed over MUSE - SMART, an application to create stop-motion animation on mobile phones, and the Digital Narrative Tool (DNT) an application to support users engaged in the video creation process which uses the mobileDNA and runs on both mobile phones and PCs.

The second research question asked in this thesis is: do the functional requirements for a MCSCL platform captured in MUSE, SMART and the DNT support constructionist, contextualised, and collaborative activities for learning? To answer this question the evaluation of the software that is the focus of this thesis will involve two parts. In the first part MUSE is evaluated in terms of a set of standard performance metrics. In the second part, MUSE is validated using an authentic learning setting, where the user applications to support digital narrative production (SMART and the DNT) are evaluated. These applications are evaluated in usability and learning terms using direct observation, interviews, questionnaires and the artifacts produced in a workshop setting.

In short there are five contributions from this thesis: A set of functional requirements for a MCSCL platform; the MUSE platform conceptual architecture; the MUSE platform reference implementation; the SMART mobile animation application; and the DNT mobile and PC application to support the mobileDNA.

Publications Related to this Ph.D.

- Peter Byrne and Brendan Tangney (2009) "*SMART: Stop-Motion Animation and Reviewing Tool*" in Multiplatform E-Learning Systems and Technologies: Mobile Devices for Ubiquitous ICT-Based Education (Ed. Dr Tiong-Thye Goh) IGI Global, Chapter 14. (pp 229-242).

- Peter Byrne, Inmaculada Arnedillo-Sánchez, and Brendan Tangney (2008). *A Mobile Computer Supported Collaborative Learning Tool for Digital Narrative Production*. Proceedings of the MLearn 2008 Conference, University of Wolverhampton, UK. (pp 66-73).

- Peter Byrne and Brendan Tangney (2007) *Works in Progress: Mobile Learning, Animation on Mobile Phones* in IEEE Distributed Systems Online, Vol. 8, No. 7.

- Peter Byrne and Brendan Tangney (2007) *SMART - An application to support animation on mobile devices* (Eds. Sánchez, I. A., et al.) The CSCL Alpine Rendez-Vous, Beyond Mobile Learning Workshop. Villars, Switzerland, Trinity College Dublin Press. (pp 66-67).

- Peter Byrne and Brendan Tangney (2006) *Animation on Mobile Phones*. IADIS International Conference on Mobile Learning. (pp 365-369).

Contents

List of Tables

List of Figures

Glossary of Terms

CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CRITE	Center for Research in IT in Education
CSCL	Computer Supported Collaborative Learning
CSCW	Computer Supported Collaborative Working
DNT	Digital Narrative Tool
DVP	Digital Video Production
ER	Entity Relationship
eVB	eMbedded Visual Basic
GSM	Global System for Mobile communications
GPS	Global Positioning System
GUI	Graphical User Interface
HTTP	Hyper-Text Transfer Protocol
ICT	Information and Communications Technology
I/O	Input / Output
IP	Internet Protocol
IrDA	Infra-red Data Access

IR	Infra-Red
ISO	International Organization for Standardization
IT	Information Technology
J2SE	Jave 2 Standard Edition
KB	KiloByte
LIME	Linda In Mobile Environments
MANET	Mobile Ad hoc NETwork
MCSCL	Mobile Computer Supported Collaborative Learning
MMS	Multimedia Messaging Service
mobileDNA	Mobile Digital Narrative Approach
ms	Milliseconds
MST	Minimum Spanning Tree
MVC	Model - View - Controller
NKF	Network Kernal Framework
OS	Operating System
OT	Operational Transformation
PC	Personal Computer
PDA	Personal Digital Assistant
QUIS	Questionnaire For User Interaction Satisfaction
RDBMS	Relation DataBase Management System
RMI	Remote Method Invocation
RPC	Remote Procedure Call

SMART	Stop-Motion Animation and Reviewing Tool
SOA	Service Orientated Architecture
SP	Service Pack
SQL	Structured Query Language
SUMI	Software Usability Measurement Inventory
SUS	System Usability Scale
TCP/IP	Transmission Control Protocol / Internet Protocol
WLAN	Wireless LAN
XEP	XXMP Extension Protocols
XXMP	eXtensible Messaging and Presence Protocol
ZPD	Zone of Proximal Development

Chapter 1

Introduction

Traditionally much research on computer supported collaborative learning, CSCL, focused on either learning through distant collaboration or face-to-face collaboration sharing a single computer. A new and emergent area within CSCL is that of mobile computer supported collaborative learning, MCSCL, e.g. [Zurita and Nussbaum, 2007]. From a pedagogical perspective, classifications of MCSCL argue that leveraging off the affordances of mobile devices allows the development of learning activities that significantly differ from desktop-based CSCL application [Roschelle, 2003]. This thesis follows the argument from Patten et al. that the applications that make the best use of mobile technology for learning are informed by constructionist, contextual, and collaborative, learning theories [Patten et al., 2006].

It is well known that the usage of mobile phones is increasing [Gartner, 2007, Morgan Stanley Global Technology & Telecom Research, 2009], including devices with built-in cameras and multiple communication options e.g. MMS, Bluetooth, and Infrared (IR) [Gartner, 2005]. More people will access the internet through mobile devices than desktop PCs in five years [Morgan Stanley Global Technology & Telecom Research, 2009]. There is also a trend towards the use of mobile phones by students, which can be exploited in educational situations [Naismith et al., 2004, Bull et al., 2002]. MCSCL applications are given further impetuous recently now that processing power on mobile devices, or smart phones, has reached a sufficient level to make useful applications feasible, for instance the applications available for the iPhone and the latest Android devices.

Yet creating sophisticated applications, encompassing both PCs and mobile devices,

raises several technical considerations including: exchanging data between mobile and fixed devices; developing appropriate user interfaces for mobile devices; heterogeneity of hardware, software and networking technology. In the mobile environment further difficulties encountered include: low memory and processor performance, small screen with reduced GUI capabilities and resolution, heterogeneous operating systems, heterogeneous networking technologies with varying network bandwidth and latency limits [Cinque et al., 2005, Guicking and Grasse, 2006, Roth, 2002a]. Therefore the above challenges of the mobile computing environment and the current lack of middleware support for developing MCSCL applications provide the motivation to create a middleware to support application developers in the mobile computing domain.

1.1 Motivation

While there is already a lot of collaborative applications available for mobile devices, typically these are built from scratch and tailored to specific platforms, e.g. Geney [Danesh et al., 2001], Cooties [Soloway et al., 2001] etc. Collaborative applications often contain common functionality, for example, networking support, which would be better provided as part of a more generic solution. Currently there is a lack of middleware providing generic support for such common functions [Springer et al., 2008, Roth, 2002b] particularly spanning both fixed and mobile contexts [Farooq et al., 2002]. As a result of this lack of middleware support for mobile collaborative applications developers must build both the application itself and a basic middleware from scratch [Roth, 2002b]. In both the CSCW and CSCL fields the trend is towards building generic systems and platforms to decrease the development cost of collaborative applications [Lonchamp, 2006, Dimitracopoulou, 2005], allowing developers to focus on application specific details [Roth, 2003].

In short, there is a lack of middleware support for MCSCL applications over both mobile and PCs, and the consequent focus on individual one-off MCSCL and mobile learning applications, which rarely outlive a particular device or study. Therefore, the principal motivation for this thesis is to derive the set of functional requirements that should inform the design of platforms to support MCSCL applications. For this reason questions asked in this thesis focus on finding these functional requirements and

testing them in practice by developing a middleware to support MCSCL applications. It is hoped that answers to these question will contribute to encouraging progress and further development in the mobile middleware domain.

A secondary motivation that underpins the work in this thesis was to provide support for developing MCSCL applications to support digital video production activities. The area of digital video production was initially chosen because digital video production can support constructionist, collaborative and contextualised learning activities. Furthermore, in the area of collaboration and creativity there is agreement that digital video production (DVP) supports collaborative learning and encourages, creativity and self-expression, deeper thinking and draws on students' out-of-school interest e.g. [Reid et al., 2002, Buckingham, 2003, Buckingham et al., 1999]. According to [Hofer and Swan, 2005] digital video presents "powerful opportunities to design student-centred, inquiry–based projects" and supports "critical inquiry, problem solving, and critical thinking" [Swain et al., 2003]. While for [Kearney and Schuck, 2005] digital video projects encourage development of media literacy, communication and presentations skills. Animation is an analogous process to digital video production that shares many of the potential educational advantages while being a simpler activity [Madden et al., 2008]. The important distinction between live action and animation is that live action records real life events as they occur whereas animation creates the illusion of life and motion from static frames. This allows users to depict scenes not possible with live action [Collins et al., 2000]. Furthermore, digital video production and animation are easily understood as they are familiar [Marsh and Thompson, 2001] and concrete activities [Hamalainen et al., 2004] for the students.

Further motivation was that currently to date little use has been made of ICT to support the collaborative, as distinct from the video production aspect, of the digital narrative creation process. Given the aforementioned benefits of digital video production and animation for learning, this thesis describes two MCSCL applications for creating digital narratives have been developed using the MUSE platform. The first application is called the Stop-Motion Animation and Reviewing Tool (SMART), which supports the shooting of small animated-movies using the stop-motion animation technique. The second application is called the Digital Narrative Tool (DNT) is inspired by the mobile Digital Narrative Approach (mobileDNA) [Arnedillo-Sanchez,

2008, Arnedillo-Sanchez and Tangney, 2006]. The DNT is an application to support users engaged in the video creation process described in mobileDNA. The DNT allows users to create mind-maps collaboratively for idea-generation, create a script for the story, and capture images and sounds as the content for the movie. A collaborative timeline editor is included so the users can create the final movie. It will be shown that the performance and functionality of the DNT degrades gracefully as users move from high performance PCs to mobile devices, replacing the rich graphical and communication support on PCs applications with alternative communications tools and simpler graphical user interfaces on mobile devices.

1.2 Research Questions

The previous section highlighted the lack of middleware support for mobile learning and MCSCL applications on mobile devices and PCs. In addition it was noted that digital video production activities present valuable opportunities to support constructionist, collaborative and contextualised learning but that there is a lack of ICT support for collaboration. Therefore, this thesis investigates two main research questions. The first question asked in this thesis is: what are the functional requirements for a platform to support MCSCL applications on both mobile and PCs?

This first question is addressed by a review of the literature relevant to CSCW, CSCL, MCSCL, mobile learning, and the pedagogical theories relevant to mobile learning and CSCL. In addition, the thesis then examines the previous work on mobile middlewares in terms of these functional requirements. The next step to answering this question is to compare the functionality provided by MUSE and the other middleware systems described in the thesis with the set of functional requirements identified earlier.

The second research question asked in this thesis is: Do the functional requirements for a MCSCL platform captured in MUSE, SMART and the DNT support constructionist, contextualised, and collaborative activities for learning? The answer to this question involves examining a number of sub-questions that arise here including: firstly from a technical perspective can we design and build an MCSCL middleware to work in a setting containing heterogeneous devices and networks? What is the performance of the middleware on mobile devices and PCs? How long is the message round-trip time

between devices and the server? How many concurrent users can the system support? and secondly from a learning perspective: what is the usability of the applications developed and to what extend to they support constructionist, contextualised and collaborative activities for learning?

To answer the technical questions several performance metrics are collected, these metrics are derived from metrics gathered in related work on mobile collaborative middlewares including [Zurita and Nussbaum, 2006, Gotthelf et al., 2007, Cotroneo et al., 2007]. Some of the metrics recorded to answer these questions in this thesis include: typical message size, round-trip time, group latency, protocol overhead, throughput, and implementation cost. Moreover the utility and functionality of the MUSE platform is further validated by building two collaborative applications for digital narrative production using the platform.

The sub-questions concerning the usability and educational merits of the applications built using MUSE include: what is the usability of the MCSCL applications subsequently developed? And to what extent do they support constructionist, contextualised, and collaborative activities for learning? To answer these questions the user applications (SMART and the DNT) are evaluated in usability and learning terms using direct observation, interviews, questionnaires and the artifacts produced.

1.3 Thesis Contributions

In short there are five contributions from this thesis: The set of functional requirements for a MCSCL platform; the MUSE platform conceptual architecture; the MUSE platform reference implementation; the SMART mobile animation application; and the DNT mobile and PC applications to support the mobileDNA.

1.3.1 Functional Requirements For A MCSCL Platform

There is a currently a lack of middleware support for developing MCSCL applications so each new application has to be developed from scratch for each device. In addition a rudimentary middleware or communications infrastructure must be created to support the applications. The first contribution of this thesis is to provide a set of functional requirements derived from the literature to inform the design of platforms to support

MCSCL applications.

1.3.2 MUSE Conceptual Architecture

By applying the above set of functional requirements this thesis provides a conceptual model for a MCSCL platform to support developers creating applications in this domain. There are four elements to the conceptual design to increase the flexibility of the software in the heterogeneous mobile environments. Firstly communications is via specified XML formats. Secondly the MUSE service-layer and MUSE networking-layer are designed according to a design-by-contract approach. Thirdly the MUSE conceptual architecture recommends using an MVC architecture and service orientated architecture. Finally the MUSE auxiliary-interface specifies a set of functions that an application can rely on regardless of OS supplied support.

1.3.3 MUSE Reference Implementation

A reference implementation of the MUSE platform was created during this thesis and tested using a set of performance metrics. These included: message round-trip time; group latency; protocol overhead; throughput; and implementation cost. These metrics uncovered the acceptable range of operation that this middleware can support. For instance the round-trip and group latency figure imply that this MUSE platform can support up to five simultaneous PC MCSCL client applications, and can support at least four mobile clients.

To provide further validation that the MUSE platform can support MCSCL applications for collaborative digital narrative production two applications were created using this middleware, SMART and the DNT. In addition, these two applications help answer the second research question: do the functional requirements for a MCSCL platform captured by MUSE, SMART and the DNT support constructionist, contextualised and collaborative activities for learning?

1.3.4 SMART

SMART which is a mobile application to enable learners to create animations on their mobile phone using the stop-motion animation technique [Byrne and Tangney, 2006,

2007a,b, 2009]. This application uses facilities of the MUSE auxiliary-interface to enable image capture on a mobile phone, and saving images on the device. It uses the service-layer and networking layers to send images and XML representing the animation to a service on the MUSE application server to generate a movie file that the learners can view on any device. This software was evaluated (see Chapter 6) and was found to be usable, and that the learners were able to engage in a constructionist, contextualised, and collaborative approach to this activity.

1.3.5 Digital Narrative Tool

The second application was the digital narrative tool, which is a mobile and PC application designed to support learners engaged in the mobileDNA [Arnedillo-Sanchez, 2008, Arnedillo-Sanchez and Tangney, 2006]. The application provides collaborative tools to create mind-maps, storyboards and edit the movie timeline. Additional services are included for chat communication and video rendering. The DNT was evaluated in a small pilot study and the users were able to follow the mobileDNA process and create digital movies by the end of the study. There is initial indicative evidence that the users were able to engage in a constructionist, contextualised, and collaborative approach to this activity. However this is a small study so further studies are needed to support this statement.

1.4 Thesis Road Map

The remainder of this thesis is as follows:

Chapter 2 - Theoretical Foundations: This chapter focuses on a review of the literature apposite to the design of a middleware for MCSCL. Consequently, four areas of particular relevance to the design and implementation of the middleware are computer supported collaborative working and computer supported collaborative learning, mobile learning, pedagogical theories relevant to mobile learning.

Chapter 3 - State Of The Art: This chapter provides a review of the literature and describes related work in the area of mobile middlewares in general and MCSCL middlewares in particular.

Chapter 4 - MUSE: Middleware For MCSCL: This chapter describes the design and implementation of the MUSE platform for mobile for MCSCL. It describes the technical challenges of developing systems in this environment, and describes the characteristics of MUSE in the distribution model, communications infrastructure, sharing model, concurrency model and synchronisation model. Aspects of MUSE specific to groupware and multimedia production are also described. Finally muse is compared with the other middleware systems described in Chapter 3.

Chapter 5 - Services And Applications: This chapter describes the literature pertinent to digital media production, with a particular focus on digital video and animation production. The mobileDNA process is described and problems encountered by users are highlighted. This chapter then outlines the generic services provided by MUSE to support MCSCL digital multimedia applications. The two applications developed to use MUSE are also described.

Chapter 6 - MUSE Evaluation: The evaluation of the software that is the focus of this thesis involves two stages. The first stage of the evaluation of the MUSE platform uses a set of standard performance metrics. This section addresses the sub-questions including: can we design and build an MCSCL middleware to work in a setting containing heterogeneous devices and networks? What is the performance of the middleware on mobile devices and PCs? How many concurrent users can the system support? In the second stage the user applications (SMART and the DNT) are evaluated according to the framework described by [Sharples et al., 2007, Vavoula, 2007, Vavoula and Sharples, 2008, Vavoula et al., 2006]. The sub-questions asked examine the usability of the applications and do the applications enable constructionist, contextualised and collaborative activities for learning?

Chapter 7 - Conclusion: This chapter contains the concluding discussion, thesis findings and restates the thesis contributions. This chapter finishes by considering the future direction of this research.

Appendix A - Survey Forms: This section includes a full copy of each of the questionnaires used in the course of this study.

Appendix B - MUSE Technical Specification: This section contains a list of the default XML formats included with the MUSE middleware and a description of the attributes of each message type. This section also includes a detailed description of the application programming interface of MUSE.

Chapter 2

Theoretical Foundations

An objective of this thesis is to develop a middleware platform to support mobile computer supported collaborative learning, with a particular focus on digital video production. Therefore, this chapter presents a review of the literature pertinent to the design of such an MCSCL platform. Consequently, areas of particular relevance to the design of the middleware are: computer supported collaborative working; computer supported collaborative learning; mobile learning; and pedagogical theories relevant to mobile learning. This chapter then derives a list of functional requirements for a MCSCL platform to support such mobile learning activities. These functional requirements are used to inform the literature review, comparison and evaluation of previous related work on mobile middlewares, which is described in the next chapter on the state of the art. The description of aspects related to digital video production and the design of MCSCL applications is deferred until Chapter 5

Computer supported collaborative working is the starting point of this review as it is the most mature field covered in this thesis and forms the basis for later collaborative applications in the computer supported collaborative learning or mobile computer supported collaborative learning fields. This section identifies a taxonomy for classifying groupware applications and derives a checklist of desirable features a middleware for CSCW should provide. In this context, awareness mechanisms are given particular attention, as such awareness mechanisms are vital in enabling collaborating users to establish a shared context. Computer supported collaborative learning is described next and the trend from text-orientated to action-orientated CSCL systems is noted. Furthermore, recent CSCL systems are being designed to provide greater generic

support for CSCL activities, which use middlewares that can be tailored to different collaborative learning tasks.

The following section then describes mobile computer supported collaborative learning (MCSCL) domain, which is more than an extension of CSCL to mobile devices. MCSCL leverages the different affordances provided by mobile devices, rather than traditional CSCL based applications which rely on affordances offered by the PC. These affordances enable new learning opportunities, for example, contextualisation in the real environment by exploiting the mobility enabled by the devices. This section also describes the pedagogy that underpins both CSCL and MCSCL, and recommends that the best use of mobile technology for learning are informed by constructionist, contextualised and collaborative learning theories. While related work in mobile learning is described, it is noted that currently most mobile learning applications are one-off designs and are not portable, as currently there is a lack of middleware support for MCSCL application [Springer et al., 2008, Roth, 2002b].

2.1 Computer Supported Collaborative Working

Computer supported collaborative work (CSCW) is a mature field that concentrates on developing tools to enable collaborative working. Ellis et al. define groupware as "a computer-based system that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment" [Ellis et al., 1991]. While Mittleman et al. define groupware technology as "a software solution which, if implemented, could help move a group toward its goals in some specific way" [Mittleman et al., 2008] and a collaboration product as a collection of "collaboration technologies offered as an integrated package" [Mittleman et al., 2008]. Grudin reserves the term CSCW to refer to the research area and groupware to refer to the technology and applications [Grudin, 1994]. However recent work has noted that the area of CSCW has become fragmented Schmidt [2009], Grudin [2010], therefore synthesising this previous work this thesis finds the following useful definitions: CSCW to refer to the research area; groupware to refer to collaborative applications, and collaboration technology to refer to the finer-grain functionality of groupware applications.

These broad definitions encompass a wide range of applications from messaging

	Same Time	**Different Time**
Same Place	face-to-face interaction	asynchronous interaction
Different Place	Synchronous distributed interaction	asynchronous distributed interaction

Table 2.1: Space - Time Classification [Ellis et al., 1991]

systems like email or chat, to multi-user editors. However, two important elements in these definitions are the idea of a shared task and a shared environment. A shared task or a common goal are prerequisites for collaboration. A shared environment is relevant to supporting a shared understanding between the collaborators and promoting the efficacy of the collaboration through the use of collaboration technologies. In brief, supporting the development of groupware applications that provide a shared environment to collaborating users engaged in a shared task is an objective of the middleware developed in the course of this thesis.

An initial taxonomy of CSCW categorised applications according to a space-time division [Johansen et al., 1991]. Ellis et. al expanded on this taxonomy categorising applications according to a space-time taxonomy and an application level taxonomy [Ellis et al., 1991] (see Table 2.1 on page 12). The space-time taxonomy classified CSCW systems according to a space-time division of the participants using the systems, with a spectrum that ranges from same time and place around a computer screen to different time and place distributed work. The application level taxonomy of CSCW systems categorised CSCW system according to their functionality and provided the following classes: message systems, multi-user editors, group decision support systems and electronic meeting rooms, computer conferencing, intelligent agents and coordination systems [Ellis et al., 1991].

A two level taxonomy of CSCW is also presented by Rodden who classifies groupware systems according to space-time separation of the users and application functionality [Rodden, 1991]. The space-time taxonomy is the same and classifies groupware systems according to whether they are synchronous or asynchronous and whether the users are remote or co-located [Rodden, 1991]. Rodden's application level taxonomy recognises four types of CSCW applications: message systems, computer conferencing, meeting rooms, and co-authoring and argumentation systems. Rama and Bishop in a survey and comparison of groupware applications follow the same approach, although they

use some different categories at the application level taxonomy [Rama and Bishop, 2006]. Alternatively Crook represents the space-time distinction as between interaction around a computer and interaction through the computer, over the network (p.189-193) [Crook, 1996].

Mittleman et al. believe that as a result of the proliferation of groupware applications in recent years, particularly with Web 2.0 technologies and social networking, classification of groupware applications is difficult [Mittleman et al., 2008]. Therefore they propose a taxonomy of groupware technologies that focuses on the capabilities of individual components of groupware applications, and then classify the groupware applications as "bundles of capabilities" [Mittleman et al., 2008]. They identify nine dimensions in groupware systems: Core functionality, content, relationships, session persistence, supported actions, action parameters, access control, alert mechanisms, and awareness indicators. This classification scheme is useful as it captures the set of actions, interactions and functionality one would expect to be present to some extent in any given CSCW system. This classification serves as a useful checklist of properties that a generic CSCW platform should support. As far back as 1996, Schmidt and Rodden proposed a similar checklist for designing a CSCW platform but concentrated on the interaction mechanisms between the users and state that the platform should support: informal interaction, information sharing and exchange, decision making, coordination and control protocols, and domain directories [Schmidt and Rodden, 1996].

Core Functionality is the primary functionality provided by a groupware application. There are many overlapping categories in the above application taxonomies that try to classify the core functionality of groupware systems, i.e. [Rodden, 1991, Mittleman et al., 2008, Rama and Bishop, 2006, Ellis et al., 1991, Crook, 1996, Johansen et al., 1991]. This thesis synthesises these application taxonomies in order to provide a useful framework and so suggests the following application categories: messaging systems, conferencing systems, group decision support systems and collaborative co-authoring systems. To clarify these categories this section describes each with some examples where appropriate.

Messaging Systems can be synchronous or asynchronous and range from email, threaded discussion groups, voice mail support systems, SMS text-messaging systems to instant messaging. To provide context, messaging systems usually track messages

and retain old messages. Regardless of time, the ability to contact colleagues efficiently is needed in any collaboration so this interaction paradigm has been included into almost all groupware systems, for instance, Quilt includes tools for threaded discussion, annotation of edits with text and a private directed messaging system [Fish et al., 1988, Leland et al., 1988]. Cosar [Jaspers et al., 2001], a more recent collaborative text-editor, contains text-chat for communication and feedback.

Conferencing systems include simple real-time tools to sophisticated systems involving integrated phone systems or web cams, the use of shared workspaces, whiteboards and meeting rooms systems. These system can allow users "who are either gathered in an electronic meeting room or physically dispersed, to interact synchronously through their workstations or terminals" [Ionescu and Marsic, 2003] or through an audio link. Meeting rooms systems are larger scale systems that may contain a physical component e.g. the PlexCenter Planning and decision support laboratory at the University of Arizona [Applegate et al., 1986], but can be purely virtual environments e.g. TeamRooms [Roseman and Greenberg, 1996].

Group decision support systems aim "to improve the productivity of decision-making meetings, either by speeding up the decision-making process or by improving the quality of the resulting decisions" [Cockburn and Jones, 1995]. Other tools in this area are voting tools or tools for issue analysis and idea generation like brainstorming tools. Group decision tools are typically incorporated into other groupware applications to improve the efficacy of collaboration activities.

Collaborative co-authoring systems are tools to support the creation of documents, text, images, or other shared object, by a group of users. These systems can be synchronous or asynchronous [Ionescu and Marsic, 2003]. Early influential examples include Quilt [Fish et al., 1988, Leland et al., 1988], a collaborative tool for cooperative writing and GROVE a simple text editor designed for use by a group of people simultaneously editing an outline document [Ellis et al., 1991].

Content is the set of possible contributions to a collaboration system, for instance, text, graphics, or video etc. produced by the collaborators. *Relationships* are the associations users can establish between content contributions, for example, to create data structures including, lists, trees, and collections. *Session persistence* defines whether a user's contribution is ephemeral, relating only to the session, or permanent.

Supported actions are the operations the users can perform on the content contributions, i.e. add, receive, associate, edit, move, delete, and judge (vote). *Action parameters* are properties that characterise or modify the supported actions, they are synchronicity and identifiability. Synchronicity concerns the expected delay between a user executing an action and other users perceiving the effects of the action. Identifiability is the degree to which users can determine who executed an action. *Access control* manages the users rights and privileges to execute a supported action.

Alert mechanisms are functions in the system that notify the users that something of someone in the system requires their attention. *Awareness indicators* are a larger set of functions to inform the users about the state of the shared space and provide the users with knowledge of the status and actions of the other users in the shared space.

2.1.1 Workspace Awareness

As awareness information is of particular relevance to synchronous collaboration it justifies further explanation in this section. Awareness, as a concept in general, is related to the individual's perception of the workspace, and the participants present in the workspace, which they can use in order to complete a task. The most widely cited definition of awareness is by Dourish and Bellotti which states that "awareness is an understanding of the activities of others, which provides a context for your own activity" [Dourish and Bellotti, 1992]. However,

> "while staying aware of others is something that we take for granted in the everyday world, maintaining this awareness has proven to be difficult in real-time distributed systems where information resources are poor and interaction mechanisms are foreign. As a result, working together through a groupware system often seems inefficient and clumsy compared to face-to-face work." [Gutwin and Greenberg, 2002].

The workspace awareness information of interest to users of groupware systems is summarised as "who is present, where they are working, and what they are doing" [Gutwin and Greenberg, 1998b]. These simple awareness requirements are expanded upon in [Gutwin and Greenberg, 2002].

Under the heading of "*who is in the system?*" Gutwin and Greenberg [Gutwin and Greenberg, 2002] ask the following sub-questions: is there anybody else using the workspace? what is their identity?, and who did what? Identity and embodiment awareness information help users answer these questions. All actions in the system can be associated with the user who performed the action and in this way the system provides identity. Embodiment is any way of consistently representing a user in the workspace and "describes the way in which users are themselves directly represented within the display space" [Benford and Fahl, 1994]. In this way embodiment is an element of identifiability of actions described in [Mittleman et al., 2008]. In short, embodiment translates identity information into something tangible in the user interface.

Under the heading of "*what are the users doing?*" Gutwin and Greenberg [Gutwin and Greenberg, 2002] ask the following sub-questions: what actions are being performed? What are the other users in the system's intentions? And what document or more generally object are the other users interacting with?

Dourish and Bellotti identify two types of awareness information, passive awareness mechanisms and active awareness mechanisms. Similarly Gutwin and Greenberg identified three key ways in which awareness information can be gathered: consequential communication, feed-through, and intentional communication [Gutwin and Greenberg, 2002]. Consequential communication and feed-through are passive awareness information, while intentional communication is an active awareness mechanism. Consequential communication, feed-through, and intentional communication can be used to answer the question, what are the users doing?

Consequential communication is awareness information that is conveyed by a user's actions in a system. To take the example from Gutwin and Greenberg [Gutwin and Greenberg, 2002] if a pilot is flying a plane and reaches over to engage the landing gear, the co-pilot will see this and will be informed that the landing gear has been engaged without any explicit communication necessary, he only has to observe the pilot's actions, or interactions in the shared workspace.

Feed-through is awareness information gathered from the system as the users interact with the objects in the shared context. For example if you cannot see the user performing the actions directly you can perceive that they have done something to the objects in the system purely by observing the changes in the shared objects. As an

example from an air traffic control system [Gutwin and Greenberg, 2002], the person controlling the departures may not be able to communicate or see the person who is controlling the arrivals of the planes. By observing the display listing the arrivals he will be able to ascertain that the arrivals controller is landing planes because the entries on the arrivals board change to landed. This can also be seen in collaborative editing environments as writers contribute to the document, the other writers can tell what the others doing by observing the document getting longer and new text appearing etc.

Intentional communication is awareness information that is exchanged by users through explicit communication. For example by sending a message or calling to another user in the system. There are several different options for this explicit communication, including email, fax, instant messaging, telephone call and so on. [Ackerman, 2000] state that "CSCW systems should have some secondary mechanism or communication back-channel to allow users to negotiate the norms of use, exceptions and breakdowns among themselves, making the system more flexible". In one study [Churchill et al., 2000] they recommend that chat or instant messaging systems should be situated in the same context of the objects being discussed. They advise that the chat client is built directly into the groupware application and not in a separate window as has traditionally been the case. Intentional communications mechanisms are an obvious way of deliberately informing the other users in the systems what you are doing.

Finally, under the heading of "*where are the users in the workspace?*" Gutwin and Greenberg [Gutwin and Greenberg, 2002] ask the following sub-questions: what are the other users current locations in the workspace? what they are currently looking at? and where they can potentially view and interact with?

The users need reference points in a shared workspace for communication and navigation to help users to answer these questions. Having information like size or bounds of the workspace and what the current state of the users and objects in the system is considered important [Yang and Olson, 2002] in this respect. Some virtual environment systems even have learning about the workspace as one of the design goals. It is useful for the users to have some landmarks in the workspace for discussing the workspace. These landmarks or common contextual reference points aid communication as they

are a shortcut to locations in the document and can be expressed orally rather than rely on telepointers which might distract from the current task you are performing [Yang and Olson, 2002].

"Where users are involved in a joint project of some kind, such as putting together a machine, viewing a shared document, or hunting down a monster, people need not only to see the environment and each other, but also to communicate in order to convey their own intentions, confirm the understanding of each others' intentions, and coordinate their joint actions." [Axelsson et al., 2003]. In an editable document the reference points could be headings or other standout features of the document. In a collaborative programming environment the system used could be line numbers and class names. In a collaborative drawing environment the reference points could be coordinates or standout shapes or objects in the drawing.

2.1.2 Summary Of Findings From CSCW

A large and diverse body of literature informs the field of CSCW. This section concentrated on defining an appropriate taxonomy to classify groupware applications and in this way decide on a set of operations, interaction mechanisms and support functionality to provide in a CSCW platform. In summary, a two level classification of groupware systems was identified, with a space - time division of the participants and a classification of the applications according to the level of support for collaboration technologies.

The following collaboration technologies were identified: core functionality; content; relationships; session persistence; supported actions; action parameters; access control; alert mechanisms; and awareness indications. This taxonomy of collaboration technologies is useful for categorising the level of support for collaboration provided by groupware applications, or alternatively to provide a checklist of features that an ideal CSCW platform should support. CSCL applications, and by extension MCSCL applications, build on groupware applications and this CSCW research. Therefore it is important to support both users working at all dimensions of the space-time classification and to provide support for these collaboration technologies in CSCL and MCSCL systems.

Furthermore, awareness mechanisms were noted as particularly important for collaboration, so features should be included in the platform to help the users answer the

questions: Who is present? Where they are working? And what they are doing?

The next two sections will ascertain the further requirements derived from CSCL and MCSCL that inform the design of a platform to support MCSCL application development.

2.2 Computer Supported Collaborative Learning

Computer supported collaborative learning, CSCL, is a related field, which developed by applying CSCW applications for educational and learning activities. Bricker et al. note that synchronous collaborative software can raise the educational value of computer-based activities [Bricker et al., 1998]. The aim of computer supported collaborative learning is to create new tools and applications to support collaborative learning while exploring the learning processes involved, ultimately developing best-practice and software for educational environments [Dimitracopoulou, 2005]. In such systems "collaborative learning is viewed as a pedagogical method that can stimulate students to discuss information and problems from different perspectives, to elaborate and refine these in order to re-construct and co-construct (new) knowledge or to solve problems" [Dimitracopoulou, 2005]. Discussion of pedagogy is deferred until the next section as it relates to MCSCL and mobile computer supported collaborative learning. This section describes a classification of CSCL applications and notes the current trends towards generic CSCL platforms for CSCL application research.

There are several taxonomies for CSCL e.g. [Jermann et al., 2001, Dimitracopoulou, 2005, Lonchamp, 2006, Dillenbourg, 1999]. This study focuses on the taxonomies described by [Dimitracopoulou, 2005] and [Lonchamp, 2006]. Dimitracopoulou's taxonomy classifies CSCL systems into two categories, text-production oriented and action-oriented collaborative systems [Dimitracopoulou, 2005]. Text-production oriented systems focus on collaboratively creating text documents or dialogues, while action-oriented collaborative systems start from a student's actions in the system, then captures the student's emerging knowledge and finally uses this knowledge-representation as the subject of an artifact-centred discussion.

The locus of CSCL research has moved from text production systems, often using CSCW tools, to action-orientated collaborative systems in the last ten years [Lonchamp, 2006]. Initial influential systems were text production systems, typically structured

chat systems. For example, Group Leader Tutor [McManus and Aiken, 1996] and C-Chene [Baker and Lund, 1996], which incorporated rigid turn-taking, built-in "sentence openings" or enforced roles to scaffold communication. Augmenting existing applications, for instance word processor applications with chat was a further avenue of research. Churchill et al. investigated embedding or annotating Microsoft Word documents with anchored conversations, embedding chat windows linked to sections of the document, so that scrolling the document view displays the conversations relevant to the displayed section of the document [Churchill et al., 2000]. Extending on this work, shared synchronous collaborative writing systems such as text-editors like Cosar [Jaspers et al., 2001] contain text-chat for communication and feedback. Coler, an Entity-Relations modelling shared workspace [Constantino-Gonzalez and Suthers, 2002] is an example of an action-oriented application, where users build Entity-Relationship (ER) diagrams as solutions to database modelling problems, later using these models as a basis for discussion and ultimately deriving a group ER solution. CoolModes [Pinkwart, 2003] a graphical modelling environment is an additional example of an action-oriented system.

Lonchamp divides CSCL systems into specific tools or applications and generic model-centred systems [Lonchamp, 2006]. Generic systems have the potential to overcome the limited reuse potential of specific tailored systems and the general trend in CSCL is towards generic system [Lonchamp, 2006]. While describing current trends relating to the design of collaborative learning systems [Dimitracopoulou, 2005] mentions that it is "important to provide flexible architectures and customisable tools".

Examples from the CSCW field of the progression from limited functionality applications to generic systems is seen in the development from early systems like GROVE [Ellis et al., 1991] to later more generic applications like TeamRooms [Roseman and Greenberg, 1996]. GROVE was an application for collaboratively creating document outlines while TeamRooms is an extensible system to share virtual meeting rooms which allows users to access multiple persistent collaborative applications and widgets, like shared text editors and notice boards.

A more recent example of this trend in CSCL is Omega+, a model-based generic system to support "the four dimensions of collaborative learning: the situation, the interaction, the process, and the way of monitoring individual and group performance

[Lonchamp, 2006]". The application is focused on text production and creating scaffolding applications for chatting in particular. The system explicitly specifies these four aspects in a set of models that serve as parameters for the generic environment. Users can use an administration client to create models to represent and specify the available actions in the collaborative environment. For example, they could specify whether the interaction is through structured chat, free chat, a debating model, or some other custom model.

Action-orientated systems are more complex and support a richer interaction model, so a platform to support CSCL should ideally support action-orientated systems because by supporting such systems it is possible to emulate support for text-production systems, as these latter systems require only a subset of the functions available to an action-orientated system. In addition action-orientated systems have at their core collaboratively constructed common artifacts, which through the users interaction serve as a basis of discussion and creating a shared understanding between the users. Furthermore, in this way action-orientated systems more fully support the constructionist, contextualised and collaborative pedagogies advocated in this thesis and described in 2.3.1.

This section was concerned with the classification and trends in computer supported collaborative learning, noting the move towards action-oriented systems and the greater emphasis on generic systems for creating flexible CSCL applications. The next section outlines the research area of mobile computer supported collaborative learning and includes an overview of the pedagogical foundations of both CSCL and MCSCL that relevant to this thesis. Furthermore the next section introduces some taxonomies for classifying MCSCL and mobile learning applications.

2.3 Mobile Computer Supported Collaborative Learning

The adoption of mobile devices is increasing, with [Bull et al., 2002] arguing that it is inevitable that every student will have a portable wireless device, and recent data on mobile phone ownership in developed countries would tend to support this claim [Gartner, 2007]. Most modern mobile phones include built-in cameras and additio-

nal communication support with MMS, Bluetooth, and Infrared (IR) [Gartner, 2005]. MCSCL is an emerging area of CSCL that aims to use mobile devices for collaboration and learning, specifically because of the level of adoption of mobile technology and the reality that processing power on mobile devices has reached a sufficient level to make useful applications feasible. In addition, there is widespread adoption of mobile phones by teenagers in the developed world which can be exploited in educational environments [Naismith et al., 2004] and people are familiar, and feel comfortable, with mobile phones. For example, many people who would not traditionally use computers and other related technologies habitually interact with a mobile phone [Geser, 2004].

The next section outlines the pedagogical foundations relevant to mobile computer supported collaborative learning applications. Then the following section provides a review of the literature in mobile learning in particular and describe related work in the mobile learning field that informs the design of the mobile learning applications developed on the MUSE platform.

2.3.1 Pedagogical Foundations

As with all technology to support learning, MCSCL software needs a sound pedagogical basis at its foundation. Several learning theories are relevant to MCSCL software design, particularly social-constructivist theories, encompassing constructivism, constructionism, and collaboration. While situated learning [Lave and Wenger, 1991, Brown et al., 1989] advocates contextualising learning in authentic settings, which is of particular interest from a mobile learning perspective.

Briefly, according to constructivist learning theories individuals construct new knowledge from their experiences, assimilating new knowledge to represent their own internal representation of the world or changing their internal representation in light of contradictions. This leads to the idea that collaboration is a powerful force for constructivist learning because in order to collaborate effectively users must negotiate a shared understanding; this process of negotiation and explanation leads learners to update their mental models until they establish a shared understanding. Particular learning theories related to social-constructivism that are relevant to MCSCL software given its collaborative emphasis include, Vygotsky's notion of the zone of proximal development, instructional scaffolding and situated learning. The zone of proximal development

(ZPD) [Vygotsky, 1978] is defined as the zone between tasks that a learner can perform alone and tasks that they can complete with help from a more knowledgeable peer. Instructional scaffolding [Wood et al., 1976] is in a sense a structured application of the ZPD theory. Scaffolding is the structured assistance provided by a teacher (or more knowledgeable peer) to support a student's learning. Scaffolding involves providing sufficient support and resources to promote learning when introducing new concepts and then gradually fading this support as the students improve. Instructional scaffolding as a structured application of the ZPD provides a model for the provision of software support for learning. From a MCSCL perspective the important aspect is that both theories have an emphasis on the collaborative construction of new knowledge, whether with an instructor or more knowledgeable peer. In addition, the role of more knowledgeable peer can change between the collaborators as different participants will have varying levels of expertise when performing specific elements of complex tasks.

The constructivist learning theory lends itself to pedagogic approaches that encourage learning by doing or experimental learning, leading to the constructionist theory of Seymour Papert who argued learning is best expressed when learners are actively creating a meaningful artifact, for example, writing a computer program, building LEGO™ objects etc. An additional benefit of the constructionist approach is that learners have a tangible object to show their peers, for example a digital video, which Kearney and Schuck mention as "an important part of the learning process" and gives the students a chance to further discuss their subject [Kearney and Schuck, 2005]. The ability to use the artifact as a basis for further discussion underlines constructionism as a social activity and is relevant to the design of action-oriented CSCL applications.

The act of creating a meaningful product during the construction process has an analogy in situated learning that recommends setting learning in an authentic context. Situated learning [Brown et al., 1989, Lave and Wenger, 1991], advocates setting learning in an authentic context which requires knowledge and involves social interaction and collaboration. The requirement to involve learning in authentic settings is particularly relevant to mobile learning (see section 2.3.2).

The common thread throughout all these learning theories and practice is the role of social interaction, negotiation of a shared understanding and collaboration in learning. The constructivist theories, including the ZPD and instruction scaffolding, clearly

demonstrate the collaborative aspect to learning. While Lave et al.'s theory of legitimate peripheral participation [Lave and Wenger, 1991], or more generally situated learning [Brown et al., 1989], involve social interaction, contextualisation in authentic activities, and collaboration.

This thesis follows the argument from Patten et al that the best use of mobile learning applications are informed by a constructionist, contextualised, and collaborative approach [Patten et al., 2006]. The tools should enable a constructionist approach and allow peers to create tangible artifacts that can form the basis for negotiation of meaning and help peers establish a shared understanding. Secondly from contextualised prospective the tools should exploit the mobility of the devices to situate the learners and learning activities in authentic contexts. While the tools should enable collaboration, essentially the *sine qua non* of a socio-constructivist approach, thereby allowing peers to work together during the construction process.

This subsection provided a high-level overview of the pedagogy related to CSCL and MCSCL. The next subsection introduces the mobile learning field and describes related work that informs the design of the MCSCL applications developed as part of this thesis.

2.3.2 Mobile Learning

The attributes of contemporary mobile devices which include, portability, small screen sizes, diverse input and output functions, and wireless communications facilities, suggest several potential affordances [Gibson, 1977, Norman, 1993] for learning. The most obvious affordance of mobile devices is the mobility of the technology. The ability to use mobile devices anytime or anywhere opens new situations and opportunities for deploying learning software. Traditional educational tools, books, pen and paper, could claim the same affordance of mobility, but are essentially passive or static technologies. For instance you cannot update a book in real-time, or you cannot access information that is not available in your surroundings when using just a pen and paper. "On the other hand, consider that all the informational and communicative power ... available to any m-learner, at any point in space and time" [Alexander, 2004].

"Mobile devices participate in a network that is overlaid in the same physical space in which students and teachers participate socially in teaching and learning" [Roschelle,

2003] so as to provide two levels of participation at the same time and same place, the normal classroom participation and the overlaid technology mediated space. Hence mobile learning research has the potential to create applications that use the affordances of mobile technology, which differ from desktop-based CSCL [Roschelle and Pea, 2002].

Furthermore, [Chen et al., 2003] identify six unique characteristics of mobile learning environments: urgency of learning need; initiative of knowledge acquisition; mobility of learning setting; interactivity of the learning process; situating of instructional activity; and integration of instructional content; of which three salient higher-level characteristics are extrapolated in this thesis. Firstly, communications facilities enable users to ask questions, of peers, experts and query other sources of information. Secondly, mobility enables situating instructional activities in authentic contexts by embedding them in daily life or on field trips in both an opportunistic or planned fashion. Thirdly, the combination of mobility and communications facilities enable users to link situations or contexts involving problem solving scenarios with the ability and knowledge to solve these problems in context, for instance communicating or collaborating with knowledgeable peers to solve a new problem encountered in the environment. To this list may be added the media capture facilities on mobile devices, which enables the use of a richer set of information.

Mobile learning is an emerging and growing field so as with CSCW and CSCL different taxonomies have been put forward for classifying, and developing, mobile learning applications, most notably by [Roschelle, 2003] and [Naismith et al., 2004]. Moreover the functional-pedagogical framework for mobile learning proposed by [Patten et al., 2006] purports that unique attributes of handheld devices support constructionist, contextual and collaborative learning experiences.

Naismith et al. structure their classification of mobile learning activities around the main themes or learning theories they regard as relevant to learning with mobile technologies, the main themes they identified are: behaviourist; constructivist; situated; collaborative; informal and lifelong; learning and teaching support [Naismith et al., 2004]. They then group mobile learning activities under one of these categories. This classification is summarised in Table 2.2 on page 26 adapted from [Naismith et al., 2004]. Rochelle's classification concentrates on the most developed mobile learning applications: classroom response systems, participatory simulations,

Theme	Activities	
Behaviourist Learning	Drill and feedback	Classroom response systems
Constructivist Learning	Participatory Simulations	
Situated Learning	Problem and case-based learning	Context Awareness
Collaborative Learning	MCSCL	
Informal and Lifelong Learning	Supporting intentional and accidental learning episodes	
Learning and Teaching Support	Personal Organisation	Support for administrative Duties

Table 2.2: An Activity-Based Categorisation Of Mobile Technologies And Learning [Naismith et al., 2004]

and collaborative data gathering (probeware). Patten et al use seven categories to classify MCSCL applications, administrative, referential, interactive, microworld, data-collection, location-aware and collaborative [Patten et al., 2006]. In addition, the categories of data collection, location aware and collaborative present the best use of mobile technology for learning especially when they are informed by constructionist, contextualised and collaborative learning theories [Patten et al., 2006].

Nevertheless "there is, as yet, no overarching 'theory of mobile learning'" [Naismith et al., 2004], however, examining the above taxonomies it is clear that there is significant overlap between them. This thesis agrees with the argument from Patten et al. which is that the applications that make the best use of mobile technology for learning are informed by constructionist, contextualised and collaborative learning theories [Patten et al., 2006]. Therefore by way of example this section describes some previous MCSCL applications under the data-collection, collaborative and location aware categories [Patten et al., 2006].

2.3.2.1 Mobile Learning Applications

In order to provide context by way of example, this section describes some previous MCSCL applications under the data-collection, collaborative and location aware categories.

The defining attribute of the data-collection category of applications is that they use the various input facilities of mobile devices to record data about their environment. Typical input options available on mobile devices include text keypad, stylus, touch screen, camera, microphone, or peripheral sensors. Ideally, the inputted or collected data is then the basis for further study, collaboration and reflection. Two different examples of such uses are probeware and multimedia creation applications.

Probeware [Tinker, 2000] is mobile technology augmented with sensors and probes to provide a contextualised and situated learning opportunities, where users can record data from their surroundings and use this as the basis for further enquiries or later reflection. Tinker states that the role of probeware is to "increase the amount of experimentation students can undertake, and to show the relationship between the experiment and an abstract representation of the data" [Tinker, 2000] so in this way connect the user with the experiment under study. For instance, the SENSE project [Fraser et al., 2005] uses this approach, where users gather environmental data, such as local pollution data, with mobile devices. As the users evaluate and analyse the resulting data, and reflect on the data-collection process itself, Fraser et al. state that it "impacts on children's understanding of the scientific process" [Fraser et al., 2005].

Applications in the collaborative category aim to use the communications facilities on mobile devices to provide collaborative and constructivist applications that build on the notion that learning is socially constructed [Vygotsky, 1978]. From a collaborative, constructivist and situated learning perspective, participatory simulations [Colella et al., 1998], like Geney [Danesh et al., 2001] and Cooties [Soloway et al., 2001] offer useful examples of successful applications for mobile learning. Geney is a collaborative mobile application to help children learn basic genetic concepts using Palm™ handheld computers, that the authors note enable rich social interactions to take place. Users are able to construct their knowledge of genetics through exchanging and mating digital fish with a simplified genome, and observing the traits of the offspring produced. The goal of the project is to produce digital fish with particular traits so that users have to collaborate to mate digital fish to accomplish this goal. In this manner the application is both constructivist and collaborative. 'Cooties' is a mobile application to model the spread of disease. Users beam messages via IR that are either germ-free or germ-laden, later the users study their interaction history to discover patterns in the spread of the disease. Users can view a concept-map representing the people they met, and "this exercise enables children to see that meeting one person really means meeting all the people who that person has met previously" [Soloway et al., 2001]. As users socialise and engage in the simulation, they grow to understand how disease spreads and experience how easily a disease spreads in a population.

In the location aware category learning applications make use of positioning in-

formation from sensors in the handheld devices. Applications in this category use positioning systems in the devices to enhance contextual learning activities by enabling the learner to interact with their environment in a context aware manner. For example presenting appropriate reference information based on location or encouraging the learners to explore their environment. Applications in this area range from museum guides to augmented environments for treasure hunts etc. For example MoULe [Arrigo et al., 2007] is a Mobile and Ubiquitous Learning system, which enables users to edit and share location based digital objects which are concept-maps and wiki pages using PCs and mobile phones with GPS and built-in cameras. All these digital objects contain location information. In addition, the system includes a learning management system so that users can access these documents online using moodle software. Ambient Wood is another example, where users explore a woodland using their handheld devices, which provides appropriate multimedia content to the users based on their location in the woodland [Cole and Stanton, 2003].

2.3.3 Summary Of Findings In Mobile Learning

In review, mobile learning allows new opportunities for learning that are not available using traditional PC based CSCL applications as it supports greater contextualisation of learning activities in authentic settings out in the field. In addition to their anywhere, anytime nature mobile devices are less intrusive than PCs and allow a technology mediated communication or information space to be overlaid on the physical setting.

There have been some attempts to use MCSCL to bridge the gap between users on PCs and mobile devices, for instance, "StudentPartner" [Hwang et al., 2007] is an integrated multimedia forum, which allows users to capture media files with a mobile device and upload them to a shared database that users access through PCs and mobile devices, to create a shared discussion forum. Typically applications are targeted as specifically mobile learning or as PC-orientated CSCL.

Further analysis of the mobile learning area results in several requirements that inform the design of a middleware to support MCSCL applications. Firstly, as outlined for CSCL, a platform to support MCSCL applications development should support constructionist, contextualised, and collaborative activities for learning. In addition, data-collection, collaborative and location aware applications are noted as the best use

of mobile technology [Patten et al., 2006] so supporting these modes of interaction with mobile technology is an advantage in an MCSCL platform. The specific case of data-collection for multimedia capture, and collaboration for digital multimedia production, enabled by mobile technology are of particular interest as together they provide a good example of constructionist, contextualised and collaborative activities.

2.4 Conclusion

This thesis is concerned with designing an platform for MCSCL applications, especially for digital narrative production. This chapter presented a review of several areas of research most pertinent to the design of such tools including computer supported collaborative working, computer supported collaborative learning, mobile learning and the pedagogical theories relevant to CSCL and mobile learning. This thesis found a set of functional requirements for creating such a platform for MCSCL applications.

In review, one goal of SMART and the DNT is to help users to create digital narratives using images and sound in a collaborative fashion. A digital narrative is equivalent to a shared document in the CSCW sense thus lessons learnt from the CSCW field are applicable to development of such tools. Computer supported collaborative work (CSCW) is a mature field that concentrates on developing tools to enable collaborative working. This thesis uses CSCW to refer to the research area, groupware to refer to collaborative applications, and collaboration technology to refer to the finer-grain functionality of groupware applications. An established taxonomy of CSCW categorises applications according to a space-time division of the participants using the systems, with a spectrum that ranges from same time and place around a computer screen to different time and place distributed work [Johansen et al., 1991, Ellis et al., 1991, Rodden, 1991, Rama and Bishop, 2006]. In addition, several studies further classified groupware applications according to their functionality e.g. [Rama and Bishop, 2006, Mittleman et al., 2008, Rodden, 1991, Ellis et al., 1991]. Schmidt and Rodden produced a synthesis of the CSCW literature and their experience to provide a list of requirements for a CSCW platform [Schmidt and Rodden, 1996]. They argue that such a platform should provide a set of generic techniques for communication accessible to groupware applications. In total they recommend including support for: informal interaction;

information sharing and exchange; decision making; coordination and control protocols; and supporting domain directories. Mittleman et al. identify nine dimensions in groupware systems: Core functionality, content, relationships, session persistence, supported actions, action parameters, access control, alert mechanisms, and awareness indicators [Mittleman et al., 2008]. The principle aim of this thesis is to design and build a middleware to support MCSCL applications. The MCSCL middleware must be flexible to support as wide a variety of collaborative mobile learning applications as possible, therefore informed by the CSCW literature it should support users working in the four dimensions of this CSCW taxonomy and provide generic application level CSCW functions, e.g. core functionality; content; relationships; session persistence; supported actions; action parameters; access control; alert mechanisms;and awareness indicators [Mittleman et al., 2008]. CSCL applications, and by extension MCSCL applications, build on groupware applications and this CSCW research. So it is important to provide support for these collaboration technologies in CSCL and MCSCL systems.

Awareness support was described in detail as a necessary feature of collaborative groupware applications and so should be supported by MUSE. Some concrete awareness mechanisms that were described include the recommendation that groupware systems should include a communication or messaging system and that specific provision for identity and embodiment should be included.

A noted recent trend in CSCL application development is towards generic extensible systems that can support several integration mechanisms. Such systems have the advantage of flexibility and so are also suited to the mobile technology field that includes many and various heterogeneous components and systems. From a CSCL perspective this section recommended developing applications and tools in the action-oriented [Dimitracopoulou, 2005] category under Dimitracopoulou's classification and as a generic system [Lonchamp, 2006] according to Lonchamp's classification.

Mobile learning or MCSCL extends on the work of CSCL but aims to leverage off the unique affordances of mobile devices for learning. Applications in this area are now feasible given the greater processing power and adoption rates of mobile devices today. The level of adoption of mobile devices continues to increase, while the latest generation of mobile devices also have sufficient processing power available to make creating usable mobile applications feasible. Furthermore, the unique combination of

attributes on mobile devices, including, small size, diverse input and out capabilities like media capture facilities, and powerful communications tools afford applications that differ to traditional desktop based CSCL.

There is no overarching pedagogical theory of mobile learning but several learning theories are relevant to the field, particularly social-constructivist theories, encompassing constructivism, constructionism, and collaboration. Instructional scaffolding is a method of supporting these learning theories in practice. In addition, situated learning advocates contextualising learning in authentic settings, which is of particular interest from a mobile learning perspective. This thesis follows Patten et al's taxonomy of mobile learning and agrees that the categories of data collection, location aware and collaborative present the best use of mobile technology for learning especially when they are informed by constructionist, contextualised, and collaborative learning theories [Patten et al., 2006]. Probeware, participatory simulations and multimedia creation applications are examples that would satisfy these criteria.

In conclusion, this chapter found a set of functional requirements for a MCSCL platform after reviewing the literature from CSCW, CSCL, MCSCL and mobile learning and related pedagogical theories. The first requirement is that the MCSCL platform should support all four dimensions of the space-time classification of groupware system. Secondly, the platform should support all nine dimensions of groupware technology Mittleman et al. [2008]. Thirdly from a CSCL and mobile learning perspective it should support constructionist, contextualised and collaborative activities. Fourthly it should support mobile applications for data collection, location awareness and collaboration. The collaborative aspect is also relevant to the previous findings from CSCW and CSCL. To support data-collection it should provide an extensible abstraction over underlying hardware so that additional functionality can be added as new hardware and sensors become available. In order to accommodate location aware applications a similar extensible abstraction over the locations sensors should be included.

These requirements provide a guide to examining related work on mobile middleware systems. The next chapter expands on the theme of designing middleware or platforms to support application development on mobile devices by describing related work that informed the design of the MCSCL platform that is the focus of this thesis.

Chapter 3

State Of The Art

While there is already a lot of collaborative applications available for mobile devices, typically these are built from scratch and tailored to specific platforms. Collaborative applications often contain common functionality, for example, networking support, which would be better provided as part of a more generic solution. It has been noted that there is a lack of middleware to provide generic support for such common functions e.g. [Springer et al., 2008, Roth, 2002b], particularly spanning fixed and mobile contexts [Farooq et al., 2002]. Springer et al. underline this lack stating: "An open and customisable environment for mobile collaborative applications is still missing" [Springer et al., 2008]. According to Roth "the lack of middleware platforms leads to monolithic and unstructured code" [Roth, 2002b] and that any changes to the communications infrastructure require applications to be re-engineered [Roth, 2002b]. In short, this lack of middleware support for mobile collaborative applications results in a twofold development process as the developer has to build the application itself and also "has to realise at least a rudimentary middleware platform" [Roth, 2002b].

The previous chapter identified the theoretical foundations relevant to developing a middleware to support MCSCL applications. These foundations included the aggregate requirements in terms of CSCW, CSCL, mobile learning and pedagogical support that such a middleware should support. This objective of this chapter is to review the state of the art in mobile middleware to support MCSCL applications. Therefore, this chapter reviews the related work on middlewares for mobile collaborative activities. These systems are then compared according to the type of middleware architecture each system uses, and the degree of flexibility afforded in the distribution model, communication

infrastructure, sharing model, concurrency model and synchronisation model used by each system. The chapter concludes by comparing the evaluation methods used in each of the studies of these systems to inform the design of the MUSE evaluation.

3.1 Middleware For Mobile Collaboration

The mobile environment comprises disparate hardware, software and networks so software development in this space must deal with heterogeneity in several dimensions. Hardware can vary from PC and laptops with plenty of processing power and memory, to PDAs and mobile phones with less resources and capacity. Operating systems vary widely from PCs, laptops, PDAs and mobile phones. Typically operating systems vary from one mobile phone handset manufacturer to another. Furthermore there is a lack of standards to access the hardware or OS services. On the PC there are reliable high speed networks using Ethernet and TCP/IP technologies. But on mobile devices networks support can range over several different protocols and technologies with different characteristics, e.g. Bluetooth, IrDA, WiFi, and 3G. Roth identifies several challenges to developing applications in the mobile technology space including: hardware limitations; missing IP support; missing platform independence and interoperability; missing support from manufacturers; and incompatible reference models [Roth, 2002b]. Furthermore the mobile technology field is developing quickly with new models (often incompatible) available all the time from most manufactures. At the moment the heterogeneity of mobile devices is increasing as several new smart phones operating systems have entered the market, including: the iPhone OS; several versions of the Android operating system; and Microsoft Windows Phone 7 Series. Developing software that can cope and last in such a rapidly changing technical landscape is a further challenge.

Now by focusing on software development for mobile devices, additional challenges become apparent i.e., heterogeneity, mobility and network or hardware limitations. This is underlined by Cinque et al. who state that these challenges "arise from the combination of heterogeneity, dynamism, context-awareness, and mobility" [Cinque et al., 2005]. Guicking and Grasse summarise that the specific characteristics of mobile environments include "the limitations of processing and battery power, memory

Hardware	Software	Network
Reduced GUI capabilities	Heterogeneous OS availability	Differing networking technologies
Small screen and low resolution	Lack of OS provided services	Poor bandwidth
Rudimentary or no keyboard	Rudimentary file systems	Poor latency time
Low processor perfomance		Poor reliability
Small memory		Mobility of devices
Battery or power issues		Data Interoperability

Table 3.1: Summary Of Mobile Challenges

and user interface capabilities, (b) unreliable network conditions, and (c) a highly dynamic environment during application run-time including for example changing user and device locations" [Guicking and Grasse, 2006]. Table 3.1 on page 34 summarises these challenges under the headings, hardware, software, and networking.

"Groupware platforms help to decrease the development costs for a synchronous groupware drastically as they take over a number of tasks of synchronous sessions" [Roth, 2003]. As a result of using such groupware platforms the application developer can concentrate on the development of user applications [Roth, 2003, Bellavista et al., 2009]. The challenges to developing applications in the mobile computing environments described above, and the aforementioned lack of middleware support for developing application in this area provide a strong motivation to develop middleware support for application developers in the mobile computing domain. Furthermore, in recent years it is starting to be widely recognised that the next crucial technological challenge is "to identify, propose, validate, and spread the adoption of open and interoperable software-support solutions specifically designed and implemented for wireless-enabled portable devices" Bellavista et al. [2009]. While there are a wide variety of frameworks for groupware they are primarily focused on the PC environment, support for heterogeneous devices is often an afterthought or later extension [Guicking and Grasse, 2006]. Examples of this approach include Pocket DreamTeam [Roth, 2003] which uses a proxy to bridge the gap between mobile and fixed networks. As the areas of groupware or mobile collaborative applications are not well supported across heterogeneous environments involving mobile devices and traditional PC-based computing environments, so a focus on middleware for these heterogeneous environments, with an emphasis on groupware, is warranted.

Previous work in the mobile domain often focused on peer-to-peer and ad hoc net-

works [Kortuem, 2002, Cattelan and da Grace Pimentel, 2008, Zurita and Nussbaum, 2006]. This thesis is concerned with supporting users collaborating between both fixed PC devices and mobile devices, however, previous research in the peer-to-peer and ad hoc networking areas does inform the design of the middleware developed during this thesis in that it serves to highlight the challenges of developing for wireless mobile devices and highlights evaluation metrics to measure middleware performance.

There is also a substantial body of work at the lower-levels focusing on the physical communications links e.g [Stuedi and Alonso, 2005, Premadasa and Landfeldt, 2005, Blum et al., 2010]. This thesis aims to support application level developers to create sophisticated MCSCL applications, so previous work at this lower-level of abstraction is outside the scope of this research. Furthermore, there has been research in extending particular distributed software paradigms to mobile or ad hoc networks, for example, distributed publish – subscribe systems [Pongthawornkamol et al., 2007], delay tolerant messaging approaches [Fall, 2003, Guidec and Mahéo, 2007] or Software-as-a-Service for mobile computer environments [Anerousis and Mohindra, 2006]. Another recent trend for mobile middleware research has been developing mobile middleware for social networking e.g. [Gupta et al., 2009]. This thesis is suggesting a large extensible middleware for MCSCL so research focusing on particular distributed software paradigms is too narrow a consideration and is not covered here in any detail. However such paradigms do provide partial solutions to implementing the MUSE middleware. Furthermore the work in the area of social networking presents significantly different challenges to MCSCL that this area is not described in detail either.

A popular approach to classifying mobile middleware systems is based on the programming paradigms that dominate a particular middleware system [Gaddah and Kunz, 2003, Hadim et al., 2006]. Gaddah and Kunz conducted a survey of middleware paradigms for mobile computing with a particular emphasis on mobile and ad hoc networks (MANET) [Gaddah and Kunz, 2003] and produced a classification of middleware systems in this domain. They classified mobile middlewares under four headings: reflective; tuple space; context-aware; and event based. Hadim et al follow a similar approach but suggest a classification with the following categories: event based and message oriented middleware; peer-to-peer middleware; component-based and mobile agents middleware; tuple spaces based middleware; information sharing based

middleware; and virtual machine based middleware [Hadim et al., 2006]. To provide context a short definition of these terms is provided here.

Reflective middlewares use the reflection programming technique to allow systems to access the middleware as a white-box thereby enabling applications to access, reason about and change its behaviour. This flexibility is useful in mobile ad hoc systems as they need to be able to adapt to heterogeneous hardware and network characteristics.

A *tuple space middleware* is an asynchronous connectionless programming paradigm that supports decoupling between client and server. Tuple spaces where first described as part of the Linda coordination language [Gelernter, 1985] and Linda in Mobile Environment (LIME) [Picco et al., 1999] is an extension of this model to mobile systems. Communication is not direct but mediated through a tuple space, which is a globally shared, associatively addressed memory space. A tuple space can be implemented as a repository of tuples, which are essentially vectors of typed values or fields. The client applications can access this shared repository to create new tuples and read or take existing tuples from the repository. These properties of asynchronous connectionless and decoupled communication are useful when dealing with dynamic networks that must deal with frequent disconnections etc.

Context-aware middleware are middleware systems that expose context information about the system to applications. Applications can then use this information to adapt their behaviour accordingly. For example, if there is no network connectivity then the application could adopt a caching strategy. In addition, context-aware middleware often provide an abstraction layer over various sensor systems. The most widely studied systems in this category are location-aware middlewares which exploit location information to discover neighbouring services or broadcast messages to nearby users for instance.

In short, *event based middlewares* or *message oriented middlewares* are asynchronous and connectionless based coordination and communications models. An event-based system uses the publish–subscribe paradigm to decouple components. In this way the communication does not rely on the request–reply communication paradigm widely employed in other approaches but is based on an event notification scheme. In this scheme event generating objects publish events as they occur, objects interested in certain events subscribe to these events. The events are then propagated to the sub-

scribers in an asynchronous, many-to-many communication model. As the subscribers and publishers are decoupled and communicate asynchronously it is a useful approach to designing middlewares for mobile systems with unreliable networks.

Information sharing based middleware approaches deal with the problem of frequent disconnections in mobile environments by maximising the availability of data by replicating data structures on the various devices. One problem with the approach is ensuring data integrity and consistency of replicas. A *peer-to-peer middleware* is a decentralised approach with each node in the system capable of acting as both as client and a server. Middlewares in this category provide abstractions to the application layer to cope with mobility, dynamic routing, and resource discovery.

Component-based and mobile agents middleware is a programming technique where the system is designed to be as modular as possible. These modules can then be distributed through the network using mobile agents. This keeps the middleware light weight as modules can be installed only when they are need. *Virtual machine based middlewares* are a related approach that use interpreters and mobile agents, which allows applications to be written in separate small modules that get injected and distributed through the network using tailored algorithms such as overall energy consumption.

Turning from the classification of middlewares, to their comparison and evaluation, the practice is to compare individual mobile middleware systems against an evaluation framework. Table 4.1 on page 83, Table 4.2 on page 85 and Table 4.3 on page 86 provides a summary of the comparison of the middlewares discussed in this thesis. Gaddah and Kunz's evaluation framework examines the following aspects: synchronous connection based or asynchronous connectionless; reconfigurable; adaptable; awareness and lightweight [Gaddah and Kunz, 2003]. While Hadim et al. use an evaluation framework that examines the following dimensions of the mobile middlewares: power awareness; openness; scalability; mobility; heterogeneity; ease of use [Hadim et al., 2006]. While these evaluation criteria are useful, the approach defined by Guicking et al. [Guicking and Grasse, 2006], and elaborated in [Guicking et al., 2008], is focused more towards groupware applications. They identify facets of groupware applications which differ from one implementation to the next. These facets they label *variation points*, which the underlying middleware should support. The five variation points listed are: the distribution model; the communication infrastructure; the sharing model; the concur-

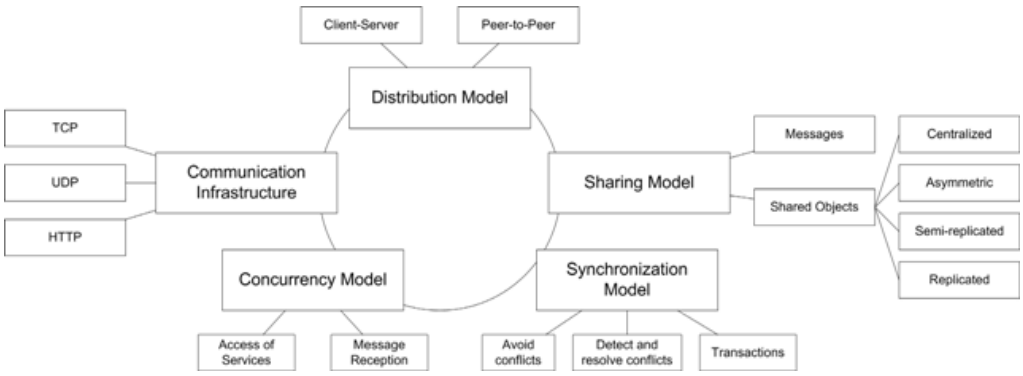

Figure 3.1: Variation Points And Realisations [Guicking et al., 2008]

rency model; and the synchronisation model. They evaluate middleware to support groupware according to the level of support provided at each of the variation points. Figure 3.1 on page 38 depicts this classification scheme, showing the variation points at the centre and possible implementations at these variation points. This thesis uses a combination of middleware types and the scheme from Guicking et al. to compare the related work discussed here.

In summary, the *distribution model* is the logical interaction model for the groupware applications, variations here are client-server and peer-to-peer models. The *communication infrastructure* is the low-level networking support the middleware provides, for example, the communications protocols and marshaling formats supported. The *sharing model* is the level of abstraction used by shared data in the groupware system, options here include simple message exchange to distributed shared objects. The *concurrency model* is the support provided for concurrent access to shared data and services. The *synchronisation model* is the support provided to ensure consistency of data in case of concurrent modification by resolving and managing conflicts, solutions under this heading include, locking mechanisms or transaction support[Guicking et al., 2008].

This section described the motivation for creating a middleware for MCSCL applications and a taxonomy for comparing related work on middlewares for mobile collaboration. This section also provided scope for the related work which is described in the next section.

3.2 Review of Systems

This section describes several middleware systems that relate to, or inform the design of MUSE presented in chronological order. The emphasis of this section is on the structure and implementation details of the middleware systems. The evaluation of these middlewares is described in the next section with respect to informing the design of the evaluation of the MUSE platform.

XMIDDLE [Mascolo et al., 2001a,b] is an information sharing based middleware designed to support collaborative applications on peer-to-peer mobile networks. Applications using this middleware share a distributed memory which is organised in a tree structure using XML. Applications can link to portions of this shared tree and in this way interact with the data in this branch or sub-branches. Peers can interact without being connected to other peers, when they later connect to the other peers a reconciliation algorithm detects and resolves any conflicts in the tree structure which occurred while the peer was disconnected. Nevertheless, the conflict resolution solution is not complete and requires application specific domain knowledge or policies to resolve some conflicts. XMIDDLE is a session and presentation layer middleware and does not explicitly deal with any networking problems, however the reference implementation is restricted to UDP at the transport layer, IP at the network layer and WiFi at the data-link and physical layer.

Proem [Kortuem, 2002] is a middleware for MANET systems. Communication is managed by a high-level transport protocol which provides an unreliable, connectionless, asynchronous messaging protocol. Above this transport layer is a data protocol to enable peers to share and synchronise data. Also at this layer is a presence protocol that enables peers to announce their presence and discover other peers, and a community protocol to support group formation. Proem uses a service oriented architecture with an event-based notification system to enable peers to access new computational resources. Applications in Proem are called peerlets, which are executed by a peerlet engine that is included as part of the middleware. Peerlets are designed as drop-in modules and can be exchanged between peers at run time. Proem was used as part of a software engineering course to enable students to create mobile peer-to-peer applications, no further evaluations were presented.

The Network Kernel Framework (NKF) [Roth, 2002b] is another middleware for mobile and ad hoc scenarios. The NKF operates between low level services provided by the OS and the application layer. There are four fixed modules that provide the basic functionality of the middleware, a look-up module to search for other communication parties inside the network, a negotiation module to decide the particular communication protocols and formats to use in the communication, a service module performs some additional setup for specialised network connections, and inspection module provides higher-level application with information about the lower-level connection. The NKF provides three types of optional modules to support communication, the network modules are the low-level implementation of the network protocols, the processing modules pre-process data before transmission, e.g. compression or security, and the codec modules ensure the data is sent and received in compatible formats. The NKF is network focused and provides transparent networking support to applications, communication is message based, RMI or RPC are not supported but must be implemented at the application layer.

Pocket DreamTeam [Roth, 2003] middleware extends DreamTeam [Roth and Unger, 2000] a decentralised peer-to-peer middleware designed to support groupware on PCs. This middleware uses the NKF described above for communication and basic services. The Pocket DreamTeam middleware adds groupware support so includes services for the coordination of the participants, group and user profile management, session management, announcement services and pre-defined elements to achieve group awareness e.g. participant lists, mouse pointers etc. Communication between mobile peers and fixed peers operates through proxies. The proxies act on behalf of a mobile peer if it is disconnected, and synchronises with the mobile device when it becomes re-connected. Communication between peers is achieved through method calls which are executed synchronously on replicated resources. A shared whiteboard drawing tool, and a user status application were built to evaluate the Pocket DreamTeam middleware [Roth, 2003]. The authors also looked at the percentage of code that could be reused while developing these applications.

Zurita and Nussbaum [Zurita and Nussbaum, 2006] describe an ad hoc wireless network architecture for face-to-face mobile collaborative applications. This is also a peer-to-peer architecture, in which each node is identical and contains an *operator*

component to mediate communication between the application layer and the other nodes in the system. This architecture emphasises group management and overlays a hierarchical group management system. Any node can contain a *coordinator* component which manages group formation and each group must contain one coordinator component. To create a group hierarchy the coordinators from each group form a new higher level group, the existing groups then form sub-networks in the overall system. Zurita and Nussbaum evaluated this middleware by measuring group formation time and transfer time between a coordinating node in the system and the end points. Three different message sizes were used, 128 bytes, 256 bytes, and 512 bytes. Communication is point to point and number of messages required for group formation increases as groups or nodes are added, therefore as number of groups increases the time taken to reliably send messages to all nodes in the system increases e.g. this study shows that with 15 groups the time taken to send a 512 byte message approaches 10 seconds. The salient feature of peer-to peer architectural approaches for MCSCL applications is that to synchronise data between nodes requires an increase in messages proportional to the number of peers. This is particularly problematic in mobile networks, which suffer from poorer latency and lower bandwidth than fixed networks. This middleware provides support for grouping and network communication, but it does not include any additional functionality in the middleware for groupware, this functionality must be implemented in the application logic of the collaborative applications.

The Agilo groupware framework [Guicking and Grasse, 2006] comprises a layered architectural model with three tiers. The bottom tier manages message exchange between peers, to provide network transparency it contains a network abstraction interface and networking protocol implementations. The middle tier contains the core framework which routes messages to application layer modules and provides a local module look-up service and a client registry to manage dynamic grouping of clients. The top tier is an application layer tier that contains optional services. Agilo is a message-based architecture, that uses a peer-to-peer distribution model. Messages are processed sequentially and multiple accesses to shared data can be encapsulated in a simple transaction. However, this method of managing access to shared resources does allow synchronisation conflicts so applications must implement their own higher level synchronisation model if the application needs to avoid or detect conflicts. The

Middleware	Type	Distribution	Communication
XMIDDLE	Information Sharing, P2P	P2P	UDP over WiFi
Proem	Event-Based, Virtual Machine + P2P	P2P	Asynchronous, Connectionless
Network Kernel Framework	Message-Orientated, P2P	P2P	Multiple Protocol support
Pocket DreamTeam	Information sharing, P2P	Client Server proxy	Network Kernel Framework
MCSCL-CS	Message-Orientated, P2P	P2P	TCP/IP over WiFi
Agilo	Message-Orientated, P2P	P2P	Asynchronous, Multiple Protocol support
GMAC	Message-Oriented, P2P	P2P	TCP/IP over Ethernet
ESPERANTO	Tuple-Space, Context Aware	Client-Server	Asynchronous, connectionless , WiFi, Bluetooth
Mobilis	Message-Orientated, Event-Based	Client-Server	Multiple Protocol support

Middleware	Sharing	Concurrency	Synchronisation
XMIDDLE	XML messages	Shared access to tree structures	Tree Reconciliation algorithms
Proem	Message-passing	N/A	Data Protocol
Network Kernel Framework	Message-passing	N/A	N/A
Pocket DreamTeam	Replicated Objects, RMI	Optimistic Concurrency control	Optimistic Concurrency control
MCSCL-CS	Message-passing	N/A	N/A
Agilo	Message-passing, Shared Objects	Incoming Message Order	Simple Transactions
GMAC	Message-passing	N/A	N/A
ESPERANTO	Shared Objects	N/A	N/A
Mobilis	Message-passing	N/A	N/A

Table 3.2: Related Work

evaluation of Agilo involved building two groupware applications to use the framework.

Gotthelf et al. describe a decentralised peer-to-peer middleware for groupware applications called GMAC [Gotthelf et al., 2007]. This middleware is a message-oriented approach, with all peers establishing two communications links, one for control messages and one for data. The control link is used to create and manage a balanced binary tree overlay network between connected peers. This middleware implements algorithms to deal with peers becoming disconnected and recovery from failure of peers. However, this is an application layer approach to supporting groupware applications and is implemented on PC and uses the Internet as the communication medium, it is not implemented in a mobile environment.

Gotthelf et al. evaluated their decentralised middleware for groupware applications in two stages, first they created two groupware applications to work on their middleware to discover how GMAC handles synchronous and asynchronous communications requirements in applications designed for end users. In the second stage they gathered data for several performance metrics using a network simulator including: End-to-end group propagation delay; throughput; protocol overhead; group latency; resource utilisation and failure recovery [Gotthelf et al., 2007]. End-to-end group propagation delay is defined as the time required by a random node to send 100 KB to all the group members. Throughput is measured as kilobytes per second transmitted between peers as the number of peers in the system increases. Protocol overhead is the network traffic that is generated by GMAC that does not represent useful application data, i.e. messages to control or maintain the overlay structure. Group latency is the longest end-to-end delay in seconds between any pair of nodes, this value is plotted against the number of nodes in the system and shows that group latency exhibits $O(log(n))$ increase in delay as nodes are added. The resource usage is a measure of the network resources consumed in the process of data delivery to all receivers and is expressed as a percentage of Minimum Spanning Tree (MST) latency. Failure recovery is the time needed to recover from simultaneous failing hosts in a a group with 100 nodes.

ESPERANTO is a middleware platform designed to support service-orientated architectures in nomadic computing environments [Cinque et al., 2005, Cotroneo et al., 2007]. To allow for disconnections ESPERANTO uses an asynchronous, connectionless, tuple-space based architecture with resources centralised on the fixed infrastructure.

To support mobility, the peers are network-aware so provide a hand-over mechanism to change between networking protocols when appropriate, e.g. WiFi and Bluetooth. Data is exchanged using a Remote Method Invocation (RMI) scheme, which uses proxies on both sides of the client-server computation so client applications can access services in a location transparent manner. However, ESPERANTO is a generic middleware solution optimised for RMI and is not specifically designed for groupware or multimedia. Therefore the platform is evaluated in terms of invocation latency and throughput using small data sizes from 1 byte to 10 kilobytes only.

The Mobilis architecture for mobile collaboration services [Springer et al., 2008] is also a layered approach with three layers: The device OS Layer; Basic Services Layer; and Mobilis Service Layer. The device OS layer contains communications and services provided by the mobile OS e.g. GPS, WLAN, GSM, and Bluetooth. The basic service layer contains implementations of services that are dependant on the OS layer. The Mobilis service layer contains services to support mobile collaboration, for example, group chat, group management, media sharing and location services. Mobilis uses an event-based and message oriented architecture. This architecture is not available on mobile phones or devices with low processor and memory as it is dependent on an implementation of eXtensible Messaging and Presence Protocol (XXMP) and the XXMP Extension Protocols (XEP) libraries to provide publish–subscribe and other collaboration support services. However, Mobilis is currently implemented only on PCS using J2SE and was tested on the Android platform. An application that allows all the members of a tourist group to be aware of each others activities on a map was created to validate the functioning of the Mobilis architecture.

The MobiVine[Agarwal et al., 2009] concentrates on hiding implementation differences and platform fragmentation in the mobile domain by providing proxy interfaces, called M-Proxies, that hide implementation differences between each platform. Their study concentrated on the Symbian S60 and Android platforms. The MobiVine system uses tooling support and code generation to convert MobiVine code into platform specific code at compile time. To use this middleware applications are coded using the common MobiVine interfaces, then at compile time the system looks up the mapping for real API provided by the manufactures and the MobiVine interface and generates the corresponding code. The system currently provides mappings for the native platform

APIs on Symbian S60 and Android for: location, SMS, call, and HTTP. This system is not designed specifically for collaboration but focuses on addressing the specific issue of fragmentation of platform APIs on mobile devices.

In summary, message-oriented middlewares [Springer et al., 2008, Guicking et al., 2008, Roth, 2002b, Zurita and Nussbaum, 2006, Gotthelf et al., 2007] and peer-to-peer based middlewares [Mascolo et al., 2001b, Gotthelf et al., 2007, Kortuem, 2002, Zurita and Nussbaum, 2006, Roth, 2002b, Guicking et al., 2008] are the most popular approaches to middleware design on mobile devices. Event-based systems [Kortuem, 2002, Springer et al., 2008] and tuple-space systems [Cinque et al., 2005, Cotroneo et al., 2007] are also used, while only one system can be classified as an information sharing middleware [Mascolo et al., 2001b]. The advantage offered by message-oriented, event-based, and tuple-space middlewares is that communication is asynchronous and connectionless, these communications paradigms are tolerant of disconnections so are a good solution when using unreliable networks.

Most of the middlewares discussed adopted a peer-to-peer distribution model, however this was because of a bias towards supporting MANETs in this domain. However MUSE involves a central server to simplify routing and service discovery, and as it is tailored for multimedia applications a server is required for dealing with computationally expensive processes for multimedia manipulation. Therefore, Pocket Dream Team [Roth, 2003] and ESPERANTO [Cotroneo et al., 2007, Cinque et al., 2005] are most relevant, in this respect, to the design of MUSE as it uses a SOA and needs to work on both fixed and mobile networks. The Pocket DreamTeam middleware is designed to extend an existing PC-based solution to the mobile environment and used a client–server approach to act as a bridge between the fixed and mobile parts of the system. While ESPERANTO used a client–server approach to communicate with reliable infrastructure providing a SOA.

All the middlewares described use a layered architectural model to middlewares yet deciding whether to provide functionality at a given layer is a difficult problem [Schmidt and Rodden, 1996]. Most of the middlewares described provide a limited functionality, predominantly restricted to the network and transport layers i.e. [Roth, 2002b, Cinque et al., 2005, Gotthelf et al., 2007, Springer et al., 2008, Zurita and Nussbaum, 2006], essentially the communication infrastructure and the sharing model respectively. As a

consequence of using asynchronous and connectionless communication infrastructure the most popular sharing model is message-passing approach, only [Cinque et al., 2005, Guicking et al., 2008, Roth, 2003, Mascolo et al., 2001b] provide support for any higher-level shared-data constructs e.g. shared objects, RPC or RMI.

Typically, concurrency and synchronisation must be programmed at the application layer. The middlewares that provide support for concurrency, synchronisation and data integration do so relatively simply and can still encounter conflicts. Agilo orders messages according to incoming message order at a given node but this still leads to inconsistencies as messages can be received in different orders at each node [Guicking et al., 2008]. A second approach adopted by Agilo is *simple transactions*, this involves encapsulating multiple messages in a single message for transmission, this guarantees that all actions in the transaction are not interleaved with messages received from other nodes, yet the problem remains the same, transactions can be executed in a different order at each node. XMIDDLE [Mascolo et al., 2001a,b] uses the most complex algorithm for concurrency control and synchronisation but requires input from the application layer to resolve some conflicts. Locking mechanisms are not used in any of the systems described as it is too complex to implement over unreliable networks, for example, Pocket DreamTeam which uses locking on the fixed network abandons it for an optimistic concurrency control approach when using mobile devices [Roth, 2003].

One aim of the MUSE middleware is to support groupware in mobile and fixed contexts and in particular to support collaborative multimedia applications. The middleware systems described here do not address multimedia and are predominantly text-oriented systems. Two systems did include services designed for groupware applications, these services include, user, group and session management services [Springer et al., 2008, Roth, 2003]. In the mobile context most of these systems concentrated on wireless computer systems, using WiFi with laptop computers rather than mobile phones. The lack of support for groupware and multimedia on smaller mobile devices and mobile phones is a problem that MUSE is designed to address.

To conduct evaluation of their middlewares three studies gathered data from performance metrics [Zurita and Nussbaum, 2006, Gotthelf et al., 2007, Cotroneo et al., 2007] and several of the studies also discussed validation of their middleware by developing applications to use the middleware e.g. [Roth, 2002b, Guicking and Grasse,

2006, Springer et al., 2008]. Similarly, the evaluation of MUSE uses a two stage process, first gathering performance data and then validating the middleware by evaluating applications designed to run on the MUSE middleware i.e. (SMART and DNT). Further details on and discussion relating to evaluation are covered in the next section.

3.3 Performance Evaluation

The related work in the area of middleware systems for mobile computing was described in the previous section in chronological order. Typically the first approach to the evaluation of the middlewares described involved creating applications to use the middleware. In this context the creation of a usable application was understood to have validated the functioning of the middleware. However this approach is limited, or the evaluation is lacking, as the applications themselves are never evaluated. So that it is not clear if the middleware provides adequate performance to support an actual usable application. The result is that this type of evaluation is generally restricted to proof-of-concept evaluation and does not provide any stress testing either explicitly in terms of performance or implicitly in terms of meeting the demands of a usable application. This approach was adopted for the evaluation of Agilo [Guicking et al., 2008], Mobilis [Springer et al., 2008], Network Kernal Framework [Roth, 2002b], Pocket DreamTeam [Roth, 2003], and XMIDDLE [Mascolo et al., 2001b]. Briefly, to provide context for the evaluation the MUSE platform this section outlines the evaluations described for these systems. This section is structured such that systems that followed this first approach are presented initially followed by the systems that did conduct some level of explicit performance evaluation.

The Agilo framework for synchronous groupware applications was evaluated by building two collaborative applications using the framework [Guicking et al., 2008]. The first tool was a collaborative application designed to support communication and coordination in emergency missions of public safety organisations. The second application was a meeting support system for facilitated meetings where the participants are co-located. The implementation of these applications was put forward as validation of the efficacy of the middleware. The meeting support system has been tested with up to 20 simultaneous users connected to the system. To date the meeting support

system has been used with up 200 participants to perform meetings. The authors state that there have been some load tests for the meeting support system, but no systemic quantitative evaluation of the framework has yet been carried out [Guicking et al., 2008]. In effect the system has been implicitly tested through the collaborative meeting support system sufficiently, but still requires a performance evaluation to determine its scalability and reliability and expected normal operating behaviour.

To evaluate the Mobilis middleware a prototype tourist application was built using the middleware [Springer et al., 2008]. The tourist application allows all members of a tourist group to be aware of the location and activities of the other users of the system as the map is updated. The middleware includes media sharing services so that they can exchange and share photos, videos and sounds of the trip. This application was the only evaluation as no performance data was recorded. This study does not report any further details on the evaluation using the tourist application.

The Network Kernal Framework was evaluated by developing two applications to use the system, DreamTeam and Quickstep [Roth, 2002b]. DreamTeam is a higher level middleware to support collaborative whiteboards and text editors built using the NKF. Quickstep is a mobile application to share application data, for example, appointments, contacts, addresses, and business cards. While both applications are described as providing verification that the NKF is useful for supporting collaborative applications, no further evaluation is presented.

Building on the NKF, Roth describes an extension to DreamTeam called Pocket DreamTeam to work in a mobile ad-hoc environment [Roth, 2003]. Pocket DreamTeam is a high level middleware to support mobile collaborative applications. To evaluate Pocket DreamTeam, four applications are described. The first application is called *online list* which is an application to display the list of currently active users on each device. The second application is called *session management tool* which is an application that lets the users create, announce and join shared sessions. The third application is called *diagram* which allows the users to create collaborative structured diagrams, e.g. class diagrams and flow charts. The final application described is called *free-hand drawing tool* which is a shared drawing tool that presents the users with a shared whiteboard interface to created shared drawings. The evaluation of these applications is given in terms of code usage statistics to gauge the implementation cost

	Stationary (lines)	Proxy (lines)	Re-used %	Mobile (lines)
Core Platform	125000	110000	98%	9500
Online List	4400	4000	95%	930
Session Management	7100	6900	93%	2100
Diagram	6900	5200	92%	600
Draw	930	520	90%	410

Table 3.3: Pocket DreamTeam Implementation Costs [Roth, 2003]

of creating the applications using Pocket DreamTeam. The results from this study are reproduced in Table 3.3 on page 49. *Stationary code* is code that is implemented on a server or fixed computing resource. *Proxy code* is code on the mobile device and server to encapsulate computationally expensive operations that can be updated even when the mobile devices are offline. The *Re-used code* is the amount of the code of the proxy application which was reused from the stationary application. *Mobile code* is code specific to the mobile device, which generally represents user interface code. Roth recognises that these figures only represent a trend and are not an absolute measure. Again as for the NKF above these applications were used to verify the concept but were not themselves evaluated at all.

The XMIDDLE middleware also adopted this approach to evaluation [Mascolo et al., 2001b]. In this research a case study of building an application to provide a shared shopping list using the XMIDDLE middleware was described. This was put forward as verification that the XMIDDLE middleware can support mobile applications. However no further evaluation was presented, either of the middleware or the application.

In summary, up to this point, most of the studies described validated their middleware systems by creating applications to implicitly verify the usefulness of their middleware. No further evaluation criteria were presented, with only the collaborative meeting support application built using Agilo reporting the total number of participants and the number of simultaneous clients supported. However, three of the studies did record some simple metrics including: start-up time (XMIDDLE); memory footprint (XMIDDLE and Agilo); and the rough cost of implementation in number of lines or code or classes needed (Agilo and Pocket DreamTeam). This suggests that the starting point for the evaluation of the MUSE platform should start proving that applications can be created using the MUSE platform, however this thesis argues that this alone does not

validate the usefulness of the middleware and so both performance testing and evaluation of the utility of the applications developed is needed. From a technical perspective three further related studies did record more detailed metrics and performance data to evaluate their middleware: ESPERANTO [Cotroneo et al., 2007, Cinque et al., 2005]; GMAC [Gotthelf et al., 2007] and MCSCL-CS [Zurita and Nussbaum, 2006]. Nevertheless there is a lacuna in these studies as any applications created to use the middlewares were not evaluated.

The ESPERANTO middleware was tested using a simple application to communicate through a *request-response* mechanism [Cotroneo et al., 2007]. This application has been instrumented with probe points to get timestamps during various stages of the process. The hardware required to set up the testbed consists of two mobile devices, two permanent hosts, and two access points. Four aspects of the system are measured: Invocation latency; throughput; invocation latency during hand-offs; and hand-off latency.

Invocation latency is measured on the client-side as the round-trip time of a method invocation. *Throughput* is measured on the mediator-side as the number of requests per second processed by the server. *Invocation latency during hand-offs* is measured as the average latency of consecutive remote invocations when one or more hand-offs does occur. The *hand-off's latency* is measured as the time taken for the Mobility Manager to reveal the need for a hand-off and to reconnect to the mediator. The last two metrics are specific to the ESPERANTO middleware, however the first two are more generic and are applicable to the performance evaluation of the MUSE platform. They test their system using payload sizes ranging from 1 byte to 10 KB. They show that invocation latency is a function of payload size ranging from 192.797 ms for 1 KB of data to 588.826 ms for 10 KB of data (these payload sizes are comparable to the message sizes used by MUSE for most communication).

In common with the evaluations described earlier GMAC is evaluated by running two applications using the GMAC middleware [Gotthelf et al., 2007]. The first application is Chatero, a synchronous collaborative application that allows people to join in a chat group, vote for actions and draw over fixed background images. The second application is Science-Peer a groupware application for asynchronous cooperation. Science-Peer includes a document sharing capability and is intended to find researchers with similar

interests. No experimental results relating to these applications were reported.

However, the GMAC middleware was tested with a performance evaluation [Gotthelf et al., 2007]. There were five main metrics presented to evaluate the performance including: end-to-end group propagation delay, throughput, protocol overhead, group latency, resource latency, and failure recovery. As the GMAC middleware is a peer-to-peer architecture, the first performance metric measured is *end-to-end group propagation delay*, which they define as the time from a random node to send 100KB to all the group. *Throughput* was the next metric described and this study reports that the throughput achieved by GMAC is equal to half the overall bandwidth. The reduced throughput is the result of protocol overhead caused by the retransmissions needed to create and maintain the overlay network used by the middleware. This study defined *protocol overhead* as network traffic that does not represent useful application data, in this case traffic to maintain the overlay network of this peer-to-peer system. *Group Latency* is defined in this study as the longest end-to-end delay between any pair of nodes. *Resource latency* as a percentage of MST latency. *Failure recovery* is expressed as the time required to recover from simultaneous failing hosts on a group with 1000 nodes.

The results for the GMAC middleware are from a software simulation and reported for a large number of simultaneous nodes (up to 1000 nodes) so are hard to compare directly with MUSE which is designed to support small groups of between four to ten simultaneous users. The two figures most directly relevant are end-to-end group propagation, and group latency. End-to-end group propagation delay for 50 simultaneous nodes is 125 seconds and maximum throughput is 8 KB/sec. Average group latency between any two nodes is four seconds when 50 nodes are connected to the system. As this is a peer-to-peer system these figures are hard to compare with the centralised architecture used with MUSE because the central server acts as a bottleneck with an upper bound on the number of clients that it can support based on the network connectivity of the server itself.

The ad-hoc wireless network architecture for face-to-face collaborative applications presented by Zurita and Nussbaum (MCSCL-CS) is evaluated by measuring the interconnection delay before group formation of an application built using this architecture [Zurita and Nussbaum, 2006]. The implementation of this architecture was built using

the eMbedded Visual Basic (eVB) run-time for the windows mobile-based pocket PC 2002 platform. The system is executed over a wireless peer-to-peer Wi-Fi network using TCP/IP on Compaq IPAQ handhelds. Microsoft SQL server is used for persistence. The evaluation of MCSCL-CS was performed with 1, 2, 9, 12 and 15 groups, where each group contained three handhelds, thereby supporting up to 45 students. They tested the system by measuring the delivery time for three message sizes (128 bytes, 256 bytes, and 512 bytes) for each of the five different group sizes. They find that group formation is proportional to the number of groups in the system, with a time of 5 seconds for 15 groups. The second aspect they measure is the time taken for sending packages to different number of groups, again using the same message sizes as above. They find that as their architecture replicates messages at each sub-network, delivery time increases as the number of groups increases until it overloads the wireless network. The transmission time ranges from less than 1 second for 1 group at 128 bytes, to 9.102 seconds for 15 groups at 512 bytes.

The second research question asked in this thesis is: do the functional requirements for a MCSCL platform captured in MUSE, SMART and the DNT support constructionist, contextualised, and collaborative activities for learning? The first part to answering this question is to compare MUSE and each of these systems with the functional requirements listed. This comparison is covered in the next chapter, 4.3.12. The second part to answering this question is to answer several sub-questions including: from a technical perspective can we design and build an MCSCL middleware to work in a setting containing heterogeneous devices and networks? What is the performance of the middleware on mobile devices and PCs? How long is the message round-trip time between devices and the server? How many concurrent users can the system support? and what are the usability and learning support provided by the applications developed?

Selecting or comparing metrics and results between such diverse systems is difficult. The approach adopted in this thesis is to chose a set of metrics that shed light on the technical challenges relevant to the implementation of a middleware for MCSCL and digital narrative application. The metrics used in this thesis are: round-trip time, group latency, protocol overhead, throughput, and implementation cost. Further details on these metrics are given in Section 6.1 which lists the technical issues and any performance metrics that stress test the MUSE middleware's approaches to overcoming

them. These metrics can then be used to answer answer the sub-research questions described.

3.4 Conclusion

This chapter described the lack of software support for developing mobile learning or MCSCL applications e.g. Springer et al. [2008]. The result is that mobile learning applications are currently implemented on a one off basis and tied to particular platforms. Rochelle notes that there is no high level shared data exchange mechanism appropriate for the pedagogical applications Roschelle [2003]. Wireless and mobile technologies in education will succeed when there is more commonalities in communication infrastructures, protocols, messaging standards, and processing capabilities across distributed devices Roschelle [2003].

However, recently there has been some efforts to provide stable communication platforms and middlewares on mobile devices that would be some help in this regard. In this context an overview of related work developing middlewares to support mobile collaboration was given. These middlewares were compared according to their middleware type and their level of support for each of the variation points usual for mobile groupware applications: distribution model; communications infrastructure; sharing model; concurrency model; and synchronisation model. MUSE aims to support flexibility at each of the variation points and provide support for each of the above requirements, the next chapter describes in detail the MUSE platform and how it supports this flexibility.

Furthermore, this chapter included a review of the evaluation practices used by these system in order to provide context and inform the design of the evaluation for MUSE used in these thesis. This chapter found that the predominant evaluation approach was to create applications using the middleware and use this as implicit validation of the usefulness of the system. The second approach was to conduct performance testing using several performance metrics.

What are the functional requirements for a platform to support MCSCL applications on both mobile and PCs? Do the functional requirements for a MCSCL platform captured in MUSE, SMART and the DNT support constructionist, contextualised, and

collaborative activities for learning? The next step to answering these research questions is to examine the level of support for the functional requirements identified in the previous chapter, including: support for the four dimensions of the space-time classification of groupware system; the nine dimensions of groupware technology from Mittleman et al. [2008]; constructionist, contextualised and collaborative activities; data collection and location awareness. Therefore, the next chapter describes MUSE in detail and provides a comparison of MUSE to the systems described in this chapter in terms of their level of support for these requirements.

Chapter 4

MUSE: A Platform for Mobile Computer Supported Collaborative Learning

To provide context this chapter includes a brief overview which expands on the technical challenges of developing software in the mobile computing domain introduced in the previous chapter. The conceptual architecture of the MUSE platform is described, with particular focus on the elements of the conceptual architecture that are intended to handle the heterogeneous nature of the mobile computing environment. Aspects of this conceptual approach include: the insistence on open XML messaging formats; a design-by-contract approach to middleware design; the use of a distributed model-view-controller (MVC) interaction paradigm; and using interfaces to hide hardware or OS dependant implementation details from the application layer.

Moving on from the conceptual design, this chapter then describes the implementation of MUSE in detail. The MUSE platform provides a reliable system for creating constructionist, contextualised and collaborative learning applications. The MUSE platform contains three core components: the MUSE application server, a collection of infrastructure services and the MUSE middleware. The MUSE middleware is composed of two core layers, the MUSE service-layer and the MUSE networking-layer. The MUSE service-layer is used to support the interactions between components and applications via an event-based and service orientated architecture. The MUSE networking-layer provides for transparent access for communication over different networks, thereby

abstracting from the client applications the differences between TCP/IP, HTTP, and MMS, etc. Messaging between components of the MUSE platform is using open XML formats.

The previous chapter outlined the elements of CSCW relevant to building MCSCL applications and described a space-time classification of groupware systems and a set of functions that groupware applications should support to various degrees, namely: core functionality, content, relationships, session persistence, supported actions, action parameters, access control, alert mechanisms, and awareness indicators [Mittleman et al., 2008]. This chapter will address how these aspects of groupware systems are handled with MUSE, later in Chapter 5 these aspects are revisited with regard to the MCSCL applications and services developed during the course of this thesis. Furthermore particular features of MUSE middleware are highlighted according to the framework of variations points [Gotthelf et al., 2007] in groupware systems. The next chapter will provide a review of the literature relevant to digital video production, for now it is enough to note that the MUSE middleware is designed to support MCSCL applications in general and collaborative digital multimedia production applications in particular. Therefore some aspects of the MUSE middleware relevant to digital multimedia production are outlined. Finally, MUSE is compared with the systems described in 3.2 in terms of their level of support for the theoretical foundations presented in Chapter 2.

4.1 General Overview Of Mobile Computing Technical Considerations

The various challenges encountered in the mobile computing domain were summarised in Table 3.1 on page 34. This section briefly outlines these technical challenges involved in developing software in a mobile computing environment, it will also describe some generic solutions and considerations relating to hardware, software and networking.

This section starts by examining the hardware limitations on mobile devices. There are two broad categories of hardware limitations on mobile devices, the limitations of the mobile computing platform and I/O limitations. The limitations of the mobile computing platform encompass low processor performance, small memory size and battery life. The second are I/O limitations including, reduced GUI capabilities, small

screen sizes with low resolutions, and rudimentary keyboards. However mobile devices do support a wide variety of alternative I/O options including cameras, sound recorders, and locations sensors for example.

For mobile devices to remain portable they need to be small and lightweight, this impacts the size of the batteries, which in turn sets a limit on the amount of power available for the devices. The implication of these physical constraints is that the amount of processing power and memory available for mobile computing is limited in order to limit power usage, so mobile devices for the foreseeable future will always be less powerful computationally than PCs and other fixed computing sources. This being the case, more careful use of the available computational resources (collectively memory and processing power) is a necessity.

There are two solutions available to deal with limited computational resources. The first solution is to create software which is frugal with computational resources. There are two principal drawbacks of this approach, first that it is difficult and demanding on the application developer to write software in this way, and second device specific characteristics must be exploited to get optimal performance from the application. These problems limit the portability of the software and it is difficult to exploit this software development paradigm consistently in practice.

The alternative solution examines the overall computing environment (including communications technology) to distribute software components over several devices. This approach involves either a symmetric or asymmetric distribution of the computational load. In the symmetric approach all the nodes in the system are equivalent, this is typically the approach used with peer-to-peer systems. This simplifies software development as all nodes are the same. However, this approach involves extra overhead to implement the peer-to-peer overlay network, (routing, service discovery, detecting node failure etc.) that must be factored in to the overall efficacy of this approach.

In the asymmetric approach the computation nodes in the system are assumed to have different characteristics so that computationally expensive operations are executed on devices better able to handle them. This solution essentially bypasses the computational shortcomings of mobile devices but at the expense of relying on the communications facilities of the devices. However, not all computation can be removed from the mobile devices and this approach has implementation overheads resulting

from distributing the software components.

The second major set of hardware challenges to developing on mobile devices are the I/O limitations including: reduced GUI capabilities; small screen sizes with low resolution; and rudimentary keyboards. These limitations are harder to avoid, and can only be ameliorated by sensible user interface design. For instance applications should require as little text input from the user if possible, and should not display too much information on the screen at any one time. The user interfaces should also scale based on the display sizes and where possible they should make use of native UI elements of the devices themselves.

While there are challenges using the I/O options appropriately there are also opportunities because mobile devices support a variety of alternative I/O options including cameras, audio recorders, and location sensors. Difficulties arise however as methods of accessing this hardware can vary between devices, as a result the software should provide some degree of transparent access to this hardware to hide precise implementation details from application developers. Taking I/O options in total, it is true that they can vary significantly between each device, therefore software designed for mobile devices should be careful to separate the presentation layer from the application logic. In this manner a large portion of application logic can be reused between devices so that only code that depends on specific I/O characteristics needs to be altered.

What is true for hardware I/O is true for software, as there is large variety of available software on mobile devices. This includes a range of heterogeneous operating systems, operating system services, and third party applications. Writing software that is able to function on multiple heterogeneous operating systems is a difficult problem to solve. Typically the only solution is to re-implement the software for each operating system, although this can be an onerous task. Another option is to design to minimise the amount of software code exposed directly to the operating system of the device, for example, by running code on a virtual machine that is designed to execute cross-platform software. Java would be a the canonical example of this approach. Yet this leaves the problem of heterogeneous OS services and third-party applications on mobile devices, for example some devices provide a local file system implementation while this is not available on other devices. Solutions to this problem range from providing an implementation of the missing services, to selectively disabling features

of the application which rely on the missing services. Again good practice is to isolate and encapsulate access to heterogeneous OS services behind an interface, so that the implementation can be changed without requiring significant modification of the user applications.

The final challenges to creating software on mobile devices are the diverse networking facilities and network characteristics available on mobile devices. There are several different networking technologies for mobile technology and each device can potentially support a different subset of the available communications technologies. The problem with this is that applications developed for mobile devices do not know what communications facilities will be available on a particular devices. Applications are often designed to use just one or two particular networking technologies and then ignore devices that do not support them. As this is a similar problem to dealing with heterogeneous OS services the alternative approach is to isolate and encapsulate access to the networking facilities of the device.

A secondary problem with networking on mobile devices is whichever communications technology is used it will suffer from lower bandwidth, poorer reliability, and greater latency than comparable fixed networks. Therefore applications developed in this space should not make over use of the network where possible and should be able to tolerate some degree of disconnection from other devices in the system.

A final problem with mobile devices is related to the very mobility of the devices themselves. For example they will typically change location regularly so routing information to the devices can be problematic.

To provide some context this section outlined some of the difficulties developing software in the mobile domain and listed some general solutions available. The next section describes the MUSE platform which is designed to support development of MCSCL applications and so must consider and address the above challenges in order to provide a reliable platform on which to develop such MCSCL software.

4.2 MUSE Conceptual Architecture

This thesis describes the design and development of MUSE *platform* to support the development of MCSCL software for multimedia applications for mobile devices and PCs.

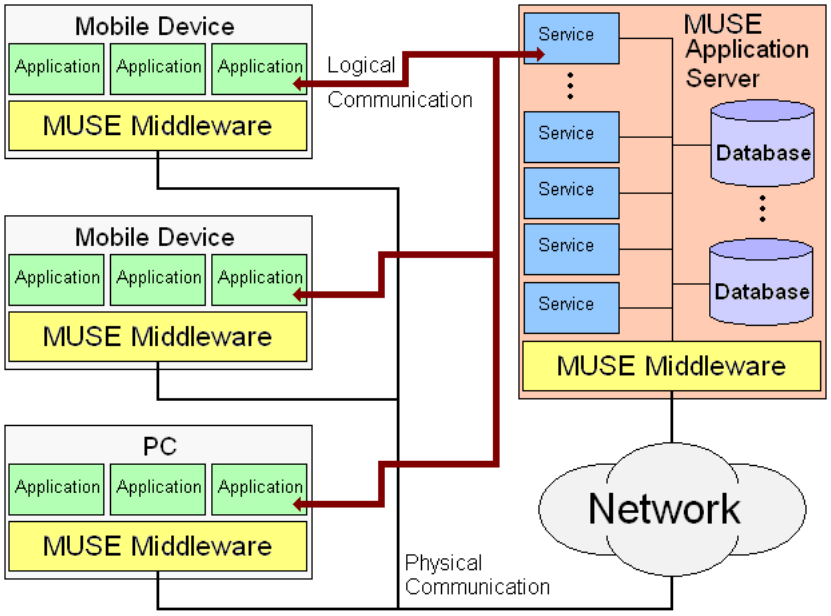

Figure 4.1: Overall Architecture

It addresses the technical and pedagogical considerations from Chapter 2. The MUSE platform has three major components, the MUSE middleware, the MUSE application server, and a set of infrastructure services used to implement core functionality of the MUSE platform. Such a generic system has the potential to overcome the limited reuse potential of specific tailored systems and the general trend in computer supported collaborative systems (CSCL) has been towards generic systems [Lonchamp, 2006]. Dimitracopoulou, while describing current trends for the design of collaborative learning systems, mentions that it is "important to provide flexible architectures and customisable tools" [Dimitracopoulou, 2005]. In addition, current trends challenge software developers "to provide a uniform and integrated user experience across the desktop, web, and mobile platforms" [Bosch et al., 2007]. The architectural goal is to minimise the platform-specific parts of the software while maximising the commonalities across the platforms, with both parts separated by simple interfaces. [Bosch et al., 2007] recommend a service-oriented architecture (SOA) as a technical solution to achieving this architectural goal. A SOA consists of loosely coupled reusable components and provides a flexible, reconfigurable platform on which to develop applications. Service-oriented architectures provide a logical way to design software systems that provide services to end-user applications or services distributed over a network [Papazoglou et al., 2007]. The overall high-level architecture is shown in Figure 4.1 on page 60.

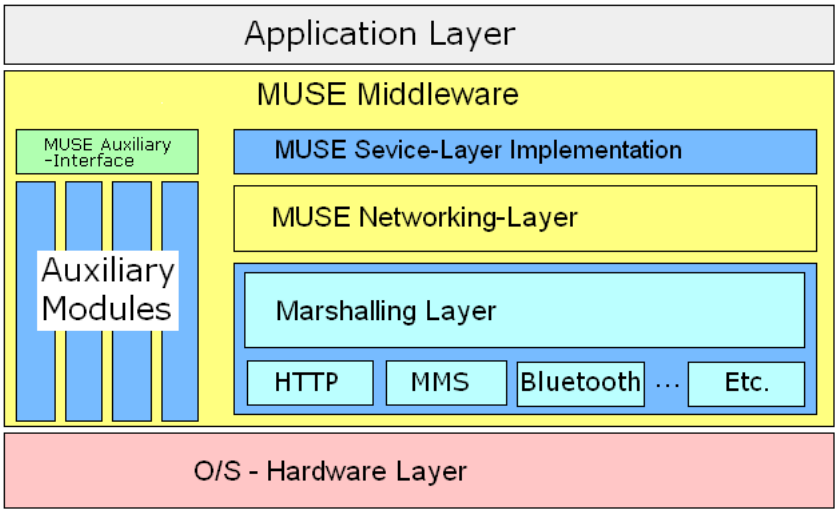

Figure 4.2: MUSE Middleware Conceptual Model

The MUSE *middleware* uses a layered architecture which includes two core tiers, the MUSE *networking-layer* and the MUSE *service-layer*. The networking-layer is the lower layer and provides a transparent networking interface to layers above. This layer acts as a reliable, connectionless, and asynchronous communications mechanism between client and the server. Networking support adapts as users move from reliable LANs to less reliable networking technologies for mobile devices. In this way, MUSE provides a transparent network layer to create a reliable communication link over varying networking technologies including: HTTP, TCP/IP, and MMS.

The top layer is the service-layer which provides an API to interact with services through the MUSE platform. This layer interacts with an event based system using a publish–subscribe paradigm. This is the upper layer and presents a simple API for use by client applications at the application layer. In addition the MUSE middleware provides a set of auxiliary modules for accessing multimedia facilities, security, advanced I/O APIs, user management and group management. The combined API from these auxiliary modules is labelled as the MUSE *auxiliary-interface*. Taken together the service-layer and auxiliary-interface comprise the encapsulated API presented to client applications. In this way lower level implementations can be changed while maintaining support for application software.

The MUSE *middleware* was described above, however the overall MUSE platform also contains the MUSE *application server* and a set of infrastructure services. The

platform uses a client-server architecture to communicate between the mobile devices or PCs and the application server. The application server hosts services, controls access to shared data, and manages connections to external resources or databases. The platform provides a distributed Model-View-Controller (MVC) paradigm for interacting with shared data on the application server. MUSE uses a service orientated architectural model that logically connects the user applications, on the PCs or mobile devices, to services on the application server. A service oriented architecture facilitates the reuse of loosely-coupled components, thereby enabling the development of flexible and reconfigurable applications suitable to this heterogeneous environment. Furthermore, this service orientated architecture enables the platform to conform to the trend towards generic open systems in CSCL [Lonchamp, 2006].

The application server executes two types of services, user services and infrastructure services. The *user services* essentially provide an implementation of the *model* and *controller* in the MVC architecture for the applications to use. The client applications should provide an implementation of the *view* aspect of the MVC architecture. The client application then uses the MUSE middleware to interact with the model and controller on the application server. In this way applications on either the PC or mobile device can provide alternative views to the same model and controller, thereby enabling software reuse and allowing for heterogeneous user interfaces on the various devices. The *infrastructure services* provide the core functionality needed by the MUSE platform. The three infrastructure services required by the MUSE platform are the registry service, the log-in service and the responder service. These services are required in order to implement, the publish–subscribe function, the user grouping function, and asynchronous communication mechanism, respectively, which are required by all implementations of the MUSE middleware.

Thus far this section has described the general architecture of the MUSE platform. This section will now focus on the specific aspects of the conceptual design of the MUSE platform that deal with the significant heterogeneity among hardware, software and networking technologies in the mobile computing domain that were outlined in the previous section. While building software that masks all aspects of heterogeneity is almost impossible this thesis is informed by related middleware research and employs established distributed system practices and design patterns, in order to develop a

stable platform to meet the specific challenge of creating groupware software across both the mobile and PC domains to support learning activities.

The MUSE platform is a conceptual architectural model separate from any given implementation, with clear interfaces between layer and system boundaries. There are four elements to the conceptual design to increase the flexibility of the software in the heterogeneous mobile environments. Firstly communications is via specified XML formats. Secondly the service-layer and networking-layer are designed according to a design-by-contract approach, which means implementations of these layers must adhere to a contract of behaviour to the other layers. Thirdly the MUSE platform uses a distributed MVC architecture, backed by service oriented architecture to support loosely coupled components. Finally the auxiliary-interface specifies a set of functions that an application can rely on regardless of OS supplied support. This approach to designing and specifying the MUSE platform focuses attention on the interface between components in the system in order to maximise software reuse and minimise development effort if re-implementation is necessary.

4.2.1 XML Message Formats

For interaction across system boundaries, usually over the network, the MUSE platform specifies the message exchange format to be used between the nodes in the system in a platform independent XML format (see Figure 4.3 on page 64). In this way any implementation that can read or write messages that conform to these XML formats can interact with the MUSE platform. Furthermore, the conceptual model specifies the expected behaviour between the layers and nodes so that alternative implementations of the MUSE platform must be extrinsically identical while they can be intrinsically different. In short this enables third parties or alternative middleware implementations to interact with MUSE in a language and platform independent manner.

4.2.2 Design-By-Contract

The conceptual design of the MUSE platform uses a design-by-contract methodology. For example, the service-layer manages the routing of the internal messages between components on the device and the network. However to do this effectively the service-

```
<internal>
<header>
<source location = "String: service address" />
<destination location = "String: service address" />
<originator clientid = "int: humanid" />
<timestamp

    timestamp = "long: milliseconds"
    vectorclock = "int: vector clock timestamp" />

<payloadreader reader = "String: Fully Quali-
fied Class Name" />
<header>
<payload>

    String: xml data

</payload>
<attachment name = "String: filename">

    Base 64 Encoded data

</attachment>
</internal>
```

Figure 4.3: Internal Message Format

layer must adhere to a contract or a set of expected behaviours that the layers above and below can interact with consistently. The service-layer has the responsibility to send and receive the internal messages logically between the client and the server. Upon receiving a message the service layer must parse the internal message and included payload message and deliver the payload message to the client components at the application layer that registered an interest in the service. This layer should also ignore any duplicate messages based on timestamp and vector-clock information. Client applications sending messages to services on the application server should send messages to the service-layer which in turn passes them on to the networking-layer for transmission to the application server.

In addition to the above role as post-master in the system the service-layer should provide an implementation of the publish–subscribe paradigm. Therefore the service-layer must manage a local service registry that records which application components have subscribed to particular services. This registry can be queried to find the set of available services on the application server, and to subscribe or unsubscribe to

```
<service servicename = "String: Address of the Service"
action = "String: REGISTER|UNREGISTER|LIST|NONE"
humanid = "int: human id" />
```

Figure 4.4: Service Message Format

messages from a service. In order to achieve this the service-layer should interpret service messages from the application server. The service-layer consults this registry to find the components interested in the message.

The networking-layer contract expects to receive XML messages using the internal message format (see Figure 4.3 on page 64). The networking-layer is responsible for transmitting messages to the application server over whichever networking protocol is available. The networking-layer must transform (*marshall*) this XML data into the appropriate format for the protocol selected. When the networking-layer receives a message from the application server it should transform (*demarshall*) it back to the XML internal message format. It must then pass this message to the service-layer which will handle further routing and processing of the message. In addition, the clients can register as connected synchronously or asynchronously. Synchronous connections imply that the server will push event notifications to the clients directly as they occur; clients registered as connected asynchronously use a polling mechanism to check for new events. However the networking contract to the service-layer operates as if all communications are asynchronous and event based. The synchronous communication is really a convenience to reduce messages generated by a polling approach.

So in summary the MUSE middleware adopts a design-by-contract approach to separate functionality between the layers of the middleware. The service-layer should interpret service messages to update the client registry or send service messages to the networking layer to update the registry on the server. The service-layer should interpret internal messages from the server and pass them on to the interested client components or should pass internal messages from client components on to the networking-layer to forward them to the application server. The networking-layer should transmit XML internal messages between the clients and server over whichever protocol is available. The networking-layer should handle marshalling and demarshalling the messages into the appropriate formats for transmission over the chosen networking protocol.

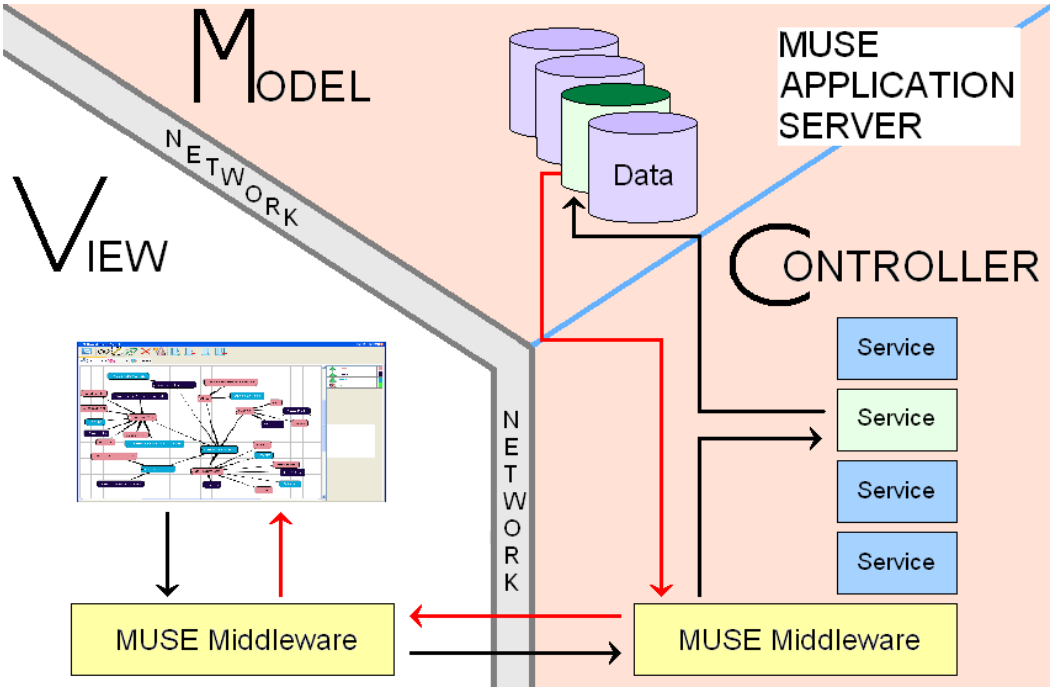

Figure 4.5: Model View Controller

4.2.3 Distributed Model-View-Controller

The third aspect of the MUSE platform that maximises software reuse is that MUSE recommends a distributed Model-View-Controller MVC paradigm (see Figure 4.5 on page 66) for application software development, with only the view residing on the mobile devices. This means in practice that the application data (model), and the application logic (controller) are located on a reliable server, while only the user-interface (view) is located on the mobile devices. Using this approach only the view needs to be adapted to take account of heterogeneity of hardware and software on mobile devices. In addition, this design approach supports developing applications using a thin client approach at the application layer. The components of the distributed MVC architecture communicate using the service-oriented architecture available in the MUSE platform, which facilitates decoupling between components using the publish–subscribe paradigm.

The typical interaction with the distributed MVC starts when the user makes a change to the shared data through the view on their device, the update is first sent to the MUSE middleware which sends the change on to the application server. The MUSE middleware on the application server, forwards this update to the appropriate services.

When the service receives the update it can either commit the change or roll it back. The service then updates the data accordingly. The application server then sends the result back to all the clients that were interested in updates to this shared data. Finally the client devices receive the updates and then update their user-interfaces to reflect the update.

4.2.4 MUSE Auxiliary-Interface

The fourth aspect that is included in the MUSE platform is a set of auxiliary modules to provide a simple interfaces to common functionality that client application need. Currently there are six types of modules: encoding; security; grouping; sensors; file-system; and additional programming language functions. The implementations of each module vary from device to device if necessary. The purpose of these APIs is to provide a more complete platform on the mobile device, as many of these features are typically not available on mobile devices, or if they are present then in a very heterogeneous fashion.

The *encoding* module provides a simple API to encode multimedia data for transmission as character data. The default implementation of this interface provides support for Base-64 encoding. Alternative implementation can be supplied at run-time using dependency injection techniques. The *security* module provides a simple API to encrypt and decrypt character data. The default implementation of this interface provides support for a simple mono-alphabetic substitution cipher (Caesar cipher). Again, alternative implementations can be supplied at run-time using dependency injection techniques. This module supports public key cryptographic key exchange. The default implementation uses the Diffie-Hellman key exchange algorithm [Diffie and Hellman, 1976]. The *language* module includes interfaces to access implementations of generic programming functionality that was lacking on some devices. There are interfaces for the *queue* data structure, a *sorting* algorithm and a *logging* system. In addition this module includes support for a few utility functions; a function to wrap text based on the screen dimensions, a function to convert bytes to integers and vice-versa, and a function to represent colours using hexadecimal digits.

The *sensor* modules provide an interface to access sensor hardware on mobile devices. This module includes interfaces to access the camera, audio recorder, and loca-

tion sensors transparently. The *file-system* module provides facilities to save and load media files. It also includes a loading tracker, that monitors the loading of files so that the application can wait for a file to load fully before continuing, if required. The *grouping* module provides an interface to interact with the user management system, this interface supports users logging in and out of the system, listing the status of the other users in the system, and searching for users details by and id number.

Full details of the MUSE auxiliary-interface APIs are listed in Appendix B using the Java programming language. This section described a general overview of the conceptual architecture of the MUSE platform, the next section will describe the reference implementation of MUSE platform using the Java programming language. It is this version that is used for implementing the MCSCL applications described in 5.

4.3 MUSE Implementation

The MUSE platform implementation contains three principal components: the MUSE middleware, the MUSE application server, and several infrastructure services. The MUSE middleware provides an API so client applications can send messages to the server, look-up services and register an interest in events from a service. The MUSE platform uses the SOA paradigm to decouple the components of the MCSCL applications and an open XML message format for all communication to allow other applications to use the services hosted on the application server. The MUSE middleware provides a transparent network layer to provide reliable network transport over several networking protocols. The application server provides a central application server to simplify the consistency maintenance algorithms and to reduce the number of messages needed.

This section outlines how client applications can connect to the MUSE platform and then describes the implementation of the MUSE service-layer and the MUSE networking-layer. The MUSE auxiliary-interface was covered in the last section, however further details of this interface are available in Appendix B. This section will then outline the operation of the application server and associated infrastructure services. This platform was developed using the Java programming language with versions of the middleware built using both J2ME and J2SE. This section describes the specific aspects of the MUSE platform as they related to the variation points framework from [Gotthelf

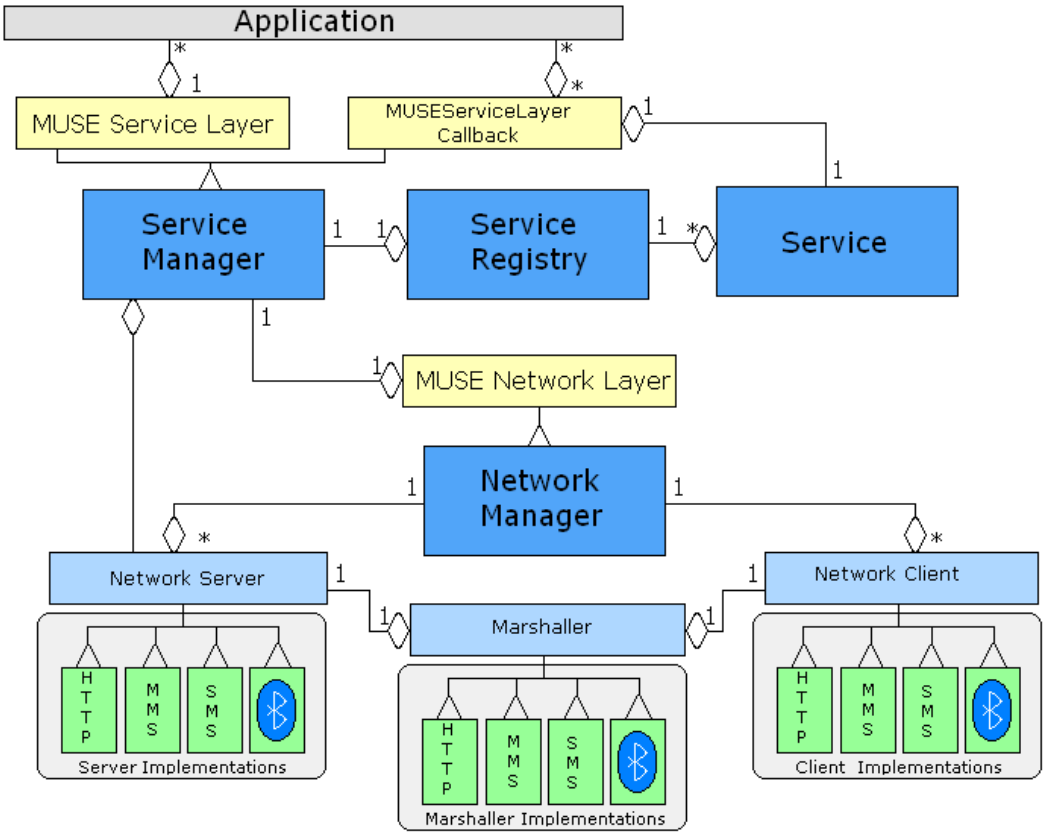

Figure 4.6: MUSE Middleware Details

et al., 2007]. This section concludes by describing the groupware aspects of the MUSE platform in terms of the space-time [Ellis et al., 1991] classification of participants and Mittleman et al.'s taxonomy of groupware systems [Mittleman et al., 2008].

4.3.1 Application Layer

When an application wishes to use the MUSE middleware it should get a reference to the service-layer, and use this to log-in to the application server. The application must follow the following steps, first it should send a message to register interest in receiving service messages and user log-in messages. It should then get a user name from the user, and use this information to log-in to the application server. However, for convenience an *abstract* application is provided in J2ME and J2SE for application developers to use, which manages this initial orchestration needed for the application to use the MUSE platform correctly. Extending this abstract application will register the application for the standard infrastructure services and provide a blocking log-in interface to the user. This information is used to automatically complete the log-in

process on the application server. The application developer can then register any Java objects that implement the MUSE*ServiceLayerCallback* interface with the MUSE middleware to receive messages from a given service. From this point on messages generated by this service are forwarded to the interested object.

4.3.2 MUSE Service-Layer

The MUSE Service-Layer is represented in Java using the `MUSEServiceLayer` interface, which is used by client applications to interact with the MUSE platform. It provides functions to interact with the service-orientated architecture. The `requestServices` method sends a message using the `ServiceMessage` format to the server to request an up-to-date list of available services. The `getAvailableServices` method queries the middleware and returns the local (cached) list of available services. These two functions are related but are decoupled to accommodate the asynchronous nature of the communication between the client and server using the MUSE middleware. If there are no services listed it will also send a message to the server to request an up-to-date list of available services.

The `register` method takes a reference to a `Service` object as a parameter and sends a message using the `ServiceMessage` format to register that a component of the application has an interest in receiving messages from the named service. The `unregister` method takes a reference to a `Service` object as a parameter and then sends a message using the `ServiceMessage` format to inform the application server that a component object of the application no longer wishes to receive messages from the named service. The `sendMessage` method takes an `InternalMessage` as a parameter, this method then passes this message onto the MUSE networking layer, which will handle its further communication to the application server. An `InternalMessage` can be addressed to any service or component depending on the direction of the communication i.e. client to server or server to client.

In addition the MUSE service-layer contains one other interface, namely the `MUSEServiceLayerCallback` interface. The `MUSEServiceLayerCallback` interface includes only one method which provides a callback point where Java objects which have registered interest in a service can receive the resulting messages. It is then up to the Java objects how to handle this information. This mechanism is key to the ability of

the MUSE platform to support alternative *view* implementations as part of the MVC architecture.

Three Java objects are used to implement the MUSE service-layer functionality. The first object is the `ServiceManager`, which is responsible for implementing all the methods of the `MUSEServiceLayer` interface and routing incoming messages to the correct Java object by passing messages to their `MUSEServiceLayerCallback` interface handler. The second object is the `ServiceRegistry`, which is a local database that records which Java objects are interested in messages from particular services. The `ServiceManager` uses the `ServiceRegistry` to route messages correctly. Both the `ServiceManager` and `ServiceRegistry` are singletons. The third object is the `Service` object, which is a Java object to represent a service on the MUSE platform. It stores a component-to-service name binding and a reference to the Java objects callback method for use by the `ServiceManager`. It is also used to represent the XML service messages used by the MUSE platform.

4.3.3 MUSE Networking-Layer

The MUSE Networking-Layer is represented in Java using the `MUSENetworkLayer` interface, which is used by the service-layer to send messages between the client and the application server. This is a simple interface with only one method, namely `sendMessage`, which method takes an `InternalMessage` as a parameter, it is up to the implementation to handle the transmission of the message. The `NetworkManager` is the implementation of the `MUSENetworkLayer` and has the responsibility of delivering messages to the application server. The `NetworkManager` contains references to several networking clients and servers which implement different networking technologies using the `NetworkClient` and `NetworkServer` interfaces. The `NetworkManager` tries to send the message with each network client until failure. The application developer can configure the failure criteria in a properties file by varying the number of retries and the timeout between retries that the `NetworkManager` attempts. Finally the order in which each network client is used can be altered based on which protocols are available on the device or manually using a policy.

The `NetworkClient` abstract interface is used to send messages to the MUSE application server. This interface is implemented by the `SocketClient, HTTPClient`,

and `MMSClient`, and these implementations make the actual connection to the MUSE application server.

The `NetworkServer` abstract interface which represents the components of the networking system that are listening for incoming connections, or polling for new messages from the server. This interface provides methods for starting and stopping the listening servers. This interface is implemented by the `SocketServer`, `HTTPServer`, and `MMSServer`, and these implementations pass incoming messages they receive on to the `ServiceManager` for further routing.

Finally the `Marshaller` abstract interface provides methods to marshall messages to and from the `InternalMessage` format to the correct format for transmission over the network. The system contains implementations of the `SocketMarshaller`, `HTTPMarshaller`, and `MMSMarshaller` for marshalling and demarshalling messages for transmission over the network. Both the network server and network client implementations should contain references to the appropriate marshaller implementation. For example the `HTTPClient` and `HTTPServer` hold references to the `HTTPMarshaller` which is used at run-time to marshall and demarshall the messages used with the HTTP network protocol.

4.3.4 MUSE Application Server

The MUSE application server contains the services used by MCSCL client applications, and references to resources required by the services, for example databases or files on the server. All messages pass through the application server, which manages routing, forwarding or filtering of the messages before queueing responses to clients.

The MUSE application server is also built on top of the MUSE middleware which handles routing incoming messages to the correct services on the server. Like the applications on the client devices the services on the application server must implement the `MUSEServiceLayerCallback` and register their name binding with the local MUSE middleware registry so that messages from the clients can be routed correctly, however the service must also publish itself to the public `RegistryService` so that the clients can find or register an interest in the service.

There are three types of service interfaces that can be implemented, user services, daemon services, and infrastructure services. A *user service* is a service that is pu-

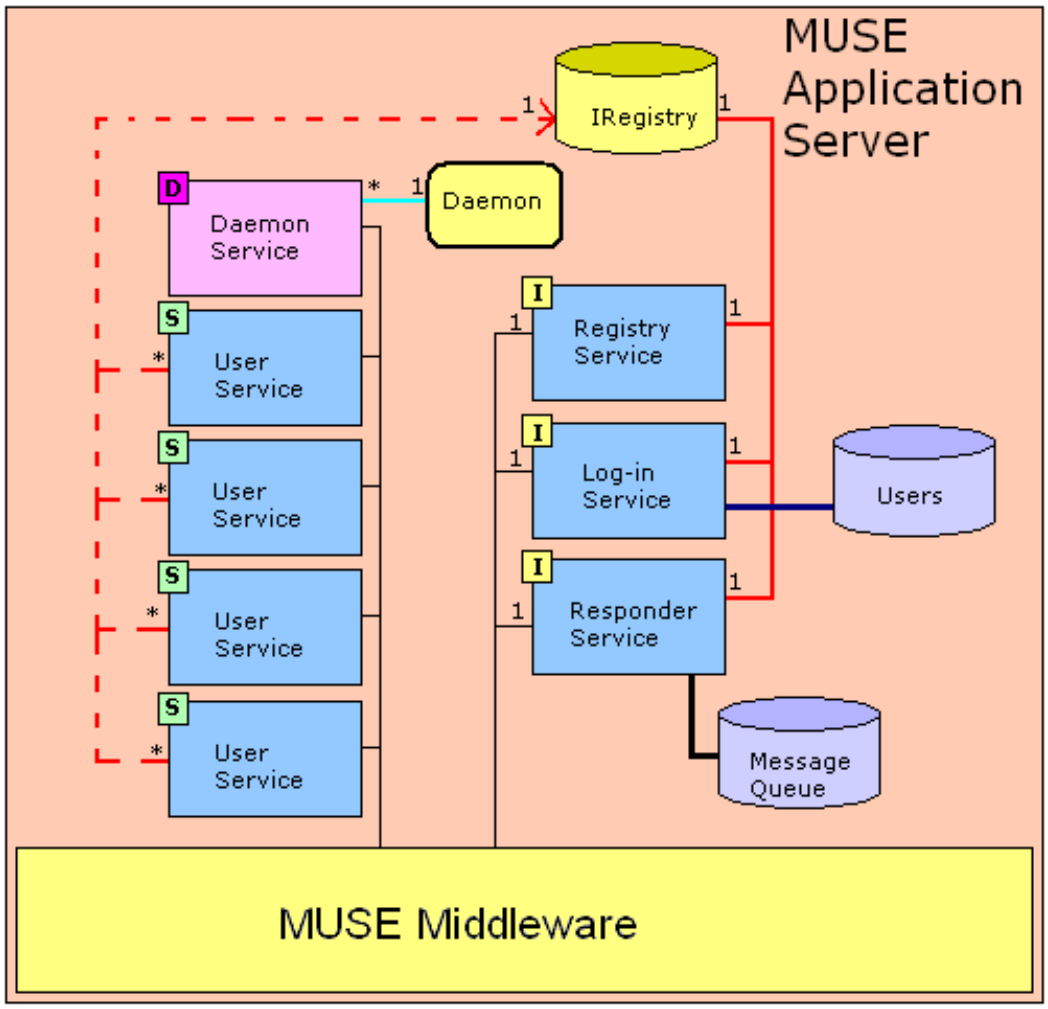

Figure 4.7: MUSE Application Server

blished so that user applications can use it through the MUSE platform. A *daemon service* is a service that executes at intervals in the background on the server without user input. Furthermore a daemon service can be configured to start after a delay and then run continuously at intervals. An *infrastructure service* is a service that can be queried directly by user services without passing through the middleware. Infrastructure services are registered in a separate registry on the application server called the `IRegistry` which is a singleton that can be accessed by all services. A given service can implement all three types of service interfaces at the same time if required. This functionality is supported to allow composite services to be created, for instance a user service can query another user service if the second service has registered as an infrastructure service, without the need to traverse the network. As an example a daemon service could run in the background and get the state of a user service periodically to archive it to a database, if the user service implements the infrastructure interface.

There are three core services that must be implemented by the MUSE application server to support the SOA and grouping functions, they are implemented as infrastructure services. They are the `RegistryService`, the `Log-inService` and the `ResponderService`. The `RegistryService` manages the publish-subscribe mechanism, and records which users are interested in receiving messages from a service. The `Log-InService` manages the users logging-in and out of the application server, and the user's location i.e. are they using a PC or mobile device. The `ResponderService` is used by all the services that wish to send replies to the clients. Services send messages to the ResponderService which queues messages in a database. This service queries the RegistryService for a list of users interested in the message and queues a copy in the database for each user. The MUSE middleware then queries this service for messages to send to the clients. The `ResponderService` keeps track of the vector timestamps received by each client and removes messages that are confirmed delivered by the clients. In this way the services and the clients are decoupled through a system that essentially uses an implementation of the producer–consumer design pattern. The `Daemon` is a background thread that ensures that each of the daemon services are executed on schedule.

4.3.5 Messaging

The MUSE middleware implementation also includes a built-in XML parser and a `MessageReader` interface. Application developers can implement the `MessageReader` interface to define custom message formats to send as payload in an `InternalMessage` envelope. However, the implementation also included a default set of message formats and readers that can be reused for any new services if required. The complete list of available formats is listed in Appendix B.

4.3.6 Distribution Model

There are two choices for the distribution model, either a peer-to-peer or client-server approach. Several of the middleware solutions described earlier used a peer-to-peer architecture e.g. [Guicking and Grasse, 2006, Roth, 2002a, Zurita and Nussbaum, 2006, Gotthelf et al., 2007, Mascolo et al., 2001a], as the authors believed that a peer-to-peer network topology was a good fit for the mobile networks which include many low-powered devices that suffer frequent disconnections. The drawback of using a peer-to-peer system is that it creates too many messages and in most cases requires multicast to all the clients in the system to work effectively. This is not so much of an issue on reliable networks using desktop computers but when using mobile devices that do not have access to cheap and reliable networks or the ability to use multicast this approach is not very practical or cost effective. Furthermore there is a lot programming overhead to implement a robust sharing, concurrency, and synchronisation model with a peer-to-peer approach. Therefore, some of the middleware solutions opted for a traditional client-server architecture as their distribution model including Mobilis [Springer et al., 2008], Pocket DreamTeam [Roth, 2003], and ESPERANTO [Cinque et al., 2005, Cotroneo et al., 2007]. In addition, several CSCL or CSCW shared editing and chat systems use a central server with a traditional client-server model for managing the shared artifacts, with examples including, COSAR [Jaspers et al., 2001] C-CHENE [Baker and Lund, 1996] and TeamRooms [Roseman and Greenberg, 1996].

In a client-server architecture for CSCW, the document or data-structure is stored on the central-server. The clients make edits remotely on this shared document and as all edits go through a central point it is possible to serialise or create a total-

ordering on edits on the document. The main problem with this architecture is the responsiveness of the client's interface as all edits are network dependent, the typical solution is to perform edits locally at the same time and only inform the clients if they performed conflicting operations, or to use caching techniques. This approach simplifies the implementation of more complex sharing models, concurrency models and synchronisation models.

The MUSE middleware natively uses a client-server architecture as its distribution model, however a peer-to-peer interaction mechanism, but not physical topology, can be emulated at the service or application levels. The MUSE platform uses a centralised server to support clients with limited connectivity in a dependable way by providing a reliable node in this system to manage interactions with shared data. Recording state information and maintaining document consistency in one place rather than using a distributed shared memory over a peer-to-peer network layer is more reliable and simpler to implement. Both Pocket DreamTeam [Roth, 2003] and ESPERANTO [Cinque et al., 2005, Cotroneo et al., 2007] used a central server for this reason. A central-server was useful from a routing and discovery perspective as this could be simplified to using a well-known address (configurable) approach to physical discovery. In addition using a centralised server was useful when implementing the system as it was convenient to attach extra networking infrastructure to the server , e.g. an MMS Gateway or RDBMS.

4.3.7 Communication Infrastructure

The middlewares for mobile collaboration described earlier, typically concentrated on providing a reliable networking layer on mobile devices. Therefore they normally focused on using one networking technology, with TCP/IP over WiFi being the most prevalent, i.e. [Zurita and Nussbaum, 2006, Gotthelf et al., 2007, Mascolo et al., 2001a, Cinque et al., 2005]. However, the Network Kernal Framework [Roth, 2002b], which focuses on networking support, provides a transparent network layer to support multiple networking protocols. Agilo [Guicking and Grasse, 2006, Guicking et al., 2008] and Mobilis [Springer et al., 2008] also provide support for multiple protocols. In order to accommodate the most heterogeneity at the networking layer the MUSE platform aims to support multiple protocols. The MUSE implementations in J2ME and J2SE currently support HTTP, TCP/IP, and MMS. The MUSE architecture provides generic

networking interfaces that can be extended to provide implementations of alternative networking technologies in future.

As well as heterogeneity in networking technologies on mobile devices, mobile networking typically encounters problems related to low bandwidth, high latency, and frequent disconnections. To overcome this issue several of the systems studied used an asynchronous, connectionless mode of communication, Agilo [Guicking et al., 2008, Guicking and Grasse, 2006], Proem [Kortuem, 2002], ESPERANTO [Cinque et al., 2005], or used an event-based notification system, i.e. Mobilis [Springer et al., 2008] and Proem [Kortuem, 2002], to reduce the number of messages that need to be transmitted over the network. The MUSE platform used an asynchronous, connectionless, and event-based networking system, in order to cope with frequent disconnections, and high latency.

In short the MUSE platform's communication infrastructure includes functionality to manage access to multiple protocols at the physical communication layer. There are additional functions to marshal and demarshall messages into formats appropriate to each networking protocol. This communication infrastructure also contains security functions to secure the payload of messages and functions to serialise or deserialise multimedia for transmission. The networking layer also includes a persistent asynchronous invocation mechanism. This invocation mechanism overcomes two problems with unreliable mobile networks, high message latency and frequent disconnections. These invocation semantics (at most once) allow the MUSE client applications to pass messages to the networking layer in a non-blocking way, allowing other operations to proceed in a high latency setting. The MUSE networking-layer queues messages to allow disconnections, the MUSE networking-layer repeatedly tries to execute the request until it has either succeeded or definitely failed. In this way, the MUSE networking-layer provides reliable network to the application layers.

4.3.8 Sharing Model

The MUSE platform has an emphasis on supporting MCSCL applications so, as stated earlier, it borrows heavily from the CSCW and CSCL fields. The output from MCSCL activities are usually shared objects and artifacts. From a distributed systems perspective, research into shared editors on desktop computers used two broad approaches

to managing shared objects: a distributed shared memory involving peer-to-peer replicated data structures or a centrally managed shared memory employing a traditional client-server model.

A distributed shared memory over peer-to-peer architecture involves each peer client maintaining a copy of the shared data-structure or document, for the DNT application, described in Chapter 5, this would be the XML document representing the concept-map, storyboard and timeline.

Shared objects can be updated with two methods, using message passing or using distributed programming constructs, e.g. CORBA objects, or RMI. The message passing approach is used by the MUSE platform, as it is flexible and easy to exchange messages in a platform independent manner, and requires less overhead. Higher-level mechanisms are constructed on top of this message passing mechanism, for instance the shared data-structures that are stored on the MUSE application server are managed using this message passing mechanism. In fact most of the previous related work opted for a message passing mechanism, with only Agilo [Guicking et al., 2008, Guicking and Grasse, 2006], Pocket DreamTeam [Roth, 2003], and ESPERANTO [Cinque et al., 2005, Cotroneo et al., 2007] using a shared object model.

Message passing and an event-based architecture are the predominant sharing model used by the MUSE platform. Alternatively if shared objects are needed they can be implemented, at the application layer by services on the MUSE application server, as centralised shared objects. Replicated shared objects are not directly supported because of the large number of messages involved and the cost of messages on mobile networks.

4.3.9 Concurrency Model

The concurrency model on the MUSE platform relies on the server controlling concurrent access to the services on the MUSE application server. The MUSE application server orders the messages using the order in which they were received. Reply messages from the server contain a vector timestamp so that clients can correctly order the messages they receive. All access to services on the application server are synchronised using Java locking and synchronisation mechanisms to ensure atomic updates to service state. Furthermore, simple transactions are supported as multiple messages to a service that

are enclosed in an enveloping message that will ensure each of the enclosed messages is executed in a fixed order at the service, and are not interleaved with other messages.

4.3.10 Synchronisation Model

The problem with a replicated structure is maintaining consistent and up-to-date copies at each of the clients while supporting concurrent editing in a distributed environment [Sun and Ellis, 1998]. Several solutions are available including turn-taking, serialisation or locking schemes, which restrict reading/writing privileges but reduce concurrency by prohibiting users editing sections of the document simultaneously [Helmer et al., 2004, Ionescu and Marsic, 2003]. Sun and Ellis review an alternative approach using operational transformation (OT) algorithms for consistency maintenance [Sun and Ellis, 1998, Ignat and Norrie, 2002]. OT algorithms replicate a shared document to all clients; users can edit any part of the document at any time. Editing operations are executed locally and then broadcast to the other clients. When clients receive a remote edit, they first transform it using the OT algorithm to repair inconsistencies, before executing it. OT is lock-free, non-blocking and is responsive because local edits are not sensitive to network latency i.e. executed locally first, therefore OT schemes are a good choice for implementing group editing on the Internet and in LAN environments.

Extensions to OT systems exploit the structure of documents to help reduce conflicts in the system, for example, in [Davis et al., 2001, 2002] they have created an operational transformation algorithm to use with XML documents, making use of the tree-structure inherent in XML documents. In a sequential flat file, an OT algorithm orders and transforms the user's actions against all other actions because a change to any part of the document affects the entire document. In tree-based schemes, consideration only needs to apply to actions on the same branch structure.

The approach adopted by the MUSE platform is predicated on the operation of the sharing model and concurrency model already described. As shared data-structures are centralised on the MUSE application server and access to them is controlled through access to the services, the MUSE application server totally orders update messages and uses Java synchronisation primitives to ensure atomic updates of shared - object state. In this way the synchronisation model used by MUSE ensures that shared objects are maintained in a consistent state, and can only move between one consistent state and

another.

4.3.11 Groupware Support

To support groupware applications in particular MUSE includes support for user and group management. Some generic collaboration services were also implemented and are also included with the MUSE platform implementation, i.e. a text chat service and history service.

Earlier in Chapter 2, this thesis recognised two important taxonomies of groupware systems. The first taxonomy was a classification by the space-time division of the participants [Ellis et al., 1991]. The second taxonomy was Mittleman et al.'s classification scheme for groupware systems, which identified nine dimensions to groupware systems: core functionality, content, relationships, session persistence, supported actions, action parameters, access control, alert mechanisms, and awareness indicators [Mittleman et al., 2008]. To emphasise the groupware support provided by the MUSE platform this subsection will apply these classification schemes to describe the groupware aspects of the MUSE platform.

In the space-time taxonomy there are four ways participants can interact with a groupware system: at the same time and a same place; at the same time and a different place; at a different time and the same place; and at a different time and a different place.

Same time, same place: This mode of operation is supported if the users are seated around a single screen or whiteboard which is displaying an application developed using MUSE.

Same time, different place: This is the standard support envisioned by the use of the MUSE platform. In this mode of operation users can interact with client applications on mobile phones or PC at the same time. The MUSE platform manages the interaction and message passing between the systems.

Different time, same place: This mode of operation is enabled through the asynchronous connectionless networking used by the MUSE platform. This enables user to log in at different times and catch up on missed messages. In addition the state of the application is persistent and stored in a database, thereby further supporting this mode of operation.

Different time, different place: This is a combination of the previous two options. Using the MUSE platform allows users to log-in from different locations on mobile devices and PCs. Using the asynchronous connectionless networking and persistence to a database allows users to log-in at different times.

The MUSE platform provides varying degrees of support for all nine dimensions of Mittleman et. al's taxonomy: core functionality, content, relationships, session persistence, supported actions, action parameters, access control, alert mechanisms, and awareness. For example, MUSE is designed with a *core-functionality* of collaborative co-authoring systems in mind, however, the eventual implementation of such a system must be at the application layer. Groupware applications can easily construct collaborative co-authoring systems, messaging systems, and group decision support systems using MUSE, however conferencing systems are not currently supported as they require real-time audio and video streaming support which is not yet part of MUSE. Elements to support these core-functionalities include: the shared and managed data-structures; native support for chat-messaging; a voting service; and an extensible service orientated architecture.

The MUSE platform is designed to use multimedia and XML *content*. The MUSE middleware is aimed toward supporting collaborative multimedia applications, so in this respect there are several features included to deal with multimedia manipulation. The MUSE middleware includes functions to serialise and deserialise multimedia files using a base-64 encoding scheme so they can be sent over any protocol that can exchange plain text. There is a service to automatically convert a series of still images into a movie, and another that can handle both images, video and audio files and create a movie file (see 5.3.1). The MUSE application server supports daemon services and has access to additional processing power not available on mobile devices to enable complex editing of multimedia files in the background on the server. Regarding XML data, the MUSE platform contains an XML parser and several readers to interpret different XML formats.

In terms of *relationships* between data the MUSE platform supports this through user services and shared data-structures on the MUSE application server. This is essentially an application layer consideration. However, services have been developed to create tree-based associations between data, named the `Tree-Service`. This service

was developed for the DNT (see Chapter 5) in order to support associations between, nodes on mind-maps, frames in a storyboard, and multimedia files in a timeline.

The MUSE platform supports *session persistence*, through its ability to connect to a external database. This is used by MUSE to store user and group status, to log or archive old messages, and to cache messages in case of disconnections. In a more general sense of session persistence, a service has been prototyped for interacting and storing output from the SMART and DNT applications (see Chapter 5) to an external learning object repository called ENLACE .

With respect to the *supported actions* available on the MUSE platform it should be noted that data is exchanged using a message passing paradigm. Therefore higher-level constructs must be implemented at the application layer using user services. The `Tree-Service` developed for use by the DNT supports a remote procedure call (RPC) mechanism with functions to add, remove, delete, edit, and move, data in a shared data-structure. Examples for other actions supported by users services include actions to insert and remove data from queues and a voting mechanism.

The above supported actions available in MUSE are parametrised with meta-data. Regarding Mittleman's *action parameters* this means that the *identifiability* aspect of actions is supported by all actions occurring on the MUSE platform including the identity of the user who executed the action. The *synchronicity* property is related to the round-trip time between a user executing an action and the response message returning to the client, so this property is network dependent.

None of the applications currently implemented on the MUSE platform make use of *access control* facilities however the MUSE platform does include facilities for implementing access control for an application. To support access control the MUSE platform includes a public key exchange function, an encryption function, and grouping functions to restrict messages on a per group basis.

The *alert mechanisms* on the MUSE platform are supported by the event-based notification system used by the platform. The MUSE platform passes on event notifications to client applications, the applications decide how to report the actual alert to the user. For example the DNT chat implementation on the PC includes an alert mechanism that provides a flashing icon and plays a sound when a chat message is received.

	Same Time / Same Place	Different Time / Same Place	Same Time / Different Place	Different Time / Different Place
MUSE	x	x	x	x
XMIDDLE	x	x	x	x
Proem	x	x	x	x
Network Kernel Framework	x		x	
Pocket DreamTeam	x		x	
MCSCL-CS	x			
Agilo	x		x	
GMAC	x		x	
ESPERANTO	x	x	x	x
Mobilis	x		x	
MobiVine				

Table 4.1: Comparison To Space-Time Classification Requirements

Awareness indicators, like alert mechanisms, are primarily an application layer consideration and are usually implemented in the GUI. The principal awareness mechanism directly built into the MUSE platform is that every user in the system has an associated colour. As every action in the system is identified by the user who executed it, this colour can be used as a source of embodiment information to represent the user in the user interface.

4.3.12 Comparison To Functional Requirements

This sections contains three tables that compare aspects of the related work described in Chapter 3 and MUSE to the functionality requirements identified in Chapter 2. In review, the previous chapter found a set of functional requirements for a MCSCL platform after reviewing the literature from CSCW, CSCL, MCSCL and mobile learning and related pedagogical theories. The first requirement is that the MCSCL platform should support all four dimensions of the space-time classification of groupware system. Secondly, the platform should support all nine dimensions of groupware technology Mittleman et al. [2008]. Thirdly from a CSCL and mobile learning perspective it should support constructionist, contextualised and collaborative activities. Finally, it should support mobile applications for data collection, location awareness and collaboration. ThereforeTable 4.1 on page 83 examines the support for the four dimensions of the space-time classification of groupware systems. Then Table 4.2 on page 85 lists the level support for the nine dimensions of groupware technology from Mittleman et al.

[2008]. Finally Table 4.3 on page 86 displays whether each middleware system explicitly supports constructionism, collaboration, contextualisation and support for location aware applications and data collection.

Starting with the four dimensions of the space-time classification of groupware systems. MUSE and three of the described systems provide some support for all four interactions mechanisms i.e. XMIDDLE Mascolo et al. [2001a,b], Proem Kortuem [2002], and ESPERANTO Cinque et al. [2005], Cotroneo et al. [2007]. However, the remainder of the systems only support synchronous operations (same time / same place and same time / different place), while MCSCL-CS Zurita and Nussbaum [2006] was designed to support synchronous operation within a classroom setting.

Next Table 4.2 on page 85 shows how MUSE compares to the middleware systems described in 3 and depicts the level of support each middleware system provides for Mittleman's nine dimensions of groupware technology: i.e. core functionality, content, relationships, session persistence, supported actions, action parameters, access control, alert indicators, and awareness mechanisms. This table contains two types of elements. The first element type is a simple checklist, where cells marked with an 'X' indicated that the system supports the given feature. The second type is sub-list of items that the system supports for a given feature.

The first type of elements are straightforward and can be easily understood from the table. The second type are bit more complex so this section will briefly describe support for: core functionality, content, relationships, and supported actions.

Core functionality was defined earlier as collaborative co-authoring systems, messaging systems, group decision support systems, and conferencing systems. MUSE is designed predominantly to support collaborative co-authoring systems but includes tools to support messaging and group decision support systems. Most of the systems are best classified as messaging systems, without explicit support for other collaboration technologies which must be completely implemented at the application layer. However Mobilis and XMIDDLE and can support simple collaborative editing systems and Agilo can support a group decision support applications.

In terms of content, MUSE is the only system to include explicit support for text, images, audio, video and raw data content. However, Mobilis comes the closest in this regard with stated support for text, images, videos, and audio. The remainder of

Middleware	Core Functionality	Content	Relationships
MUSE	Collaborative Editor Messaging System Group Decision Support	text, image, audio, video, data	collection, list tree, map
XMIDDLE	Collaborative Editor	text	tree
Proem	Messaging System	data	data spaces
Network Kernel Framework	Messaging System	text,data	tree
Pocket DreamTeam	Collaborative Editor	text,data	list, tree
MCSCL-CS	Messaging System	data	
Agilo	Messaging System, Group Decision Support	text, data	
GMAC	Messaging System	text	tree, list, collection
ESPERANTO	Messaging System	data	tuple space
Mobilis	Collaborative Editor Messaging System	text, image, audio, video	map
MobiVine		data	

Middleware	Session Persistance	Supported Actions	Action Parameters	Access Control	Alert Mechanisms	Awareness Indicators
MUSE	x	add, receive, associate, edit, move, delete, judge	x	x	x	x
XMIDDLE	x	add, receive, associate, edit, move, delete				
Proem	x	add, receive	x	x		
Network Kernel Framework		send, receive		x		
Pocket DreamTeam	x	send, receive				
MCSCL-CS	x	send, receive				
Agilo		add, receive				
GMAC		add, receive, associate, judge			x	
ESPERANTO	x	add, receive,				
Mobilis	x	add, receive, associate			x	
MobiVine		add, receive				

Table 4.2: Comparison To Functional Requirements From Mittleman et al. [2008]

Middleware	Constructionism	Collaboration	Contextualisation	Location awareness	Data collection
MUSE	x	x	x	x	x
XMIDDLE		x			
Proem		x		x	
Network Kernel Framework		x			
Pocket DreamTeam		x			
MCSCL-CS		x			
Agilo		x			
GMAC		x			
ESPERANTO		x			
Mobilis	x	x	x	x	x
MobiVine				x	

Table 4.3: Comparison To CSCL And Mobile Learning Requirements

the systems described are designed for text or raw data and therefore require that the support for additional content types must be provided at the application layer by the application developer.

Regarding support for relationships between content, MUSE supports the widest variety of relationships including: collections, lists, trees, and maps. GMAC supports trees, lists, and collections and PocketDream Team supports lists and trees. All other systems only natively support one data structure type to express relationships between data.

In terms of supported actions, Mittlemans classifications describes seven types of supported actions that can be performed with content: add, receive, associated, edit, move, delete and judge. MUSE supports all seven supported action types. The rest of the other systems only support add and receive and rely on the application developer to provide support for these actions.

The level of constructionism, collaboration, collaboration, location-aware and data-collection activities is depicted in Table 4.3 on page 86. All the systems surveyed included some support for collaboration, but this is not surprising as support for collaboration was the *sine qua non* feature for inclusion in this review. MUSE include support for all of these features. In addition, Mobilis provides some level of support for all these features. Proem includes support for location aware applications. None of the other systems described included any direct support for constructionism, contextualisation, location-awareness and data-collection activities.

In conclusion, MUSE includes support all dimensions of the space-time classification. Furthermore the main result that is clearly visible from Table 4.2 on page 85 is that MUSE is the only system that is able to support all nine as described earlier in this section. Finally it is the only system that could be found to include full support for constructionist, collaborative, contextualised, location aware and data-collection activities.

4.4 Conclusion

To provide context this chapter briefly outlined the technical challenges of developing software in the mobile computing domain. The conceptual architecture of the MUSE platform was described. To deal the heterogeneous nature of the mobile computing environment the conceptual approach includes: the insistence on open XML messaging formats; a design-by-contract approach to middleware design; the use of a distributed model-view-controller (MVC) interaction paradigm; and using interfaces to hide hardware or OS dependant implementation details from the application layer.

This chapter described an implementation of the MUSE platform using J2ME and J2SE in detail. In short the MUSE platform contains three core components: the MUSE application server, a collection of infrastructure services and the MUSE middleware. The MUSE middleware is composed of the MUSE service-layer and the MUSE networking-layer. The MUSE service-layer supports the event-based notifications and service orientated architecture aspects of the platform. The MUSE networking-layer provides for transparent communication between the client and server over different networks, i.e. TCP/IP, HTTP, and MMS.

In Chapter 3 it was noted that lessons learnt from CSCW, CSCL and mobile learning were important to the design of platform to support MCSCL applications. Starting with the CSCW field, two taxonomies were described to classify groupware applications. The first taxonomy is the space-time [Ellis et al., 1991] classification of the participants. The second taxonomy classified groupware applications according to their functionalities [Mittleman et al., 2008]. To highlight the groupware support of the MUSE platform this chapter outlined how the MUSE platform enables applications to support the four categories of the space-time division, and the nine dimensions of Mittleman et. al.'s

classification. In addition, particular features of MUSE middleware are highlighted according to the framework of variations points [Gotthelf et al., 2007] in groupware systems.

Furthermore, the trend is towards generic open systems in CSCL [Lonchamp, 2006]. This influenced the design of MUSE platform to use a service-orientated architecture to enable development of flexible tools for learning. While [Dimitracopoulou, 2005] classify CSCL systems as either text-production or action-oriented. The messaging system on the MUSE platform makes developing text-production orientated systems simple. Alternatively the MVC paradigm facilitates an action-oriented approach to user interaction with the applications. For instance, when a user updates the state of the system it is sent to the server for further processing and the result is displayed as an update to their view, this then provides context for their next interaction with the system.

From a technical perspective the MUSE platform is informed by previous research on middleware design, distributed systems and uses established design patterns to extend support from the PC to the mobile computing domain to provide a suitable platform on which to create mobile learning applications. In Chapter 5 this thesis will describe two such MCSCL applications built to enable constructionist, contextualised and collaborative approaches to multimedia production and learning.

Chapter 5

Services And Applications

MUSE was developed for MCSCL applications in general but with the area of animation and digital video production particularly in mind. There is a growing body of evidence in the literature that digital video production (DVP) can facilitate powerful learning experiences, including collaborative learning, problem solving, critical thinking, and creativity, while encouraging development of media literacy, communication, and presentations skills e.g. [Buckingham, 2003, Buckingham et al., 1999, Hofer and Swan, 2005, Kearney and Schuck, 2005, Posner et al., 1997, Reid et al., 2002, Swain et al., 2003]. Animation is an analogous process to digital video production that shares many of the educational benefits, for example, Collins et al. consider animation a subset of video [Collins et al., 2000].

Yet digital video production and animation are not without their problems. While the cost of digital video cameras continues to decrease, they remain a relatively expensive resource and more crucially DVP is a time consuming activity with resulting issues relating to access and control [Burden and Kuechel, 2004]. Filming and editing, which offer the most learning benefits, involve time-consuming classroom management, making DVP somewhat impractical as a whole class activity leading to scheduling problems in the classroom [Arnedillo-Sanchez and Tangney, 2006, Reid et al., 2002]. To date little use has been made of ICT to support the collaborative, as distinct from the video production aspect of the process, and would therefore benefit from applications in the MCSCL space.

This section will outline the educational benefits and current problems with digital video production and animation learning activities. Then the mobile Digital Narra-

tive Approach (mobileDNA) [Arnedillo-Sanchez and Tangney, 2006, Arnedillo-Sanchez, 2008] is described as an approach to support and encourage collaboration in the digital narrative and mobile learning spaces. Difficulties experienced with this approach are outlined.

Then two MCSCL applications, SMART [Byrne and Tangney, 2006, 2007a,b, 2009] and the DNT [Byrne et al., 2008, Arnedillo-Sanchez, 2008], are described. Next the services that are provided on the MUSE application server to support such applications are described in turn. Both applications were designed with a view to supporting constructionist, contextualised, and collaborative learning activities.

SMART is an application for creating animations on a mobile phone using the stop-motion animation technique. The DNT is an application that works on mobile devices and PCs to support users following the mobileDNA script. SMART uses MUSE for sending images and XML data to the MUSE application server. There is a SMARTService, running on the MUSE application server, which converts the images and XML data-structure into a completed movie file. The DNT is a more fully featured groupware application for digital video production. It makes full use of the facilities provided by the MUSE platform and several services on the MUSE application server. In addition, the design of the user interface of the DNT application was informed by CSCW, CSCL and mobile learning domains, and these aspects are outlined in the context of describing the GUI.

5.1 Digital Video Production And Animation

There is a growing body of evidence in the literature that Digital Video Production (DVP) can facilitate powerful learning experiences. [Buckingham et al., 1999] argue that "the high quality of digital video can encourage pride and a sense of ownership" and can be very motivating for students. While for [Kearney and Schuck, 2005] digital video projects encourage development of media literacy, communication and presentations skills. Clayton found that DVP boosted the confidence of students while fostering, understanding of film making, technical skills, and comfort with technical language [Clayton, 2002]. Swain et al. argue that video production is a "Mindtool that promotes critical inquiry, problem solving, and critical thinking" [Swain et al.,

2003]. [Buckingham, 2003] emphasises the collaborative learning aspect of DVP while [Reid et al., 2002] highlights that it encourages creativity and self-expression. According to [Hofer and Swan, 2005] digital video presents "powerful opportunities to design student-centred, inquiry–based projects". Potter consider digital video production a new literacy practice based on case studies of young children engaged in digital video production Potter [2010]. Given this research, unsurprisingly the adoption of DVP in schools is increasing [Burden and Kuechel, 2004].

Animation is an analogous process to digital video production and it shares many of the potential educational advantages while being a simpler activity. Collin et al. consider animation a subset of video, of which they recognise three such divisions: live action; animation; and talking heads, e.g. face-to-face video conferencing [Collins et al., 2000]. The important distinction between live action and animation is that live action records real life events as they occur whereas animation creates the illusion of life and motion from static frames. This allows users to depict scenes not possible with live action, e.g. making objects appear to move by themselves or illustrate dynamic or scientific processes like blood moving around the body [Collins et al., 2000]. Digital video production and animation, and more generally moving image media, are familiar even to pre-school children [Marsh and Thompson, 2001] and "learning activities which incorporate them may help to connect school life with the wider world" [Madden et al., 2008].

Earlier studies of animation for learning largely focused on studying how viewing animations of dynamic systems helped users to understand complex systems. For example, one study of the use of animation on student understanding of computer algorithms [Byrne et al., 1999] notes that their "experiments show a trend towards a benefit of animations and predictions on students' ability to solve procedural problems about algorithms". On the other hand, recent research has supported interactive and constructionist approaches to using animation. [Tatar et al., 2003] describe a project to animate scientific processes dynamically using Sketchy, an animation and drawing tool for PDAs. An advantage of this system is that users engage in a constructivist process, for instance exchanging drawings and animations to uncover misunderstandings of scientific phenomena [Tatar et al., 2003] . In a different subject domain [Zagal et al., 2004] use animation to support storytelling by children aged 11 – 12 using software cal-

led Alice. Alice is a 3D programming environment and teaching tool for introductory computing that enables users to tell stories using animation. To control the animation, and on-screen characters, the users drag-and-drop graphical objects, which represent statements in a programming language. While [Zagal et al., 2004] describe this animation activity as a success they note that a supportive social context is important for children to become authors of multimedia in an educational context, e.g. collaborative skills are necessary as is providing structure to scaffold the animation activity.

There are several different animation techniques, for instance, cell animation or computer animation; however this study uses the stop-motion animation technique because it is "concrete and easy to approach for the beginner" while supporting development of additional skills including hand-eye coordination [Hamalainen et al., 2004]. Traditional stop motion animation involves shooting a movie one frame at a time, changing drawings of characters slightly between each, thereby creating the impression of movement. Stop-motion animation is not limited to drawings, with variations of the technique using clay models, LEGO bricks, everyday household objects, and people. For example Animaatiokone [Hamalainen et al., 2004] is a system for creating clay animation and learning about stop-motion animation. The Animaatiokone installation consists of a desk to stage the animations, a mounted camera to capture the images, and a mounted screen to view the animations and timeline. In addition, Animaatiokone supports collaboration by allowing users to share clay models and extend previous users' animations. One limitation is that the Animaatiokone installation is large and fixed to one location therefore animations can only include objects and drawings that can fit into the machine.

To date little use has been made of ICT to support the collaborative, as distinct from the video production aspect, of DVP. DVP is typically a sequential process involving planning, storyboarding, scripting, filming, and editing. Though all group members can be involved in the early stages, filming and editing are a bottleneck with only some group members participating at these stages, which means that at the editing stage only a few of the participants are engaged and benefiting educationally from the digital video production process. For instance, in one study [Kearney and Schuck, 2005] of the effect of digital video production on pedagogy, students used DV cameras for filming and used iMovie to edit the film. However, the production process followed

a predominantly linear structure from storyboarding, scripting, filming through to editing. The study supports the pedagogical benefits of DVP nevertheless the use of ICT is concerned with productivity and ease of use of the technology for the students rather than enhancing collaboration in the process. In addition, Reid et al. describe a study [Reid et al., 2002] where students produced documentaries on history topics using digital video, yet this process again followed a linear progression from idea-generation through to final editing, with the ICT being used for content production, rather than enhancing collaborative opportunities.

DVP and animation are of course not without their problems. While the cost of digital video cameras continues to decrease, they remain a relatively expensive piece of equipment with resulting issues relating to access and control [Burden and Kuechel, 2004]. While limitations imposed by the availability of equipment may actually work to the advantage in fostering group work, DVP is perceived by many as a disruptive practice requiring "herculean classroom management efforts" to overcome the technological shortcomings [Reid et al., 2002]. Filming and editing, which offer the most learning benefits, involve time-consuming classroom management, making DVP somewhat impractical as a whole class activity leading to scheduling problems in the classroom [Reid et al., 2002, Arnedillo-Sanchez and Tangney, 2006]. Using a digital camera means that images have to be transferred to a desktop machine and loaded into another application to create the final movie or animation. For a whole class activity access to the desktop may prove to be an issue, images are processed in a different physical location to which they were captured and the learner is required to master two pieces of technology. Carrying out the image capture and animation editing on the single mobile phone device means that the ready-at-hand nature of the technology is being exploited, only a single application needs to be mastered and where a large number of learners are involved it is much easier for different groups to work in parallel. Furthermore, the in-built communication facilities of a phone mean that the learners can easily exchange images, and completed animations.

5.1.1 The Mobile Digital Narrative Approach (mobileDNA)

The mobile digital narrative approach (mobileDNA) [Arnedillo-Sanchez and Tangney, 2006, Arnedillo-Sanchez, 2008, 2009] is a pedagogical methodology specifically designed

to scaffold collaborative creativity among distributed learners engaged in the creation of digital narratives with mobile technologies. Although it is informed by traditional DVP approaches, it distinctly differs from these in that the mobileDNA short-circuits the planning, scripting, and storyboarding stages and parallelises shooting and editing (see Figure 5.1 on page 95). This parallel approach has the advantage of keeping more learners engaged throughout all stages of the process as compared with the sequential approach. Typically during the sequential process only a few learners remain involved during the editing and filming stages, as seen in Figure 5.1 on page 95. Furthermore the mobileDNA uses ideas from theatrical studies and improv-theatre to help with story generation and borrows from computer science, promoting a rapid prototyping philosophy to creating the completed video, aiming to have a finished product in a few hours rather than the longer (days) timescale with traditional sequential approaches. Thus, it enables the synchronous participation of an entire group of learners in all phases of the production and lowers the time demands. At the heart of the mobileDNA methodology there is a three-phase task-oriented digital narrative creation activity involving Story Generation, Shooting and Editing, and Production and Screening.

During *Story Generation* the participants, collaboratively and face-to-face, create the story to be told. When *Shooting and Editing*, they are divided into three groups: the *Image Group*, who shoot the visuals and act the parts in the movie; the *Sound Group*, who record the audio (dialogues, narrations, sound effects, etc); and the *Editing Group*, in charge of assembling the media created by the other two groups. A peculiarity of this phase is that the Image and Sound groups separately go on location to shoot the story while the editing group stays in the editing station. As the media is being captured with mobile phones this is transferred via Multimedia Messaging Service (MMS) to the editors who can start editing shortly after the image and sound groups have arrived on location and started shooting and recording. By the time the crew and cast are back at the editing station, the first version of the digital narrative is ready for viewing. One consequence of shooting and editing simultaneously is that it allows more users to stay engaged in the process compared to the sequential approach, which often results in some users being idle or disengaged. During the *Production and Screening* phase, the participants review the digital narrative in the making and engage in a critical evaluation of their work, which may lead to additional cycles of

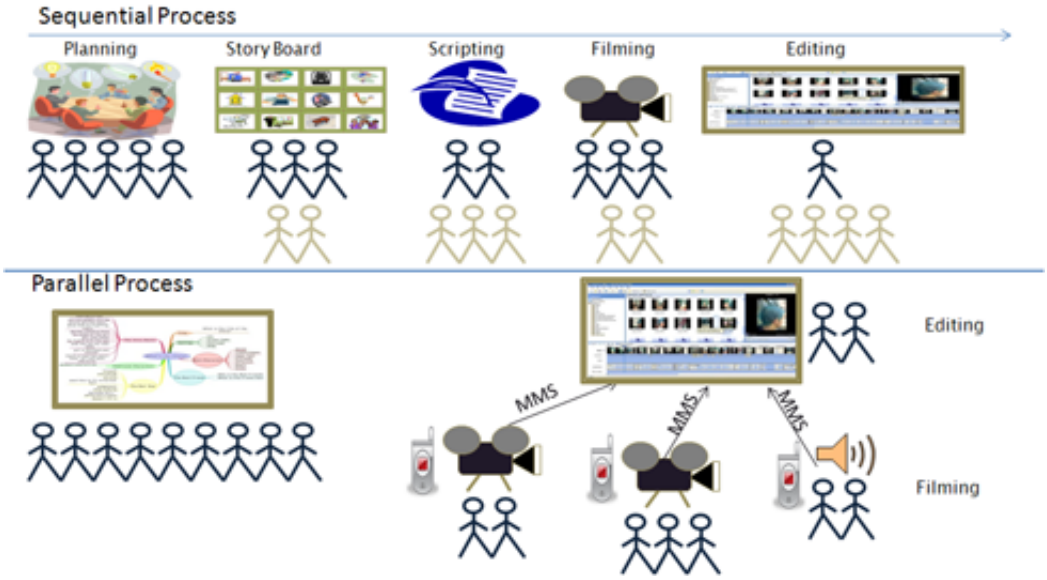

Figure 5.1: Sequential vs. Parallel Process

targeted shooting and recording. Editing continues until the group is satisfied that their digital narrative is ready for screening.

In order to implement and evaluate the mobileDNA and its three-phase digital narrative creation activity, a prototype of the DNT was developed by combining several pieces of proprietary software. These include: a scripting tool (developed with a mind-mapping application) that scaffolds the story generation, and outputs a paper-print of the story script; an MMS gateway to transfer files from the mobile devices to the computer; a PC operating system to manage the media files; a video editing application to edit the movie; and the native interfaces on the mobile devices for media capture. The next section expands on the difficulties experienced by the learners that participated in previous mobileDNA workshops using this prototype. The remainder of the chapter then describes the design of SMART and the DNT and highlights aspects of their design to ameliorate these difficulties.

5.1.2 Difficulties Encountered With The mobileDNA

To date, 200 people have participated in mobileDNA workshops and successfully created collective digital narratives by following the mobileDNA script and using the DNT prototype [Arnedillo-Sanchez and Tangney, 2006, Arnedillo-Sanchez, 2008, 2009]. The evaluation of the methodology and the participants' experience is outside the scope

of this thesis. However, findings from multiple case studies clearly indicate that the data and work-flow underlying the digital narrative creation activity and the DNT prototype efficiently support the emergence of productive collaborative creative interactions among participants [Arnedillo-Sanchez, 2009]. Notwithstanding the previous, the lack of integration among the various tools that make up the DNT prototype present difficulties for the users. These include: maintaining a shared understanding between the filming and editing groups; managing media files; communication between groups; and scaffolding the process when there is no integration between software. The following paragraphs further describe the findings reported by Arnedillo-Sánchez [Arnedillo-Sanchez, 2008, 2009].

Maintaining a shared understanding is a problem in all collaborative activities. During the first phase of the mobileDNA activity, the users create a story script with the scripting tool. To compensate for the lack of explicit connection between the application used to create the story script and the native interfaces of the mobile phones used to capture the media, the participants carry with them on location a paper-copy of the script. However, this is not ideal in practice and leads to lack of a shared understanding between the filming and editing groups. Such misunderstandings largely occur if there are any changes between the story generation and shooting, and between shooting and editing.

As part of their task, the image and sound groups create visuals and sounds that they consider will contribute to the digital narrative creation. These groups often shoot and record scenes taking advantage of what is available in their surroundings and hence, not necessarily in the same order as in the script. Additionally, the filming groups may create several media pieces together and send them to the editor in a batch process later. The implication of this is that the editing group does not receive media in timeline order and can receive multiple similar media assets for the same scene. Consequently, the editors must attempt to assemble the media into a digital narrative without the context and access to the original reason or motivation for capturing a piece of media.

Conversely, there is little feedback from the editors to the filming groups so users capturing media are unaware of the progress of the overall digital narrative with the result that they often do not capture enough images for a scene. As the filming groups

use MMS in a one-way fashion to send media to the editors, the editors must wait until the filming groups return to the base location to inform them of this dearth of images. Providing this feedback involves conscious effort by the editors to inform the filming groups of the lack of images. Another factor related to a lack of shared understanding is the difficulties with file management experienced by editors, as they cannot directly relate media files received from the filming groups with the story script. For instance, without additional information it is impossible to judge what a sound file contains before actually listening to it.

The mobileDNA uses several pieces of proprietary software, including: a mind-mapping application to create the mind-maps; an MMS gateway to transfer files from the mobile devices to the computer; a PC operating system to manage the media files; a video editing application to edit the movie; and the native interfaces on the mobile devices for media capture. As there is no integration between each of these programs, there is no support for users as they switch from one program to another. For instance the participants use the mindmap application on a PC and then print a copy for each group. They then use the mobile devices to capture media. Then they either send the media via MMS to a MMS gateway or transfer the files to the PC using a cable. The users then must find the media on the file system and move it to their working directory. Lastly they use Microsoft movie make to create the video. Therefore the users must learn to use several pieces of proprietary software, which is not central to the digital narrative activity itself.

In review, digital video production and animation are established activities for learning that would benefit from greater support for collaboration. The mobileDNA advocates a parallel approach to filming and editing, so aiming to increase user participation and engagement throughout the process while promoting greater collaboration between the users. The mobileDNA uses MMS on the mobile phone to provide additional support for collaboration in the filming and editing stages.

SMART and the DNT are two applications designed to add greater support for collaboration in these activities, therefore the following sections introduce the design and implementations of SMART and the DNT in turn. Both applications are built on the MUSE platform described in Chapter 4. The DNT application is informed on the DNT prototype (Arnedillo-Sanchez, 2008), additional data and work-flow requirements dis-

tilled from the evaluation of the mobileDNA workshops, and user interface features suggested by research into CSCW and CSCL.

5.2 Stop-Motion Animation And Reviewing Tool

As described earlier, digital video production can provide many opportunities for learning [Buckingham et al., 1999, Buckingham, 2003, Hofer and Swan, 2005, Kearney and Schuck, 2005, Posner et al., 1997, Reid et al., 2002]. Animation is a related, yet simpler, activity that shares many of the educational advantages of digital video production [Madden et al., 2008]. However, both activities can be time consuming, involve using a diversity of devices and are non-trivial to implement as whole class activities. This chapter advocates developing a dedicated application for mobile phones that uses the cameras, communications facilities, and ready-at-hand nature of mobile phones to help overcome these problems. The specific focus of this section is the design and implementation of a mobile learning application called the Stop-Motion Animation and Reviewing Tool (SMART) [Byrne and Tangney, 2006, 2007a,b].

The application enables mobile phone users to create animations using the stop-motion animation technique. SMART adheres to the constructionist, contextualised, and collaborative approach to developing learning applications for mobile devices argued for by [Patten et al., 2006]. SMART allows users to capture images, sequence the images using a filmstrip paradigm, insert title cards, and view the completed movie, all on the mobile phone. From a technical perspective, an XML document represents the animations, which can be transferred to a PC for rendering, using the networking facilities and video generation services provided by MUSE [Byrne et al., 2008].

5.2.1 The SMART Implementation

Animation supported by mobile devices is an activity that lends itself to the argument of [Patten et al., 2006] that an MCSCL tool should encourage elements of a constructionist, contextualised, and collaborative approach to learning. Animation is an inherently constructionist activity with the learner required to create scenes and characters at the physical level and the actual animation itself at a higher level of abstraction. A level of contextualisation is achieved through the choice of topic the animation addresses and

Figure 5.2: SMART Graphical User Interface

the mobility facilitated by the phones, which enables the users to incorporate elements from their surroundings in their animations. The activity lends itself to collaboration since there are easily separated tasks, which different participants can undertake. Furthermore, requiring the learners to display their finished animation to their peers and to reflect on the product, and the process used to create it, promotes a constructivist approach to learning.

Mobile phones generally include support for J2ME. By using J2ME, it is possible to create sophisticated applications that use the multi-media, image capture and communications features of the devices. In addition, the communications facilities enable users to collaborate and exchange animations, thereby supporting a collaborative, and constructivist approach to animation.

SMART, supports the shooting of small animated-movies using the stop-motion animation technique. A user can capture frames, containing images from their surroundings, drawings or clay models (created by themselves or others) and additional

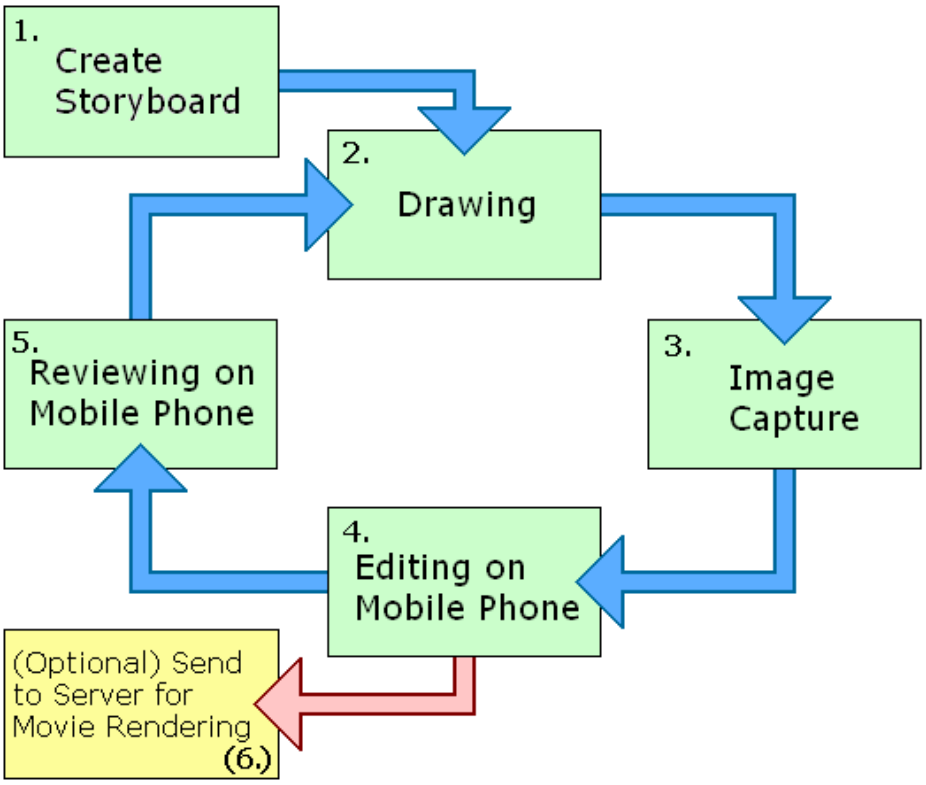

Figure 5.3: SMART Process

real world objects. Several frames form a scene, and an animated movie can contain numerous scenes. When users select a scene to edit, the application displays a filmstrip. Users can use the filmstrip screen to add, reorder, or delete frames. Similarly, users can add, reorder, or delete scenes. They can then review their work by playing the full movie or previewing individual scenes. SMART also includes a facility to add text frames to the filmstrip, which is a concept borrowed from silent movies and helps the users tell the story.

Described here is a typical interaction and usage of the system from starting a project to creating the final movie. Initially the users in a group create a simple storyboard, typically containing three to six story elements, to represent the story for the animation. Then they create and gather the artifacts to use in the animation, for example, drawing images, or sculpting clay models. The users then switch to SMART to create the animation. First, the users create a new movie, the top-level unit in SMART. The users then create a number of scenes that will comprise the completed animation. Once a scene is selected a series of frames for the scene can be captured using the integrated camera. When the users are happy that they have captured the

appropriate frames, they can edit the scene, reordering or removing frames as desired. The users can then preview the scene or view the complete movie on the mobile device. When the movie is complete, users have the option of sending the movie to a server where it can be rendered into a format that allows it to be distributed to non-SMART users. In addition, on the server the users can add sound to the animation if desired.

The application was developed on Sony-Ericsson™ K750s, Nokia™ N73s, and Nokia™ N95s mobile phones, all of which have integrated cameras and support Bluetooth, MMS, and J2ME. SMART can be ported to any other similar mobile phones. SMART is deployed as a J2ME MIDLet using the connected limited device configuration 1.1 (CLDC) and the media information device profile 2.0 (MIDP). In addition, SMART requires access to the PDA optional packages and the Wireless Messaging API 2.0. SMART uses XML to represent the animations. XML is used as it is an open format and third-party developers can easily parse the XML to create tools to manipulate the animations, for example, adding a soundtrack.

However, the J2ME platform has several technical constraints, which include a lack of memory and processing power for multimedia manipulation, the restrictions of the J2ME security model, and difficulties ensuring that applications are portable on multiple mobile phones.

Currently the processing power and memory limitations on small mobile devices are a constraint on performing advanced image and video editing tasks. Additionally, the lack of support for the Java Media Framework APIs is an obstacle to converting a series of JPEG images into a portable movie file format. There is also a deficit of third party tools supporting file conversion operations on mobile devices. Therefore, if the user wishes to convert the completed animation into a portable movie file format, for example 3GP, they must send it to a server for further processing using the MUSE platform. 3GP is a multimedia container format defined by the Third Generation Partnership Project (3GPP) for use on 3G mobile phones. As this is a standard file format, its use supports portability of the application and allows users who do not have J2ME, or SMART, installed on their mobile phone to view the animations. In addition, it was impossible to implement an onion-skinning function on the mobile phones because they lacked the processing power, memory, and advanced media manipulation facilities. Onion skinning is an animation technique that displays the current frame overlaid on

the previous frames to help the animator to align the next frame.

Java MIDLets use a sandbox model similar to Java Applets and therefore require that the application be signed with a valid digital certificate before permission to use many for the functions of the phone can be granted, e.g. local file system access, camera functions, and MMS facilities. This is a practical barrier to deployment because each phone model and each mobile network operator can require a different set of valid certificates. Each certificate costs between €200 and €400 per year, so it can be expensive to buy the correct certificates for wide deployment on users' mobile phones.

The graphical user interface (GUI) for the application (see Figure 5.2 on page 99) is built predominantly using the high-level API of the MIDP. This has the advantage that as the GUI components are defined in a device independent manner it simplifies porting the application to other mobile devices. A supplementary advantage is that menus and buttons in the application will operate using the same interface paradigm as the native interface on the mobile device, thereby simplifying the task of learning to use the application. However, some of the unique requirements of the application necessitated creating specific interfaces using the low-level Java API of the MIDP, for instance, the filmstrip screen. These APIs control drawing directly to the display on the phone. The user interface design and development must consider this when dealing with the varying display sizes on different mobile phone models.

5.2.2 Future Work

SMART is predominantly used as a standalone mobile application, which the users can use to create and view animations all on the mobile device, however, SMART does use the networking facilities of MUSE to communicate with the SMARTService on the MUSE application server to convert the animation into a movie file. It also uses the *encoding, sensor* and *file-system* components of the MUSE auxiliary-layer to convert images to text for transport over the network, capture the images and save them to the device, respectively.

The DNT is a more fully featured application for digital video production and is described in the next section. The DNT makes use of several services available on the MUSE application server, including chat facilities for communication, a mind-mapping tool for idea-generation, and a timeline editor tool for advanced editing the movie and

adding a sound track. At this point it is relevant to ask if integrating future versions of SMART to use MUSE as a middleware platform and to use the DNT to edit the animations would improve the software and further support learning. According to the original design considerations for SMART, the software should support a constructionist, contextualised, collaborative approach to learning. This section will take each design consideration in turn and discuss the benefits of adopting the MUSE open architecture to provide common support across the desktop, web, and mobile platforms and using the DNT shared workspaces to edit the animations.

If SMART used MUSE to communicate with the DNT application then the DNT GUI would provide an interface for editing animations on the PC, in addition to the current mobile only interface. This configuration would support and simplify further 'post-production' of the animations, which is not technically feasible on mobile devices now. The first activity that would benefit from this approach is the ability to simplify adding and editing a soundtrack to the animation. This additional facility would enhance the finished animation while continuing to support a constructionist approach to animation.

Further integration with MUSE and the DNT would present options for the users to contextualise their work, as there would be a more diverse range of platforms from which the users could interact with the animation. Besides the DNT editor would enable the users to include not only elements from their surroundings in their animations but previous animations and other media.

Increasing opportunities for collaboration is an advantage of using an open system like MUSE. While a central benefit of SMART is that the application is self-contained on the mobile device, supporting stop-motion animation from start to finish all on the device, additional collaboration involving MUSE and the DNT application on the PC would provide new opportunities for collaboration. Firstly, the users could split into groups with some users creating characters for the animation and capturing images, while other users would use the DNT to edit the animation or allow the groups to work on the scenes in parallel. A second option is to increase collaboration by enabling users on multiple PCs to edit the animation. This is possible as the DNT contains shared-workspaces that allow users on separate PCs to edit the filmstrip simultaneously.

MUSE can send any images captured by SMART automatically to the DNT editor

if required. This enables faster feedback between users creating the animations and users editing the animation on the PC. This feedback together with the other communications facilities, including text chatting tools, available within MUSE enables the users to engage a constructivist process including the negotiation of meaning between the collaborators, even if they are not co-located.

As mentioned earlier, SMART operates principally as a standalone application mainly using MUSE to transfer images and files to a PC to generate 3GP movie files if required. However due to the limitation of this process SMART is currently being integrated to work with the DNT via the MUSE platform. In this case, the DNT GUI will provide an interface for editing animations on the PC, in addition to the current mobile only interface. This configuration supports further 'post-production' of the animations, which is not technically feasible on mobile devices now. In particular, this gives the ability to add and edit a soundtrack for the animation. The DNT's shared workspaces support additional levels of collaboration. Therefore, integrating SMART with MUSE and the DNT could benefit and enhance each of the pedagogical underpinnings of SMART. However, SMART should also retain the ability to operate as a standalone application on a mobile phone to satisfy an original design goal to support animation without the need for access to computers.

5.2.3 Conclusion

This section argues that stop motion animation can provide the potential educational advantages of digital video production while also being a simpler and less expensive activity in which to engage. However, both digital video production and animation are time-consuming activities with problems regarding access to expensive equipment. This chapter recommended using mobile technology to overcome these issues. Therefore, this chapter described SMART, an application designed to support a constructionist, contextualised, and collaborative approaches to making animations on mobile phones. A noted advantage of which is that as the application is implemented on a mobile phone it has, among others, the benefit of being relatively inexpensive, can exploit the ready-at-hand nature of the device, and it is a familiar technology.

In addition, this thesis noted the trend in software development towards open-systems and therefore described the current state of integration of SMART and the

MUSE platform, which is developed to facilitate the development of cross platform applications in the area of digital narrative production. The chapter noted some current limitations with SMART and described how further integration between SMART, the MUSE platform and the DNT would provide additional opportunities to support constructionist, contextualised, and collaborative approaches to animation. Thus, the next section describes the design and implementation of the DNT in detail.

5.3 Digital Narrative Tool

The mobileDNA was introduced earlier as a pedagogical methodology specifically designed to scaffold collaborative creativity among distributed learners engaged in the creation of digital narratives with mobile technologies. Two notable feature of the mobileDNA are the use of mind-maps for idea-generation and that filming and editing occur in parallel. At the moment the mobileDNA uses many disparate applications, software and hardware to achieve its learning objectives. Although, it is possible to create digital narratives using the mobileDNA and off-the-shelf technology the process is not seamless. The digital narrative tool (DNT) described here is designed to support this pedagogical methodology within a single shared workspace.

With CSCL applications, the challenge of managing a shared workspace manifests itself in the users' ability to maintain a shared understanding or a shared conception of the problem to be solved [Roschelle and Teasley, 1995]. While maintaining shared understanding is a difficult problem in collaborative activities this problem can be exasperated by the use of such disparate software. As there is no integration between each of these programs, there is no support for users as they switch from one program to another. The result is that much of the user's time is spent dealing with essentially infrastructure or software issues, i.e. transferring files at each stage, rather than concentrating on creating a digital narrative. Furthermore, Inkpen argues "it is important that we begin to investigate alternative technologies to seamlessly support children's collaborative knowledge constructions" [Inkpen, 1999]. While Rochelle et al. identified division of labour as a key strategy for collaboration, for instance, "students may choose to divide up multiple representations among multiple devices, to provide a larger overall screen space" [Roschelle and Pea, 2002]. Furthermore, as outlined in

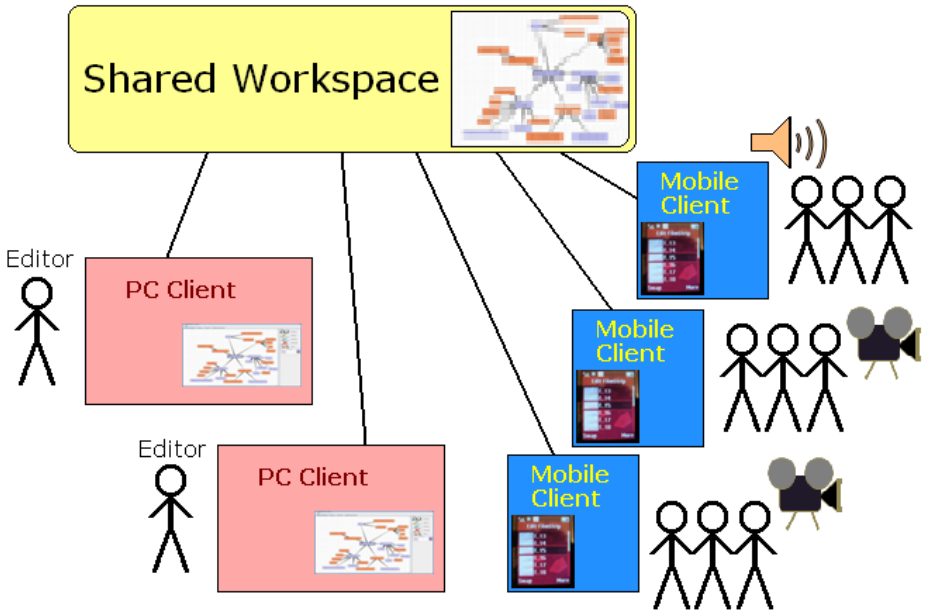

Figure 5.4: Shared Workspace

section 5.1.2, the distributed nature of the activity also contributes to the difficulties of maintaining a shared understanding, for instance, between the filming group out in the field and the editing group working back on the PC.

A recognised solution to these problems of maintaining a shared understanding and supporting division of labour involves using a shared workspace. A shared workspace is a distributed but connected environment in which all the users can interact with the workspace, the artifacts in the workspace and each other as if inside a single interface (see Figure 5.4 on page 106 for an illustration of this concept). This is a mature concept in CSCW with numerous studies in this area over the last fifteen years, e.g. [Dourish and Bellotti, 1992, Gutwin and Greenberg, 1998a, Ishii, 1990, Scott et al., 2003].

The DNT contains several interfaces to interact with three shared workspaces: a shared mind-map, a shared storyboard, and a shared timeline. The first interface screen contains a shared mind-map for idea-generation, the second interface screen contains a shared storyboard to refine the story to be told; the third interface is the shared timeline screen to create the digital narrative; the fourth interface is the shared chat screen to support communication within the application. A fifth interface screen is included on the PC version of the DNT which provides a built-in media player interface to display the completed movies. Alternatively the mobile version of the DNT provides

a media capture interface to enable mobile filming and sound recording groups to collect the appropriate media with their mobile devices. Section 5.3.2 includes labelled screenshots for all these interfaces as they appear on both the PC and a mobile phone.

In addition the shared workspaces include several awareness mechanisms to improve the efficacy of the shared workspace. Awareness mechanisms provide users with the, "who, what, where, and how" information about other users' interaction with the workspaces and help users arrive at a shared understanding. The DNT PC client provides a rich graphical interface and the MUSE mobile client provides a subset of this functionality. Specific awareness mechanisms in the DNT interface include: embodiment of users in the interface with colour; chat communication; consequential and feed-through awareness information through updates to the GUI; intentional communication through a text chat system, and awareness of the past through a view of the interaction history visible in the workspace.

There are several services on the MUSE application server to manage the user interactions in these shared workspaces. This section describes each of the services on the MUSE application server used by the DNT including: a tree-service; a chat service; a logging service; a video generation service. This section then describes the PC and mobile versions of the DNT application. Awareness mechanisms in the interface are highlighted where relevant. Finally the aspects of the DNT interfaces and services are discussed in terms of Mittleman et al.'s classification scheme [Mittleman et al., 2008] and in terms of supporting the mobileDNA.

5.3.1 DNT Services

The DNT application relies on the core infrastructure services described in Chapter 4 i.e. `Log inService`, `RegistryService`, and the `ResponderService`. In addition a second set of infrastructure services and user services are needed to support the DNT client application including: `Tree Service`, `VideoGenerationService`, `VideoManagementService`, `ChatService`, `HistoryService`, `LORepositoryService`, and `DatabaseLogService`.

The `TreeService` is central to the DNT application and manages data-structures used to seamlessly represent the mind-map, storyboard and timeline and ultimately the complete digital narrative. This service orders all actions from the clients before

making changes to the documents, keeping the data-structures consistent. The `VideoGenerationService` is responsible for creating the videos from the XML representing a digital narrative and converting it into a movie file format. This service queries the `TreeService` for this XML data, which contains the current state of the data in the timeline. This service can be executed as a daemon service so that it can automatically create new videos periodically. This service currently uses FFmpeg[1] as the video encoder, however all access to FFmpeg is hidden behind an interface to facilitate changing this in future if necessary. As this is a processor intensive process it should not be scheduled to run too often, the default setting is every two minutes.

The `VideoManagementService` contains a list of previously generated videos. The users of the DNT client can send a request for a list of videos, and then query this service to send them a particular video from this list. The users can also send a request to this service to request that a new up-to-date video, this service then forces the `VideoGenerationService` to generate a new video and add it to the list.

The `ChatService` handles all text communication between users, both the mobile and PC versions of the DNT client subscribe to this service. In addition, the MUSE application server includes an infrastructure service named the `HistoryService` which any service can use to record all messages sent to it. This service can later be queried to retrieve messages by a time range and service name criteria. There is also a `DatabaseLogService` that records all interaction with the MUSE platform, this could be used as a source of information for the researcher using the MUSE platform. `LORepositoryService` is a prototype service tested with the DNT and SMART to share artifacts produced from these learning activities (mind-maps, storyboards, and movies) with later groups of learners.

5.3.2 DNT Interface

The `LoginScreen` is the first screen presented to the user, default implementations of this screen are provided with the implementation of the MUSE middleware for J2ME (Figure 5.5 on page 110) and J2SE (Figure 5.6 on page 110). The user input gathered from this screen is sent to the `LoginService` to log the user into the DNT application.

[1] FFmpeg is an open source set of libraries available under the GNU Lesser General Public License from www.ffmpeg.org)

An important piece of awareness information that users need is concerned with who did what action. User interfaces should display information about what actions each user performs at any given time. Embodiment is a way to show users in a workspace and is how "users are themselves directly represented within the display space" [Benford and Fahl, 1994]. A typical source of embodiment information is to colour code all actions by the user who committed the action. The `LoginScreen` together with the `LoginService` on the MUSE application service are the central components used to provide this embodiment information in the DNT interface (Figure 5.15 on page 117). In the PC client a list of users and their status is always visible on screen. Due to the smaller screen space available to the mobile client this user list interface must be accessed by selecting a menu item, however this option is available from all screens, see detail (D) in Figure 5.5 on page 110).

When the login is complete the user is presented with the `MindMapScreen` on the PC version of the DNT. On mobile devices the users are presented with the `StoryboardScreen` interface of the DNT client. This design choice reflects the different default usage of the applications on the two platforms. On the PC the mobileDNA emphasises using the mind-map for idea-generation, while on the mobile client is focused on filming and recording sound for the digital narrative. Typically the mind-map on the mobile device will provide a relatively static interface that the mobile groups will consult for context, but rarely edit. However it is possible to edit the mind-map on the mobile device as indicated in Figure 5.8 on page 111 detail (B).

The DNT contains three shared workspaces for creating and editing collaborative mind-maps, storyboards, and timelines. The first shared workspace is the `MindMapScreen` which is a collaborative tool used in the brainstorming and idea generation phase of the mobileDNA. This screen (see Figure 5.7 on page 111) allows users to add, delete, edit and move nodes in a mind-map. The same view will be represented on all the user's screens so moving a node on one automatically updates the location of the node on the other screen. The nodes of the mind-map are displayed in the same way on the mind-map interface of the mobile DNT client (Figure 5.8 on page 111).

The second shared workspace is a `StoryboardScreen` which is a collaborative editor to support convergent thinking so that the learners can refine their ideas into a *script* for their movie. This screen (Figure 5.9 on page 112) allows the users to add, delete,

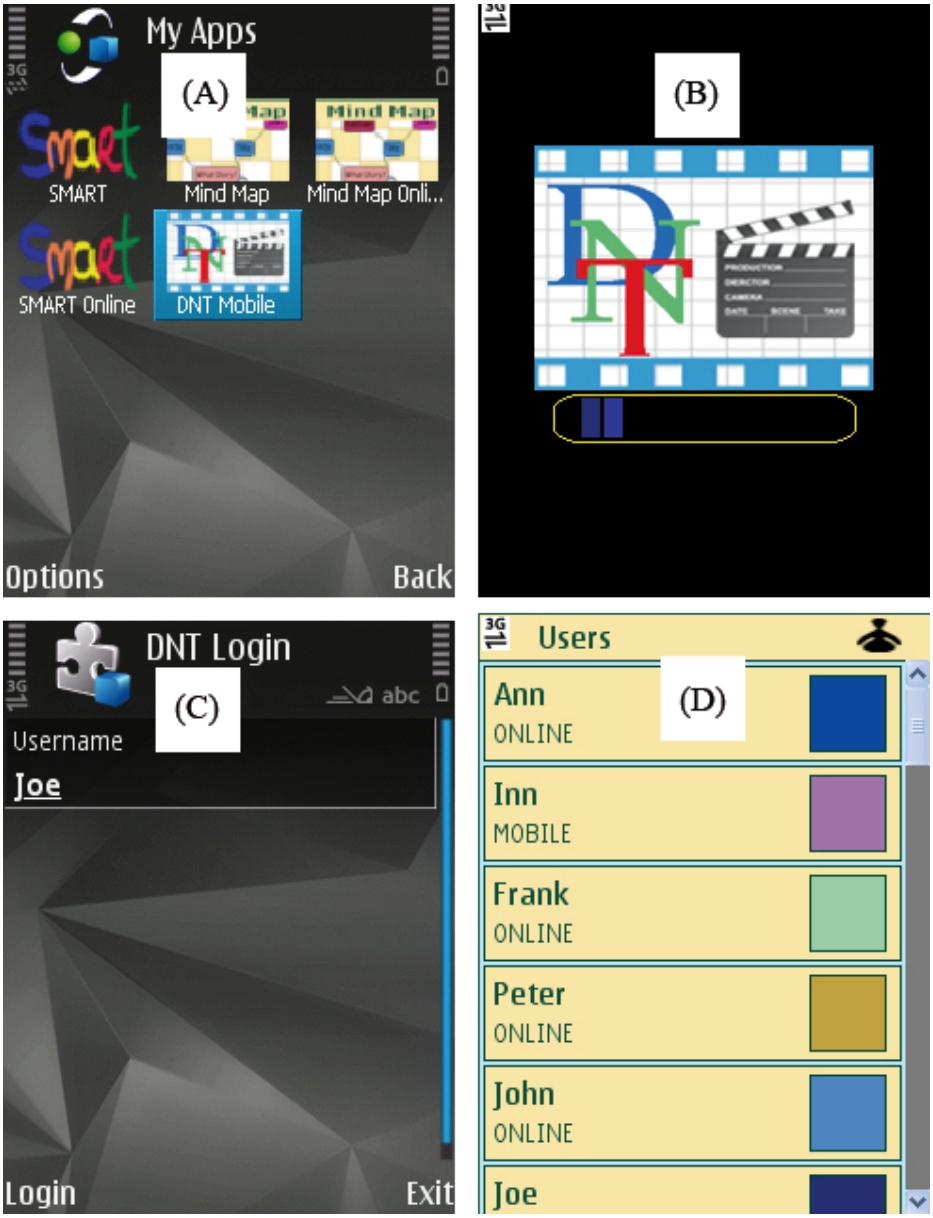

Figure 5.5: DNT Mobile Login Interface (A) Select Application, (B) Launch Screen, (C) Login Prompt, (D) User Status List Screen

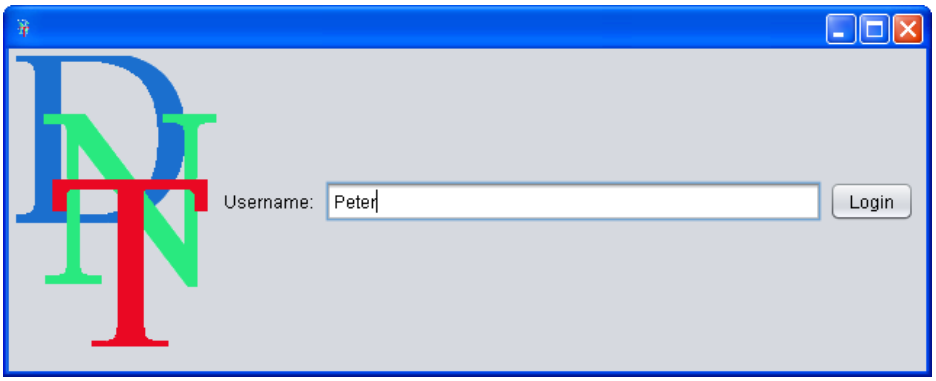

Figure 5.6: DNT Login Screen

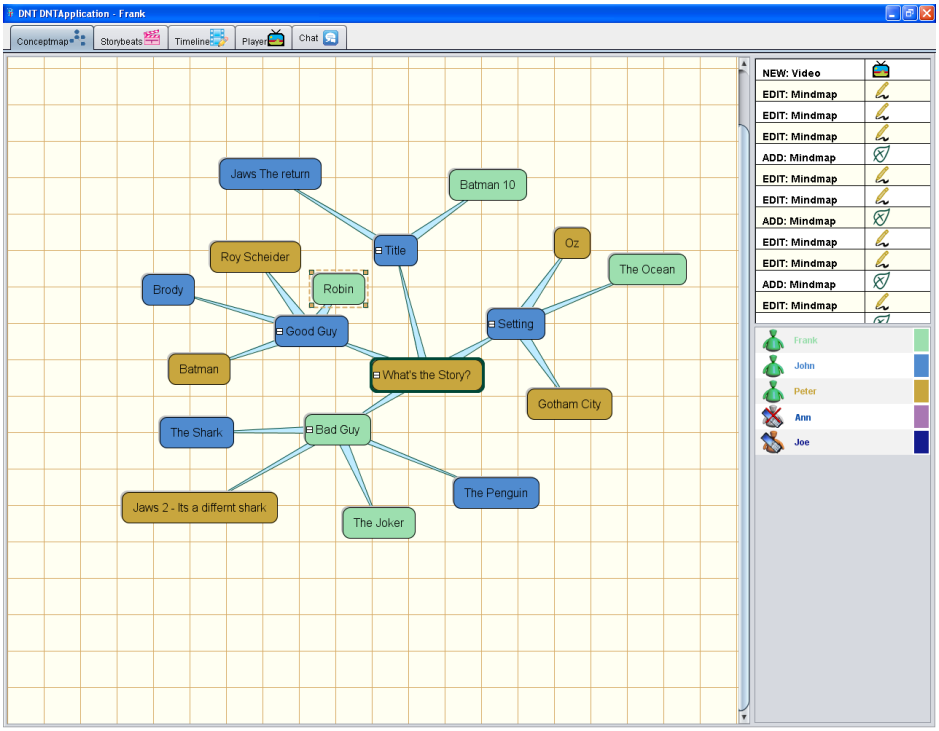

Figure 5.7: DNT Mindmap Screen

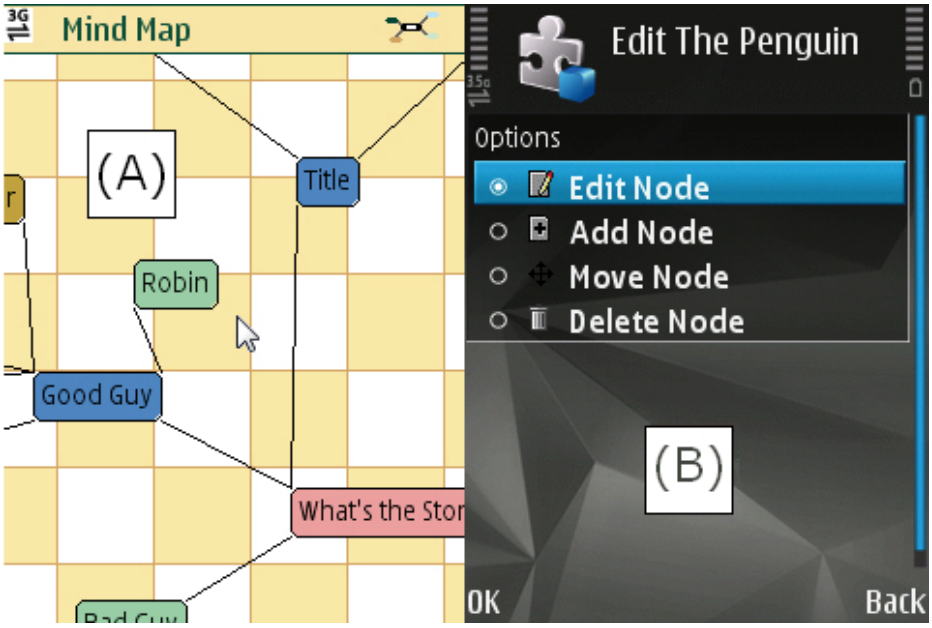

Figure 5.8: DNT Mobile Interface Detail (A) Mindmap Screen (B) Editing node options.

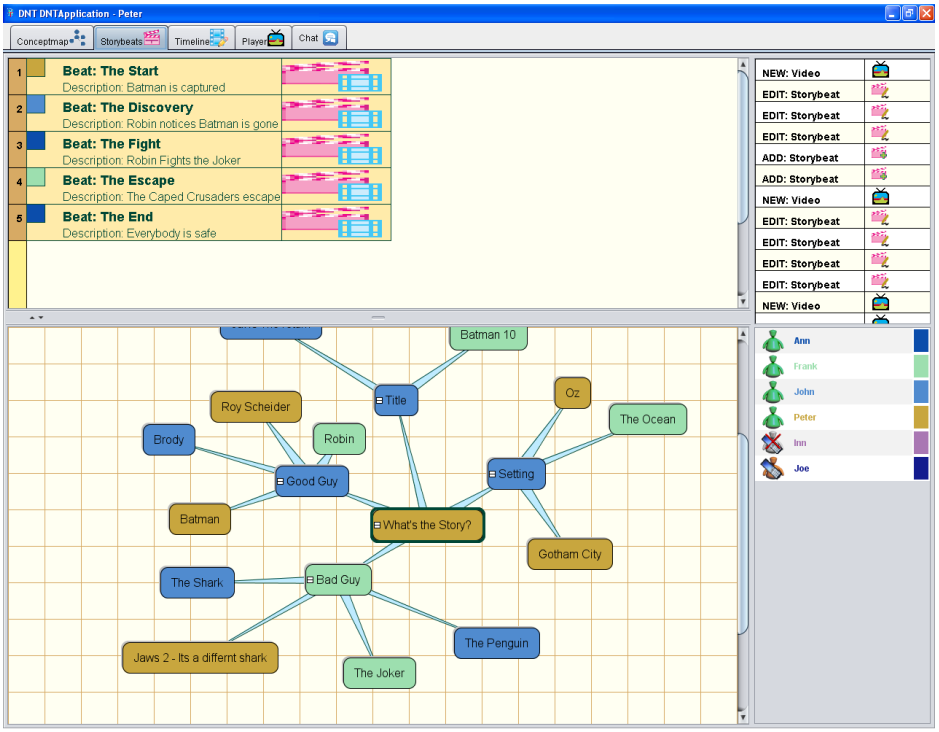

Figure 5.9: DNT Storyboard Screen

edit or reorder the scenes that comprise the story to be told. This is the first screen (Figure 5.10 on page 113) displayed on the DNT mobile client and provides the same functionality, however on the mobile client the users can also select a scene to provide context for the filming and sound recording element of the mobileDNA. In addition the PC client always displays a non-editable representation of the mind-map on this screen to provide context to the users while they refine their story.

The third shared workspace is the `TimelineScreen` which is collaborative editor for editing the digital narrative using the images and other media captured by mobile users. There is one `TimelineScreen` for each scene in the narrative; allowing a division of labour hierarchically where each user can edit a separate scene for example, so supporting cooperation. The users can change between timelines using a drop-down menu at any time. Some difficulties with the mobileDNA noted earlier were that editor had problems managing the media files and understanding which part of the narrative the groups recording sound or capturing images intended to place the media file. This interface solves this problem as all captured media is tagged with the scene in the narrative and placed in the correct timeline screen that the user captured the media intended. In addition, this `TimelineScreen` interface simplifies media management for

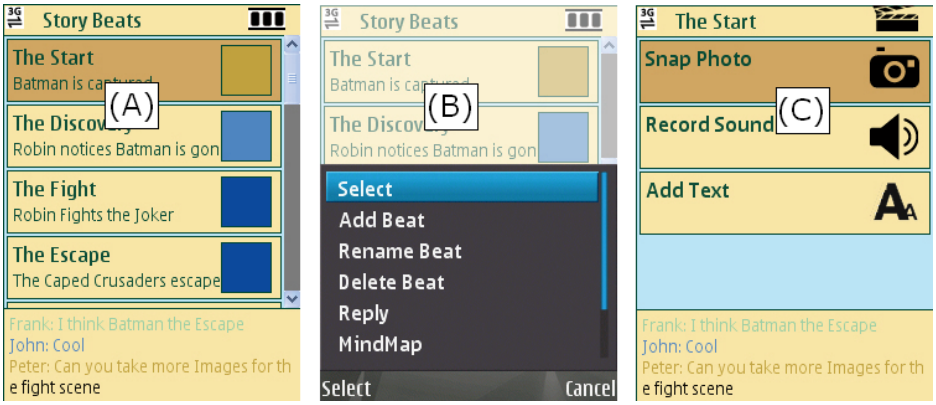

Figure 5.10: DNT Storyboard Interface (A) Storyboard (B) Storyboard editing options (C) Media capture screen

the editor as only the media relevant to a particular scene is displayed in the interface.

The `TimelineScreen` is only available on the PC client (Figure 5.11 on page 114). The image and sound files are displayed at the top of the screen, from here, the users can select elements from this pool of media files and either copy them to another scene or add them to the timeline used to represent the digital narrative. Using a drag and drop facility the users can reorder frames in the timeline, alter the length an image will play for, and remove an image or sound from the timeline. The users can also preview sound files by double-clicking on them.

The counterpart to the `TimelineScreen` on the mobile DNT client is the `MediaCaptureScreen` (Figure 5.12 on page 115). This screen provides facilities for users to capture images, sounds, and create text frames. The media files are then automatically sent to the MUSE application server which adds them to the pool of media available for the timeline of the selected scene.

The `Tree-Service` manages the data-structures on the server that represents the mind-map, storyboard, and timeline. The shared workspaces provide facilities to allow users collaboratively interact and edit the mind-map, storyboard and timeline, and provide support for users who are not co-located or are interacting using a mobile device. Users in the field capturing media, and the editors collating this media into the digital narrative, use the shared workspaces to inform their actions and provide feed-back to each other. The workspaces use a relaxed-WYSIWIS system for the display space to accommodate users on multiple different devices with differing capabilities and display sizes etc. While the "relaxed-What You See Is What I See" (WYSIWIS)

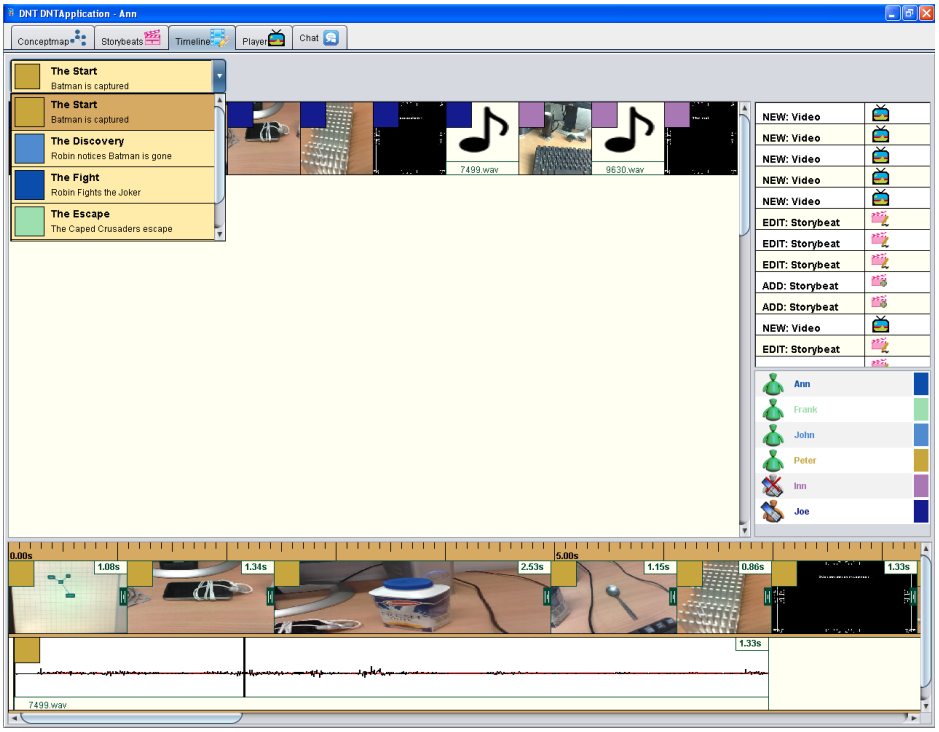

Figure 5.11: DNT Timeline Screen

approach sacrifices some awareness of the shared context, it allows better individual control [Gutwin et al., 1996] so supporting division of labour and introducing flexibility in the collaboration methods adopted by the users.

To provide additional support to the users several awareness mechanism are supported in the shared workspaces including: *feed-through* as a carrier of implicit awareness information; workspace support for communication that relies on a *frame of reference* or spacial information; *embodiment* in the workspace through colour and status information; *awareness of past* actions or interaction history; and *communications facilities*.

To facilitate the implicit communication of *feed-through* awareness information, the DNT system immediately sends all changes to shared workspaces, e.g. the mind-map or timeline, to all users. Therefore, while viewing the mind-map, the users will notice any changes as other users add, edit or remove elements from the workspace; consequently, they will be aware that other users are present and interacting with the workspace.

However, often a user's actions are not immediately obvious to other users, so additional support in the user interface is needed, e.g., they occurred outside a user's view. If a user wishes to clarify or highlight an action, they can inform the other users

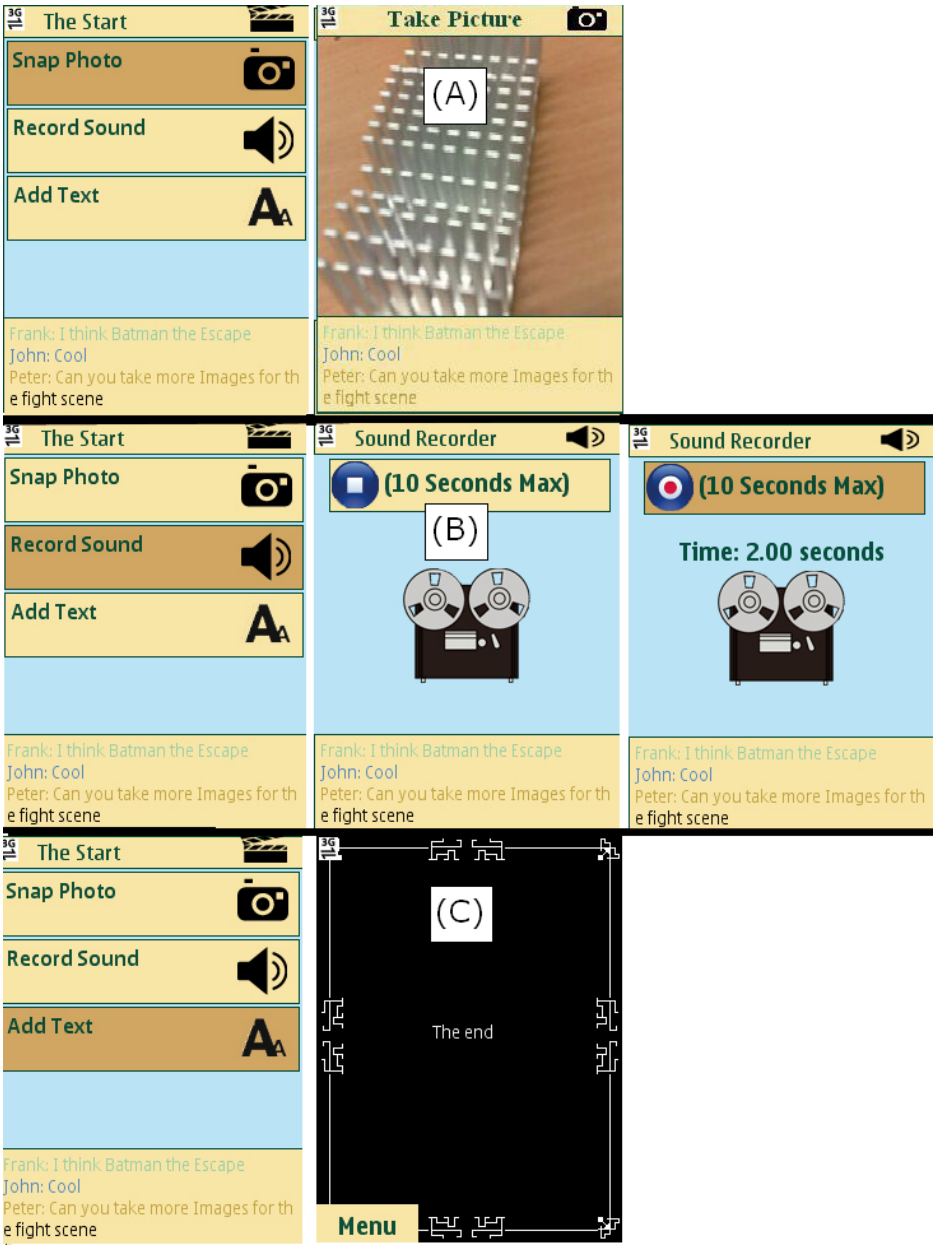

Figure 5.12: DNT Mobile Media Capture Interface (A) Image Capture Sceens, (B) Sound Capture Screens, (C) Text Frame Creation Screens

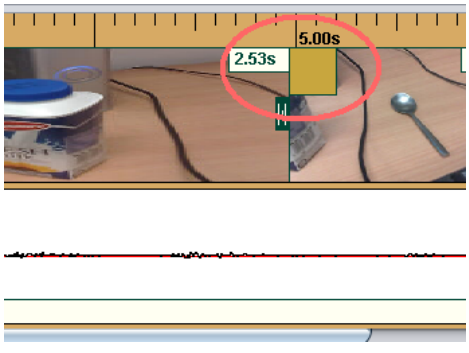

Figure 5.13: DNT PC Timeline Detail For Spacial Referencing

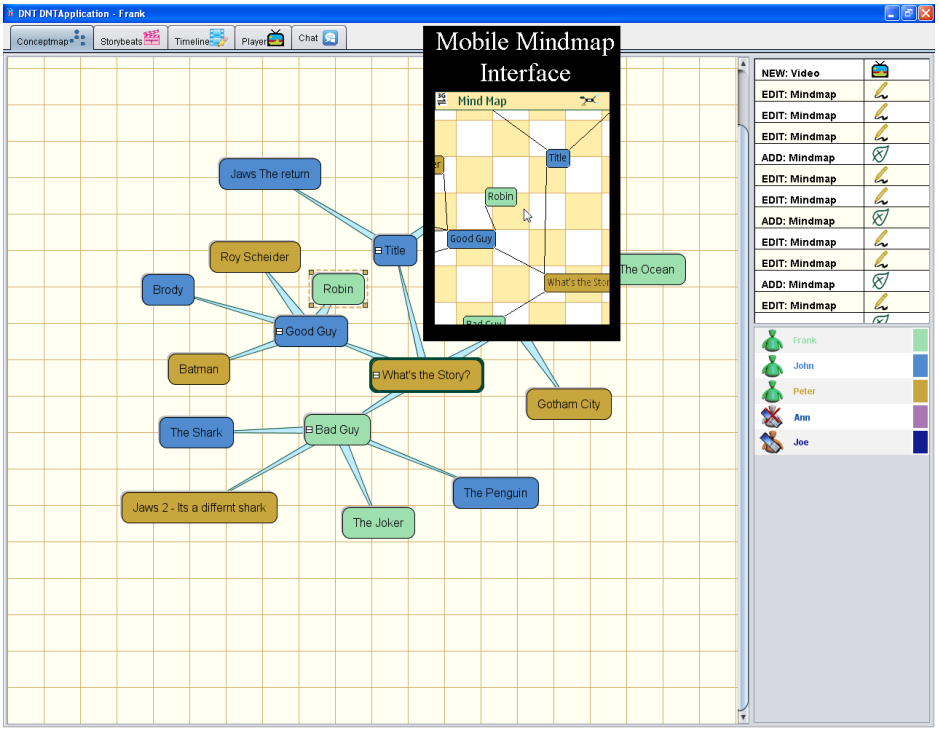

Figure 5.14: Representation Of DNT Mindmap On Mobile And PC clients

via one of the communications mechanisms but in order for this to be efficient users need a *frame of reference* in the workspace to indicate where the change occurred. The DNT provides support for such deictic language by including common frames of reference in the workspace. The mind-map workspace achieves this by representing the relationships in the mind-map the same way on each client (Figure 5.14 on page 116). The DNT applies the same concept, a uniform representation on each client, with the storyboard, and timeline interfaces. The timeline workspace has the added convenience that it includes a ruler representing the time in seconds to play each piece of media; this serves a secondary function as a coordinate system (Figure 5.13 on page 115).

The DNT uses two features to carry *embodiment* information, the use of colour, and represents a user's actions on another user's interface with icons [Beaudouin-Lafon and Karsenty, 1992]. There are suitable icons for all actions in the system and the DNT links a colour with each user and applies this consistently throughout the interface, marking all changes to the shared workspace with the colour of whoever made the change, so that the users can see who is doing what. For example if a user edits a node of the mind-map, it will be marked with an "Edit" icon and highlighted in that user's

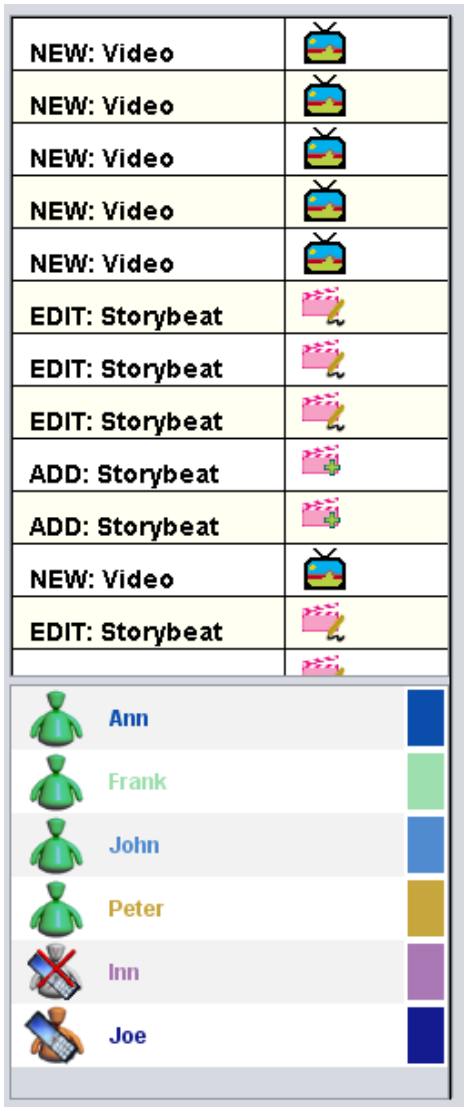

Figure 5.15: Detail Showing DNT Action History And User Status Components

colour.

To allow users to identify when an action occurred is important, Gutwin et al. call this "*awareness of the past*" [Gutwin and Greenberg, 2002], and list presence history, location history and action history as the three elements of awareness of the past. The DNT records and time stamps all actions the users perform including: all edits to the mind-map and timeline; capturing images and recording sound; communication messages via text chat; and interactions with interface elements. The PC version of the DNT client includes an interface component that is always visible, which displays a list of actions as they occur. This interface component is used to display the icons associated with actions in the DNT (Figure 5.15 on page 117).

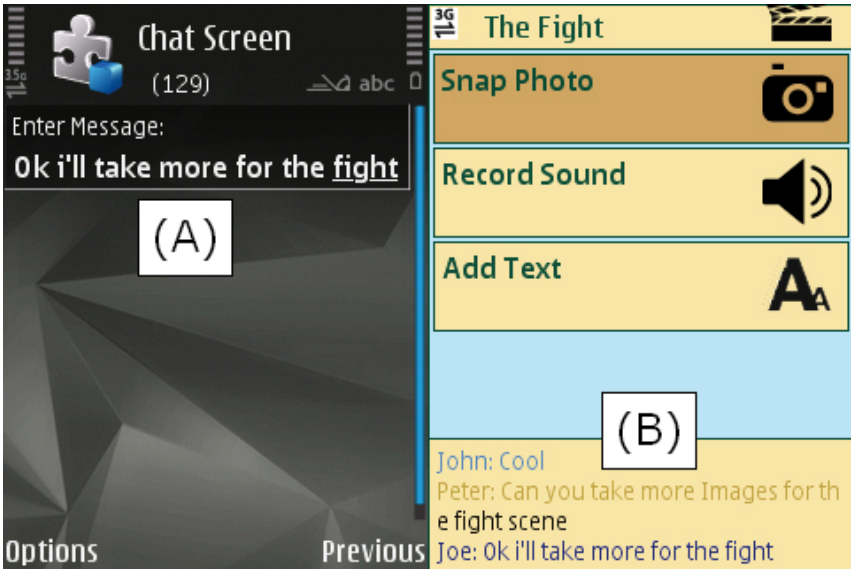

Figure 5.16: DNT Mobile Chat Interface (A) Enter Chat Message (B) Received Chat Interface

Studies of CSCW [Churchill et al., 2000, Ellis et al., 1991, Axelsson et al., 2003] note it is useful to support some form of informal *communication facilities* to clarify issues and help avoid conflict. [Churchill et al., 2000] recommend embedding instant messaging systems in the workspace. In addition, Scholl et al. note that "chat will not become obsolete as audio/video becomes more widely available" [Scholl et al., 2006]. As communications tools are the key ingredient of collaborative activities the DNT includes a contextualised communication system in the form of a built in text chat, which can be accessed on the `ChatScreen` (Figure 5.17 on page 119) in the PC version of the DNT. If there are new incoming chat messages they are indicated by playing a sound and flashing an icon, as depicted in Figure 5.18 on page 119. In the mobile version of the DNT chat messages are always visible on screen (see Figure 5.16 on page 118 for an example).

The DNT PC client contains one final screen called the `MediaPlayerScreen` (Figure 5.19 on page 120). This screen provides a list of media from the `VideoManagement-Service` which is visible on the left-hand side of the screen. If a users wishes to view a

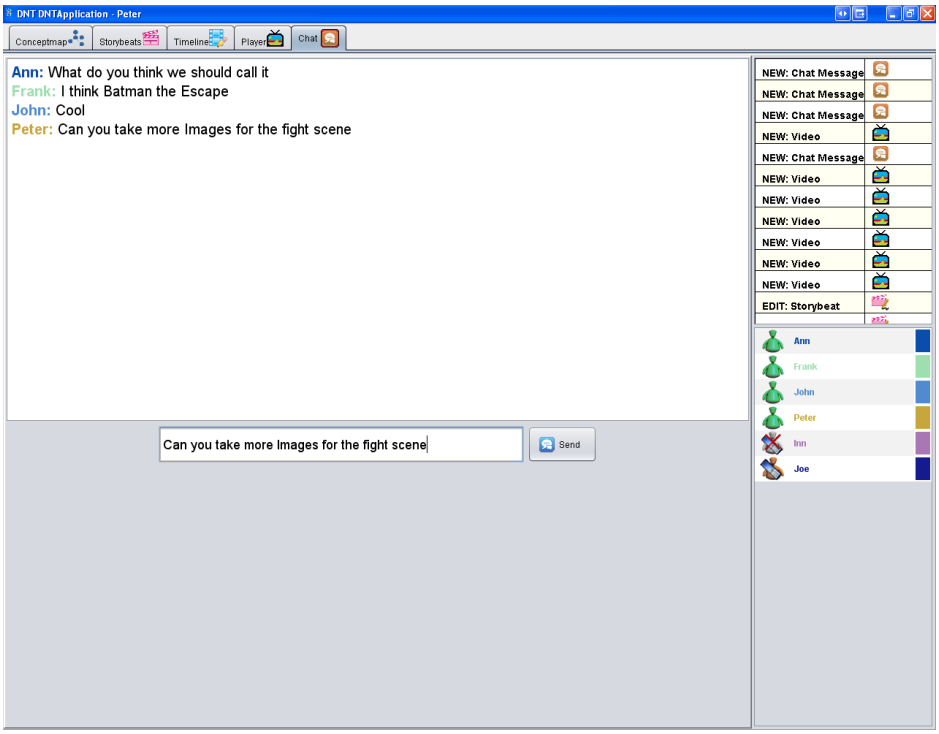

Figure 5.17: DNT Chat Screen

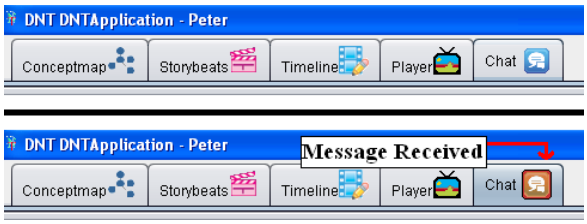

Figure 5.18: DNT Chat Alert - New Message Received

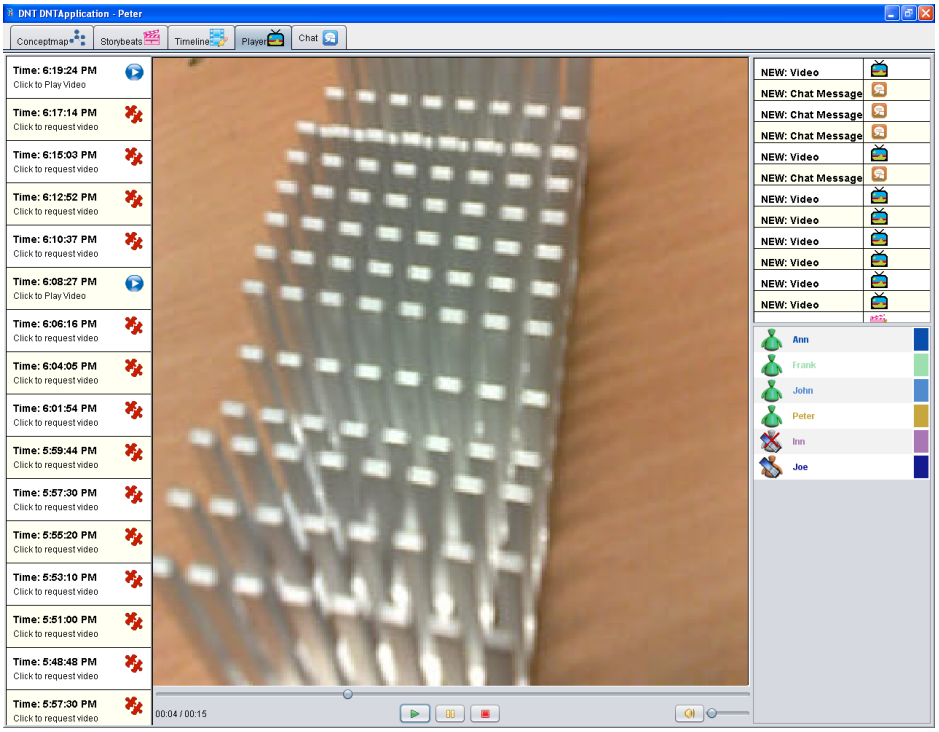

Figure 5.19: DNT Media Player Screen

movie they click on the item in the list, however if the file is not currently located on the user's computer then it will request it from the `VideoManagementService`. Once the movie is on the user's computer they can view it with the built in media player.

5.3.3 DNT Collaboration And mobileDNA Support

The previous subsection described the DNT user interfaces for the PC a mobile clients. However this section will now discuss the functionality of the DNT application in terms of the level of support it provides for the nine dimensions of Mittleman et al.'s classification scheme for groupware described earlier [Mittleman et al., 2008]. Furthermore, the DNT was developed to supported the mobileDNA process, as several difficulties experiences by users were outlined earlier, this section will discuss the aspects of the DNT designed to overcome these difficulties.

In review, the nine dimensions of Mittleman et al.'s classification scheme include: core functionality, content, relationships, session persistence, supported actions, action parameters, access control, alert mechanisms, awareness indicators.

In terms of *core functionality* the DNT is a collaborative co-authoring system. It provides a shared workspace that is used to produce three main shared artifacts: a sha-

red mind-map to support divergent thinking for idea-generation; a shared storyboard to support convergent thinking to refine the story to be told; and a shared timeline to produce the finished movie. The DNT includes the secondary functionality of a messaging system.

The *content* that the DNT is used to manipulate is a shared tree-based data structure. The DNT interacts with this data-structure using the `TreeService` on the MUSE application server using an RPC type mechanism. This service provides the core functions which are used to implement the shared mind-map, shared storyboard and shared timeline. The DNT also handles and manipulates multimedia content using this service to associate images and sound with frames in the shared timeline. In addition the DNT produces movie files from the shared timeline data using the `VideoGenerationService` on the MUSE application server. The *relationships* between elements in these shared artifacts is the standard *parent–child* relationship used by the tree data-structure. The DNT uses a secondary relationship between elements in the storyboard and the timeline, i.e. there is one timeline in the system for each element of the storyboard. The DNT includes *session persistence* by storing user information, and logging previous interactions with the application in a database. It also saves XML files on the MUSE application server that represent the state of the tree data-structures used by the mind-map, storyboard, and timeline. This allows users to access the system at anytime and it will still be up-to-date.

The *supported actions* are service dependent. The `TreeService` supports the following operations on nodes: add, move, edit, delete, copy and swap. An initialisation action is also available to request a full up-to-date copy of the data. The `ChatService` is a simpler messaging system and only supports adding messages to the group chat. The `VideoManagementService` supports actions to request the latest video file, to force the system to render a new movie or to add a new video to the list of available videos. All actions and messages using the MUSE platform contain the identity of the user who initiated the action and vector timestamp of when it occurred to supported the synchronicity and identifiability *action parameters*. The DNT does not use any *access control* functions currently.

The MUSE middleware notifies the DNT when messages from services it is interested in are received. The chat interface uses this information to provide an *alert mechanism*

(it plays a sound and lights up an icon) to inform the user of new chat messages. The `ActionHistory` component used on the PC version of the DNT uses this alert dimension of the MUSE platform to provide an aggregate list of recent updates in the shared workspace on the user interface (see Figure 5.15 on page 117).

The DNT *awareness indicators* try to convey the following information: Who is in the system? What are they users doing? And where are the users in the workspace?

Who is in the system? The PC version of the DNT includes a list of the users in the system, with their status (online, offline, mobile) and the colour used to carry embodiment information in the system, on screen at all times. This information is available in a separate screen on the mobile DNT client but can be accessed from a menu item available on every screen.

What are the users doing? This information is visible in the DNT PC client from the `ActionHistory` component which is always visible on screen and contains a list of recent actions. Another source of this information is feed-through information carried by all actions being replicated on the other users screens, for instance, changes to the mind-map are propagated to all DNT client interfaces. Finally this awareness information can be passed on through intentional communication via the `ChatService` and the chat GUI component that is available on the PC and mobile versions of the DNT clients.

Where are the users in the workspace? The users need reference points in a shared workspace for communication and navigation to help users to answer this question. The layout of artifacts in the shared workspace are represented in the same way on each client. Taking the mind-map as an example, it is depicted in the same way on all the DNT clients (mobile and PC) so that users can meaningfully discuss elements of the interface (this concept was illustrated in Figure 5.14 on page 116). Another example taken from the timeline interface, is that the time is listed at the top of the timeline interface and this can be used as a reference point in intentional communication (this concept is illustrated in Figure 5.13 on page 115).

Moving on now from this classification scheme, this section will now address the difficulties raised by the earlier studies with the mobileDNA. In summary the difficulties experienced by the users of the mobileDNA include: maintaining a shared understanding between filming and editing groups, managing media files, and communication

between groups. To address these issues the system provides shared workspaces that all the groups can simultaneously view or edit, automatic file management facilities and an integrated environment. To facilitate a shared understanding between filming and editing groups the users can view the mind map at all times but unlike the printed version, the DNT sends any changes made by the editing group to all users. In addition, the filming groups must also make an explicit decision regarding which scene of the narrative a piece of media is relevant to, in this way reinforcing the connection between the storyboard and the image capture process. Filming groups often take too few images and must wait until the editors inform them of this fact before they can act on this information. If editors were able to contact the filming groups in the field quicker, it would be easy for them to remedy this lack and capture more images.

To improve file-management the DNT automatically annotates the captured media with the scene name that identifies the media with the correct scene. This media is then automatically included in the storyboard at the relevant scene without needing the editors to manage the files. Furthermore, during the mobileDNA, the filming groups typically capture multiple images before sending them via MMS in a batch process to the editors thereby reducing the amount of parallel editing possible, as the editors must wait for the media. The DNT automatically transfers the captured media, so freeing the filming group to concentrate on image capture rather than routine tasks like transferring files to the editors. In this way, the editors get greater feedback regarding the progress of the filming groups as they receive the media directly. This is a feed-though [Gutwin and Greenberg, 2002] awareness mechanism and results from the users' interactions with the shared workspace which are observable by all the participants. The users experience this feedback without requiring active communicating between the participants.

In addition, the editors get greater feedback regarding the progress of the filming groups as they receive the media directly. The system uses the same shared data-structure to represent all phases of the process from the mind-map, the storyboard to the timeline that represents the digital narrative. The main advantages of this seamless approach are that it is possible to have similar interfaces across platforms, include more immediate feedback between groups and to include awareness mechanisms throughout the system.

Finally, the DNT includes contextualised chat system to facilitate communication between the groups. In [Kerr and Murthy, 2004] they describe the role of communications facilities in conflict-avoidance and negotiation. Another study [Handel and Herbsleb, 2002] stressed the role of communications in establishing shared understandings. This system allows users to communicate directly in the shared workspace as it is embedded in the workspace with the shared artifacts.

5.4 Conclusion

Animation and digital video activities can facilitate powerful learning experiences, however they are not without their problems, particularly in relation to cost and access to the appropriate technology. Furthermore, to date little use has been made of ICT to support the collaborative, as distinct from the video production aspect of the process, so would benefit from applications in the MCSCL space. The mobileDNA [Arnedillo-Sanchez and Tangney, 2006, Arnedillo-Sanchez, 2008, 2009] is a existing pedagogical methodology specifically designed to scaffold collaborative creativity among distributed learners engaged in the creation of digital narratives with mobile technologies. Previous studies using off-the-shelf software to implement the mobileDNA found that the users experienced difficulties including: maintaining a shared understanding between filming and editing groups, managing media files, and communication between groups [Byrne et al., 2008, Arnedillo-Sanchez, 2008, 2009].

Given this time-consuming nature of digital video production and animation this chapter described two applications to support these applications for learning. The first application described is the Stop-Motion Animation and Reviewing Tool (SMART), which is designed to support a constructionist, contextualised and collaborative learning approach to making animations on mobile phones using the stop motion animation technique. A noted advantage of which is that as the application is implemented on a mobile phone it has the benefit of being relatively inexpensive, can exploit the ready-at-hand nature of the device, and it is a familiar technology.

The second application described was the digital narrative tool (DNT), which is a more fully featured groupware application for digital video production. The DNT works on mobile devices and PC to support users engaged in the mobileDNA. Both

applications make use of the MUSE platform and services that are provided on the MUSE application server, including services to manage shared workspaces, generate video files and provide communication mechanisms. One aspect of the DNT is that it is designed to address some of the difficulties experienced by the users engaged in the mobileDNA. Therefore the DNT includes shared workspaces for editing the mind-map, storyboard, and timeline, to help maintain a shared understanding between users. It automatically manages file transfers and other infrastructures aspects of the process to simplify managing media files, and includes awareness mechanisms and chat communication tools to improve communication between users.

The next chapter moves onto the evaluation of the MUSE platform and an evaluation of SMART and the DNT. The evaluation methodology is described and results of the evaluation of the MUSE platform in terms of software performance metrics are presented. The chapter then describes the evaluation of SMART and the DNT in terms of usability and learning outcomes.

Chapter 6

MUSE Evaluation

This thesis investigates two main research questions. The first question asked in this thesis is: what are the functional requirements for a platform to support MCSCL applications on both mobile and PCs? This first question was addressed in Chapter 2 and Chapter 3. Initially in Chapter 2 by a review of the literature relevant to CSCW, CSCL, MCSCL, mobile learning, and the pedagogical theories relevant to mobile learning and CSCL. And then by review of the related work on collaborative middlewares in terms of these functional requirements in Chapter 3.

The second question is: Do the functional requirements for a MCSCL platform captured by MUSE, SMART and the DNT support constructionist, contextualised and collaborative activities for learning? The first part to answering this question was described in Chapter 4 which examined the level of support for the following functional requirements identified in Chapter 3, including: support for the four dimensions of the space-time classification of groupware system; the nine dimensions of groupware technology from Mittleman et al. [2008]; constructionist, contextualised and collaborative activities; data collection and location awareness. Therefore, Chapter 4 described MUSE in detail and provided a comparison of MUSE to the systems described in Chapter 3 in terms of their level of support for these requirements.

The next stage to answering this question involves addressing several sub-questions: from a technical perspective can we design and build an MCSCL middleware to work in a setting containing heterogeneous devices and networks? What is the performance of the middleware on mobile devices and PCs? How many concurrent users can the system support? What are the usability and learning support provided by the applications

developed?

To answer these sub-questions the evaluation of the software that is the focus of this thesis will involve two stages. From a technical perspective, MUSE is evaluated in terms of a set of standard performance metrics. While there are many challenges developing software for mobile devices (summarised in Table 3.1 on page 34), the subset of technical challenges relevant to the MUSE middleware are: low processor and memory; battery life; heterogeneous os services; varying networking technologies; and poor network bandwidth, latency, and reliability. Therefore the technical evaluation examines how MUSE dealt with or overcame these problems. During this evaluation the following metrics are examined: round-trip time, group latency, protocol overhead, throughput, and implementation cost. This chapter starts by focusing on answering these sub-questions. The second part is to validate the functioning of the middleware with two applications for digital narrative production that were built using the MUSE platform. These user applications (SMART and the DNT) are then evaluated in the context of the framework described by [Sharples et al., 2007], and further expanded on in [Vavoula, 2007, Vavoula and Sharples, 2008], which advocates evaluating mobile learning projects according to three levels of granularity, the micro level (usability), the meso level (educational), and the macro level (organisational). This evaluation will focus on the micro and meso levels from this framework, with the macro level being outside the scope of the research. At the micro level, the questions asked examine the usability of the applications. At the meso level the question asked is: do the applications support constructionist, contextualised, and collaborative activities for learning?

The first section focuses on the performance evaluation of MUSE and describes the performance metrics used in the evaluation of MUSE. The results of these performance metrics are then presented. Finally an interpretation of these results is presented in terms of addressing the second research question covered by this thesis. The second section will then focus on the usability and learning evaluation of the applications developed on the MUSE middleware. This section starts by describing the usability metrics and learning rubrics used in this study. Then the results of two workshops to evaluate SMART and one pilot study to evaluate the DNT are presented to help answer the second research question covered in this thesis.

6.1 Performance Evaluation

Selecting or comparing metrics and results between such diverse systems is difficult. The approach adopted in this thesis is to chose a set of metrics that highlight the technical challenges relevant to the implementation of a middleware for MCSCL and digital narrative application. The next section lists these technical issues and any performance metrics that stress test the MUSE middleware's approaches to overcoming them. Where applicable any results from the above related middleware evaluations are compared.

6.1.1 Performance Metrics

In short, the technical challenges relevant to the design of the MUSE middleware, as opposed to the mobile applications, includes: varying networking technologies; poor network bandwidth, latency, and reliability; battery life; low processor and memory; and heterogeneous OS services. These can be grouped in to three sets covering networking problems, computational resources and heterogeneous os services.

As mentioned earlier the above systems are evaluated through a range of different and in most cases not easily comparable metrics. For instance, most of the systems discussed used a peer-to-peer topology. Nevertheless this thesis attempts to generalise from these studies to derive a set of useful metrics. The metrics that are used to evaluate the networking problems encountered by the MUSE platform include: round-trip time; group latency; throughput. The metrics relevant to computation resources are, protocol overhead and implementation cost. There is no metric to measure heterogeneous OS services as this is impossible to measure in any meaningful way.

The first metric examined was *round-trip time,* which is time for a message to be sent from the server, retrieved and processed by the client and sent back. This time is the full round-trip time from a service through the middleware, including marshalling and demarshalling to the an application and back to the service through the middleware again. In this respect it is similar to the invocation latency metric used by ESPERANTO [Cinque et al., 2005, Cotroneo et al., 2007]. An advantage of recording round-trip time over end-to-end delay is that it avoids clock synchronisation problems as the time stamps are consistently recorded on the one device. A thousand messages are sent

to test round-trip time and the average recorded. This test is repeated until there are eight simultaneous clients connected to the system. This is an important test as it benchmarks the system from end-to-end completely, the resulting figure consists of measurement of the networking performance, the MUSE networking-layer and MUSE service-layer on both the PC and mobile devices. This figure is measured by a service on the application server and a client on the target device that sends messages back to this service. In this respect it is a useful measure of the performance that application developers can expect because the time is measured from the application layer on the server to the application layer on the client and back.

The second metric is *group latency*, which in this thesis is defined as the round-trip time for a message to reach all clients in the system. To gather this information a message is sent to all of the clients registered with the system. The messages sent to all the clients have the same group identification number, the service gathers the replies by this group ID together. The resulting value is equal to the longest delay recorded. As before this value is then averaged over a thousand trials and repeated until there are eight simultaneous clients connected to the system. As described earlier versions of group latency were gathered for GMAC [Gotthelf et al., 2007], and MCSCL-CS [Zurita and Nussbaum, 2006] but are not directly comparable as those studies were focused on peer-to-peer communication and encounter a lot of protocol overhead on this metric.

The third metric is *protocol overhead*. The definition from [Gotthelf et al., 2007] is the time taken that is not directly related to exchanging information. While in the other studies [Gotthelf et al., 2007, Cotroneo et al., 2007] this typically related to the cost of maintain overlay networks in peer-to-peer systems. This thesis assumes it to relate to the time required to prepare and queue a message to be sent over the communications network or to receive an incoming message from the network and pass it on to relevant service or application. In other words this relates to the marshalling and demarshalling times and the time to route messages using the service orientated architecture. This figure also acts as a proxy for the processor and memory usage of the middleware itself, as a large figure here would indicate that the middleware was placing a heavy load on the device.

The fourth metric used the measurement of *throughput* which in this thesis is recorded in KB/sec to transfer a file. In this case a file of 2 MB is used as it is bigger

than any of the actually files expected to be exchanged using this middleware. This trial is repeated fifty times on each protocol. This is gathered to measure the upper bound of network performance that the users can expect. As MUSE is designed with a particular focus on digital media production it should be able to deal with a bursty communication profile as a result of larger media files being periodically transferred, this metric will examine if the system can handle such communication.

The fifth metric is *implementation cost*. This is necessarily a rather imprecise measurement e.g. previous studies listed the number of classes [Guicking et al., 2008], lines of code [Roth, 2003]. However it is useful as a guide to the cost of developing an application with the middleware and the potential memory footprint that applications can expect to occupy on the mobile devices. This metric is relevant in respect to the low processor and memory size available on mobile devices.

Another aspect to the performance evaluation is to stress test the middleware to discover the maximum number of synchronous client applications that can use the MUSE platform and expect reasonable responsiveness at the user interface level. This figure will differ for mobile and fixed PC clients. In addition, the mobile devices will principally be producers of multimedia while the PC clients will mainly be consumers of multimedia files. Furthermore, the PC clients application will normally expect synchronous operation, while on the mobile device there are not such strict time bounds.

For handheld applications a GUI response time of about one second is specified by Wilson et al. [2004], but this only impacts direct effects on the GUI which usually involve small messages sizes. In terms of the DNT mobile application this would be relevant to chat messages, and updates to the storyboard or mind-map. Larger file transfers are in one direction, from the mobile to the PC, so do not have the same timeliness requirements. There is no specific time recommended for this so this thesis understands anything up to the range of about 20 - 30 seconds to transfer captured multimedia as acceptable. As group latency is the maximum time to send a round-trip message to all the participants so a time that is close to the round-trip time implies that all clients can expect the same performance and responsiveness.

In this thesis throughput is used to find the maximum number of users the MUSE platform can handle before performance and responsiveness becomes unreasonable. The middleware should minimise protocol overhead so that responsiveness is related

Equipment	Operating System	CPU	Memory	Java Version
PC Server	Microsoft Windows XP Professional (SP 3)	Intel Pentium 4 - 3.4 GHz	3 GB	J2SE 6.0
PC Client 1	Microsoft Windows XP Professional (SP 3)	Intel Pentium 4 - 2.8 GHz	512 MB	J2SE 6.0
PC Client 2	Microsoft Windows XP Professional (SP 3)	Intel Pentium 4 - 3.4 GHz	3 GB	J2SE 6.0
PC Client 3	Microsoft Windows XP Professional (SP 3)	Intel Pentium 4 - 2.8 GHz	512 MB	J2SE 6.0
PC Client 4	Microsoft Windows XP Professional (SP 3)	Intel Core 2 - 2 * 2.4 GHz	2 GB	J2SE 6.0
PC Client 5	Microsoft Windows Vista Home Premium (SP 3)	Intel Core 2 Duo - 2 * 2.4 GHz	4 GB	J2SE 6.0
PC Client 6	Microsoft Windows Home Edition (SP 3)	Intel Pentium M 1.73 GHz	1 GB	J2SE 6.0
PC Client 7	Microsoft Windows XP Professional (SP 3)	Intel Core 2 - 2 * 2.4 GHz	2 GB	J2SE 6.0
PC Client 8	Microsoft Windows XP Professional (SP 3)	Intel Core 2 - 2 * 2.4 GHz	2 GB	J2SE 6.0
Nokia N95	Symbian S60 3rd Edition	ARM 300 MHz	128 MB	J2ME, MIDP (2.0), CLDC (1.1)
Nokia N73	Symbian S60 3rd Edition	ARM 220 MHz	128 MB	J2ME, MIDP (2.0), CLDC (1.1)
Nokia N73	Symbian S60 3rd Edition	ARM 220 MHz	128 MB	J2ME, MIDP (2.0), CLDC (1.1)
Nokia N73	Symbian S60 3rd Edition	ARM 220 MHz	128 MB	J2ME, MIDP (2.0), CLDC (1.1)

Table 6.1: Equipment Table

to the time to send messages over the underlying network and not being held up by the middleware itself. In addition a low figure here implies that the middleware is not making undue demands on the processor.

Two aspects of implementation cost are the file size of the middleware and the amount of support it provides to application developers to simplify development. This measure is imprecise and just a guide to expected memory usage on the devices. The next section describes the evaluation setting and the equipment used to gather data from these metrics.

6.1.2 Evaluation Setting

The equipment used for the performance evaluation is listed in Table 6.1 on page 131. The MUSE application-server runs on one PC (this computer also ran a supporting database). During the evaluation up to eight PC clients are connected to the server at the same time, the computers are connected with a 100MB Ethernet network. In addition up to four simultaneously connected mobile devices were tested. A small application was built on the MUSE platform to gather performance data, and each test was performed by sending a thousand messages for each evaluation scenario. Following the approach from [Cotroneo et al., 2007] this application was instrumented to gather timestamps to record the time to marshall or demarshall messages and to route messages with the service orientated architecture.

The evaluation of the MUSE platform used three messages that test the range of expected usage for applications built on the platform. These messages sizes were 368-bytes, 17KB and 100KB. The message size of 368-bytes represents the typically size

of simple text-based messages, for instance, chat or mind-map messages used by the DNT. This message size is comparable to the message sizes used in the middleware evaluation by Zurita and Nussbaum [Zurita and Nussbaum, 2006]. The message size of 17KB is equal to the file size of an image captured by SMART or the DNT. The message size of 100KB represents the average audio file size and is the same message size used to evaluate GMAC [Gotthelf et al., 2007]. In addition, a message size of 2 MB was used to measure throughput. To gather the data on round-trip delay, group latency, and throughput, a thousand messages were sent over each of the protocols, with up to eight clients connected simultaneously. However, it was impossible to gather this data for MMS as this would be a prohibitively expensive and is network operator dependent so varies from seconds to several minutes or more.

6.1.3 Performance Evaluation Results

This chapter focused on addressing one research questions: Do the functional requirements for a MCSCL platform captured by MUSE, SMART and the DNT support constructionist, contextualised, and collaborative activities for learning?

This chapter used a second approach to answering this question by focusing on several sub-questions including: from a technical perspective can we design and build an MCSCL middleware to work in a setting containing heterogeneous devices and networks? What is the performance of the middleware on mobile devices and PCs? How long is the message round-trip time between devices and the server? How many concurrent users can the system support? To answer to these questions this section measured the performance of the MUSE platform described in this thesis and examined the data from a set of performance metrics to ascertain if it could provide an adequate platform to support MCSCL applications for digital narrative production.

The next section will provide further support to answer this question by examining the usability of two MCSCL applications built on the MUSE platform, SMART and the DNT. The next section will also look at whether these applications can support constructionist, contextualised, and collaborative learning activities.

To provide some context for this discussion this section starts by describing the ratio between the different sized messages used by MUSE (368-byte, 17KB, and 100KB) and then describes the timeliness constraints for each of these various message types.

Then the results from the round-trip and group latency metrics are described. Next the protocol overhead and throughput of the MUSE platform are outlined. This section finishes by describing the implementation cost of using the MUSE platform.

The 368-byte sized messages are the most widely used messages in the system as the main operations in the system are short text-based instructions to alter XML-based data-structures. Therefore, it is the performance of the system when handling these messages that will have the greatest impact on the user experience of the applications developed on MUSE. Furthermore the 17KB is the next most used message, and finally 100KB is the least used message size. Based on examining the logs during both user studies and during development and testing the typical measured ratio is that for every 100KB message sent there are eleven 17KB messages and fifty-two 368-byte messages.

Number of devices:	1	2	3	4	5	6	7	8
TCP-IP Round Trip Time (368 bytes)	0.086	0.155	0.490	0.830	1.264	1.839	2.484	3.218
TCP-IP Group Delay Time (368 bytes)	0.086	0.171	0.591	0.981	1.463	2.212	2.785	3.567
TCP-IP Round Trip Time (17.0 KB)	0.104	0.275	0.513	0.861	1.289	1.872	2.575	3.657
TCP-IP Group Delay Time (17.0 KB)	0.104	0.331	0.614	1.012	1.492	2.121	2.884	4.018
TCP-IP Round Trip Time (100.0 KB)	0.239	0.608	1.051	1.630	2.364	3.132	3.907	5.105
TCP-IP Group Delay Time (100.0 KB)	0.239	0.706	1.239	1.878	2.724	3.556	4.380	5.617
HTTP Round Trip Time (368 bytes)	1.685	2.705	5.055	7.542	N/A	N/A	N/A	N/A
HTTP Group Delay Time (368 bytes)	1.685	3.248	5.539	8.328	N/A	N/A	N/A	N/A
HTTP Round Trip Time (17.0 KB)	2.572	3.976	6.702	9.823	N/A	N/A	N/A	N/A
HTTP Group Delay Time (17.0 KB)	2.572	4.613	8.163	11.940	N/A	N/A	N/A	N/A
HTTP Round Trip Time (100.0 KB)	4.178	5.635	9.190	14.887	N/A	N/A	N/A	N/A
HTTP Group Delay Time (100.0 KB)	4.178	6.935	10.948	17.503	N/A	N/A	N/A	N/A

Table 6.2: MUSE Messaging Performance Table

While the next section (Chapter 6.2) describes the usability and learning studies of SMART and the DNT using MUSE, the number and type of media files captured during these studies is relevant here. During the studies using SMART, the average number of images captured was thirty-two, with an average file size of about 17KB. In the DNT study the users captured sixty-six images and twelve sound files, with files average files sizes of 17KB and 57KB respectively. For these reasons the MUSE implementation prioritises short messages relating to user interface or data structure changes visible to the user. Larger media files are sent through a scheduling background process transparent to the user. This is because the timeliness constraints for the messages containing larger media files are lower than messages that carry data that directly interacts with the GUI.

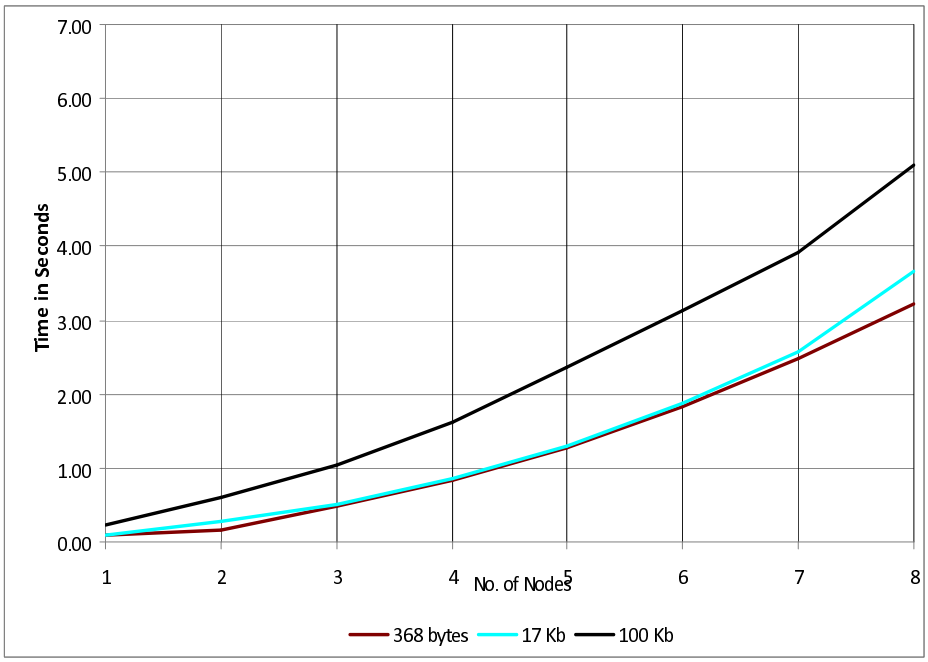

Figure 6.1: TCP/IP Round-trip Time

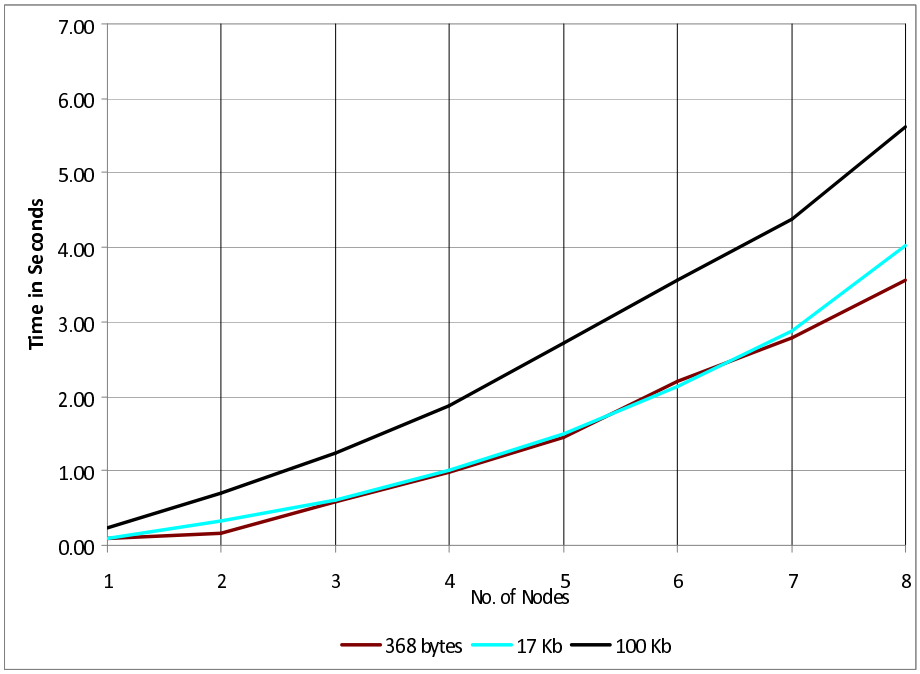

Figure 6.2: TCP/IP Group Latency

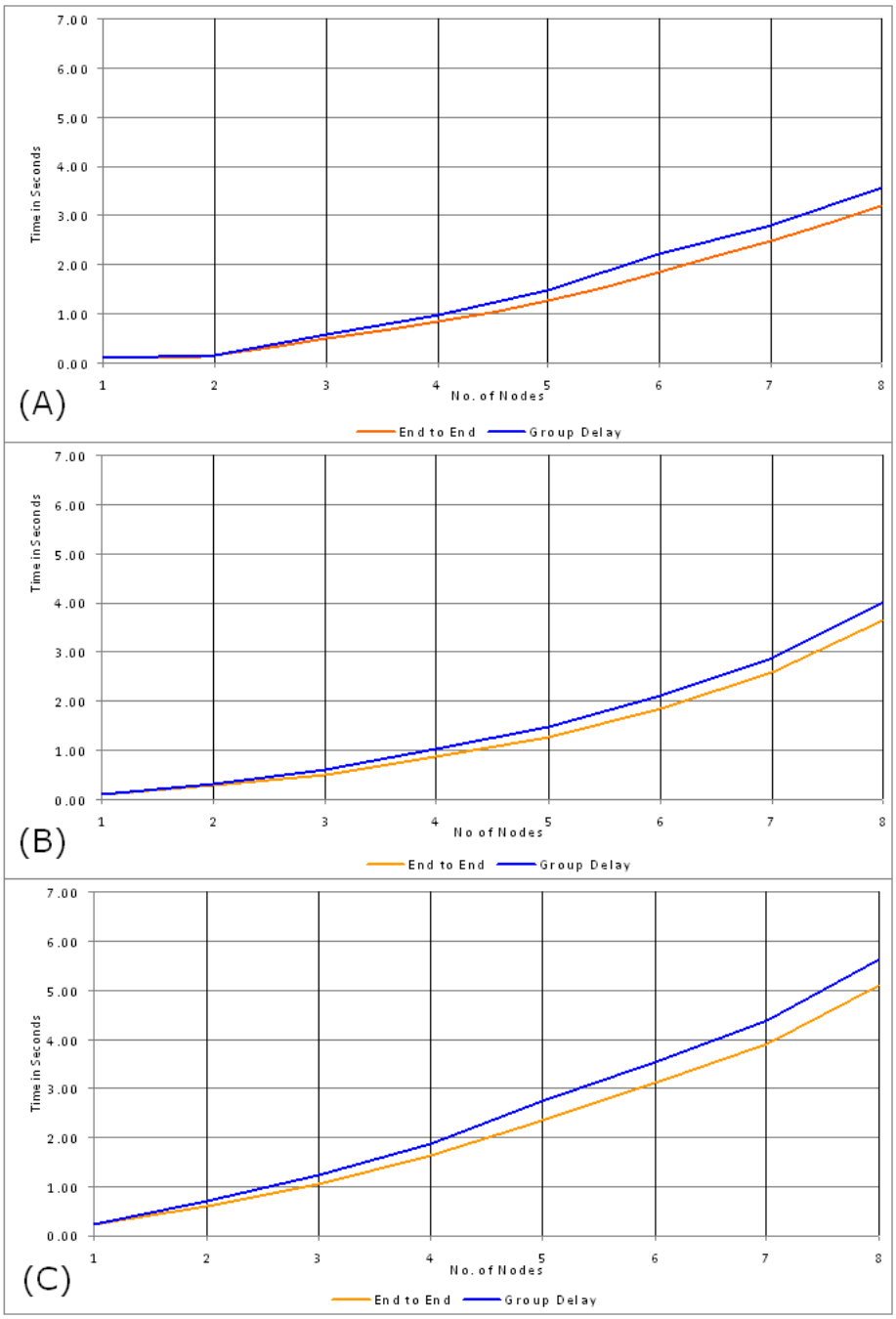

Figure 6.3: TCP/IP Round-trip Time Compared To Group Latency (A) 368 bytes message, (B) 17 KB message, (C) 100 KB message

The assumption employed in this thesis is that for actions that are noticeable by the end use and directly impact the GUI then a response time of less than 1 second is required Wilson et al. [2004]. For actions that are not immediately noticeable to the user then a time bound for 30 seconds is defined. An example of such an action would be a remote user capturing an image, the editor back in the office would have no idea of when this occurred and will not be waiting on it, so will not notice a 30 second delay. Longer delays could still be tolerated but would imply a general degradation of performance so a 30 second bound was selected as an acceptable delay for such an activity.

So given these constraints this section will now examine the round-trip and group latency performance of the MUSE platform (Table 6.2 on page 133). When dealing with 368-byte messages over TCP/IP the system can easily handle five simultaneous connected PC client applications, under heavy usage, with adequate user-interface responsiveness of around a second (Table 6.2 on page 133). With four or less clients the responsiveness improves further to below a second. Furthermore the operation of MUSE scales to support five simultaneously collaborating clients over TCP/IP with acceptable response times with 17KB and 100KB message payloads. With more than five clients the performance starts to deteriorate as round-trip time increases significantly with each additional client (Figure 6.1 on page 134).

In terms of group delay, MUSE scales well over TCP/IP as the maximum group delay times closely tracks the round-trip timeFigure 6.2 on page 134. This implies that all of the clients can expect the same performance on average (Figure 6.3 on page 135). The gap between round-trip time and group latency is a measure of the range of times in which a given message on average is likely to arrive, the larger this value the more variance expected in the performance of the system. However as the number of clients increases the gap between round-trip time and group delay increases by a greater amount each time. This trend is more pronounced over HTTP (Figure 6.6 on page 139), which implies that the system probably could not support more than about eight or ten mobile devices simultaneously and continue to provide reliable and predictable response times on all clients.

In the heaviest loaded performance test of HTTP, transferring a 100KB file to the mobile device and back with four simultaneous mobile clients, the maximum group

latency time was about 17 seconds to complete all four transfers. This time is far within the 30 second bound for transferring media specified earlier. Furthermore, in typical usage the image is only transferred asymmetrically in one direction from the mobile device to the server, halving this time. The most common case, transferring a 17KB image file, can be achieved under 10 seconds with a maximum group delay of under 12 seconds when four mobile client applications connected. These times are not noticeable between the distributed groups especially as the time between setting up new shots and capturing the new image are significantly longer than the time to transfer the images. The larger sound file size of 100KB represents a 10 second audio clip, so worst case round-trip time of 17 seconds is not noticeable in comparison to recording time.

This section will now switch focus from round-trip time and group delay to protocol overhead and throughput. This study initially looked at two sources of protocol overhead, first the time to interpret and distribute the messages using the service orientated architecture and second the time to marshall and demarshall messages when sending and receiving respectively. It was found that the times recorded in relation to the operation of the service orientated architecture approach were insignificant with times of less than a 10th of a millisecond or less in most cases. Therefore they can be ignored here.

The second component of protocol overhead (Table 6.3 on page 140) was found to be orders of magnitude more significant. Times recorded here on PC for TCP/IP range from 5ms to marshal a 368-byte message and pass it to the communications channel, to 450ms to marshal a 100KB message. The marshalling and demarshalling time on the mobile devices were longer with a range from 30ms for 368-bytes to 1290ms for 100KB. However when examined as a percentage of the round-trip time the marshalling and demarshalling time was more significant over TCP/IP (Figure 6.8 on page 140).

As Figure 6.8 on page 140 makes clear, the overhead associated with the MUSE middleware is relatively small in comparison to round-trip time. The protocol overhead for 368-byte sized messages for both HTTP and TCP/IP is less than one percentage of the round-trip time, so does not have a significant impact on the applications. When looking at the 100KB messages over TCP/IP it is clear that the marshalling and demarshalling times are adding significantly to the time. The source of this overhead

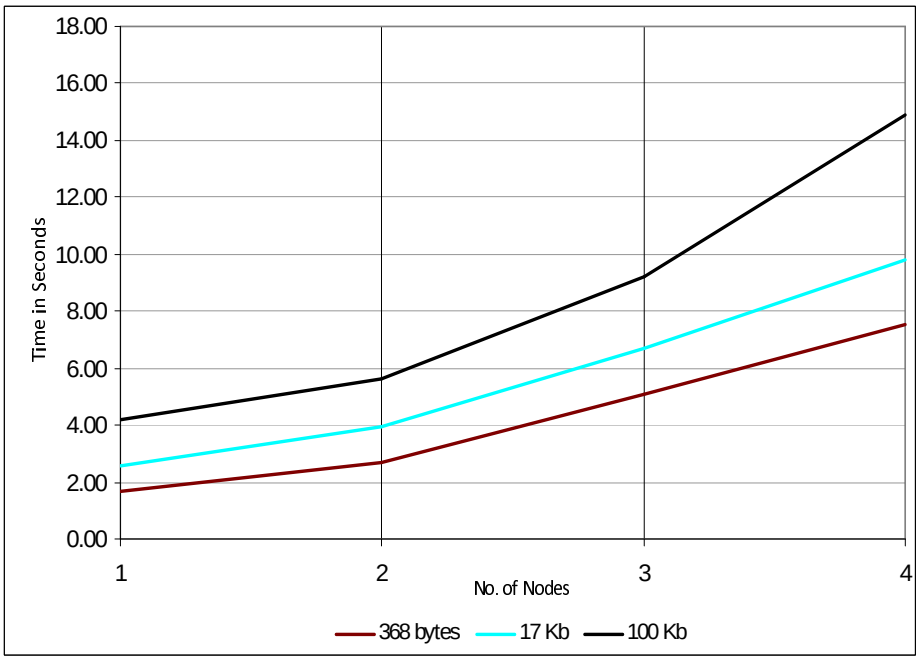

Figure 6.4: HTTP Round-trip Time

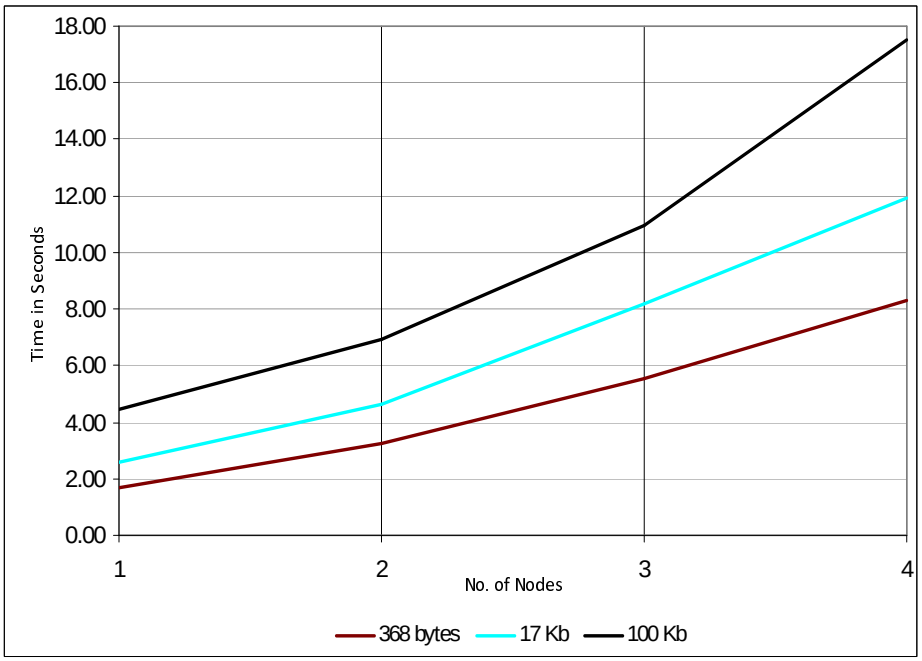

Figure 6.5: HTTP Group Latency

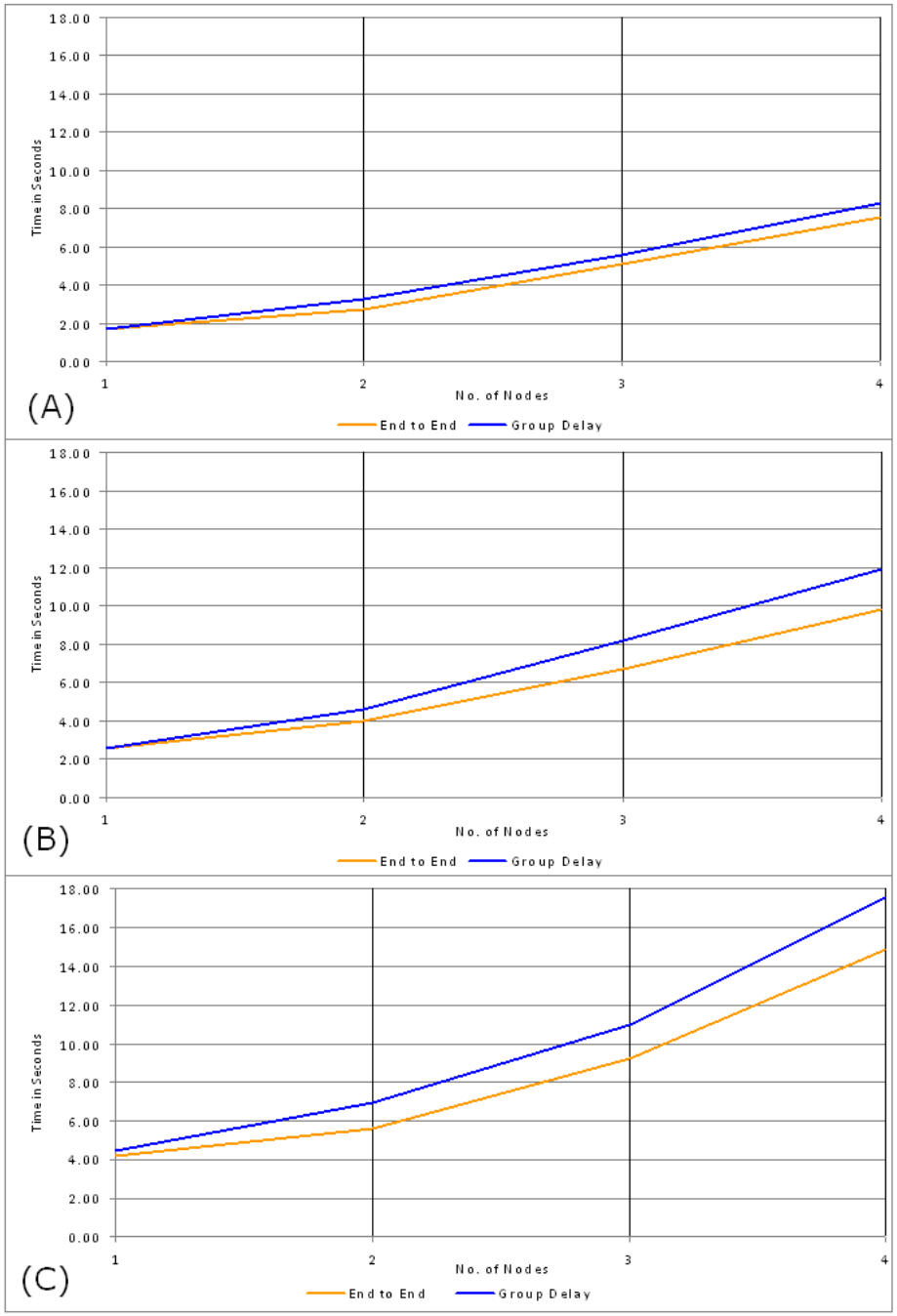

Figure 6.6: HTTP Round-trip Time Compared To Group Latency (A) 368 bytes message, (B) 17 KB message, (C) 100 KB message

	Marshalling	Round-Trip Time	Percent of roundtrip
TCP/IP (368bytes)	0.0005	0.086	0.58%
TCP/IP (17KB)	0.0040	0.104	3.84%
TCP/IP (100KB)	0.0450	0.239	18.82%
HTTP (368bytes)	0.0030	1.685	0.19%
HTTP (17KB)	0.0570	2.572	2.25%
HTTP (100KB)	0.1290	4.478	2.89 %

Table 6.3: Protocol Overhead

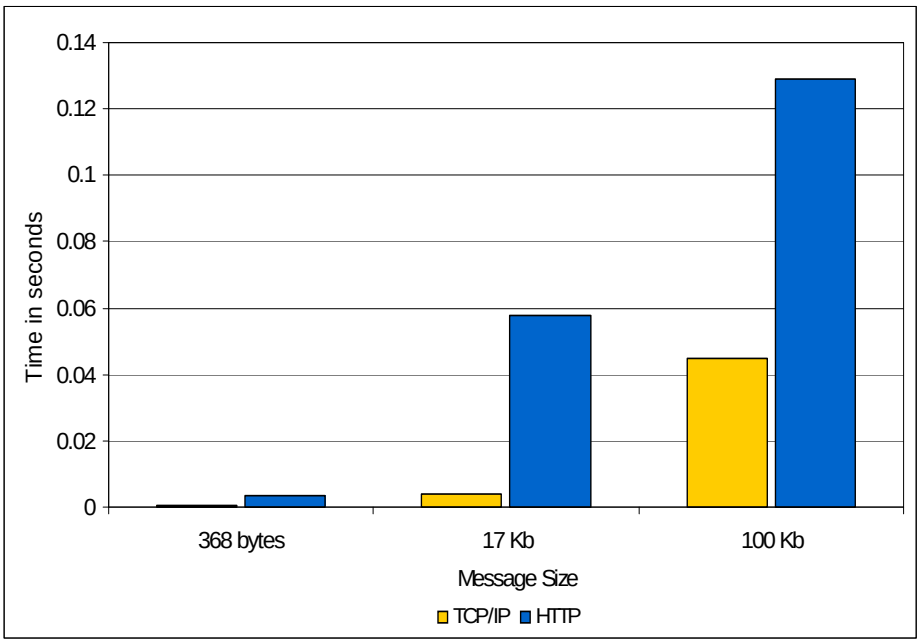

Figure 6.7: HTTP And TCP/IP Protocol Overhead

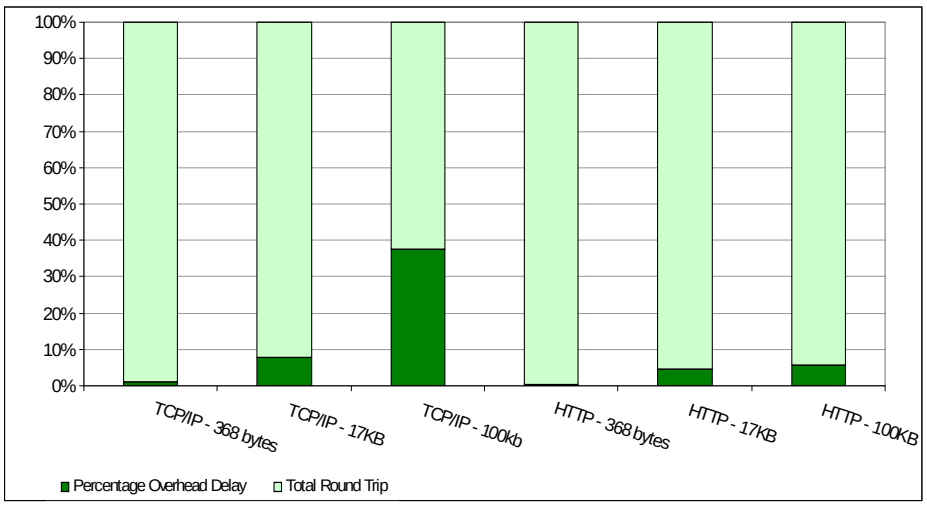

Figure 6.8: HTTP and TCP/IP Protocol Overhead Delay Comparison

	1	2	3	4	5	6	7	8
TCP-IP Throughput	443.200	278.568	205.360	173.440	146.120	119.264	94.608	78.456
HTTP Throughput	192.456	152.632	93.656	57.712	N/A	N/A	N/A	N/A

Table 6.4: MUSE Throughput (Kilobits/sec)

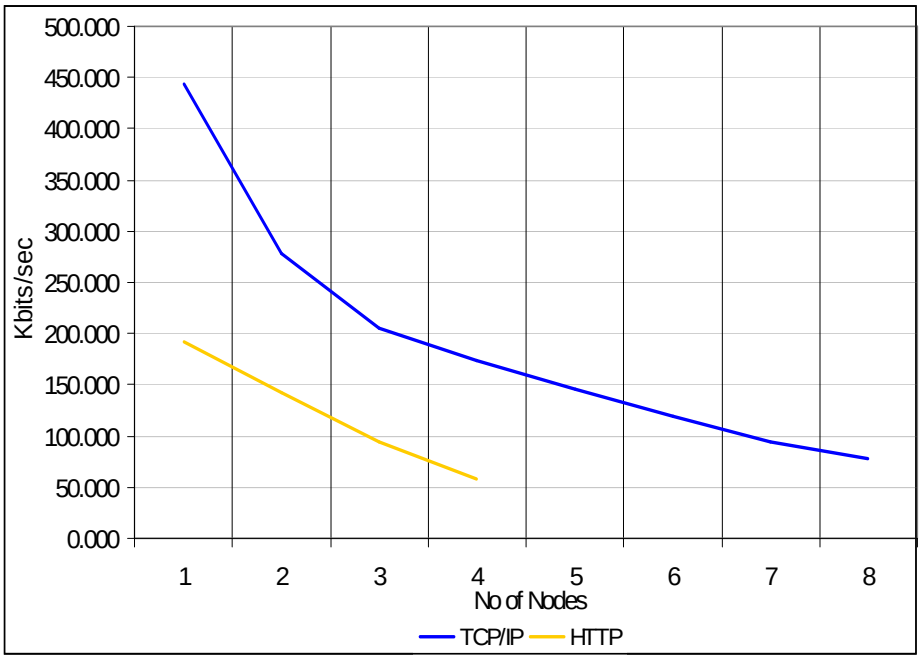

Figure 6.9: Graph Of TCP/IP And HTTP Throughput

is the encoding module that encodes and decodes binary media files to base64 encoded text. The time to encode the file into base64 grows linearly with file size. With larger file sizes this has a significant impact on performance. Therefore replacing this algorithm with a more efficient implementation would improve performance and throughput of the system.

The main impact of this delay is on throughput as encoding takes up a significant amount of time as the file size increases. The result is that as the number of clients increases, the demand of marshalling and demarshalling media files quickly decrease the maximum throughput of the system (Figure 6.9 on page 141). This suggests that the default encoding scheme should be replaced in future versions of the MUSE software for more efficient schemes to improve throughput.

The final metric, implementation cost, provides the developer with a rough guide to how much memory will be required to build applications using the middleware. It also gives an indication of how difficult in terms of developer effort it will be to create

	Memory	Packages	Classes	Lines
MUSE Middleware	270 KB	20	92	8682
MUSE Mobile Middleware	136 KB	19	87	7831
MUSE Application Server	196 KB	12	42	6180
MUSE Middleware PC + Server	N/A	32	134	14862
MUSE Mobile Middleware + Server	N/A	51	221	14011

Application Name	Memory	Packages	Classes	Lines	Total Lines with MUSE platform	% Reused Code
DNT Mobile	277 KB	12	27	5172	19183	73%
SMART	220 KB	14	31	5271	19282	73%
DNT PC	901 KB	20	68	11104	25966	57%

Total	N/A	105	326	46513

Table 6.5: MUSE And Application Implementation Cost

new collaborative applications using the middleware. More than seventy percent of the code used in the SMART and DNT mobile collaborative applications was reused from the MUSE platform directly (Table 6.5 on page 142). This is a major percent of the code required to implement the applications, significantly shortening the time required to create sophisticated collaborative applications on mobile devices. The memory footprint of the middleware compares favourable with the Agilo framework [Guicking et al., 2008]. The Agilo framework core consists of 112 classes for clients and 132 classes for the server which results in framework binaries of around 200 kb and 230 kb, respectively. The equivalent figures for MUSE mobile middleware are 136 kb with 87 classes, and 196 kb and 42 classes for the MUSE server.

The focus of this chapter was to provide support to answer the question: do the functional requirements for a MCSCL platform captured by MUSE, SMART, and the DNT support constructionist, contextualised, and collaborative activities for learning? The approach taken in this chapter was to address several sub-questions: from a technical perspective can we design and build an MCSCL middleware to work in a setting containing heterogeneous devices and networks? What is the performance of the middleware on mobile devices and PCs? How long is the message round-trip time between devices and the server? How many concurrent users can the system support? and from an educational perspective, what is the usability and learning support provided by the applications. Note that the approach to answering this question was described in Chapter 4 which examined the level of support for the following functional requirements identified in Chapter 3, including: support for the four dimensions of the space-time classification of groupware system; the nine dimensions of groupware technology from Mittleman et al. [2008]; constructionist, contextualised and collaborative

activities; data collection and location awareness.

To answer these sub-questions, the evaluation of MUSE involved two stages. From a technical perspective, in the first stage MUSE was evaluated in terms of a set of standard performance metrics. During this evaluation the following metrics were examined: round-trip time, group latency, protocol overhead, throughput, and implementation cost. The second part to answering this question is to validate the function of the middleware by building MCSCL applications on the MUSE platform and evaluating them in terms of usability and learning support. The next section focuses on the usability and learning evaluation.

6.2 Usability And Learning Evaluation

The evaluation for MCSCL applications, as with other complex systems including CSCW and CSCL systems, is difficult with many interdependent variables to examine [Lonchamp, 2006, Salomon, 1992]. Typically the evaluation of earlier systems focused on testing the individual collaboration facilities, for instance Baker and Lund compare the effects of using free-text chat as opposed to using structured chat "in order to exchange domain-related information, coordinate actions and reach agreement" when creating shared artifacts, in this case Energy-Chain Models in physics [Baker and Lund, 1996]. However, in another example, Lonchamp while discussing the evaluation of their Omega+ system states that "classical experimental evaluations are not well adapted for such a comprehensive environment because it is meaningless to isolate the effects of a particular design feature on learning" [Lonchamp, 2006].

One suggested approach to evaluating a complex CSCL system involves studying the whole learning environment examining the interaction and change between the interdependent variables over time [Salomon, 1992]. Lonchamp believes that one way of evaluating a generic system is to verify if such a system brings "the same kind of support as do existing specialised tools" [Lonchamp, 2006]. One practical suggestion from this study is that complex tools be evaluated by creating scenarios, complete with user tasks and goals, and then measuring their performance in these scenarios [Lonchamp, 2006]. Such a whole system and task-orientated approaches are often used in usability evaluation studies [Brooke, 1996, Mayhew, 1999] as it measures a system

in an authentic context in which it is expected to be used, thereby providing practical, applicable results.

The sub-question examined in this section is: can the applications (SMART and the DNT) developed on the MUSE MCSCL platform support constructionist, contextualised, and collaborative activities for learning? To address this question this section focuses on the usability and learning aspects of the MCSCL applications developed in the course of this thesis. The main body of this evaluation will focus on SMART with the initial results from a pilot study to evaluate the DNT also presented.

These applications are evaluated according to the framework described by [Vavoula, 2007, Vavoula and Sharples, 2008, Sharples et al., 2007], which advocates evaluating mobile learning projects according to three levels of granularity, the micro level, the meso level, and the macro level. The micro level examines activities of the users and assesses usability and utility of the technology used. The meso level examines the learning experience of the activity and technologies used. The macro level relates to the longer-term impact of the technology on educational and learning practice. This framework was developed in the context of the MyArtSpace [Sharples et al., 2007, Vavoula et al., 2006] project, which is an attempt to support structured inquiry learning with mobile technology to connect learning in the classroom with learning in museums. MyArtSpace enables users to take pictures, record sound, and write comments using the supplied multimedia mobile phones in order to reflect upon and share their experiences upon returning to the classroom. In addition, this framework was used to evaluate other mobile learning applications for example, by [Spikol, 2007] to evaluating their mobile game "Skattjakt", a collaborative treasure hunt game using mobile phones with GPS to navigate through the game. They found that this framework helped to identify problems, understand the learning processes, and identify further requirements.

This evaluation will focus on the micro and meso levels from this framework, with the macro level being outside the scope of the research. At the micro level, the questions asked examine the usability of the applications. At the meso level the questions asked is can the applications support constructionist, contextualised, and collaborative activities for learning? This section starts by outlining the usability metrics that are used for the usability evaluation at the micro level. This section then describes the

rubrics used to classify the learning at the meso level. Next the evaluation setting is described. Finally this section describes two studies to evaluate SMART and one small pilot study to evaluate the DNT.

6.2.1 Usability And Learning Metrics

This section describes the metrics and criteria used to evaluate the usability and learning outcomes of the MCSCL applications that were built using the MUSE platform. First the methods for gathering the usability data on the applications is described. Then the rubric used to gauge evidence for constructionism, contextualisation, and collaboration during the activities are outlined.

The main definition of usability is taken from ISO 9241-11 [ISO, 1998] "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". The definition implies that usability requirements are based on measures of users performing tasks with the product to be developed. ISO 9241-11 [ISO, 1998] mentions that the context selected should be representative of typical usage when evaluating usability. In addition, Brooke states that "usability of an artifact is defined by the context in which that artifact is used" [Brooke, 1996].

Usability measurement can be an expensive and time consuming process to undertake so several studies suggest strategies to reduce this cost. For example, Neilson advocates using a heuristic approach to evaluating usability [Nielsen, 1992, 1994] and Mayhew [Mayhew, 1999] suggests using broad task-orientated goals while developing completely new products. So the first dimension of the usability evaluation asks were the users able to achieve the goal of the task used in the study.

The studies in this thesis use a questionnaire to gather user feedback on what they liked or did not about the applications, and whether they had any problems with the applications. Questionnaires are a useful method to get rich data from such studies, e.g Bell [2005], Cohen et al. [2007]. The questionnaire in these studies uses open questions that allow the users the freedom to give feedback and mention specific usability problems they encountered with the applications (Appendix A).

In addition, usability questionnaires are a standard approach to ascertain usability quickly. There are a several questionnaires available for quantitative usability evalua-

System Usability Scale (©Digital Equipment Corporation, 1986)
1. I think that I would like to use this system frequently
2. I found the system unnecessarily complex
3. I thought the system was easy to use
4. I think that I would need the support of a technical person to be able to use this system
5. I found the various functions in this system were well integrated
6. I thought there was too much inconsistency in this system
7. I would imagine that most people would learn to use this system very quickly
8. I found the system very cumbersome to use
9. I felt very confident using the system
10. I needed to learn a lot of things before I could get going with this system

Table 6.6: System Usability Scale [Brooke, 1996]

tion, including SUMI [Kirakowski and Corbett, 1993], QUIS [Chin et al., 1988] and SUS [Brooke, 1996]. However, QUIS and SUMI are proprietary commercial solutions and so subject to licencing fees. Therefore, the final study described in this thesis uses the system usability scale (SUS) questionnaire which complements the above mentioned questionnaires as it measures the general usability of the application. The system usability scale is a simple, ten-item Likert scale that gives a global view of the subjective assessment of usability. In brief, a Likert scale is one based on forced choice questions, where respondents indicate the degree of their agreement or disagreement with a statement, typically on a 5 or 7 point scale. SUS uses a 5 point scale from "strongly disagree" to "strongly agree" with statements alternating between statements which typically get positive and negative responses. SUS provides a single score "representing a composite measure of the overall usability of the system being studied" [Brooke, 1996] in the range 0 to 100.

A recent study [Bangor et al., 2008] presents an analysis of 10 years worth of SUS data indicates that "the SUS is a highly robust and versatile tool for usability professionals" and acknowledges that the scale is easily understood by a wide range of people involved in the development of products and services [Bangor et al., 2008]. This study also attempts to answer the question: what is an acceptable score? They believe that the data and their own collective experience show that the *university-grade* rule-of-thumb (A =90-100, B=80-90 etc.) provides a good analog [Bangor et al., 2008] to interpret the scores. They conclude that acceptable products have SUS scores above 70, better products score in the high 70s to high 80s and superior products score more

than 90 however "products with scores of less than 70 should be considered candidates for increased scrutiny and continued improvement" [Bangor et al., 2008].

So in summary three methods are used in this thesis to gather usability information. The first method follows the task / goal-oriented aspect to usability evaluation, advocated in [Mayhew, 1999, ISO, 1998, Salomon, 1992], therefore to gauge the usability of the application the first question asked is: *were the users able to achieve their goals using the applications?* i.e. create the animation or digital movie in the case of SMART and DNT respectively. The second method is an open questionnaire used to uncover specific usability issues with the software. The third method is the system usability scale which is used to score general usability for the applications described in this thesis. Now turning to the learning aspects of the activities using the applications, this thesis uses a simple rubric to look for evidence of constructionism, contextualisation and collaboration.

Constructionism was earlier approximated as the theory that learning is best expressed when learners are actively creating a meaningful artifact that results in a tangible object that the learners can show to their peers resulting in further opportunities for discussion and reflection on the learning activity. This thesis considers an application to be constructionist in nature if it enables learners to create tangible objects that they can present to their peers. In the case of SMART this is principally the animation, but also the characters, sets and objects used in the animations. In the case of the DNT this is the completed movies. Furthermore, constructionism was examined by looking at the reflections of the learners on their artifact and the activity, using a questionnaire after the activity.

Collaboration was understood to be present when two or more people engaged in a specific activity to achieve a common goal. Furthermore, constructivism was briefly described earlier as a learning theory that states that individuals construct new knowledge from their experiences, assimilating new knowledge to represent their own internal representation of the world or changing their internal representation in light of contradictions. This leads to the idea that collaboration is a powerful force for constructivist learning because in order to collaborate effectively users must negotiate a shared understanding; this process of negotiation and explanation leads learners to update their mental models until they establish a shared understanding. This also

encompasses cooperation or division of labour strategies to achieving a goal. Direct evidence of this was taken from observations and indirect evidence through the learners reflection in the questionnaires on their experience of working in teams.

Contextualisation was understood to mean situating learning in an authentic context or as the ability of the learners to make use of the mobility of the devices to exploit opportunities available in their surroundings. This was mainly recorded through observation and examining the content of the artifacts produced by the learners.

In summary this rubric provides a checklist that can be used during and after conducting the workshops to examine the questionnaires and to set any observations in context. The next sections will describe two workshops to evaluate SMART in this context and one workshop to examine the DNT.

6.2.2 SMART Usability And Learning Evaluation

This section describes two workshops conducted to evaluated the SMART mobile application built on MUSE. In particular this section will attempt to answer two questions: can SMART support constructionist, contextualised, collaborative learning activities? and what is the usability of the SMART application.

As described above several studies indicate [Brooke, 1996, ISO, 1998] that usability evaluation should be carried out in an authentic context where users perform representative tasks with the artifact under investigation. To that end, a social outreach programme run within our university called the Bridge To College (B2C)[1] provides the setting for the evaluation of the SMART application described in this thesis. The programme provides students (typically 16 years of age) from designated disadvantaged second-level schools with an innovative technology-mediated learning experience. Volunteer third-level students provide the mentoring for the programme.

The format of the animation activity that is the focus of these evaluations contains four stages. In the first stage, the mentors introduce the activity and software. Then the participants break-up into groups of two to five people. Each group was supplied with a Nokia N73 mobile phone with SMART installed on it. In the second stage, each group creates a storyboard on paper, outlining the story elements that will form the basis of their animation. They create the drawings, backgrounds and other artifacts

[1]Bridge to College: https://www.cs.tcd.ie/research_groups/crite/b2c/

that they will need to tell their story and create the animations using SMART. During the third stage, the participants upload their animations to the application server using MUSE in order to generate a movie file with the `SMARTService`. They then present this completed animation to all the participants. The fourth and final stage is a discussion involving the participants, allowing them the chance to reflect on the activity.

The evaluation workshops described here lasted for three hours, at the end of which participants were asked to complete a short questionnaire (Section A.1). The questions fell into three broad categories that examined the participants' technology and multimedia background, their reflection on the activity, and solicited their feedback on the application. This section includes descriptions of two workshops to evaluate SMART. Each section starts by outlining the profile of the participants in the study including their previous computer experience. Then the results from the usability questions from the survey are outlined. Finally the results of the survey questions, and observations, related to examining learning and the user's reflections on the activity are presented. After both studies are described this section finishes by describing the conclusions from the two studies in aggregate.

6.2.2.1 First SMART Evaluation Workshop

The group in this first study consisted of fourteen participants between the ages of thirteen and sixteen. The participants worked in four self selected groups containing three participants and one group containing two participants. All participants completed the survey. The participants were familiar with technology, spending on average over eleven hours per week using computers. Typical activities they engaged in included YouTube™ (thirteen participants); playing games (eleven participants); using social networking websites (nine participants); editing photographs (four participants) and editing movies (two participants). From a mobile technology perspective, all participants had their own mobile phone with built-in camera, while twelve participants had a phone potentially able to run MUSE and SMART. To ascertain if they had engaged in similar activities before the participants were asked if they had created animations or digital videos previously, thirteen of the participants had recorded digital videos while seven participants had created animations on their PC. The most popular activity with digital videos was to share the video with friends (six participants) or share them

online (two participants uploaded them to YouTube).

As previously mentioned the evaluation follows the framework for evaluation described by [Sharples et al., 2007, Vavoula and Sharples, 2008, Vavoula, 2007] and focuses on the micro and meso levels. Taking the micro level first, this section discusses the usability of SMART. Questions at this level include, were the participants able to create animations, what did they like and dislike about the application under investigation and were there any problems with the application.

Firstly, the application did prove usable. All five groups were able to produce a short animation ranging from 10-15 seconds long during the short workshop session. When asked "what did you like about the animation activity today?" seven of the participants liked the hands-on artistic aspect to the activity, for example making the characters for the movies with modelling clay. Five of the participants mentioned that it was fun, and one participant liked the novelty of the activity. Two of the participants liked the fact it was simple while two others liked the social aspect of the activity of working in teams . Finally, one participant enjoyed the storytelling aspect.

In related responses to the question "how would you improve the software?", three participants suggested making the application faster with two being so specific as to suggest that more memory should be added to the phones to improve the software. A common complaint from the participants was that the final animations were moving too quickly. SMART sets an immutable default frame-rate at four frames a second. This proved too fast in practice and resulted in jumpy animations as the participants took too few images for this frame rate. While four frames a second can produce smooth animations, it requires many images to advance the story. Therefore, to improve SMART the frame rate should be configurable from the SMART user-interface, allowing the users to experiment with different frame rates to suit their animation.

At the meso level or learning evaluation, the focus was upon the degree to which constructionism, contextualisation, and collaboration were supported by the activity. The users were observed engaging in a constructionist process by creating the physical objects for the animation and creating the actual animations themselves. All of the groups managed to produce an animated movie during the three hours of the activity.

In order to gauge the participants' reflections on their movies and the animation activity two further questions were asked, "If you had more time, how would you

improve your movie?" and "if you were starting again with a new movie, how would you make it better?" Eleven responses advocate making better drawings and characters. Six responses suggest either more planning from the start or better storyboards would improve the animations. Both sets of responses show that the participants came to realise if they gave more consideration to the story to be told, before any artifacts or animations were created, it would have resulted in an improved animated story. Several of the movies that the participants created were not very smooth and contained jumpy animation. Three of the responses indicate cognisance of that fact, as they recommend adding more frames. Similarly, one response suggests re-shooting the movie. As the movies were typically 10-15 seconds in duration, four responses favour creating longer movies if they had another chance at this activity.

The setting of this evaluation provided opportunities for users to contextualise their work by using tools and artifacts available in their surroundings to improve their animations. There were computers and printers available in the room and three of the groups took advantage of these facilities to draw characters and backgrounds that they could print out and use in their animations.

As regards collaboration, the users were observed collaborating to create the objects, the storyboard, and the movie. In addition, while creating the animation, the users often employed a division-of-labour strategy, with one user holding the phone to capture the images, while the other users moved the characters in the movie. Again when one of the mentors asked, "What are you working on?" the participant replied, "He is doing the characters and I'm doing the storyboard".

6.2.2.2 Second SMART Evaluation Workshop

This study involved nineteen participants, who were students on a post-graduate course focused on the use of IT in education (M.Sc. Technology & Learning) run from the Centre for Research in IT in Education (CRITE), in the school of computer science and statistics, Trinity College Dublin. The participants worked in four self-selected groups containing four participants and one containing three participants. This group of participants were experienced with technology, spending on average forty hours per week using computers. All the participants routinely used word processing and email software, while social networking (seventeen participants), YouTube (fifteen participants)

and photo editing (thirteen participants) were also very popular with this group. From a mobile technology perspective, all participants had their own mobile phone, eighteen including built in camera, while thirteen had a phone that could potentially run MUSE and SMART. Furthermore, fifteen participants previously created digital videos, sharing them with friends (nine participants) or online (two participants) were the most popular activities with the completed movies.

At the micro level the application was usable as all the groups were able to produce a completed animation. When asked "what did you like about the animation activity today?" nine of the participants liked the team work and collaborative aspect of the activity, eight that it was fun, and three like the fact that they learnt about animation. Two of the participants liked the fact it was simple while two others liked the creative aspect of the activity. Finally, the concrete, novel and inspiring nature of the activity were mentioned by one participant each. Conversely when asked "what did you not like about the animation activity today?" the most common answer was technical problems as the phone froze on several occasions with the result that the application didn't save some image frames or lost a couple of frames.

In related responses to the question "how would you improve the software?", three participants suggested making the image creation quicker, and three participants suggested changes to the user interface. Two participants would have liked to have been able to edit the animation on a PC. Specific suggestions where to change the hierarchical menus to a flat menu system, to add a duplicate frame option, and to add a function to add sound to the animation.

Again looking now at the meso level, the focus was upon the degree to which constructionism, contextualisation and collaboration were supported by the activity. The observed behaviour and outcomes were the same at this level as for the first evaluation workshop. All users managed to create an animation, were engaged in a constructionist process, making both animations and artifacts for the animation, and were observed collaborating to create the artifacts, the storyboard, and movie. Similarly the users employed a division-of-labour strategy, with one user holding the phone to capture the images, while the other users moved the characters in the movie.

The participants were asked two further questions to ascertain their reflections on the animation they created: "If you had more time, how would you improve your

movie?" and "if you were starting again with a new movie, how would you make it better?". Fourteen responses said that taking more images would improve their animation. Ten responses mentioned that making better drawings and characters would improve their animation. Eight responses suggest either more planning from the start or better storyboards would improve the animations. As in the earlier study both sets of responses show that the participants came to realise if they gave more consideration to the story to be told, before any artifacts or animations were created, it would have resulted in an improved animated story. The animations were not smooth, so six responses recommended using a tripod or stand camera to improve their animation. Furthermore one group used the supplied art materials to create a make-shift tripod for their phone.

The setting of this evaluation provided opportunities for users to contextualise their work by using tools and artifacts available in their surroundings to improve their animations. In this evaluation one of the groups used a mural painted on the wall of the room as a setting for their animation. They also made use of the computers and printers in the room to create characters, props, and backgrounds for their animations.

6.2.2.3 Summary And Conclusions

The question asked was does the SMART application support a constructionist, contextualised, and collaborative approach to learning? A related question is then what is the usability of the SMART?

Taking the usability question first, the application proved usable by one significant measure because it allowed the users to achieve their goal in this task-oriented activity [Mayhew, 1999, ISO, 1998]. In other words all the groups in both workshops were able to produce animations during the three-hour animation activity. However over the course of the study twenty-two of the participants registered some degree of problems with the application or phone. The main complaint was that the phone froze (9 participants). Furthermore the second workshop uncovered a bug in the MUSE auxiliary-layer used by SMART that managed saving images on the mobile phone, with eight participants reporting this problem occurring. The two remaining participants only reported that there was a problem with the application, but never specified. There were no other usability problems raised.

In addition the observation of the participants in the first study revealed one specific usability problem with SMART because the participants complained that the final animations were moving too quickly. The fixed frame rate of SMART was too fast resulting in jumpy animations as the participants took too few images for this frame rate. Therefore, to improve SMART, the frame rate was made configurable from the SMART user-interface in the second study, allowing the users to experiment with different frame rates to suit their animation.

Some useful additions to the application where suggested including: The ability to duplicate frames, add sound and to edit the animations on a PC. These suggestions hint that integrating SMART to use the DNT editing functions for post-production would be a good idea as it can facilitate all three options. Furthermore in both cases the participants liked to share digital videos with their friends when they created digital videos previously, whether on their computer, social networking websites and YouTube. Therefore, providing additional services on the MUSE application server to interact with third-party services like YouTube might be an interesting avenue for further research and development. Two participants suggested changing the application to remove the division of the animation in to multiple scenes and instead group all frames in the same scene. However this was used by seven of the ten groups to structure their story, so it provides scaffolding for the animation creation process.

So in summary at the micro level, SMART did prove a usable application in the context of the animation activity. Nevertheless there were some bugs in this version of SMART and MUSE that caused the phone to freeze that should be addressed in future. The application could be further enhanced by integrating SMART with the editing tools available in the DNT. This section next looks at the meso level question: can the SMART application support constructionist, contextualised, collaborative learning activities?

From a constructionist perspective all the groups from the two studies were able to produce an artifact that they could present to the other participants in the workshop. In addition the questionnaire captured the participants reflections and feedback on the activity and their animations. When asked how they would improve their animation, three responses stand out: more frames, better planning and storyboard, and steady camera. These responses suggest that the participants had updated their understanding of how animation works and that they would approach the task differently if they

were to undertake it again.

In terms of collaboration during the activity, this was observed in all the teams. The typical approach was for one person to concentrate on capturing the images while the other group members created the characters, backgrounds and sets to use in the animation. Furthermore when the participants were asked *"What did you like about the animation activity today?"* the most popular answer (nine participants) was working in teams which can be taken as a proxy for collaboration in this context.

In terms of contextualisation this was noted in the fact that the participants made use of the computers and printers in their surroundings to create characters and backgrounds for their animation. The other aspect is one group made use of the physical environment as a backdrop to the story they were telling with the animation, in this case a mural on the wall provided the setting for their story.

These workshops, and others pilot trials carried out, indicates that the use of SMART does lead to successful learning experiences and offers a practical way in which mobile technology can be used to support animation as a whole class activity. However, SMART would benefit from some future improvements including: the ability to duplicate frames, add sound, and improve application stability.

6.2.3 DNT Usability And Learning Evaluation

The objective of this workshop was to pilot the use of the mobile and PC versions of the DNT application to create a digital narrative following the mobileDNA process. Three questions were of interest to this study. First what is the current usability of the application, and were there any specific improvements to the application necessary? Secondly, can the application support a constructionist, contextualised, and collaborative learning activity? Thirdly, does the application resolve any of the difficulties experienced by the users using the existing approach to the mobileDNA?

For the first stage of this workshop the group used the DNT PC client to create a mind-map for the brainstorming stage of the mobileDNA. For the second stage the participants used the storyboard interface of the DNT PC client to refine the story to be told. For the third stage the users used the mobile DNT client on the mobile phones to capture the images and sounds for the movie. For the final stage they used the timeline interface on the DNT PC client to create the completed movie. They then

presented their movie to other groups who were involved in other projects on the day. In addition, they completed a short questionnaire and completed the system usability scale form. A short unstructured interview was conducted after the workshop.

This was a small study involving five participants, who were students between 15 and 16 who were attending the Bridge To College described earlier. Due to the small number of participants in the group, the mobileDNA had to be modified so that all the participants were involved with filming stage and only split into smaller groups for sound recording and editing, i.e. they all worked in one group to create the digital narrative that was the objective of the workshop. This group of participants were experienced with technology, spending on average seven and a half hours per week using computers. From a mobile technology perspective, all participants had their own mobile phone with built in camera potentially capable of running MUSE and the DNT.

At the micro level, the first thing to note is that the DNT application did prove usable as the participants were able to create a finished digital narrative in the time allowed (three hours) for the workshop. They achieved the task despite the fact that they were not able to parallelise as much of the exercise as in the usual mobileDNA, because of the small group size there were not enough people to divide the group into filming, sound recording and editing teams at the same time.

The second aspect to evaluating the usability of the DNT application was the participants were asked to rate the DNT application using the system usability scale. Obviously because of the small group size the results from this assessment are just a rough guide for future work. The raw scores were 57.5, 62.5, 75, and 80 percent (the fifth participant did not complete the survey). The only result to draw from these scores is to focus on the reasons for the two low scores to improve future iterations of the software. A starting point is in this respect are the answers to the question: *What did you not like about the activity today?* The two responses to this question were: "It was too complicated and it was slow" and "the program that we used was a bit slow and awkward". Therefore investigating how to simplify the mobile interface further might help in this regard. However, the slowness experienced by the participants using the application was during the image capture stage, which is hardware dependant. One feature that the participants mentioned should be included in the mobile DNT client

is the ability to view captured images or listen to captured sounds on the device. This is a good point and should be included in a future iteration of the software.

From the learning perspective or meso level of the evaluation framework the study asked can the application support a constructionist, contextualised, and collaborative learning activity. To provide context the following paragraph describes the way the participants actually engaged with the mobileDNA during the workshop.

The start of the mobileDNA process was to create the mind-map for brainstorming and to support divergent thinking. The participants did use the mind-map tools for this idea-generation stage. The second stage of the mobileDNA was refining the story into a number of story beats that will provide a script for the story. The participants used the storyboard on the DNT PC application for this aspect of the process while at the same time switching to consult the mind map they created on several occasions to help focus their storyboard. The third stage of the mobileDNA is the filming and editing stage. Normally this involves performing filming, sound recording and editing simultaneously. However due to the small group size the participants first did all the filming together, then they created a rough edit of the story using the DNT timeline, which enabled them to discover if they needed to film any additional images. The whole group then did a second round of targeted filming to capture the extra images they needed. Once they had all the images that they needed they split up again with one group of two participants editing the images into the completed movie, while another group of two participants wrote a script that would be used to narrate their story. This group then recorded the script in another room while the editing group added this soundtrack as the media arrived. The final participant had to leave the workshop before the editing stage.

Starting from a constructionist and collaboration aspects, the participants were able to create a digital narrative movie in the three hours time allowed for the workshop. Furthermore they collaborated to create the mind-map to get ideas for their story, to create the storyboard to refine their story, and to capture the images for the movie. They also co-operated to record the soundtrack and edit the movie simultaneously. From a contextualised perspective they used buildings in their surroundings of Dublin city centre to highlight aspects of their story. In addition, two participants reflecting on the activity said they would have made the movie a bit longer, and one participant

said that if they had more time they would have "edited the sound to perfection".

The final objective of the workshop was to evaluate to what extent the DNT application could resolve problems experienced by the mobileDNA. These difficulties included: maintaining a shared understanding between the filming and editing groups; managing media files; communication between groups; and scaffolding the process when there was no integration between software.

First the main caveat here is that as this pilot study involved a small group size, smaller than the usual group size for the mobileDNA activity, therefore the actual workflow of the activity had to be adapted with the users participating in most of the tasks rather than splitting up into groups with distinct roles, i.e. filming, sound recording or editing. However the groups did parallelise the sound recording and editing aspects of the process. This mainly impacted the problems of maintaining a shared understanding between the filming and editing groups, and communication as all the participants were always together during the filming stage. For instance when asked did you use the built in chat tool, two participants state "no because we were always together". However if we look at the filming and editing as tasks separated in time, then the application did help as the participants used the storyboard on the phone when capturing images, so that when they went to edit the timeline later the media was grouped into the appropriate scenes. Regarding problems managing media files none of the participants had any problems and when asked what did you like about the application one of the participants mentioned this in particular, stating: "That all the pictures and sounds when straight to the computer".

Regarding the benefits of an integrated application over the diverse software used in the mobileDNA workshops there is not enough data, the only points of interest are that the integrated application simplified media management and that the participants did use the storyboard on the mobile client to organise the capture of media for the story.

6.3 Conclusion

The main research question examined in this chapter is: do the functional requirements for a MCSCL platform captured by MUSE, SMART and the DNT support constructio-

nist, contextualised and collaborative activities for learning? The focus of this chapter involved addressing several sub-questions: from a technical perspective can we design and build an MCSCL middleware to work in a setting containing heterogeneous devices and networks? to what extent does the MUSE platform support the functional requirements identified in Chapter 2? What is the performance of the middleware on mobile devices and PCs? How long is the message round-trip time between devices and the server? How many concurrent users can the system support? and from a educational perspective, can MUSE support constructionist, contextualised, and collaborative activities for learning? and what is the usability of the applications developed?

While Chapter 4 addressed the sub-question: to what extent does the MUSE platform support the functional requirements identified in Chapter 2, this chapter focused on answering the other sub-questions with a two stage evaluation of MUSE. From a technical perspective, in the first stage MUSE is evaluated in terms of a set of standard performance metrics. During this evaluation the following metrics are examined: round-trip time, group latency, protocol overhead, throughput, and implementation cost. The second stage is to validate the function of the middleware by building MCSCL applications on the MUSE platform. This second stage involved building two applications to use the MUSE platform, SMART and the DNT. These applications were then used to validate the functionality of the MUSE platform and evaluate the level of support for constructionist, contextualised, and collaborative activities for learning.

The first sub-question was: can we design and build an MCSCL middleware to work in a setting containing heterogeneous devices and networks? During this thesis MUSE was able to support the MCSCL applications SMART and DNT. Taken in terms of the responsiveness of the system another question of interest was: How long is the message round-trip time between devices and the server? For both TCP/IP and HTTP the system could support client applications within the bounding constraint of one second over TCP/IP and thirty seconds over HTTP for both round-trip time and group delay. In addition for group latency it was shown in both Figure 6.3 on page 135 and Figure 6.6 on page 139 that group latency closely tracks the round-trip time, indicating that all clients can expect equal performance when connected to the platform.

How many concurrent users can the system support? In terms of round-trip and group delay times, when connected via TCP/IP the MUSE platform can support four

simultaneous users within the specified constraints, five clients can be connected and receive performance only slightly above these time constraints. Therefore five clients is defined as the maximum TCP/IP clients for the system to perform satisfactorily, although more can be supported, performance can no longer be guaranteed to be adequate. Four clients can also be supported over HTTP within these bounds. However it is possible that more clients can be supported, probably between six and eight, but further equipment and testing would be required.

What is the performance of the middleware on mobile devices and PCs? Firstly, protocol overhead was found to be insignificant in relation to the SOA, therefore this component of the protocol overhead can be ignored. The second element of protocol overhead is the marshalling and demarshalling time. This was found to be negligible on the mobile devices. However on the PC using TCP/IP the marshalling and demarshalling time consumed a large percentage of the total round-trip time (see Figure 6.7 on page 140). This was accounted for by the fact that the algorithm used to encoded the binary media files was not very efficient. The modular design of MUSE will allow for this base64 encoding module to be replaced later transparently. The impact of this delay was most evident on throughput as the encoding delay grew with file size and resulted in throughput falling dramatically as more devices were added. Improving throughput and encoding should be a main priority for future versions of the software, as this would have the most bearing on increasing the maximum number of clients that could use the MUSE platform simultaneously.

Furthermore, the short round-trip times for the 368-byte messages, and the fact that larger file sizes are transferred using lower priority background process, results in a middleware that remains responsive for up to five PC clients or at least four mobile clients. With this number of devices the middleware would be able to provide adequate support for exchanging data, awareness information, and multimedia files between MCSCL client applications.

Finally, in terms of implementation cost, the main finding is that for the MCSCL applications over 70 percent of their code base can be reused from the MUSE platform. The DNT PC application has a lower percentage of reused code as there is a lot of code to implement the sophisticated GUI, it is still over 50 percent. In order to provide a further validation of the middleware the next step is to evaluate the MCSCL application

built using the middleware.

The framework from [Sharples et al., 2007, Vavoula et al., 2006, Vavoula and Sharples, 2008] provides an evaluation context for the study of the applications (SMART and the DNT) in this thesis with the MUSE middleware. Broadly this involves evaluating such software in three levels, micro (usability), meso (learning) and macro (organisational). The study focused on the micro and meso levels with the macro considered outside the scope of this evaluation.

The evaluation of the applications involved users engaged in goal-oriented tasks in the context in which the applications are expected to be used. Therefore the setting for the evaluations involved a workshop where participants were expected to work in groups to produce multimedia artifacts by the end of the workshop, for example an animation with SMART or a digital movie with the DNT. To ascertain the usability the first question was could they achieve the goal and produce the artifact during the workshop. The second method used an open questionnaire to gather specific usability problems with the software.

Two evaluation workshops to test SMART were described, which involved thirty-three participants in ten groups using the software to create short animated movies. All the groups were able to produce a movie by the end of the workshop. The first study did uncover one usability problem that resulted from the frame rate for the animation being fixed. This was corrected in the next iteration of the software and was not mentioned as a problem in the second workshop. The only other usability problem noted was a bug when saving images using the MUSE middleware that resulted in the frames going missing.

To evaluate the DNT a small pilot study was carried out, which involved five participants working in one group to create a digital narrative movie using the DNT PC and mobile applications. The group were able to produce a movie during the workshop. However some of the participants in this study mentioned that the DNT mobile application was a bit complicated or awkward so further research to simplify this interface is needed.

From a educational perspective the studies examined if the applications could be used for constructionist, contextualised, and collaborative activities. This thesis uses a rubric to classify observations and survey responses according to these categories.

Accordingly the studies reveal that both SMART and DNT can support learning activities with each of these aspects present. For the DNT this thesis was also interested in whether the DNT application can solve some of the problems experiences by users with the mobileDNA process. Due to the nature of the small pilot study it was only possible to find evidence to support the case that it appears to eliminate the problem with managing media files.

Chapter 7

Conclusion

A emerging area in CSCL is mobile-CSCL. MCSCL is a broad area, borrowing heavily form the wider CSCL, CSCW, and mobile learning fields. Starting from a pedagogical perspective, classifications of MCSCL argue that leveraging the affordances of mobile devices allows the development of leaning activities that significantly differ from desktop-based CSCL applications. One particular advantage is the increased opportunity to contextualise a learning activity in authentic settings outside the classroom. Furthermore there is general agreement in the mobile learning community that applications informed by the principles of constructionism, contextualisation, and collaboration are of particular value for exploiting mobile devices for learning.

For this reason this thesis highlighted that animation and digital video production are activities that facilitate constructionist, contextualised and collaboration learning experiences. However the equipment required for these activities can be expensive leading to problems with access to the appropriate technology. Furthermore, to date little use has been made of ICT to support the collaborative aspect of the process. Therefore these activities would benefit from applications in the MCSCL space, especially given the wide availability of phones with built-in cameras.

The lessons learnt in the broader CSCW and CSCL have a major impact on the design of MCSCL platforms and applications. Two important ideas in CSCW are the classification of CSCW applications according to the space-time divisions between the participants using collaborative applications (see Table 2.1 on page 12) and the classification of CSCW application according to the groupware functionality the applications provide. For example Mittleman et al. use nine dimensions to classify groupware

systems according to: core functionality, content, relationships, session persistence, supported actions, action parameters, access control, alert mechanisms, and awareness indicators [Mittleman et al., 2008].

Furthermore, it was noted in thesis that the trend in CSCW and CSCL application development is towards generic extensible systems that can support several interaction mechanisms. In order to provide a generic MCSCL platform, and building on this trend, a middleware to support CSCW applications and by extension MCSCL applications should aim to support as complete a subset of the above mentioned interaction and groupware functionalities as possible.

Mobile learning is a fast growing research field, particularly as ever more powerful mobile devices and smart-phones are on the market, which makes creating sophisticated learning applications on mobile devices more feasible. This multiplication of powerful devices, while a boon to some application developers also has a major downside. The current set of mobile devices available encompass a heterogeneous set of operating systems, app stores, computational resources, network connectivity options, screen sizes, input devices and other hardware characteristics. Note the wide variety of choices available when listing only operating systems, including: iPhone OS, Windows Phone 7 Series, several branches of Android OS, Symbian S60 and S40, Palm webOS, and legacy versions of Symbian and windows mobile. Therefore in this environment it is a substantial challenge to develop software to target more than one device.

Another noted problem in the mobile software development area is the dearth of middleware available to support software development. The result is that when users wish to develop mobile learning applications they have to choose one device on which to develop their applications. This has the problem of undermining a key potential of mobile learning, which is the ability to take advantage of users existing technology to deploy new learning applications. For example, this often results in having to hand out special devices for a particular learning activity, and segregating it from the rest of the learning experience in the classroom. Because of the quick rate of change in mobile technology such devices can become obsolete quickly, necessitating re-implementing the application again from scratch on the new generation of devices.

While several middlewares were compared during this thesis, none of the middlewares described supported the aggregate requirements from CSCW, CSCL, mobile

learning, pedagogical considerations, and multimedia production described during the thesis. For these reasons a generic middleware that provides certain guaranteed behaviour and functionality; that has a technology agnostic definition; that provides for smooth porting of applications to other devices; and provides a migration path to new devices; would be a substantial aid to the developer of mobile learning software, or mobile software in general. Therefore, this thesis focused on the design and development of the MUSE platform to support development of MCSCL applications, with a particular focus on digital video production. Consequently, a secondary aspect of this thesis is the design and development of two MCSCL applications for digital video production called SMART and the DNT. The next section will describe these contributions and review any findings from evaluating the MUSE platform and the MCSCL applications.

7.1 Thesis Contributions And Findings

The previous section highlighted the lack of middleware for support for mobile learning and MCSCL applications on mobile devices and PCs. In addition it was noted that digital video production activities present valuable opportunities to support constructionist, collaborative and contextualised learning but that there is a lack of ICT support for collaboration. Therefore, this thesis investigates two main research questions. The first question asked in this thesis was: what are the functional requirements for a platform to support MCSCL applications on both mobile and PCs? The second question is: do the functional requirements for a MCSCL platform captured by MUSE, SMART and the DNT support constructionist, contextualised and collaborative activities for learning?

In short, there are five contributions from this thesis which help to answer these questions: The set of functional requirements for a MCSCL platform; The MUSE platform conceptual architecture; the MUSE platform reference implementation; the SMART mobile animation application; and the DNT mobile and PC application to support the mobileDNA.

7.1.1 Functional Requirements For A MCSCL Platform

What are the functional requirements for a platform to support MCSCL applications on both mobile and PCs? This first question is addressed by a review of the literature relevant to CSCW, CSCL, MCSCL, mobile learning, and the pedagogical theories relevant to mobile learning and CSCL. In addition, the thesis then examines the previous work on mobile middlewares in terms of these functional requirements.

This thesis found that the first requirement was that the MCSCL platform should support all four dimensions of the space-time classification of groupware system. Secondly, the platform should support all nine dimensions of groupware technology Mittleman et al. [2008]. These nine dimensions are: core functionality; content; relationships; session persistence; supported actions; action parameters; access control; alert mechanisms; and awareness indicators. Thirdly from a CSCL and mobile learning perspective it should support constructionist, contextualised and collaborative activities. Fourthly it should support mobile applications for data collection, location awareness and collaboration. The collaborative aspect is also relevant to the previous findings from CSCW and CSCL. To support data-collection it should provide an extensible abstraction over underlying hardware so that additional functionality can be added as new hardware and sensors become available. In order to accommodate location aware applications a similar extensible abstraction over the locations sensors should be included. Finally this set of functional requirements were then compared against the existing related work on mobile middlewares. It was found that none of them provided support for the complete set of requirements. This in turn informed the design of the MUSE conceptual architecture and ultimately the MUSE reference implementation.

7.1.2 MUSE Conceptual Architecture

The lack of middleware support for MCSCL applications was noted earlier, therefore this thesis provided a conceptual model for a MCSCL platform to support developers creating applications in this domain. In addition the conceptual model facilitates deploying the MUSE middleware on multiple platforms. There are four elements to the conceptual design to increase the flexibility of the software in the heterogeneous mobile environments. Firstly, all communication uses specified XML formats so that applications can

easily communicate in a language independent way. Second the MUSE service-layer and MUSE networking-layer are designed according to a design-by-contract approach, which means implementations of these layers must adhere to a contract of behaviour to the other layers. Thirdly the MUSE conceptual architecture recommends the use of an MVC architecture, backed by a SOA, to support loosely coupled components. Finally the MUSE auxiliary-interface specifies a set of functions that an application can rely on regardless of OS supplied support.

This approach to designing and specifying the MUSE platform focuses attention on the interface between components in the system in order to maximise software reuse and minimise development effort if re-implementation is necessary. This approach allows alternative implementations of MUSE to interact with each other, thereby reducing the development effort extending an application to alternative devices. This approach allows components of the MUSE system to be upgraded to new devices or use new programming languages on a component by component basis. For instance the server could be reimplemented in C++ for performance reasons, without needing to change any of the clients. Alternately a new version of the middleware could be created on the iPhone for instance without needing any updates to existing clients connecting to the platform. Given this conceptual model, it was necessary to build a reference implementation to test the soundness of the model and to provide an implementation that new versions can test their correct operation against.

The second research question asked in this thesis was: do the functional requirements for a MCSCL platform captured by MUSE, SMART and the DNT support constructionist, contextualised, and collaborative activities for learning? To start to answer this question MUSE is compared with the other middlewares described in the state of the art (Chapter 3) and against the set of functional requirements (4.3.12). It is found that MUSE is the only middleware described that can support all the functional requirements listed. A second approach to addressing this question is outlined in the next section, which reviews the reference implementation and describes the findings from the performance evaluation of this implementation.

7.1.3 MUSE Reference Implementation

The reference implementation of the MUSE platform provides an instantiation of the MUSE conceptional design. Therefore it includes the following salient features: parsers for all the XML formats in Appendix B; communication between the distinct layers uses the defined XML messages specified as part of the Design-by-Contract pattern; includes support for an MVC approach to distributed applications; uses a services orientated architecture; and contains implementations of all auxiliary-interface classes. The reference implementation was developed using J2ME and J2SE.

Briefly restated, the MUSE platform has three major components, the MUSE middleware, the MUSE application server, and a set of infrastructure services used to implement core functionality of the MUSE platform. The MUSE middleware uses a layered architecture which includes two core tiers, the networking-layer and the service-layer. This networking-layer layer acts as a reliable, connectionless, and asynchronous communications mechanism between client and the server. The service-layer layer is the service oriented architecture layer which provides an API to interact with services through the MUSE architecture. In addition the MUSE middleware includes an auxiliary-interface which provides a set of auxiliary modules for accessing multimedia facilities, security, advanced I/O APIs, user management and group management. The application server hosts services, controls access to shared data, and manages connections to external resources or databases. The infrastructure services provide the core functionality needed by the MUSE platform including: the publish-subscribe implementation, user management, and routing service replies to the clients.

This thesis uses Guickings et al.s taxonomy to classify middleware systems [Guicking and Grasse, 2006, Guicking et al., 2008]. In terms of this model the MUSE middleware uses a client-server architecture as its *distribution model*. The MUSE middleware's *communication infrastructure* supports HTTP, TCP/IP and MMS at the physical communication layer. Message passing and an event-based architecture are used as the *sharing model* in MUSE. The *concurrency model* in MUSE relies on the server ordering messages using the order in which they were received. All access to services on the application server are synchronised using Java locking and synchronisation mechanisms to ensure atomic updates to service state. The MUSE middleware uses a central server to mediate access to shared data, so data consistency or the *synchronisation model* is

ensured by the implementation of the concurrency model.

In addition, to support groupware and multimedia production applications in particular MUSE includes: support for user and group management; generic collaboration services e.g. text chat communication; functions to serialise and deserialise multimedia files to transfer as plain text; and services to convert image and audio files into a video file.

Do the functional requirements for a MCSCL platform captured by MUSE, SMART and the DNT support constructionist, contextualised, and collaborative activities for learning? The second approach to answer this question is to divided it into several sub-questions: from a technical perspective can we design and build an MCSCL middleware to work in a setting containing heterogeneous devices and networks? What is the performance of the middleware on mobile devices and PCs? How long is the message round-trip time between devices and the server? How many concurrent users can the system support? and from a educational perspective, can MUSE support constructionist, contextualised, and collaborative activities for learning? and what is the usability of the applications developed? To answer these questions the reference implementation of the MUSE platform was tested using the following performance metrics: average message size; message round-trip time; group latency; protocol overhead; throughput and implementation cost. These metrics uncovered the acceptable range of operation that this middleware can support. During user studies and development it was found that messages sizes clustered around three discrete sizes, 386bytes, 17KB, and 57KB. These sizes correspond to: system messages, used to interact with data-structures and core functions in the middleware; the average image size; and the average sound file size respectively. However during the performance evaluation 100KB was used in place of 57KB in order to measure performance because this equals the maximum size of audio files used by the DNT.

This thesis defined a one second delay as acceptable on the desktop and a round-trip time of 30 seconds as acceptable on mobile devices. Using this measure it was found that the MUSE platform could acceptably handle four simultaneously connected PC clients or at least four simultaneously connected mobile clients. Furthermore, after examining group latency it was found that group latency closely tracks the round-trip time. This indicates that there is low variance between the performance received by

each of the clients using MUSE. This is significant as it means that the performance of the clients can be predicted with a good degree of certainty at each client.

To measure protocol overhead the execution times for marshalling and demarshalling messages and routing messages using the SOA were recorded. Protocol overhead was found to be insignificant in relation to the SOA as execution times were measured in nano-seconds on the PC and were less than the measurement resolution on the mobile devices and so were recorded as zero. The marshalling and demarshalling time was found to be relatively unimportant on mobile networks as the marshalling time was less than three percent of the round-trip time. However on the PC using TCP/IP the marshalling and demarshalling time consumed a large percentage of the total round-trip time, totalling 18.82 percent in each direction. Most of the marshalling time was caused by the encoding algorithm used to convert binary multimedia messages into base64 format for transmission. This encoding time increased in relation to file size. The main impact was that as the number of clients increased throughput decreased because the message encoding became a bottleneck in the system.

Implementation cost was measured in terms of application memory footprint, and the number of packages, classes and lines of code used. It was found that the MCSCL applications build using MUSE, over 70 percent of their code base is reused from the MUSE platform. This implies a significant reduction in developer effort when using the platform as this functionality doesn't need to be re-implemented from scratch. In addition to use the MUSE platform a developer only needs to implement four simple interfaces to access the core functionality of the MUSE platform. All the auxiliary-interface modules contain default implementations so no development work is required to use them with the reference implementation.

A final aspect to answering this second research question and to provide further validation that the MUSE platform can support applications for collaborative digital narrative production two applications were created using this middleware, SMART and the DNT. These applications were then evaluated in the context of the framework from [Vavoula, 2007, Vavoula and Sharples, 2008, Sharples et al., 2007]. This involves evaluating such software at three levels, micro (usability), meso (learning) and macro (organisational). The study focused on the micro and meso levels with the macro considered outside the scope of this evaluation.

7.1.4 SMART

The first application described was SMART, which is a mobile application to enable learners to create animations on their mobile phone using the stop-motion animation technique. This application uses facilities of the MUSE-auxiliary interface to enable image capture on a mobile phone, and save images on the device. It uses the service-layer and networking layers to send images and XML representing the animation to a service on the MUSE application server to generate a movie file that the learners can view on any device.

The SMART application was evaluated in two workshops described in this thesis. There were two aspects to the evaluation, firstly what is the usability of the application (meso level) and secondly did the application support a constructionist, collaborative, and contextualised activity (micro level).

The application was usable as the users were able to complete their goals during an authentic learning activity using the application. However one usability problem reported in the first study. This problem resulted because the frame rate used by the SMART application for the animations was immutable, so the users could not control the speed of the movie playback. This caused the users to create jumpy animations. The problem was resolved before the second workshop, and no problems in this respect were reported in the second study.

The second aspect examined in this study is can the MCSCL application support constructionist, contextualised and collaborative activities. In brief from a constructionist perspective all the groups from the two studies were able to produce an artifact that they could present to the other participants in the workshop. When asked how they would improve their animation, three responses stand out: more frames, better planning and storyboard, and steady camera. These responses suggest that the participants had updated their understanding of animation and would approach the task differently in future.

During the activity, collaboration was observed in all the teams. The typical approach was for one person to concentrate on capturing the images while the other group members created the characters and the backgrounds and sets to use in the animation. In addition, the most popular aspect of the activity with the participants was working in teams, which can be taken as a synonym for collaboration in this context. In terms

of contextualisation, this was noted by the fact that the participants made use of the computers and printers in their surroundings to create characters and backgrounds for their animation. The other aspect is one group made used of the physical environment as a backdrop to the story they were telling with the animation, in this case a mural on the wall provided the setting for their story.

In summary, the SMART MCSCL application achieved it's design goals and validated the practical function of the MUSE middleware during an authentic goal-orientated mobile collaborative learning activity. In addition the second research question asked in this thesis was addressed by the usability and learning studies using SMART highlight that MUSE can positively support a MCSCL application that enables constructionist, collaborative and contextualised learning activity.

7.1.5 Digital Narrative Tool

The second application described was the digital narrative tool (DNT), which is a more fully featured groupware application for digital video production. The DNT works on mobile devices and PCs to support users engaged in the mobileDNA [Arnedillo-Sanchez, 2008, Arnedillo-Sanchez and Tangney, 2006]. This application also makes use of the MUSE platform and services that are provided on the MUSE application server, including services to manage shared workspaces, generate video files and provide communication mechanisms. One aspect of the DNT is that it is designed to address some of the difficulties experienced by the users engaged in the mobileDNA. Therefore the DNT includes shared workspaces for editing the mind-map, storyboard, and timeline to help maintain a shared understanding between users. It automatically manages file transfers and other infrastructures aspects of the process to simplify managing media files, and includes awareness mechanisms and chat communication tools to improve communication between users.

The DNT was evaluated in a small pilot study that examined the usability of the application and whether it could support a constructionist, collaborative, and contextualised learning activity, in this case an activity to create a digital narrative. The evaluation found that from a constructionist and collaboration aspects the participants were able to create a digital narrative movie in the three hours time allowed for the workshop. Furthermore they collaborated to create the mind-map to generate

ideas for their story, to create the storyboard to refine their story and to capture the images for the movie. They also co-operated to record the soundtrack and edit the movie simultaneously. From a contextualised perspective they used buildings in their surroundings of Dublin city centre to highlight aspects of their story.

This thesis was also interested in whether the DNT application can solve some of the problems experiences by users with the mobileDNA process. Due to the nature of the small pilot study it was only possible to find evidence to support the case that it appears to eliminate the problem with managing media files.

Like SMART the DNT helped to successfully validate the functioning of the MUSE platform during a collaborative learning activity as the participants were able to achieve their goals in the time allowed using the software built on the platform. The pilot study of the DNT pointed towards future research to examine the extent that the application could resolve the difficulties experienced with the mobileDNA process. The next section will concentrate on future research and development opportunities with the MUSE platform and MCSCL applications build in the MUSE platform. Furthermore this study underlines that MUSE can positively support another MCSCL application that enables constructionist, collaborative and contextualised learning activity

7.2 Future Research And Development Work

There are several areas for future research and development resulting from this work. In this section, they are divided into the following four categories: evaluation and data collection; optimisation; new feature development; and new MCSCL application development. This section will describe each in turn starting with further evaluation and data collection.

A priority for more evaluation and data collection is the DNT. Up to this point it has been the subject of a small pilot study as described in this thesis. However a larger workshop involving more participants that more fully approximates the mobileDNA, would be useful in order to provide additional support for the hypothesis that MUSE and the DNT can resolved the difficulties experienced by users engaged in the existing mobileDNA process. This workshop would also collect additional usability data that would inform the design of future iterations of the mobile and desktop DNT

applications.

An unanswered question uncovered during this thesis is what is the maximum mobile client applications that can be supported on the MUSE platform? The current evaluation found that five of PC client applications is the maximum number that can connect to the MUSE platform and continue to receive adequate performance, according to the metrics and constraints defined in this thesis. For mobile clients, it was found that the system could easily support four mobile clients according to the same terms of reference. However, due to lack of available equipment, it was impossible to test this metric to failure in order to find the maximum number of clients that the MUSE platform could support.

During the testing and performance evaluation of the MUSE platform a few bottlenecks were discovered including, the base64 encoding algorithm and the `Responder` server component. Removing these bottlenecks would improve the responsiveness and stability of the MUSE platform and the MCSCL applications. Improvements in the MUSE platform should be bench-marked against the performance metrics identified earlier to find the characteristics of the new system to find how many concurrent mobile or PC clients that the platform can support. Furthermore this would reveal the timeliness guarantees of the new platform in order to support MCSCL developers using the MUSE platform.

Improving and optimising performance of the middleware must be a priority for any new development on the MUSE platform. The major obstacle to improving performance of MUSE is the encoding algorithm used to convert binary media files into base64 data so that it can be sent over the network. Improving or replacing this encoder would provide a significant boost to performance as marshalling using the current encoder consumes over eighteen percent of the time to transmit a 100KB message over TCP/IP.

A second approach to improving performance, again relating to file transfers over the network, is to cache media files once the are converted to base64 format. The message would only need to be encoded once on the server, therefore saving almost half the protocol overhead when transferring files. The server would only incur this marshalling penalty once only, and all clients requesting the file would benefit each time a file requested.

The final bottleneck in the MUSE platform is a dependency on a database look-up

by the responder component on the server for every reply. This can be improved by caching and performing the database look-ups in a background batch process. Removing such bottlenecks should have a significant impact on the MCSCL applications that use the platform. The MUSE platform would also benefit from some additional development. The main area for new development would be to add support for additional networking protocols in the MUSE platform implementation. Furthermore integrating and expanding MUSE with 3rd party services would be an interesting area of research, for example automatically adding animations or digital narratives to YouTube. The final area of future work would be to encourage other developers of mobile learning software to create MCSCL applications using the MUSE platform.

7.3 Concluding Remarks

In conclusion, this thesis set out to address two main research questions. The first question is: what are the functional requirements for a platform to support MCSCL applications on both mobile and PCs? The second question is: do the functional requirements for a MCSCL platform captured by MUSE, SMART and the DNT support constructionist, contextualised and collaborative activities for learning?

To address these questions this thesis started with a review of the literature in CSCW, CSCL, MCSCL, mobile learning and the pedagogy relevant to CSCL and mobile learning. Then it described the design and development of the MUSE platform to support MCSCL applications for digital video production applications. Furthermore, the MUSE design and implementation was compared with the related work on mobile middlewares and found to be the only system that supported the aggregate requirements established by the earlier literature review. Finally, the design and implementation of two MCSCL applications for digital video production where described, SMART and the DNT.

To evaluate MUSE, performance metrics were collected and found that it could support four simultaneously connected desktop MCSCL applications or at least four simultaneously connected mobile MCSCL applications. To validate the function of the middleware in an authentic context two MCSCL applications for digital video production were built on the platform. These applications where found to be usable in practice and

supported two constructionist, contextualised and collaborative activities for animation and digital narrative production.

Future development on the MUSE platform should aim to increase its performance by removing bottlenecks relating to marshalling and demarshalling messages, and some inefficiencies in the design of the MUSE application server. Future directions for research should concentrate on a larger scale evaluation of the DNT. A further avenue of research would involved studying the ease of use of the platform from the perspective of developers creating new MCSCL applications using the MUSE.

Bibliography

Mark S. Ackerman. The intellectual challenge of cscw: The gap between social requirements and technical feasibility. *Human-Computer Interaction*, 15(2/3):179–203, 2000. URL `http://www.eecs.umich.edu/~ackerm/pub/00a10/hci.final.pdf`.

Vikas Agarwal, Sunil Goyal, Sumit Mittal, and Sougata Mukherjea. Mobivine: a middleware layer to handle fragmentation of platform interfaces for mobile applications. In *Middleware '09: Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*, pages 1–10, New York, NY, USA, 2009. Springer-Verlag New York, Inc.

B. Alexander. Going nomadic: Mobile learning in higher education. *EDUCAUSE Review*, 39(5):28–35, September 2004 2004. URL `http://www.educause.edu/ir/library/pdf/erm0451.pdf`.

Nikos Anerousis and Ajay Mohindra. The software-as-a-service model for mobile and ubiquitous computing environments. In *Mobile and Ubiquitous Systems: Networking & Services, 2006 Third Annual International Conference on*, pages 1–6, July 2006. doi: 10.1109/MOBIQ.2006.340383.

Lynda M. Applegate, Benn R. Konsynski, and J. F. Nunamaker. A group decision support system for idea generation and issue analysis in organization planning. In *CSCW '86: Proceedings of the 1986 ACM conference on Computer-supported cooperative work*, pages 16–34, New York, NY, USA, 1986. ACM. ISBN 1-23-456789-0. doi: http://doi.acm.org/10.1145/637069.637073.

Inmaculada Arnedillo-Sanchez. The mobile digital narrative tool. In Inmaculada Arnedillo-Sanchez and Pedro Isaias, editors, *IADIS Mobile Learning 2008*, pages 77–83, Lisbon, Portugal, 2008. IADIS Press.

Inmaculada Arnedillo-Sanchez. *TO BE ANNOUNCED*. Phd thesis, Trinity College Dublin, 2009.

Inmaculada Arnedillo-Sanchez and Brendan Tangney. Mobile technology: Towards overcoming technology & time constraints in digital video production. In *IADIS International Conference on Mobile Learning*, pages 256 – 259, Trinity College Dublin, July 2006 2006. ISBN 972-8924-15-1. URL N/A.

M. Arrigo, O. Di Giuseppe, G. Fulantelli, M. Gentile, G. Novara, L. Seta, and D. Taibi. A collaborative mlearning environment. In Austin Norman and Jon Pearce, editors, *mLearn*, pages 13–21, Melbourne, Australia, 16-19 October 2007 2007. University of Melbourne. ISBN 978 0 7340 3893 7. URL http://dspace.edna.edu.au/dspace/bitstream/2150/41669/1/A%20Collaborative%20mLearning%20Environment.pdf.

Ann-Sofie Axelsson, Asa Abelin, and Ralph Schroeder. Communication in virtual environments: Establishing common ground for a collaborative spatial task, tuesday, october 7 2003. URL http://www.mot.chalmers.se/dept/tso/axelsson.pdf.

M. J. Baker and K. Lund. Flexibly structuring the interaction in a cscl environment. *Proceedings of the European Conference on Artificial Intelligence in Education*, 0: 401–407, 1996. URL http://newhttpd.vjf.cnrs.fr/umr8606/FichExt/mbaker/publications/ArticlesBakerPDF/1996/1996EtAl-b.pdf.

Aaron Bangor, Philip T. Kortum, and James T. Miller. An empirical evaluation of the system usability scale. *International Journal of Human-Computer Interaction*, 24 (6):574–594, 2008. doi: 10.1080/10447310802205776. URL http://dx.doi.org/10.1080/10447310802205776.

Michel Beaudouin-Lafon and Alain Karsenty. Transparency and awareness in a real-time groupware system. In *Proceedings of the 5th annual ACM symposium on User interface software and technology*, 5th Annual ACM Symposium on User Interface Software and Technology, pages 171–180, Monteray, USA, 1992. ACM Press. ISBN 0-89791-549-6. URL http://delivery.acm.org/10.1145/150000/142646/p171-beaudouin-lafon.pdf?key1=142646&key2=8190107711&coll=portal&dl=ACM&CFID=20421862&CFTOKEN=63842174.

Judith Bell. *Doing your Research Project: A guide for first-time researchers in education, health, and social science.* Open University Press, 2005.

Paolo Bellavista, Jiang Xie, and Tuna Tugcu. Recent advances in mobile middleware for wireless systems and services. *Mobile and Network Applications.*, 14(1):1–3, 2009. ISSN 1383-469X. doi: http://dx.doi.org/10.1007/s11036-008-0118-5.

Steve Benford and Lennart E. Fahl. Viewpoints, actionpoints and spatial frames for collaborative user interfaces. In *People and computers IX*, pages 409–423, Glasgow, 1994. Cambridge University Press. ISBN 0-521-48557-6. URL `http://www.crg.cs.nott.ac.uk/research/publications/papers/HCI94.pdf`. 211417.

N. Blum, I. Boldea, T. Magedanz, and T. Margaria. Service-oriented access to next generation network from service creation to execution. *Mobile networks and applications*, 15(3):356–365, 2010.

Jan Bosch, Stefan Friedrichs, Stefan Jung, Johannes Helbig, and Alexander Scherdin. Service orientation in the enterprise. *Computer*, 40(11):51–56, 2007. ISSN 0018-9162. 1320642.

Lauren Bricker, Steven Tanimoto, and Earl Hunt. Effects of cooperative interactions on verbal communication, 1998. URL `http://citeseer.ist.psu.edu/cache/papers/cs/9735/http:zSzzSzwww.hitl.washington.eduzSzpeoplezSzbrickerzSzpubszSzchi99.pdf/bricker98effects.pdf`.

J. Brooke. Sus: A quick and dirty usability scale. In P. W. Jordan, B. Weerdmeester, A. Thomas, and I. L. Mclelland, editors, *Usability evaluation in industry.* Taylor and Francis, London, 1996.

John Steely Brown, Allan Collins, and Paul Duguid. Situated cognition and the culture of learning. *Educational Researcher*, 18(1):32, 1989. URL `http://edr.sagepub.com/cgi/reprint/18/1/32`.

David Buckingham. *Media Education: Literacy, Learning and Contemporary Culture.* Polity Press., Cambridge, 2003. ISBN 0745628303. URL `http://www.amazon.com/Media-Education-Literacy-Learning-Contemporary/dp/0745628303`.

David Buckingham, I. Harvey, and J. Sefton-Green. The difference is digital: digital technology and student media production. *Convergence: The Journal of Research into New Media Technologies*, 5(4):10–21, 1999. URL http://con.sagepub.com/cgi/reprint/5/4/10.

Glen Bull, Gina Bull, Joe Garofalo, and Judi Harris. Grand challenges: Preparing for the technological tipping point. *Learning and Leading with Technology*, 29(8):6–12, 2002. URL http://www.iste.org/Content/NavigationMenu/Publications/LL/LLIssues/Volume_29_2001_2002_/May15/Bonus_Feature_Grand_Challenges_Vignettes.htm.

Kevin Burden and Theo Kuechel. Evaluation report of the teaching and learning with digital video assets pilot 2003-2004. Technical report, Becta, March 2004 2004. URL http://www.becta.org.uk/page_documents/research/evaluation_dv_assets03.pdf.

Michael D. Byrne, Richard Catrambone, and John T. Stasko. Evaluating animations as student aids in learning computer algorithms. *Computers & Education*, 33(4):253–278, 1999. URL http://www.sciencedirect.com/science/article/B6VCJ-4078M0H-3/2/9219d00e0d8a43e0535508cae730b6a3.

P. Byrne and B. Tangney. *SMART: Stop-Motion Animation and Reviewing Tool*, page in press. IGI Global, 2009.

Peter Byrne and Brendan Tangney. Animation on mobile phones. In Pedro Isaias, Piet Kommers, and Inmaculada Arnedillo-Sanchez, editors, *IADIS International Conference on Mobile Learning*, pages 365–369, Trinity College Dublin, July 2006 2006. ISBN 972-8924-15-1. URL http://www.iadis.net/dl/Search_list_open.asp?code=2981.

Peter Byrne and Brendan Tangney. Works in progress: Mobile learning, animation on mobile phones. *IEEE Distributed Systems Online*, 8(Issue 7):1, 2007a. ISSN ISSN: 1541-4922. URL http://dsonline.computer.org/portal/site/dsonline/menuitem.9ed3d9924aeb0dcd82ccc6716bbe36ec/index.jsp?&pName=dso_level1&path=dsonline/2007/07&file=o7003wip.xml&xsl=article.xsl&.

Peter Byrne and Brendan Tangney. Smart - an application to support animation on mobile devices. In Inmaculada Arnedillo-Sanchez, Mike Sharples, and Giasemi Vavoula, editors, *The CSCL Alpine Rendez-Vous*, pages 66–67, Villars, Switzerland, 2007 2007b. Published by Trinity College Dublin Press in association with Environmental Publications with sponsoship by Kaleidoscope. ISBN 1871408482. URL `http://mlearning.noe-kaleidoscope.org/repository/Beyond%20Mobile%20Learning%20Book%20Proceedings%2011.1.07.pdf`. NOTE: link to pdf is full proceedings.

Peter Byrne, Inmaculada Arnedillo-Sanchez, and Brendan Tangney. A mobile computer supported collaborative learning tool for digital narrative production. In John Traxler, Brendan Riordan, and Chris Dennet, editors, *Proceedings of the MLearn2008 Conference: The Bridge From Text To Context*, pages 66–73, University of Wolverhampton, UK, 07 October 2008 2008. ISBN 9780956027207.

Renan G. Cattelan and Maria da Grace Pimentel. Supporting multimedia capture in mobile computing environments through a peer-to-peer platform. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 1649–1650, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-753-7. doi: http://doi.acm.org/10.1145/1363686.1364076.

Y. S. Chen, T. C. Kao, and J. P. Sheu. A mobile learning system for scaffolding bird watching learning. *Journal of Computer Assisted Learning*, 19(3):347–359, 2003. Blackwell Synergy.

John P. Chin, Virginia A. Diehl, and Kent L. Norman. Development of an instrument measuring user satisfaction of the human-computer interface. In *CHI '88: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 213–218, New York, NY, USA, 1988. ACM. ISBN 0-201-14237-6. doi: http://doi.acm.org/10.1145/57167.57203.

Elizabeth F. Churchill, Jonathan. Trevor, Sara Bly, Les Nelson, and Davor Cubranic. Anchored conversations: chatting in the context of a document. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 454–461, 2000. URL `http://www.fxpal.com/publications/FXPAL-PR-00-073.pdf`.

M. Cinque, D. Cotroneo, C. di Flora, A. Migliaccio, and S. Russo. Esperanto: a middleware platform to achieve interoperability in nomadic computing domains. In *Computer Systems and Applications, 2005. The 3rd ACS/IEEE International Conference on*, pages 106–, 2005. doi: 10.1109/AICCSA.2005.1387095.

Gill Clayton. Using avio in the classroom dfes best practice research scholarship. Technical report, BFI Education, 2002. URL `http://www.bfi.org.uk/education/research/teachlearn/digied/`.

A. Cockburn and S. Jones. Four principles of groupware design. *Interacting with Computers*, 7(2):195–210, 1995.

L. Cohen, L. Manion, and K.R.B. Morrison. *Research methods in education*. Psychology Press, 2007. ISBN 0415368782.

Helen Cole and Danae Stanton. Designing mobile technologies to support co-present collaboration. *Personal and Ubiquitous Computing*, 7(6):365 – 371, Dec 2003 2003. ISSN 1617-4909 or 1617-4917 (online). URL `http://www.springerlink.com/media/gpt919xutq4ywj6kducy/contributions/b/c/5/y/bc5ypbjx27jeq031.pdf`.

Vanessa Colella, Richard Borovoy, and Mitchel Resnick. Participatory simulations: using computational objects to learn about dynamic systems. In *Human Factors in Computing Systems*, pages 9–10, Los Angeles, USA, 18-23 April 1998 1998. ACM Press. ISBN 1-58113-028-7. URL `http://delivery.acm.org/10.1145/290000/286503/p9-colella.pdf?key1=286503&key2=1526492811&coll=GUIDE&dl=ACM&CFID=26793466&CFTOKEN=33122526`.

A. Collins, P. Neville, and K. Bielaczyc. The role of different media in designing learning environments. *International Journal of Artificial Intelligence in Education*, 11:144–162, 2000. URL `http://aied.inf.ed.ac.uk/members00/archive/vol_11/collins/paper.pdf`.

Maria de los Angeles Constantino-Gonzalez and Daniel D. Suthers. Coaching collaboration in a computer-mediated learning environment. In *Computer support for collaborative learning: foundations for a CSCL community*,

pages 583–586, 2002. URL `http://lilt.ics.hawaii.edu/lilt/papers/2002/Constantino-Suthers-CSCL-2002.pdf`.

Domenico Cotroneo, Armando Migliaccio, and Stefano Russo. The esperanto broker: a communication platform for nomadic computing systems. *Software: Practice and Experience*, 37(10):1017–1046, 2007. doi: 10.1002/spe.794. URL `http://dx.doi.org/10.1002/spe.794`.

C. Crook. *Computers and the Collaborative Experience of Learning*. Routledge, 1996.

Arman Danesh, Kori M. Inkpen, Felix Lau, Kieth Shu, and Kellogg Booth. Geney: Designing a collaborative activity for the palm handheld computer. In *SIGCHI Conference on Human Factors in Computing Systems*, volume 3, pages 388–395. ACM Press, March 2001 2001. URL `http://www.ece.ubc.ca/~elec418/resources/geney.pdf`. SIGCHI: ACM Special Interest Group on Computer-Human Interaction.

A. H. Davis, C. Sun, and J. Lu. Collaborative editing for xml documents: An operational transformation approach. In *Third Annual Collaborative Editing Workshop, ACM Group*, 2001.

A. H. Davis, C. Sun, and J. Lu. Generalizing operational transformation to the standard general markup language. In *SCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 58–67. ACM Press New York, NY, USA, 2002.

W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on information Theory*, 22(6):644–654, 1976.

Pierre Dillenbourg. What do you mean by collaborative learning. In *Collaborative learning: Cognitive and Computational Approaches*, pages 1–19. Oxford: Elsevier, 1999. ISBN 0-08-043073-2. URL `http://www2.hawaii.edu/~pdonohue/Courses/Dillenbourg_99_opt.pdf`.

Angelique Dimitracopoulou. Designing collaborative learning systems: current trends & future research agenda. In *Computer Support for Collaborative Learning: Learning 2005: The Next 10 Years!*, pages 115–124, Taipei, Taiwan, 2005. International Society

of the Learning Sciences. ISBN 0-8058-5782-6. URL http://delivery.acm.org/10.1145/1150000/1149309/p115-dimitracopoulou.pdf?key1=1149309&key2=6960247711&coll=GUIDE&dl=GUIDE&CFID=17234557&CFTOKEN=98800785. 1149309.

Paul Dourish and Victoria Bellotti. Awareness and coordination in shared workspaces. In *ACM Conference on Computer Supported Cooperative Work*, pages 107–114, Toronto, Ontario, Canada, 1992. ACM Press. ISBN 0-89791-542-9. URL http://delivery.acm.org/10.1145/150000/143468/p107-dourish.pdf?key1=143468&key2=7643236711&coll=portal&dl=ACM&CFID=16248777&CFTOKEN=56953727. 143468.

C. A. Ellis, S. J. Gibbs, and G. L. Rein. Groupware: Some issues and experiences. *Communications of the ACM*, 34(1):38–58, 1991. URL http://delivery.acm.org/10.1145/100000/99987/p39-ellis.pdf?key1=99987&key2=2929836711&coll=GUIDE&dl=GUIDE&CFID=16312417&CFTOKEN=40150179.

Kevin Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, New York, NY, USA, 2003. ACM. ISBN 1-58113-735-4. doi: http://doi.acm.org/10.1145/863955.863960.

U. Farooq, W. Schafer, M.B. Rosson, and J.M. Carroll. M-education: bridging the gap of mobile and desktop computing. In *Proceedings IEEE International Workshop on Wireless and Mobile Technologies in Education*, pages 91–94. IEEE Computer Society Washington, DC, USA, 2002.

S. Fish, Robert, E. Kraut, Robert, and D. P. Leland, Mary. Quilt: a collaborative tool for cooperative writing. In *ACM SIGOIS and IEEECS TC-OA Conference on Office information systems*, pages 30–37, Palo Alto, California, United States, 1988. ACM Press. ISBN 0-89791-261-6. URL http://delivery.acm.org/10.1145/50000/45414/p30-fish.pdf?key1=45414&key2=3903912811&coll=GUIDE&dl=ACM&CFID=25918791&CFTOKEN=42036407. 45414.

D.S. Fraser, H. Smith, E. Tallyn, D. Kirk, S. Benford, D. Rowland, M. Paxton, S. Price, and G. Fitzpatrick. The sense project: a context-inclusive approach to

studying environmental science within and across schools. *Proceedings of th 2005 conference on Computer support for collaborative learning: learning 2005: the next 10 years!*, pages 155–159, 2005.

Abdulbaset Gaddah and Thomas Kunz. A survey of middleware paradigms for mobile computing. Technical report, Department of Systems and Computer Engineering, Carleton University, Ottawa, Ontario, Canada, 2003.

Gartner. Gartner says worldwide sales of camera phones will reach nearly 300 million in 2005. Technical report, Gartner, 2005. URL `http://www.gartner.com/press_releases/asset_141163_11.html`.

Gartner. Gartner says strong results in asia/pacific and japan drove worldwide mobile phone sales to 14 percent growth in the first quarter of 2007. Technical report, Gartner Research, 2007. URL `http://www.gartner.com/it/page.jsp?id=506573`.

David Gelernter. Generative communication in linda. *ACM Trans. Program. Lang. Syst.*, 7(1):80–112, 1985. ISSN 0164-0925. doi: http://doi.acm.org/10.1145/2363.2433.

Hans Geser. Towards a sociological theory of the mobile phone. Online, May 2004 2004. URL `http://socio.ch/mobile/t_geser1.pdf`.

J. J. Gibson. The theory of affordances. In R. Shaw and J. Bransford, editors, *Perceiving, Acting, and Knowing*. Erlbaum, New Jersey, 1977.

Pablo Gotthelf, Alejandro Zunino, and Marcelo Campo. *Groupware: Design, Implementation, and Use*, volume 4715 of *Lecture Notes in Computer Science*, chapter A Decentralized Middleware for Groupware Applications, pages 191–206. Springer Berlin / Heidelberg, August 2007. doi: 10.1007/978-3-540-74812-0_15. URL `http://www.springerlink.com/content/r5578t7p7152713p`.

J. Grudin. Timelines: CSCW: time passed, tempest, and time past. *interactions*, 17 (4):38–40, 2010. ISSN 1072-5520.

Jonathan Grudin. Computer-supported cooperative work: history and focus. *Computer*, 27(5):19–26, 1994. doi: 10.1109/2.291294. URL `http://dx.doi.org/10.1109/2.291294`.

Axel Guicking and Thomas Grasse. *Groupware: Design, Implementation, and Use*, volume 4154 of *Lecture Notes in Computer Science*, chapter A Framework Designed for Synchronous Groupware Applications in Heterogeneous Environments, pages 203–218. Springer Berlin / Heidelberg, 2006. doi: 10.1007/11853862_17. URL `http://www.springerlink.com/content/66r7843778224198`.

Axel Guicking, Peter Tandler, and Paris Avgeriou. *Groupware: Design, Implementation, and Use*, volume 5411 of *Lecture Notes in Computer Science*, chapter Agilo: A Highly Flexible Groupware Framework, pages 49–56. Springer Berlin / Heidelberg, 1 edition, 2008. doi: 10.1007/11560296_4. URL `http://www.springerlink.com/content/85nvcfb9bkmv9g3e`.

F. Guidec and Y. Mahéo. Opportunistic content-based dissemination in disconnected mobile ad hoc networks. In *International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2007)*, pages 49–54, 2007.

A. Gupta, A. Kalra, D. Boston, and C. Borcea. Mobisoc: a middleware for mobile social computing applications. *Mobile Networks and Applications*, 14(1):35–52, 2009.

Carl Gutwin and Saul Greenberg. Design for individuals, design for groups: tradeoffs between power and workspace awareness. In *ACM Conference on Computer Supported Cooperative Work*, pages 207–216, Seattle, USA, 1998a. ACM Press. ISBN 1-58113-009-0. URL `http://delivery.acm.org/10.1145/290000/289495/p207-gutwin.pdf?key1=289495&key2=4625247711&coll=GUIDE&dl=ACM&CFID=17239978&CFTOKEN=26789573`. 289495.

Carl Gutwin and Saul Greenberg. Effects of awareness support on groupware usability. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 511–518, New York, NY, USA, 1998b. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-30987-4. doi: http://doi.acm.org/10.1145/274644.274713.

Carl Gutwin and Saul Greenberg. A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work*, 11 (3):411–446, 2002. URL `http://grouplab.cpsc.ucalgary.ca/papers/2002/02-AwarenessFramework.JCSCW/awareness_framework.jcscw.pdf`. Springer.

Carl Gutwin, Mark Roseman, and Saul Greenberg. A usability study of awareness widgets in a shared workspace groupware system. In *Proceedings of ACM Conference on Computer Supported Collaborative Learning 1996*, ACM Conference on Computer Supported Collaborative Learning, Boston, Massachusetts, United States, 1996. ACM Press. ISBN 0-89791-765-0. URL `http://delivery.acm.org/10.1145/250000/240298/p258-gutwin.pdf?key1=240298&key2=6012236711&coll=portal&dl=ACM&CFID=16248777&CFTOKEN=56953727`. 240298.

Salem Hadim, Jameela Al-Jaroodi, and Nader Mohamed. Trends in middleware for mobile ad hoc networks. *The Journal of Communications*, 1(4):11–21, 2006.

Perttu Hamalainen, Mikko Lindholm, Ari Nykanen, and Johanna Hoysniemi. Animaatiokone: an installation for creating clay animation. In *CHI*, pages 24–29, Vienna, Austria, April 2004 2004. URL `http://delivery.acm.org/10.1145/990000/985695/p17-hamalainen.pdf?key1=985695&key2=8183534411&coll=GUIDE&dl=ACM&CFID=68902885&CFTOKEN=85185410`.

Mark Handel and James D. Herbsleb. What is chat doing in the workplace? In *ACM Conference on Computer Supported Cooperative Work*, New Orleans, Louisiana, USA, 2002. ACM Press. URL `http://delivery.acm.org/10.1145/590000/587080/p1-handel.pdf?key1=587080&key2=6102707711&coll=GUIDE&dl=GUIDE&CFID=20481516&CFTOKEN=73733411`. 587080 1-10.

S. Helmer, C. C. Kanne, and G. Moerkotte. Evaluating lock-based protocols for cooperation on xml documents. *SIGMOD Record*, 33(1):58, 2004.

Mark Hofer and Kathleen Owings Swan. Digital moviemaking - the harmonization of technology, pedagogy and content. *International Journal of Technology in Teaching and Learning & Leading with Technology*, 1(2):102–110, 2005 2005. URL `http://ijttl.sicet.org/issue0502/Hofer.Vol1.Iss2.pdf`.

Wu-Yuin Hwang, Jung-Lung Hsu, and Hui-Ju Huang. A study on ubiquitous computer supported collaborative learning with hybrid mobile discussion forum. In Austin Norman and Jon Pearce, editors, *mLearn*, pages 101–109, Melbourne, Australia, 16-19 October 2007 2007. University of Melbourne. ISBN 978 0 7340 3893 7. URL `Notavailable-seemlearn2007doc`.

C. Ignat and M. C. Norrie. Tree-based model algorithm for maintaining consistency in real-time collaborative editing systems. In *The Fourth International Workshop on Collaborative Editing Systems, CSCW 2002*, 2002.

Kori M. Inkpen. Designing handheld technologies for kids. *Personal Technologies*, 3(3(1&2)):81–89, 1999. URL `http://www.cs.sfu.ca/people/Faculty/inkpen/Papers/hcscw_inkpen.pdf`.

M. Ionescu and I. Marsic. Tree-based concurrency control in distributed groupware. *Computer Supported Cooperative Work*, 12(3):329–350, 2003. Springer.

H. Ishii. Teamworkstation: towards a seamless shared workspace. In *ACM Conference on Computer Supported Cooperative Work*, pages 13–26, Los Angeles, USA, 1990. ACM Press. http://doi.acm.org/10.1145/99332.99337.

ISO. Guidance on usability specification and measures. iso, cd 9241-11, 1992., 1998.

J. G. M. Jaspers, G. Erkens, and G. Kanselaar. Cosar: collaborative writing of argumentative texts. In *Second IEEE International Conference on Advanced Learning Technologies (ICALT'01)*, pages 269–272, 2001.

Patrick Jermann, Amy Soller, and Martin Muehlenbrock. From mirroring to guiding: A review of the state of art technology for supporting collaborative learning. In Pierre Dillenbourg, A. Eurelings, and K. Hakkarainen, editors, *European Perspectives on Computer-Supported Collaborative Learning*, pages 324–331, Maastricht, 2001 2001. URL `http://infoscience.epfl.ch/getfile.py?docid=1568&name=197&format=pdf&version=1`.

Robert Johansen, David Sibbet, Suzyan Benson, Alixia Martin, Robert Mittman, and Paul Saffo. *Leading Business Teams: How Teams Can Use Technology and Group Process Tools to Enhance Performance*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1991. ISBN 0201528290.

Matthew Kearney and Sandy Schuck. Students in the director's seat: Teaching and learning with student-generated video. In Piet Kommers and Griff Richards, editors, *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pages 2864–2871, Montreal, Canada, 2005.

AACE. URL http://www.aace.org/newdl/index.cfm/files/paper_20518.pdf?fuseaction=Reader.DownloadFullText&paper_id=20518.

David S. Kerr and Uday S. Murthy. The effectiveness of group support systems for negotiation versus idea-generation tasks: An experimental investigation, 1st - 3rd July 2004 2004. URL http://dsslab.infotech.monash.edu.au/dss2004/proceedings/pdf/40_Kerr_Murthy.pdf.

Jurek Kirakowski and Mary Corbett. Sumi: the software usability measurement inventory. *British Journal of Educational Technology*, 24:210–212, 1993. doi: 10.1111/j.1467-8535.1993.tb00076.x. URL http://dx.doi.org/10.1111/j.1467-8535.1993.tb00076.x.

Gerd Kortuem. Proem: a middleware platform for mobile peer-to-peer computing. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4):62–64, 2002. ISSN 1559-1662. doi: http://doi.acm.org/10.1145/643550.643557.

J. Lave and E. Wenger. *Situated Learning: legitimate peripheral participation*. Cambridge University Press, 1991.

Mary D. P. Leland, Robert S. Fish, and Robert E. Kraut. Collaborative document production using quilt. In *CSCW '88: Proceedings of the 1988 ACM conference on Computer-supported cooperative work*, pages 206–215, New York, NY, USA, 1988. ACM. ISBN 0-89791-282-9. doi: http://doi.acm.org/10.1145/62266.62282.

Jacques Lonchamp. Supporting synchronous collaborative learning: A generic, multi-dimensional model. *International Journal of Computer-Supported Collaborative Learning*, 1(2):247–276, 2006. URL http://dx.doi.org/10.1007/s11412-006-8996-7. 10.1007/s11412-006-8996-7.

M. Madden, P.W.H. Chung, and C.W. Dawson. The effect of a computer-based cartooning tool on children's cartoons and written stories,. *Computers & Education*, 51(2):900–925, September 2008 2008. URL http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6VCJ-4R1186T-1-15&_cdi=5956&_user=103681&_orig=na&_coverDate=09%2F30%2F2008&_sk=999489997&view=c&wchp=dGLzVlz-zSkzS&md5=8fafc6952aeb0c87f565edafdc84a41d&ie=/sdarticle.pdf.

Jackie Marsh and Philippa Thompson. Parental involvement in literacy development: using media texts. *Journal of Research in Reading*, 24(3):266–278, 2001. ISSN 1467-9817. URL `http://dx.doi.org/10.1111/1467-9817.00148`. 10.1111/1467-9817.00148.

Cecilia Mascolo, Licia Capra, and Wolfgang Emmerich. An xml-based middleware for peer-to-peer computing. In *First International Conference on Peer-to-Peer Computing (P2P'01)*, volume 0, page 0069. IEEE Computer Society, Los Alamitos, CA, USA, 2001a. ISBN 0-7695-1503-7. doi: http://doi.ieeecomputersociety.org/10.1109/P2P.2001.990428.

Cecilia Mascolo, Licia Capra, Stefanos Zachariadis, and Wolfgang Emmerich. Xmiddle: A data-sharing middleware for mobile computing. *International Journal on Personal and Wireless Communications*, 21(1):77–103, 2001b. ISSN 0929-6212. doi: http://dx.doi.org/10.1023/A:1015584805733.

Deborah J. Mayhew. The usability engineering lifecycle. In *CHI '99: CHI '99 extended abstracts on Human factors in computing systems*, pages 147–148, New York, NY, USA, 1999. ACM. ISBN 1-58113-158-5. doi: http://doi.acm.org/10.1145/632716.632805.

M. M. McManus and R. M. Aiken. Teaching collaborative skills with a group leader computer tutor. *Education and Information Technologies*, 1(1):75–96, 1996. Springer.

Daniel D. Mittleman, Robert O. Briggs, John Murphy, and Alanah Davis. *Groupware: Design, Implementation, and Use*, volume 5411 of *Lecture Notes in Computer Science*, chapter Toward a Taxonomy of Groupware Technologies, pages 305 – 317. Springer Berlin / Heidelberg, 1 edition, 2008. doi: 10.1007/978-3-540-92831-7_25.

Morgan Stanley Global Technology & Telecom Research. *The Mobile Internet Report*. Morgan Stanley Research, 1st edition, December 2009. doi: 1450700357.

Laura Naismith, Peter Lonsdale, Giasemi Vavoula, and Mike Sharples. Report 11: Literature review in mobile technologies and learning. Technical Report 11, Nesta Futurelab, 2004. URL `http://www.nestafuturelab.org/download/pdfs/research/lit_reviews/futurelab_review_11.pdf`.

Jakob Nielsen. Finding usability problems through heuristic evaluation. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 373–380, New York, NY, USA, 1992. ACM. ISBN 0-89791-513-5. doi: http://doi.acm.org/10.1145/142750.142834.

Jakob Nielsen. *Usability Inspection Methods*, chapter Heuristic Evaluation, pages 25–61. John Wiley & Sons, New York, NY, 1994.

D. A. Norman. *Things that make us smart*. Addison-Wesley Pub. Co Reading, Mass, 1993.

M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenges. *Computer*, 40(11):38–45, 2007. ISSN 0018-9162.

Bryan Patten, Inmaculada Arnedillo-Sanchez, and Brendan Tangney. Designing collaborative, constructionist and contextual applications for handheld devices. *Computers and Education*, 46(3):294–308, August 2006 2006. URL `file:///C:/Documents%20and%20Settings/Peter/My%20Documents/2.%20PhD/Research%20Papers/CAL05BryanPatten.pdf`.

Gian Pietro Picco, Amy L. Murphy, and Gruia-Catalin Roman. Lime: Linda meets mobility. In *ICSE '99: Proceedings of the 21st international conference on Software engineering*, pages 368–377, New York, NY, USA, 1999. ACM. ISBN 1-58113-074-0. doi: http://doi.acm.org/10.1145/302405.302659.

Niels Pinkwart. A plug-in architecture for graph based collaborative modeling systems. In *Shaping the Future of Learning through Intelligent Technologies. Proceedings of the 11th Conference on Artificial Intelligence in Education*, pages 535–536, 2003. URL `http://www.cs.usyd.edu.au/~aied/vol3/vol3_pinkwart.pdf`.

Thadpong Pongthawornkamol, Klara Nahrstedt, and Guijun Wang. The analysis of publish/subscribe systems over mobile wireless ad hoc networks. In *Mobile and Ubiquitous Systems: Networking & Services, 2007. MobiQuitous 2007. Fourth Annual International Conference on*, pages 1–8, Aug. 2007. doi: 10.1109/MOBIQ.2007.4451024.

Ilona Posner, Ronald Baecker, and Bruce Homer. Children learning filmmaking using multimedia tools. In *ED-MEDIA/ED-TELECOM*, Calgary, Canada., 1997. URL `http://www.kmdi.toronto.edu/rmb/papers/p17.pdf`.

J. Potter. Embodied memory and curatorship in childrens digital video production. *English Teaching: Practice and Critique*, 9(1):22–35, 2010.

K. Premadasa and B. Landfeldt. Dynamically re-configurable communication protocols using key identifiers. In *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*, pages 471–473, 2005.

Jiten Rama and Judith Bishop. A survey and comparison of cscw groupware applications. In *SAICSIT '06: Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, pages 198–205, , Republic of South Africa, 2006. South African Institute for Computer Scientists and Information Technologists. ISBN 1-59593-567-3. doi: http://doi.acm.org/10.1145/1216262.1216284.

M. Reid, A. Burn, and D. Parker. Evaluation report of the becta digital video pilot project. Technical report, Becta, 2002. URL `http://www.becta.org.uk/research/reports/digitalvideo/`.

Tom Rodden. A survey of cscw systems. *Interacting with Computers*, 3:319–353, 1991. URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.39.8704&rep=rep1&type=pdf`.

Jeremy Roschelle. Keynote paper: Unlocking the learning value of wireless mobile devices. *Journal of Computer Assisted Learning*, 19(3):260–272, 19/5/2003 2003. URL `http://www.blackwell-synergy.com/doi/pdf/10.1111/j.1365-2729.2004.00089.x`.

Jeremy Roschelle and Roy Pea. A walk on the wild side: How wireless handhelds may change cscl. In *ACM Conference on Computer Supported Collaborative Learning*, pages 51–60, Colorado, USA, January

2002 2002. URL http://citeseer.ist.psu.edu/cache/papers/cs/25944/http:zSzzSznewmedia.colorado.eduzSzcsclzSz79.pdf/roschelle02walk.pdf.

Jeremy Roschelle and S. D. Teasley. The construction of shared knowledge in collaborative problem solving. *Computer Supported Collaborative Learning*, 128(S 69):97, 1995.

Mark Roseman and Saul Greenberg. Teamrooms: network places for collaboration. In *ACM Conference on Computer Supported Cooperative Work*, pages 325–333, Boston, Massachusetts, United States, 1996. ACM Press. ISBN 0-89791-765-0. URL http://delivery.acm.org/10.1145/250000/240319/p325-roseman.pdf?key1=240319&key2=6416867711&coll=portal&dl=ACM&CFID=21128052&CFTOKEN=98508325. 240319.

Jorg Roth. Seven challenges for developers of mobile groupware. In *CHI*, Minneapolis., 2002a. URL http://www.informatik.fernuni-hagen.de/import/pi2/paper/chi02ws.pdf.

Jorg Roth. A communication middleware for mobile and ad-hoc scenarios. In *International Conference on Internet Computing (IC'02)*, pages 24–27, 2002b.

Jorg Roth. The resource framework for mobile applications: Enabling collaboration between mobile users. *Proc. ICEIS 03. Volume*, 4:87–94, 2003.

Jorg Roth and Claus Unger. Dreamteam - a platform for synchronous collaborative applications. In *AI Society*, pages 153–165. Springer Verlag, 2000.

Gavriel Salomon. What does the design of effective cscl require and how do we study its effects? *ACM SIGCUE Outlook*, 21(3):62–68, 1992. ISSN 0163-5735. URL http://delivery.acm.org/10.1145/140000/130909/p62-salomon.pdf?key1=130909&key2=9011432811&coll=GUIDE&dl=GUIDE&CFID=26120064&CFTOKEN=82355752. 130909 SIGCUE Outlook 3.

Kjeld Schmidt. Divided by a common acronym: On the fragmentation of cscw. *ECSCW 2009*, pages 223–242, 2009.

Kjeld Schmidt and Tom Rodden. Putting it all together: Requirements for a cscw platform. *Human Factors in Information Technology*, 12:157–176, 1996.

Jermiah Scholl, John McCarthy, and Rikard Harr. A comparison of chat and audio in media rich environments. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 323–332, New York, NY, USA, 2006. ACM. ISBN 1-59593-249-6.

S.D Scott, Regan L. Mandryk, and Kori M. Inkpen. Understanding children's collaborative interactions in shared environments. *Journal of Computer Assisted Learning*, 19(2):220–228, 15 January 2003 2003. URL `http://www.blackwell-synergy.com/doi/pdf/10.1046/j.0266-4909.2003.00022.x`.

Mike Sharples, Peter Lonsdale, Julia Meek, Paul Rudman, and Giasemi Vavoula. An evaluation of myartspace: A mobile learning service for school museum trips. In Austin Norman and Jon Pearce, editors, *mLearn*, pages 238–244, Melbourne, Australia, 19 October 2007 2007. University of Melbourne, Parkville Victoria 3010, Australia. ISBN 978 0 7340 3893 7. URL `http://www.mlearn2007.org/files/mLearn_2007_Conference_Proceedings.pdf`.

Elliot Soloway, Cathleen Norris, Phylis Blumenfeld, Barry Fishman, Joseph Krajcik, and Ron Max. Log on education: Handheld devices are ready-at-hand. *Communications of the ACM*, 44(6):15–20, 2001. ISSN 0001-0782. URL `http://portal.acm.org/ft_gateway.cfm?id=376140&type=pdf&coll=GUIDE&dl=GUIDE&CFID=62015559&CFTOKEN=84397459`. 376140.

Daniel Spikol. Designing mobile games that explore novel learning practices with co-design. In Giasemi Vavoula, A. Kukulska-Hulme, and N. Pachler, editors, *Research Methods in Informal and Mobile Learning (MIL-RM)*, pages 41–47, Institute of Education, University of London, UK, December 2007 2007. WLE Centre. ISBN 1753-3385.

Thomas Springer, Daniel Schuster, Iris Braun, Jordan Janeiro, Markus Endler, and Antonio A. F. Loureiro. A flexible architecture for mobile collaboration services. In *Companion '08: Proceedings of the ACM/IFIP/USENIX Middleware '08 Conference Companion*, pages 118–120, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-369-3. doi: http://doi.acm.org/10.1145/1462735.1462770.

P. Stuedi and G. Alonso. Transparent heterogeneous mobile ad hoc networks. In *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*, pages 237–246, July 2005. doi: 10.1109/MOBIQUITOUS.2005.63.

Chengzheng Sun and Clarence Ellis. Operational transformation in real-time group editors: issues, algorithms, and achievements. In *CSCW '98: Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 59–68. ACM Press New York, NY, USA, 1998. doi: http://doi.acm.org/10.1145/289444.289469.

Colleen Swain, Rachael Sharpe, and Kara Dawson. Using digital video to study history. *Social Education*, 67(3):154–158, 2003. URL `http://www.questia.com/PM.qst?a=o&se=gglsc&d=5001920161`.

Deborah Tatar, Jeremy Roschelle, Phil Vahey, and William R. Penuel. Handhelds go to school:lessons learned. *Computer*, 36(9):30–37, September 2003 2003. doi: 10.1109/MC.2003.1231192. URL `http://ieeexplore.ieee.org/iel5/2/27578/01231192.pdf?arnumber=1231192`.

R. Tinker. A history of probeware. *Stanford University MakingSENS Web site*, 12:2006, 2000. URL `http://www.concord.org/resources/files/probeware_history.pdf`.

Giasemi Vavoula. Evaluating mobile learning, 1-2 Nov 2007 2007.

Giasemi Vavoula and Mike Sharples. Challenges in evaluating mobile learning. In Brendan Dennet Chris Traxler, John Riordan, editor, *Proceedings of the MLearn2008 Conference: The Bridge From Text To Context*, pages 296–303, University of Wolverhampton, UK, 07-10th October 2008 2008. ISBN 9780956027207.

Giasemi Vavoula, Julia Meek, Mike Sharples, Peter Lonsdale, and Paul and Rudman. A lifecycle approach to evaluating myartspace. IEEE Computer Society, 2006. 1194506 18-22.

S. Vygotsky, Lev. *Mind in society : the development of higher psychological processes.* Harvard University Press, Cambridge [Mass.] ; London, 1978. ISBN 0674576284

0674576292 (pbk). by L.S. Vygotsky ; [translated from the Russian]; edited by Michael Cole [et al.]. ill., facsim., port. ; 25 cm.

Greg Wilson, Jean Ostrem, and Christopher Bey. *Programmers Companion*, volume 1. PalmSource, Inc., 2004.

D. Wood, J. S. Bruner, and G. and Ross. The role of tutoring in problem solving. *J Child Psychol Psychiatry*, 17(2):89–100, 1976.

Huahai Yang and M. Olson, Gary. Exploring collaborative navigation: the effect of perspectives on group performance. ACM Press, 2002. URL `http://delivery.acm.org/10.1145/580000/571899/p135-yang.pdf?key1=571899&key2=2451236711&coll=portal&dl=ACM&CFID=19700698&CFTOKEN=65984335`. 571899.

Jose Pablo Zagal, Anne Marie Piper, and Amy S. and Brukman. Kids telling fables through 3d animation. Technical report, Georgia Institute of Technology, 2004 2004. URL `http://hdl.handle.net/1853/3732`.

Gustavo Zurita and Miguel Nussbaum. *An Ad-Hoc Wireless Network Architecture for Face-to-Face Mobile Collaborative Applications*, volume 3894 of *Lecture Notes in Computer Science*, chapter An Ad-Hoc Wireless Network Architecture for Face-to-Face Mobile Collaborative Applications, pages 42–55. Springer Berlin / Heidelberg, 2006. doi: 10.1007/11682127_4. URL `http://dx.doi.org/10.1007/11682127_4http://www.springerlink.com/content/mn232g2225424138/fulltext.pdf`.

Gustavo Zurita and Miguel Nussbaum. A conceptual framework based on activity theory for mobile cscl. *British Journal of Educational Technology*, 38(2):211–235, 2007.

Appendix A

Survey Forms

A.1 SMART Evaluation Questionnaire

Please circle your answer

1. Do you have a mobile phone? Yes or No

   (a) If so, what type of mobile do you have? for example: Nokia N73

2. Does your mobile phone Java a camera? Yes or No

3. Does your mobile phone have Bluetooth? Yes or No

4. Does your mobile phone have Java? Yes or No

5. Do you study Art at school? Yes or No

6. How good are you at Art? Excellent, Good, Average, Not Good

7. Have you made animations before?

8. Have you made videos on mobile phones or digital cameras before?

9. What did you do with these videos? *For example, did you send them to friends or upload them to YouTube*

10. How many hours do you spend using a computer each week?

11. Do you use the computer for any of the following activities? *Please circle all options that you have used.*

(a) YouTube

(b) Facebook

(c) Bebo

(d) MySpace

(e) Flickr

(f) Photo Editing

(g) Movie Editing

(h) Email

(i) Writing

(j) Games

12. What did you like about the animation activity today?

13. What did you not like about the animation activity today?

14. If you had more time, how would you improve your movie?

15. If you were starting again with a new movie, how would you make it better?

16. Which animation did you think was the best today?

17. Did you have any problems with the animation tool?

18. How would you make the animation tool better?

19. If the animation tool were available on you phone, would you use it? yes or No

20. What would you use it for?

A.2 Digital Narrative Tool Evaluation Questionnaire

Please circle your answer

1. Do you have a mobile phone? Yes or No

   (a) If so, what type of mobile do you have? for example: Nokia N73

2. Does your mobile phone have a camera? Yes or No
3. Does your mobile phone have Java? Yes or No
4. Does your mobile phone have Bluetooth? Yes or No
5. Have you made videos on mobile phones or digital cameras before?
6. What did you do with these videos? *For example, did you send them to friends or upload them to YouTube*
7. How many hours do you spend using a computer each week?
8. Do you use the computer for any of the following activities? *Please circle all options that you have used.*
   (a) Video Sharing Site
   (b) Photo Sharing Site
   (c) Social Networking
   (d) Instant Messaging
   (e) Video Editing
   (f) Photo Editing
   (g) Email
   (h) Blogging
   (i) Games
   (j) Writing
9. What did you like about the activity today?
10. What did you not like about the activity today?
11. If you had more time, how would you improve your movie?
12. If the Digital Narrative Tool were available on you phone, would you use it? Yes or No
13. Did you have any problems with the Digital Narrative Tool?

14. How would you make the Digital Narrative Tool better?
15. Did you use the Mind Map of the mobile application?
16. Did you use the storyboard of the mobile application?
17. did you use the Chat tool of the mobile application?

A.3 System Usability Scale (©Digital Equipment Corporation 1986)

The users are asked to start their agreement to the following statements on a five point scale indicating that they either: Strongly agree, agree, neutral, disagree or strongly disagree.

1. I think that I would like to use this system frequently
2. I found the system unnecessarily complex
3. I thought the system was easy to use
4. I think that I would need the support of a technical person to be able to use this system
5. I found the various functions in this system were well integrated
6. I thought there was too much inconsistency in this system
7. I would imagine that most people would learn to use this system very quickly
8. I found the system very cumbersome to use
9. I felt very confident using the system
10. I needed to learn a lot of things before I could get going with this system

Appendix B

MUSE Message Formats And APIs

B.1 Application Programming Interface

B.1.1 MUSEServiceLayer API

```
public void requestServices();
public ArrayList<String> getAvailableServices();
public void register(Service service);
public void unregister(Service service);
public void sendMessage(InternalMessage message);
public void sendMessage(final InternalMessage internalMes-
sage, String address);
public void setUser(Human me);
public Human getUser();
```

B.1.2 MUSEServiceLayerCallback API

```
public void deliverMessage(InternalMessage message);
```

B.1.3 MUSENetworkLayer

```
public void sendMessage(final InternalMessage internalMes-
sage, String address);
public void sendMessage(final InternalMessage internalMes-
```

```
sage);
```

B.1.4 MUSE Auxilary-interface APIs

B.1.4.1 package muse.auxiliary.encoding

```
public interface Encoder {

    public String encode(byte[] data);
    public byte[] decode(String s);
    public String encodeFromURL(String s);

}
```

package muse.auxiliary.security

```
public interface Encrypter {

    public String decrypt(String data);
    public String encrypt(String s);
    public String encryptFromURL(String s);

}
```

package muse.auxiliary.grouping

```
public interface UserList {

    public int getMyID();
    public Human getMe();
    public void setMe(Human human);
    public void getUsers();
    public void goOffline();
    public void logoff();
    public Human getHuman(int userID);
    public Vector getUsersList();
    public Human getHumanByIndex(int index);
    public int getNumberOfUsers();
    public int getIndex(int userID);
```

```
}
```

package muse.auxiliary.lang

```
public interface Queue {

    public abstract void add(final Object o);

    public abstract int count();

    public abstract Enumeration elements();

    public abstract boolean empty();

    public abstract Object front();

    public abstract boolean inQueue(final Object o);

    public abstract Object remove();

}

public interface Sort {

    public int [] sort();

    public int [] sort(Vector vec);

}

public interface Logger {

    public static final int DEBUG = 5;

    public static final int INFO = 4;

    public static final int WARN = 3;

    public static final int ERROR = 2;

    public static final int FATAL = 1;

    public static final int OFF = 0;

    public void debug(String message);

    public void error(String message);

    public void error(Exception exception);

    public void fatal(String message);

    public void info(String message);

    public void warn(String message);

    public void setLevel(int level);

    public void objectState(Object value);
```

```
}
public interface ByteConverter {

    public int toInt(byte[] b);
    public byte[] toBytes(int i);

}
public interface Color {

    public static final int BLACK = 0;
    public static final int DARKBLUE = 0;
    public static final int BLUE = 1;
    public static final int MAUVE = 2;
    public static final int PEACH = 3;
    public static final int ORANGE = 4;
    public static final int YELLOW = 5;
    public static final int DIRTYORANGE = 6;
    public static final int PINKRED = 7;
    public static final int DIRTYGREEN = 8;
    public static final int GREEN = 9;
    public static final int BLUE2 = 10;
    public static final int[] RAINBOW =
    {
        2895482,2961330,8867962,16023117,
        15052832,16182016,12422669,15537995,
        4544576,35143,2961330
    };
    public byte getAlpha();
    public byte getBlue();
    public byte getGreen();
    public byte getRed();
    public int getRGB();

}
```

package muse.auxiliary.sensors

```
public interface Camera {

    public void createCamera();
    public void captureImage(Object target, CameraProper-
    ties props);
    public void closeCamera();

}
public interface SoundRecorder {

    public void createRecorder(SoundProperties soundPro-
    perties);
    public void startRecording();
    public void stopRecording();
    public void closeRecorder();
    public boolean isRecording();

}
public interface LocationSensor {

    public Object getX();
    public Object getY();
    public Object getZ() ;

}
```

package muse.auxiliary.filesystem

```
public interface FileSystem {

    public String saveMediaFile(final byte[] fileData);
    public byte [] loadFile(String location);

}
public interface ImageLoader {
```

```
    public Image getImage(String key);
    public void loadImages(Hashtable images, Tracker tra-
    cker)
}
public interface Tracker{
    public static boolean isLoaded();
    public static boolean loadedWatchedItem(Tracker tra-
    cker);
    public static boolean addWatchedItem(Tracker tra-
    cker);
}
```

B.2 Formats

B.2.1 InternalMessage

```
<internal>
    <header>
        <source location = "String: service name" />
        <destination location = "String: ser-
        vice name" />
        <originator clientid = "int: humanid" />
        <timestamp
            timestamp = "long: milliseconds"
            vectorclock = "int: vector clock times-
            tamp" />
        <payloadreader reader = "String: Fully Quali-
        fied Class Name" />
    </header>
    <payload>
        String: XML data
```

```
        </payload>
        <attachment name = ‘‘String: filename’’ >
            Base-64 Encoded character data
        </attachment>
    </internal>
```

B.2.2 MindMapMessage

```
    <operation
        type = ‘‘String: ADD|DELETE|INITIALIZE|MOVE|SWAP|GUI_-
        MOVE’’
    >
        XMLNode data
    </operation>
```

B.2.3 XMLNodeMessage

```
    <node
        name = ‘‘String: node name’’
        parent = ‘‘int:
        parent node id’’
        id = ‘‘int: node
        id’’ human = ‘‘id: human id’’
        type = ‘‘String: MIND_MAP|STORY_BEAT|STORYBOARD’’
        x = ‘‘int: x coordinate’’
        y = ‘‘int: y coordinate’’
        description = ‘‘String: node description’’
        media = ‘‘String: file type’’
        location = ‘‘String: filelocation’’
        starttime = ‘‘int: seconds from the start’’
        endtime = ‘‘int: seconds duration’’
```

```
>

    Nested XMLNode data

</node>
```

B.2.4 HumanMessage

```
<human

    username = ‘‘String: username’’
    id = ‘‘int: human id’’
    status = ‘‘String: OFFLINE|ONLINE’’
    logoff = ‘‘String: true|false’’
    color = ‘‘String: 6 digit Hex number’’
    location = ‘‘String: e.g. ip address of user’’
    group = ‘‘String: group name’’

/>
```

B.2.5 ChatMessage

```
<chat

    id = ‘‘int: human id’’
    type = ‘‘String: MIND_MAP|STORY_BEAT|STORYBOARD|ALL
    chatmessage = ‘‘String: the contents of the chat mes-
    sage’’
    messageid = ‘‘int: unique id of the message’’
    priority = ‘‘int: 0-5 priority value’’

/>
```

B.2.6 ServiceMessage

```
<service

    servicename = ‘‘String: Address of the Service’’
    action = ‘‘String: REGISTER|UNREGISTER|LIST|NONE’’
```

```
    humanid = ‘‘int: human id’’

/>
```

B.2.7 QueryMessage

```
<query

    type = ‘‘String: REQUEST_USER|REQUEST_MESSAGE’’
    humanid = ‘‘int: human id’’
    vctimestamp = ‘‘int: vector timestamp’’

/>
```

B.2.8 VideoRendererMessage

```
<renderer

    type = ‘‘String: INITIALIZE|REQUEST_MEDIA|ADD|RENDER’’
    id = ‘‘int: id of the video file’’

>

    MediaElement: XML data type

</renderer>
```

B.2.9 MediaElementMessage

```
<media

    timestamp = ‘‘long: milliseconds’’
    id= ‘‘int: id of media file’’
    name= ‘‘String: name of the file’’

/>
```

B.2.10 VoteMessage

```
<vote
```

```
    type = ‘‘String: RESULT|SETUP|VOTE’’
    quorum = ‘‘int: minimum number of clients required’’
    success = ‘‘int: percent of clients requi-
    red to vote yes to pass vote’’
    id= ‘‘int: id of user’’
    name= ‘‘String: name of vote’’
    vote =’’String: AYE|NAY|ABSTAIN’’
/>
```

B.2.11 HistoryMessage

```
<history
    type = ‘‘String: ARCHIVE|RETRIEVE_ALL|RETRIEVE_DATE_-
    RANGE| RETRIEVE_NUMBER’’
    number = ‘‘int: number of records required required’’
    startdate = ‘‘date: start of date range of interest’’
    enddate = ‘‘date: end of date range of interest’’
    servicename= ‘‘String: service name of messages’’
/>
```