

ResearchOnline@JCU

This file is part of the following reference:

Shatte, Adrian Brian Royce (2016) *Collaborative emergency management report writing: an application for differential synchronisation*. PhD thesis, James Cook University.

Access to this file is available from:

<http://researchonline.jcu.edu.au/46991/>

The author has certified to JCU that they have made a reasonable effort to gain permission and acknowledge the owner of any third party copyright material included in this document. If you believe that this is not the case, please contact

*ResearchOnline@jcu.edu.au and quote
<http://researchonline.jcu.edu.au/46991/>*

**Collaborative Emergency Management Report Writing:
An Application for Differential Synchronisation**

by

Adrian Brian Royce Shatte

Bachelor of Information Technology, James Cook University, Australia, 2012.

Bachelor of Information Technology (Hons), James Cook University, Australia, 2013.

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy.



Information Technology Academy
College of Business, Law & Governance
James Cook University
Townsville QLD 4811

Submitted July, 2016

Declaration of Originality

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

(Signed)

Adrian Shatte

Acknowledgements

This project could not have been completed without the support of many people.

First, I would like to acknowledge the support of my advisors, Dr Jason Holdsworth and Professor Ickjai Lee, for their guidance throughout my candidature.

I would also like to extend my gratitude to the emergency management practitioners who volunteered their time to participate in this research. Appreciate goes to the local councils of Northwest Queensland, including Carpentaria Shire Council, Bourketown Shire Council, and Etheridge Shire Council, for participating and sharing this research project with their colleagues.

I would also like to thank the reviewers of papers published from this work prior to thesis submission. Their valuable comments assisted me to think critically about my work and make improvements.

I gratefully acknowledge the funding support for this project from James Cook University and the Australian Government in the form of an Australian Postgraduate Scholarship.

Finally, I would like to thank my family for their continued support in all that I endeavour.

Statement of Contribution of Others

Research Funding

- APA Scholarship, 36 months, \$75,000

Thesis Committee

- Dr Jason Holdsworth, Information Technology Academy, James Cook University
- Professor Ickjai Lee, Information Technology Academy, James Cook University

Editorial Support

- Dr Jason Holdsworth
- Professor Ickjai Lee
- Samantha Teague
- Anonymous reviewers of the published papers derived from this thesis

Ethics

- Research associated with this thesis complies with the current laws of Australia and all permits necessary for the project were obtained (JCU Human Ethics approval H6412, see Appendix A).

Publications Arising from this Thesis

Conference papers:

A. Shatte, J. Holdsworth and I. Lee, “The impact of dynamic locking on collaborative programming,” *25th Australasian Conference on Information Systems (ACIS)*, Auckland, 2014.

A. Shatte, J. Holdsworth and I. Lee, “Multisynchronous collaboration between desktop and mobile users: A case study of report writing for emergency management,” *26th Australasian Conference on Information Systems (ACIS)*, Adelaide, 2015.

A. Shatte, J. Holdsworth and I. Lee, “Web-based collaborative document writing for emergency management,” *49th Hawaii International Conference on System Sciences (HICSS)*, Koloa, HI, 2016, pp. 217-226.

A. Shatte, J. Holdsworth and I. Lee, “Untangling the edits: User attribution in collaborative report writing for emergency management,” *16th International Conference on Computational Science and Its Applications (ICCSA)*, Beijing, 2016.

Journal Articles:

A. Shatte, J. Holdsworth and I. Lee, “Collaborative report writing for emergency management,” submitted to the *Australasian Journal of Information Systems*, June, 2016 (Under Review).

Abstract

Effective collaboration is fundamental in fast changing environments such as emergencies. Emergency management (EM) requires timely and accurate information from a range of stakeholders to ensure adequate response. Problems arising from poor communication and lack of information sharing between disaster management groups can have a negative impact on response. For remote EM practitioners, a lack of resources combined with the large geographic areas affected by disasters can make effective EM response even more challenging. Modern technologies and advances in networking can provide new methods for sharing real-time information about a disaster with stakeholders regardless of their location. Thus, research should investigate methods for making this technology available for remote EM practitioners.

This thesis aims to develop a research artefact that meets the needs of remote EM practitioners for collaborative report writing tasks. The research methodology employed throughout the thesis to achieve this aim is Design Science. First, a survey of EM practitioners from remote Northwest Queensland (QLD) investigated their current information sharing and report writing practices. Limitations of current practices were identified, including the duplication of effort and the difficulty in sharing information with diverse stakeholders. It was also demonstrated that EM practitioners in remote Northwest QLD are interested in utilising collaborative technologies to improve information sharing both within and between organisations. The results of the survey derive several key features that should be supported in such systems, including support for real-time synchronisation, automatic convergence of information, flexible locking, user attribution, maintaining report history, as well as support for collaboration with users in the field.

Based on the findings of the survey of remote EM practitioners, an artefact was developed iteratively following the principles of Design Science. A comparison of text synchronisation algorithms demonstrated that the Differential Synchronisation technique (diffsync) was most suited for supporting synchronisation in remote EM, due to its natural convergence and robustness to poor network environments. However diffsync lacked support for several of the features suggested by the EM community. Thus, the development of the artefact focused on new techniques and frameworks for achieving flexible locking, user attribution, multi-synchronous editing and crowdsourcing within the diffsync technique. These features combined with diffsync provided a toolkit for developing collaborative EM report writing tools.

To determine whether the artefact met the needs of the remote EM practitioners, it was subjected to several Design Science evaluation methods. These methods included benchmarking, a static analysis based on groupware heuristics, and the instantiation of a prototype. This instantiation was subject to additional evaluations with the target EM community, including a

comparison study, a technology acceptance study, and an enhanced cognitive walkthrough. The results of these evaluations determined that the artefact meets the needs specified by the remote EM practitioners surveyed earlier in the study.

The outcome of this thesis has many contributions and implications. First, the artefact developed provides a framework on which collaborative EM report writing tools can be developed, for example the prototype developed for evaluation. This framework is based on the needs of remote EM practitioners and assists with sharing timely and accurate information between EM responders. This framework could be utilised for further research in EM or other similar fields that require sharing of timely and accurate information. Finally, there were also several technical contributions to the diffsync technique including flexible locking and user attribution which provide a toolkit for emerging work paradigms such as multisynchronous editing and crowdsourcing.

Table of Contents

Declaration of Originality.....	iii
Acknowledgements.....	iv
Statement of Contribution of Others.....	v
Publications Arising from this Thesis.....	vi
Abstract.....	vii
List of Tables.....	xii
List of Figures.....	xiii
1. Introduction.....	1
1.1. Background.....	1
1.2. Scope.....	3
1.3. Research Aims.....	4
1.4. Contributions.....	5
1.5. Methodologies.....	5
1.6. Structure of thesis.....	7
2. Overview of groupware and collaboration for EM.....	9
2.1. Overview.....	9
2.2. Groupware and collaborative editing.....	10
2.3. ICT-based groupware for emergency management.....	12
2.4. Summary.....	16
3. Research methodologies.....	17
3.1. Overview.....	17
3.2. Research Aims and Objectives.....	17
3.3. Design Science.....	18
3.4. User-centred design.....	20
3.5. Evolutionary development.....	21
3.6. Thesis structure relating to research methodologies.....	22
3.7. Summary.....	23
4. Report writing for emergency management.....	24
4.1. Overview.....	24
4.2. Emergency Management Survey.....	24

4.3. Personas	34
4.4. Summary and list of requirements	36
5. Selecting the System Architecture	38
5.1. Overview	38
5.2. Text synchronisation	38
5.3. Current text synchronisation techniques	40
5.4 Selecting the system architecture	55
5.5. Summary	56
6. Flexible Locking Techniques for Diffsync	58
6.1. Overview	58
6.2. Flexible locking	58
6.3. Benefits of flexible locking for collaborative EM report writing	59
6.4. Challenges in supporting collaborative locking	60
6.5. Implementing Flexible Locking in Diffsync	62
6.6. Support for derived features of flexible locking in diffsync	65
6.7. Summary	68
7. User Attribution Techniques for Diffsync	69
7.1. Overview	69
7.2. User attribution in collaborative editing	69
7.3. Benefits of user attribution for collaborative EM report writing	71
7.4. Developing a user attribution framework for diffsync	72
7.5. Demonstration of new user attribution framework for diffsync	75
7.6. Summary	77
8. Developing a multisynchronous framework for diffsync	78
8.1. Overview	78
8.2. Multisynchronous collaboration	78
8.3. Benefits of multisynchronous editing for collaborative EM report writing	79
8.4. Framework for multisynchronous collaborative report writing	80
8.5. Summary	85
9. Crowdsourcing framework for diffsync	86
9.1. Overview	86
9.2. Crowdsourcing for emergency management	86

9.3. Benefits of crowdsourcing for EM report writing	88
9.4. Implementing the crowdsourcing framework in diffsync.....	89
9.5. Summary.....	93
10. Evaluation.....	94
10.1. Overview	94
10.2. Evaluation Methods.....	94
10.3. Benchmarking.....	97
10.4. Static Analysis: Groupware Heuristic Evaluation	102
10.5. Prototype Instantiation.....	107
10.6. Comparison Study	109
10.7. Perceived Usability and Ease of Use (PUEU).....	113
10.8. Enhanced Cognitive Walkthrough (ECW)	116
10.9. Summary.....	125
11. Discussion and Conclusion.....	127
11.1. Summary of results	127
11.2. Research contributions and implications	127
11.3. Generalisability of results	128
11.4. Limitations and recommendations for future work	130
11.5. Conclusion	131
References	132
Appendix A: Ethics Approval H6412.....	142
Appendix B: Requirements Gathering Survey (Chapter 4).....	143
Appendix C: Hierarchical Task Analysis for ECW (Chapter 10).....	146
Appendix D: Specification of user interface for ECW (Chapter 10).....	149
Appendix E: ECW Analysis Tables (Chapter 10)	157

List of Tables

Table 1.1: Research aim and objectives.....	5
Table 2.1: Existing ICT solutions for EM.....	15
Table 3.1: Research aims and objectives for this research.....	17
Table 4.1: EM experience of participants.....	26
Table 4.2: Intra and inter-organisational communication.....	27
Table 4.3: Report-writing collaboration style.....	28
Table 4.4: Software used in EM report writing.....	29
Table 4.5: Strengths and weaknesses of current report-writing software.....	30
Table 4.6: Hardware used in EM report-writing.....	30
Table 4.7: Challenges in completing situation reports.....	31
Table 4.8: Suggested features for collaborative report writing system.....	33
Table 4.9: Personas for the collaborative report writing system based on survey results.....	34
Table 5.1: Implementations of OT and key features.....	48
Table 5.2: Implementations of collaborative editing applications based on diffsync.....	53
Table 5.3: Overview of synchronisation techniques based on support for required features.....	55
Table 9.1: Existing solutions that support crowdsourcing for EM.....	87
Table 10.1: Hypotheses for the artefact evaluations.....	96
Table 10.2: Software features compared in the evaluation survey.....	111
Table 10.3: Positive and negative aspects listed by participants with frequency of response.....	115
Table 10.4: Grading of key tasks in the collaborative EM report writing prototype.....	117
Table 10.5: Analysis of functions for the single-user locking task.....	121
Table 10.6: Analysis of operations for the single-user locking task.....	121
Table 10.7: Evaluation results based on supported hypotheses.....	125
Table 11.1: Research contributions of this thesis.....	128

List of Figures

Figure 2.1: Life cycle of a sitrep.....	13
Figure 3.4: The evolutionary development process.....	22
Figure 4.2: Mean and SD of participants' usefulness rating of key collaborative features.	32
Figure 5.1: A typical collaborative editing scenario involving four users.....	39
Figure 5.2: Three-way merge.	43
Figure 5.3: A common scenario of distributed document editing	45
Figure 5.4: Initial state and final document state for 3 sites based on four operations.....	45
Figure 5.5: Demonstration of the OT algorithm.	47
Figure 6.1: Conflicting lock requests on a sentence.	61
Figure 6.2: Overview of client and server-side locking methods in the implementation.	63
Figure 6.3: Conceptual structure of Global Locking Table (GLT) and Lock objects.	63
Figure 6.4: Example of dynamic region adjustment after one user adds a line break.	65
Figure 6.5: Visual representation of locks using two colours.....	65
Figure 7.1: A document has an index-based structure, e.g. 'a' is at index 10.	73
Figure 7.2: Attributions are colour highlighted and associated with a username.	76
Figure 8.1: The multisynchronous framework.	81
Figure 8.2: The web-based interface showing several tasks locked to mobile users.....	83
Figure 8.4: Automatic summarisation and keyword extraction using RAKE.	85
Figure 9.1: The crowdsourcing framework.	89
Figure 9.3a: User can submit a report through a mobile interface.....	91
Figure 9.3b: User can access a list of verification tasks that have been assigned to them.	91
Figure 9.3c: The verification task requires the user to vote True/False.....	91
Figure 9.4: An example of the web-based crowdsourcing interface.....	92
Figure 9.5a: Incoming microreports can be added to report or verified.	92
Figure 9.5b: Microreports awaiting verification from other users.....	92
Figure 9.5c: Archived microreports are stored for future reference.	92
Figure 9.6: Options for submitting a verification task to mobile users.	93
Figure 10.2: The average synchronisation time followed a linear trend.....	99
Figure 10.3: Average synchronisation time of non-locking vs. locking.....	99
Figure 10.4: The synchronisation time followed a linear trend for scalability.	101
Figure 10.5: The user attribution mechanism maintained accuracy (with random variation). .	102
Figure 10.7: Results of the software comparison section of the questionnaire.....	113
Figure 10.8: Results of the PUEU section of the questionnaire.	114

Figure 10.9. HTA for single-user locking task.	118
Figure 10.10. Specification of the user interface for single user locking task.	119
Figure 10.11. Problem seriousness versus task number.	123
Figure 10.12. Problem type versus task number.	124

1. Introduction

This chapter begins with the objectives and scope of the project, and justifies the approaches and methodologies chosen to achieve the project objectives. Further, a brief summary of the contributions of this thesis is provided. Finally, the structure of the thesis is outlined.

1.1. Background

Effective knowledge management is fundamental in fast changing environments such as emergencies [1]. Decision-making in emergencies requires timely and accurate information from a range of stakeholders including law enforcement, emergency response personnel, and other experts [2]. Traditionally, information management for disaster response has been a centralised effort; however modern technologies such as social media have provided new methods for gaining accurate information in real-time and supporting collaboration between responders [1].

Report writing is an important collaborative task conducted by emergency management (EM) practitioners to collate and share information between relevant stakeholders [2]. Such reports (e.g. Incident Reports and Situation Reports) include vital information about the emergency situation, including type of incident, location of incident, contact details of the relevant incident management centre, casualties (including dead, injured, evacuated, and homeless), general details about the situation and damage, actions in progress, assistance required, future intentions, and the overall prognosis of the situation [3]. There are many different strategies and technologies used by organisations to compile reports in the context of EM.

Collaborative report writing is a form of collaborative writing, which describes and process in which multiple authors complete a single document in collaboration. According to Lowry [4], there are five main strategies used in collaborative writing: *single-author* in which one individual writes a report on behalf of a larger group; *sequential-single* where the writing task is divided between a group and one author writes their section before passing it on to the next participant; *reactive* in which authors work on the writing task synchronously and make changes based on the document's evolution over time; *parallel* where users are designated roles (e.g. author and editor) or specific sections of the document to write and work on that section in parallel to other collaborators; and *mixed mode* which denotes any collaborative writing task that combines at least two of the aforementioned strategies.

Several factors can inhibit EM practitioners in providing timely and accurate for reports. One factor is the need to collaborate and share information between multiple organisations during an emergency [2]. There are also different strategies employed for emergency preparation and response between organisations, for example all states and territories in Australia are responsible for their own resources [5]. Within the scope of writing strategies, *sequential-single* and *parallel* writing strategies encourage multiple versions of a report to exist simultaneously with out-of-date information and a lack of group consensus [4]. Additionally, *single-author* strategies can be

prone to bias and human error if there are inadequate processes for accuracy and fact-checking [4]. In addition, lack of information sharing between organisations can result in duplication of responsibility and response efforts. Finally, there are additional challenges involved in obtaining timely information about the emergency from frontline responders in the field. This is due to the communication issues that can be caused by a disaster and difficulty in verifying accuracy or reports made by the public on social media [2], [6].

In recent years, the rise of the web and increased availability of high-speed internet has seen the development of several web-based tools for collaborative writing, for example Google Drive [7]. Web-based collaborative writing tools are those that allow multiple users who may be geographically separated to work on a shared document (e.g. distributed coding [8], [9]) simultaneously, with the system handling convergence and fault tolerance as necessary [10]. These systems can also include additional inbuilt features such as chat (text, voice and/or video), file sharing, user attribution, optional locking, and version control to further support collaboration when writing a document [11]–[14].

Recent research into software-based collaboration systems has also identified the need to support collaboration between different user types to achieve a single cohesive document [15], [16]. Research paradigms such as *multisynchronous* collaboration and *microtasks* attempt to define these problems and investigate solutions. Multisynchronous collaboration is a process in which some users work in real-time (e.g. desktop-based users) while other users work in isolation and commit updates when necessary (e.g. mobile users) [15]. On the other hand, *microtasks* are small, context-free steps that are given to a user which culminate in a small contribution to a greater task such as a collaborative document [17]. The combination of multisynchronous collaboration and microtasks would provide an effective platform for collaborative report writing in EM that allows for contributions from different types of users, including those located in an emergency control centre and other users in the field.

Web-based collaborative writing systems can assist in overcoming the challenges of report writing for EM by providing a centralised interface and repository for collaborative writing, maintaining a history of changes made within reports for further analysis, and providing direct communication channels between control centres and responders in the field [1], [2], [5], [6], [18], [19]. Existing systems such as *ERIC* [20], *ESA* [5], [21], *uEmergency* [18], and *MoRep* [2] attempt to solve the challenges of collaboration and report writing for emergency response, but none of these systems provide a platform for multisynchronous collaboration and microtasks within a shared report between field agents and a control centre.

Based on the need to support these different types of users in the challenging field of collaborative report writing for EM, the aim of this dissertation is to investigate and develop a research artefact that can be used to develop robust, multisynchronous systems to support remote EM practitioners in collaborative report writing tasks.

1.2. Scope

Modern web-based collaborative writing tools are powered by synchronisation techniques, which are algorithms that ensure concurrency control and provide mechanisms to handle any conflicts resulting from the merging of shared text [10]. Popular techniques used for synchronisation can be categorised as being either *synchronous*, which refers to techniques that support real-time synchronisation of text, or *asynchronous*, which refers to techniques that do not support real-time synchronisation of text [16]. Within these two categories, exist techniques such as pessimistic methods [22], shared wikis [15], edit-based algorithms [23], and state-based algorithms [24]. While all of these techniques strive for the similar goal of providing a consistent workspace for collaboration, each technique has different strengths and weaknesses that make them suitable for development of systems in different contexts. For example, some scenarios do not require support for real-time collaboration and synchronisation, such as wikis [25]. Alternatively, collaborative EM is a scenario in which real-time synchronisation is important as users are working together to achieve effective response within time constraints [2].

Researchers in the field of text synchronisation have developed several additional features within these synchronisation techniques that are useful for collaborative text editing. These include: *flexible locking*, a feature that allows participants of a collaborative text-editing session to claim exclusive editing privileges on subsections of text such that no other users are permitted to edit those subsections of text until the locks are removed [13], [26]; and *user attribution*, referring to any additional methods employed to track the contributions made by individual users to the shared text, as well as other methods to display the actions of a user to other participants as they engage with the shared workspace [27].

A novel synchronisation technique that has been introduced in the literature in recent years is Differential Synchronisation (diffsync), which is a state-based technique for synchronisation of text in distributed systems [24]. Diffsync differs from other synchronisation techniques in that it runs a continuous cycle of *diff* and *patch* operations between clients and a server. Changes are detected by performing a *diff* of the current document state against the previously known state, and these concurrent changes are reconciled by patching the changes from one peer into the copy on another (see Figure 1.1) [24]. Other benefits of diffsync include: 1) nearly identical code on both the client-side and the server-side; 2) a state-based approach that naturally provides an edit history; 3) asynchronous updates and scalability; 4) fault tolerance on unreliable or high-latency networks; 5) all clients are guaranteed to converge to the same document; and 6) compatibility with various document types. Moreover, diffsync is designed so that it can be appended to existing applications, thus introducing real-time collaboration to existing non-collaborative systems [24].

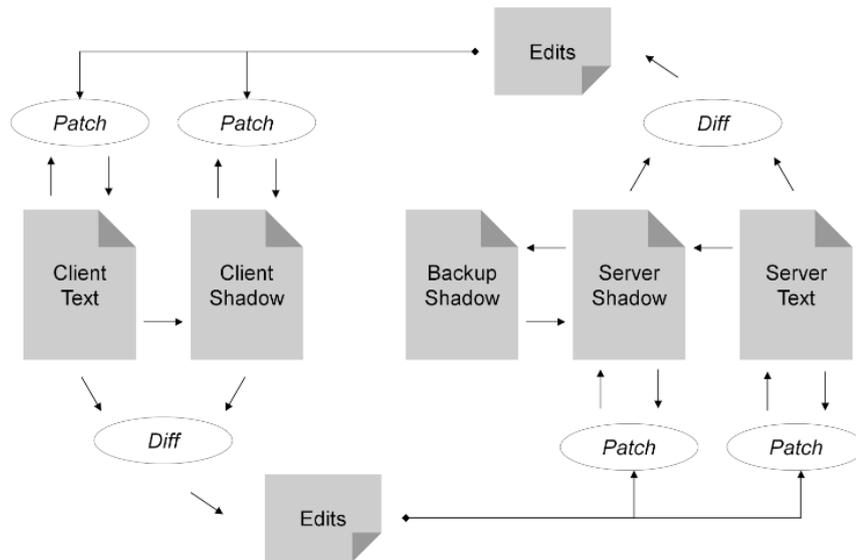


Figure 1.1: Diffsync runs an unending cycle of diff and patch operations to maintain consistency.

In contrast to edit-based techniques such as Operational Transformation (OT) [28], [29], diffsync was designed to be robust to poor network environments, including slow speeds caused by poor infrastructure or high latency environments. This makes diffsync an ideal solution for developing collaborative text-editing applications for fields in which convergence in poor network environments is important, such as the field of EM. However, currently diffsync does not support some of the other key features determined as necessary for collaborative text-editing applications, such as flexible locking, user attribution, and support for multisynchronous editing of a shared document [24].

Thus, based on this background, the scope of the research presented in this dissertation is limited to the diffsync technique and developing methods to support multisynchronous editing and support for microtasks within diffsync. Features such as flexible locking and user attribution have been identified as key tools in developing a framework for multisynchronous editing with support for microtasks, as subtasks should be locked to individual users and their contributions should be identifiable in the scheme of the greater report.

1.3. Research Aims

Based on the background and scope of the project as presented in Section 1.1 and 1.2 respectively, the aim and objectives presented in Table 1.1 were defined to guide the progress of the project discussed in this thesis.

Table 1.1: Research aim and objectives.

<i>Research Aim:</i>	To develop a solution for collaborative report writing that meets the needs of remote EM practitioners
<i>Objective 1:</i>	Determine the requirements for an improved collaborative report writing system to support remote EM practitioners
<i>Objective 2:</i>	Develop a research artefact that meets the needs of remote EM practitioners for collaborative report writing
<i>Objective 3:</i>	Evaluate the research artefact for performance, robustness, usability and acceptance by the target EM community

1.4. Contributions

Based on the aims and objectives, the contributions of this thesis to the current state of research in the field of collaborative EM report writing and diffsync are as follows. First, a list of design requirements for EM collaborative technologies is derived from the results of a survey of EM practitioners from remote Northwest QLD. Second, a real-time collaborative report writing artefact is developed for the remote EM domain that meets these design requirements and is evaluated to be usable and accepted by the EM practitioners.

In addition, several technological contributions were the result of developing the research artefact. A conceptual framework for implementing both flexible locking user attribution into the diffsync technique is defined. These two techniques comprise a third technological contribution by providing a toolkit for implementing multisynchronous editing and crowdsourcing in diffsync. These technological contributions are evaluated in terms of performance, robustness, scalability and efficiency, as well as usability measures.

These contributions have implications for both EM response and research into collaborative systems. First, the report writing artefact addresses the challenge of collaborative report writing for the remote EM practitioner user group and could be further developed into a complete application. It also formalises current processes to support both control-centre based users and field officers in working on a shared report. Additionally, this artefact can be applied to other EM domains or non-EM domains that require sharing of timely and accurate information. This toolkit could also be used for other emerging collaboration paradigms to support collaboration between diverse user types.

1.5. Methodologies

The work presented in this thesis followed the Design Science methodology for research. Design Science is a research paradigm that seeks to create and evaluate research artefacts that are

intended to solve identified organisational problems [30]. In the case of this project, the identified problem is the need for better collaboration between EM practitioners in report writing tasks. Hevner et al. [30] state seven guidelines for Design Science research:

- *Design as an artefact*: The research must result in the production of a viable artefact in the form of a model, method, framework or implementation of a system;
- *Problem relevance*: The research should develop technology-based solutions to important and relevant business problems;
- *Design evaluation*: Well-executed evaluations should be conducted to demonstrate the utility, quality and efficacy of the design artefact that is produced;
- *Research contributions*: The research must provide clear contributions in terms of the developed artefact, the foundations of the artefact's design, and/or the methodologies used to develop the artefact;
- *Research rigor*: Rigorous methods should be applied throughout the process of research, including in the design and evaluation of the artefact;
- *Design as a search process*: Available means should be utilised to reach the desired goal of the research while ensuring that the rules and laws of the problem environment are adhered to; and
- *Communication of research*: The research should be presented effectively in a manner that is accessible to both technology and management-oriented audiences.

Within these seven guidelines, the work conducted in the scope of this thesis comprises seven major steps documented in the following list:

1. Determine the needs of the EM community for systems to support collaborative report writing;
2. Study existing solutions for text synchronisation to determine the main techniques currently outlined in the literature, and determine a subset of features supported by these techniques that are beneficial to develop real-time text editing systems on the web;
3. Investigate existing solutions for flexible locking in the field of text synchronisation, and develop a conceptual solution that can be used within the difflib framework to support optional locking (e.g. locks can be optionally activated or deactivated by an entity within the system, such as a user) and dynamic (e.g. locks will adapt and update to reflect the changes made to the shared text);
4. Investigate existing solutions for user attribution in the field of text synchronisation, and develop a conceptual solution that can be used within the difflib framework to attribute edits to authors, including other forms of attribution such as calculating relative user contribution and storing document history;

5. Investigate existing solutions for multisynchronous editing in the field of text synchronisation using flexible locking and user attribution, and develop a conceptual solution that can be used within the diffsync framework to facilitate collaboration and synchronisation between online users and offline users (or those with more limited network connectivity such as mobile users);
6. Define and implement a software artefact using the previously designed conceptual solutions to demonstrate a web-based, collaborative text-editing system for the field of EM, with specific focus on collaborative report writing.
7. Conduct evaluation of the system throughout the research process and on the developed artefact, in the form of benchmarking, static analysis based on groupware heuristics, implementation of a prototype, comparison study, Perceived Usability and Ease of Use study, and enhanced cognitive walkthrough.

1.6. Structure of thesis

This thesis has eleven chapters including this introduction, background, research methods, artefact implementation, evaluations, and conclusion. The following provides an overview of the content of each chapter:

- *Chapter 2* provides an overview of groupware and collaborative systems for EM in general. The purpose of this chapter is to demonstrate the need for further research into systems to support collaborative EM report writing for remote EM practitioners.
- *Chapter 3* describes the research methodologies used to guide the project, including design science, user-centred design and evolutionary development. As this project aims to solve the organisational problem of collaborative report writing for remote EM practitioners, these research methodologies justify the techniques used for developing and evaluating the research artefact devised in this thesis.
- *Chapter 4* reports on the results of a requirements gathering survey completed by remote North QLD EM practitioners. These results describe the current report writing practices of EM practitioners and a list of features required for an improved collaborative report writing system. These features are investigated in the implementation chapters.
- *Chapter 5* presents an overview of text synchronisation techniques. The goal of this chapter is to determine whether there are existing techniques for text synchronisation that are suitable to support the collaborative EM report writing system developed in this thesis. It is concluded that no existing synchronisation technique meets all of the needs of EM practitioners, however diffsync is selected due to its strengths for EM and a set of features to be developed are outlined, including flexible locking, user attribution, multisynchronous editing and crowdsourcing.

- *Chapter 6* defines the concept of flexible locking and outlines techniques and approaches used to integrate flexible locking techniques into the diffsync method of text synchronisation.
- *Chapter 7* presents a conceptual framework to extract user attribution information from diffsync in order to track the contributions made by authors in a collaborative text-editing environment. This includes novel uses of the framework to attribute content to authors, calculate the relative contribution made by authors in a shared document, and methods to store document history in diffsync for future analysis.
- *Chapter 8* investigates techniques for supporting multisynchronous collaboration between mobile and desktop users within the diffsync technique. This culminates in a novel technique to support contributions from both online users in real-time, as well as mobile users using a push-based synchronisation mechanism.
- *Chapter 9* extends the multisynchronous framework demonstrated in Chapter 8 to include crowdsourcing features. These features allow mobile users to contribute microreports back to the emergency control centre staff working within a collaborative report. It also supports information verification by allowing mobile users to vote on the accuracy of information.
- *Chapter 10* describes the rigorous techniques used to evaluate the artefact developed throughout the preceding chapters. This includes benchmarking, static analysis based on groupware heuristics, and the development of a prototype to demonstrate the applicability of the framework artefact. The prototype becomes an additional instantiation artefact that is subjected to further evaluation, including a comparison study, technology acceptance study, and enhanced cognitive walkthrough.
- *Chapter 11* moves on to a general discussion on the work presented in this thesis, including a discussion of the findings and the potential implications this has for other areas in the field of collaborative EM. The chapter ends with a conclusion by summarising the work presented in this thesis and reiterating the main findings and contributions.

2. Overview of groupware and collaboration for EM

This chapter describes the current state of systems that support collaborative report writing for EM. Firstly, an overview of groupware and the challenges of collaborative editing are described. Further, an overview of collaborative technologies for EM is provided. The aim of this chapter is to demonstrate that there is no comprehensive system to support the needs of remote EM practitioners in collaborative report writing.

2.1. Overview

Effective knowledge management is fundamental in fast changing environments such as emergencies [1]. Decision-making in an emergency requires timely and accurate information from a range of stakeholders including law enforcement, emergency response personnel, and other experts [2]. Traditionally, information management for disaster response has been a centralised effort within individual organisations, however modern technologies have provided new methods for gaining accurate information in real-time and supporting collaboration between stakeholders [1].

Report writing is an important task conducted by emergency management (EM) practitioners to collate and share information between relevant stakeholders [2]. Documents such as *Situation Reports* (sitreps) include vital information about an emergency, including the type of incident, location of the incident, contact details of the relevant incident management centre, details of any casualties (including dead, injured, evacuated, and homeless), general details about the situation and damage, actions in progress, assistance required, future intentions, and the overall prognosis of the situation [3]. There are many different strategies and technologies used by organisations to compile reports in the context of EM.

EM reports like sitreps are completed by experts, field officers and control centre staff in collaboration. According to Lowry [4], there are five main strategies used in collaborative writing: *single-author* in which one individual writes a report on behalf of a larger group; *sequential-single* where the writing task is divided between a group and one author writes their section before passing it on to the next participant; *reactive* in which authors work on the writing task synchronously and reactively make changes as the document evolves over time; *parallel* where users are designated roles (e.g. author, editor, and so on) or specific sections of the document to write and work on that section in parallel to other collaborators; and *mixed mode* which denotes any collaborative writing task that combines at least two of the aforementioned strategies.

Due to the need for collaboration and information sharing between multiple organisations in an emergency [2] and the different strategies employed for preparation and response (e.g. in Australia, all states and territories are responsible for their own resources [5]), there can be problems in providing timely and accurate information in reports. For example, *sequential-single*

and *parallel* writing strategies can encourage multiple versions of a report to exist simultaneously with out-of-date information and a lack of group consensus [4]. Additionally, *single-author* strategies can be prone to bias and human error if there are inadequate processes for accuracy and fact-checking [4]. Also, lack of information sharing between organisations can result in duplication of responsibility and response efforts. Finally, there are additional challenges involved in obtaining timely information about the emergency from frontline responders in the field, due to potential communication issues caused by a disaster and difficulty in verifying accuracy or reports made by the public on social media [2], [6].

In recent years, the rise of the web and increased availability of high speed internet has seen the development of several web-based tools for collaborative writing, for example Google Drive [7]. Web-based collaborative writing tools are those that allow multiple users who may be geographically separated to work on a shared document (e.g. distributed coding [8], [9]) simultaneously, with the system handling consistency and fault tolerance as necessary [10]. These systems can also include additional inbuilt features including chat (text, voice and/or video), file sharing, user attribution, optional locking, and version control to further support collaboration when writing a document [11]–[14].

Recent research into software-based collaboration systems has also identified the need to support collaboration between different user types (e.g. real-time collaboration and non-real-time collaboration based users) to achieve a single cohesive document [15], [16]. Within the scope of EM, web-based collaborative writing systems can assist in overcoming the challenges of report writing for EM by providing a centralised interface and repository for collaborative writing, maintaining a history of changes made within reports for further analysis, and providing direct communication channels between control centres and responders in the field [1], [2], [5], [6], [18], [19].

This chapter provides an overview of groupware with a specific focus on techniques for supporting collaborative editing. First, Section 2.2 defines groupware, its challenges and lists some examples of collaborative technologies. Next, Section 2.3 provides an overview of the challenges facing collaborative technologies for EM as well as an overview of existing systems for collaborative EM. Finally, Section 2.4 concludes with a summary on the current collaborative technologies for EM, prompting the need for further investigation into techniques for supporting collaborative report writing for EM practitioners.

2.2. Groupware and collaborative editing

According to Ellis and Gibbs [31], the term groupware system refers to any computer-based system that is designed to engage two or more users in a common task, completed through a shared interface or environment. Over the past decade, a multitude of groupware applications have been developed to support teamwork and collaboration in many fields. These applications

can be broadly categorised into supporting either *synchronous* collaboration, or *asynchronous* collaboration [16].

Asynchronous collaborative editing applications are those that do not allow for real-time collaboration (e.g. wikis and version control systems), instead providing a platform for users to upload and converge newer versions of a document that was edited on an isolated computer. In these systems, the changes performed by other users are typically not immediately observable [16]. In contrast, *synchronous* collaboration applications are those that allow for real-time collaboration. For example, cloud-based document editors (e.g. Google Docs [7]) or cloud-based IDEs (e.g. Cloud9IDE [32]) are synchronous collaboration applications. Synchronous applications are commonly built on edit-based synchronisation techniques such as OT [33], which refers to a group of edit-based algorithms that can merge concurrent updates with immediately observable results (OT is discussed more in Section 3.4.5). Alternatively, diffsync [24] is a state-based algorithm that supports synchronous editing (discussed more in Section 3.4.6).

The benefits of collaboration have been widely researched within the psychology literature. According to several researchers [34], [35], for collaboration to be more effective than competitive or individualistic methods, the following five conditions must be met: (1) positive interdependence must be clearly perceived by all group members; (2) promotive interaction should be encouraged, such that resources are shared and group members promote each other's work; (3) individual accountability and personal responsibility towards the group goal must be clearly perceived; (4) group members should be motivated to provide effective leadership, decision-making, trust building, communication, and conflict management; and (5) regular reflection on the group's current functioning to improve the future effectiveness of the group. The sharing of knowledge and expertise allow for diversity of skills (e.g. a multidisciplinary approach), higher productivity due to division of labour, and increased visibility of the resulting work [36].

However, collaboration is not without its drawbacks. For example, collaborative writing requires strong integration between co-authors to ensure a coherent piece of writing is achieved [36]. There can also be time costs involved in dividing the workload and keeping all collaborators up-to-date with the current progress of the shared work [36]. As collaboration is a highly social enterprise, disagreements and conflicting opinions can also arise which can negatively affect the outcome of a project [36]. Finally, there may be additional negative costs where a collaborator is required to move to different sites away from the rest of the team [36]. Thus, a successful collaborative editing system should enhance the benefits of collaboration while also mitigating the drawbacks.

One area in which collaborative editing systems have had a major impact is software engineering [9], [37], [38]. Software development is regularly a team-based effort with different

groups of developers working together to design and implement solutions, fix each other's bugs, and produce quality code to meet deadlines. This teamwork can be extended to other areas of an organisation, including project managers, software testers, and interface designers [39]. Collaboration has been shown to improve the problem solving process [40], reduce errors, and improve code quality [41].

As the number of collaborators on a project increases, so too does the importance of clear communication between those collaborators [42]. Non-technical solutions to establish a steady flow of communication include team-building exercises, regular meetings, agile software development principles, and support from management [39]. However, meetings and exercises can also hinder the progress of development, due to its separation from the development environment. For this reason, some researchers have investigated and developed solutions that integrate and promote collaboration within the development environment being used by collaborative programmers [37], [38].

Simple approaches to collaboration using technology include tools such as version control, screen sharing, email, and instant messaging programs [39], however these solutions can still cause distractions because they may require the user to minimise the development environment while focusing on another application. On the other hand, version control tools are limited in that they do not often provide real-time collaboration. Another approach is to integrate some of these tools directly into an Integrated Development Environment (IDE) [39], which is a specialised type of program used to write and develop software programs. Research has shown that implementing collaborative tools directly into IDEs has several benefits, including reduced friction in the development process, improved context, and immediate traceability between collaborative artefacts and code artefacts [39]. A noteworthy example of a real-time collaborative editing plugin for the Eclipse IDE is Saros [43].

2.3. ICT-based groupware for emergency management

Collaboration is a fundamental technique used in the creation and development of work artefacts over time [44]. For many tasks, the participation of multiple users provides skill diversity, sharing of the workload, and keeps interested stakeholders up to date with the progress and direction of a project [44]. EM is a complex task that requires knowledge sharing between many separate entities including law enforcement, government, and the public [45]. In an emergency, communication between different agencies is important to ensure that accurate information is relayed to stakeholders in a timely manner.

One of the important collaborative tasks conducted by agencies during an emergency is shared report writing. An important collaborative document used by agencies in an emergency is the sitrep. A sitrep is a document that records an ongoing description of all events occurring during an incident (e.g. emergency, disaster, and so on). The content of a sitrep must be both

accurate and timely, as these details are passed on to adjacent agencies or higher headquarters and Emergency Operations Centres. The information contained in the report is used to make critical decisions during an emergency [3].

The sitrep can appear in many formats. The Australian Government suggests that a sitrep contains a report number, date and time, type of incident, location of incident, contact details of the relevant incident management centre, casualties (including dead, injured, evacuated, and homeless), general details about the situation and damage, actions in progress, assistance required, future intentions, and the overall prognosis of the situation [3]. Other guidelines state that a sitrep should contain factual information only, be brief, and should be specific to a single functional area (e.g. roads). Sitreps can also include graphics such as maps. The life cycle of a sitrep is displayed in Figure 2.1, spanning collection of information from various agencies, constructing knowledge and intelligence based on that information, and sharing it with relevant stakeholders.

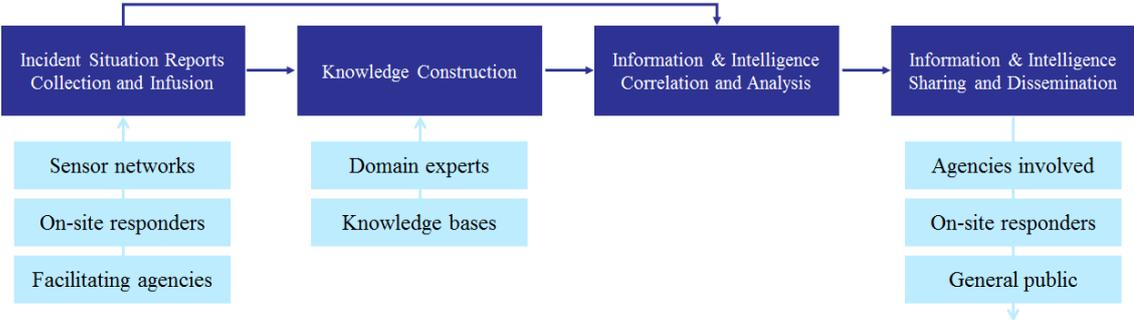


Figure 2.1: Life cycle of a sitrep (adapted from [46]).

While sitreps can be useful for providing brief but informative facts about an emergency, there can be challenges in using them effectively. For example, a report on the response to the 2011 Christchurch earthquake noted that sitreps were not always shared with the appropriate parties, resulting in important information being unavailable to assist with decision-making [47]. In addition, many parties wasted time compiling duplicate reports and responses for different levels of government when that information was already available in previous versions of sitreps [47]. In addition, supporting technology can also negatively affect information sharing in an emergency. The Royal Commission into the 2009 Victoria Bushfires concluded that key agencies used different technologies, which caused access difficulties for staff at incident control centres. This inhibited the propagation of important information including sitreps, notifications, warnings, and maps [48]. These examples demonstrate that a lack of effective coordination, poor communication and confusion over situation awareness result in ineffective sitreps.

Unexpected problems, dynamic changes of situations or environmental and knowledge

limitations lead to the need for improvisation in crises and emergencies [49]. For this reason, it is important to ensure that EM systems are flexible to the characteristics of disasters. Based on the analysis of previous disasters, six characteristics were identified that contribute to the challenging nature of EM [2], [50]:

- *Rarity of disasters*: it is rare for major disasters to occur which limits opportunities for training and learning in real situations;
- *Time pressure*: the steps of planning and executing solutions are forced to converge due to time constraints in an emergency, meaning that some decisions must be made with limited information available;
- *Uncertainty*: it is difficult to predict how a crisis will develop over time so predictions and improvisation are necessary based on intelligence;
- *High and broad consequences*: often the effects of a large disaster are high and broad, which means that interdependencies among a wide range of physical and social systems must be managed to ensure adequate response;
- *Multiple decision makers*: due to the devastating impact of disasters and their effect on multiple facets of society, disasters require the negotiation and collaboration of multiple stakeholders that may have competing priorities;
- *Complexity*: based on the combination of the previous challenges, complexity arises.

Beyond the challenges that practitioners face in all types of disasters, there are additional challenges that can affect practitioners in remote communities [39]. The most prominent of these challenges is the limited resources available to remote communities. Remote EM typically requires collaboration between local responders and other stakeholders that may work offsite (e.g. from the state capital). Further, remote areas may suffer from poorer network infrastructure which can affect the quality of network connectivity and introduce mobile blackspots. Finally, remote areas may span larger, less accessible geographic areas than urban or regional areas, posing additional challenges in mitigating, monitoring and responding to disasters. These additional challenges amplify the complexity of emergencies in remote communities.

Information and communication technologies (ICT) can be employed to assist in different tasks and phases of EM. There are four phases of the EM process: *mitigation*, *preparedness*, *response*, and *recovery*. *Mitigation* refers to the pre-disaster identification of risks and steps taken to reduce potential negative effects of a disaster on property and people. *Preparedness* refers to warning systems and other processes engaged before a disaster to ensure an adequate response by emergency managers and the public once a disaster actually occurs. *Response* refers to any actions taken immediately prior, during, and after a foretold disaster which assist in reducing human and property loss. Finally, *recovery* refers to the process of returning the affected

population back to a “normal” state after a disaster event [51]–[53]. The main benefits of ICT for EM are: 1) diverse communication channels can be utilised for effective warnings and notifications; 2) information from diverse sources can be easily integrated into a single system; 3) disaster relief efforts can be centrally coordinated; 4) systems can encourage both public and institutional input; and 5) damage caused and other effects of a disaster can be analysed to inform recovery [19].

Table 2.1: Existing ICT solutions for EM.

<i>Name of system</i>	<i>Main Features</i>	<i>Reference</i>
<i>WebEOC</i>	Event reporting, situational awareness throughout the lifecycle of an incident, resource management, generation of sitreps and after action reports.	[54]
Emergency Response Intelligence Capability (<i>ERIC</i>)	Integrates data from state and federal agencies into a single web-based map interface, historical snapshots, and automated situation reports.	[20]
Emergency Situation Awareness (<i>ESA</i>)	Detects unusual behaviour on Twitter to quickly alert the user when a disaster event is being broadcast,	[5]
uEmergency	Interactive tabletop system that displays maps and supports annotations. History slider allows users to view different situations over time. Supports single-user, multi-user, and mixed collaboration methods.	[18]
MoRep	Android application for viewing reports, requesting reports from subordinates, and creating reports with multimedia during an emergency. Information is displayed as annotations on a map.	[2]
TwiddleNet	Uses smartphones as personal servers to enable instant content capture and dissemination for first-responders in emergencies. Automatic tagging of content and distribution to several networking channels.	[55]
MobileMap	Complements radio communication to allow for ad hoc communication (when network availability is unreliable), decisions support and collaboration among workers in the field using mobile devices.	[6]

With increased availability of the internet and new ICT technologies, several software systems for collaborative EM have been developed in the past few years. Table 2.1 displays some of the systems currently being used in EM and the main features of these systems. While the

systems presented in Table 2.1 cover a range of features for collaborative EM, none of these systems provides a dedicated platform for collaborative report writing. For modern EM, it is important to support the diverse types of users that respond to an emergency. Users working in an emergency control centre are typically information providers and consumers, whereas field officers who are responding on the scene are only information providers [56]. ERIC [20] attempts to generate sitreps automatically using data from various government feeds, but it does not take into account the information provided in real-time by field officers. On the other hand, systems such as TwiddleNet [55] and MobileMap [6] provide support for communication and report sharing between first responders in the field, but neglects support for collaboration with users in a control centre.

In addition, none of these systems appear to provide a complete solution to the specific challenges faced by EM practitioners in remote communities, as mentioned throughout this section. There are several benefits for remote EM collaboration in these systems, for example TwiddleNet [55] and MobileMap [6] provide an alternative means for communication when network connectivity is limited. Further, systems like uEmergency [18] allow for visual organisation of information based on geographic location, ESA [5] can detect potential incidents based on Twitter posts, and ERIC [20] can compile reports automatically based on feeds from government organisation. Despite these benefits, none of these systems were designed or evaluated within the scope of remote EM practitioners. This demonstrates that remote EM practitioners are an underserved community within EM information systems. Thus, further research is needed to develop a framework that can support collaborative report writing for the diverse user types involved in emergency response while also considering the needs of remote EM practitioners. The needs of remote EM practitioners are explored further in Chapter 4.

2.4. Summary

In this chapter an overview of groupware and its applications to the challenges of EM was provided. Several existing technologies to support collaboration between EM practitioners were considered, however it was noted that none of these solutions provides support for collaborative report writing between control centre-based users and field officers. Further, it was noted that none of these existing solutions considers the needs of remote EM practitioners. This outlines a clear area for research to investigate and develop a solution for collaborative report writing for remote EM practitioners. Based on this need for further research, Chapter 3 presents the results of a survey of remote EM practitioners to determine their exact requirements for collaborative report writing.

3. Research methodologies

This chapter provides an overview of the methodologies that guide the research presented in this thesis. The chapter begins by outlining the research aims and objectives. A discussion of the research methodologies is then provided, including Design Science, User-centred design, and Evolutionary development. The chapter concludes by developing a plan to address the research aims and objectives using the research methodologies.

3.1. Overview

In Chapter 2 a background was provided on groupware and its benefits for EM collaboration. While several existing collaborative EM systems were identified, it was noted that none of these systems provides support for the specific needs of remote EM practitioners for collaborative report writing. As such, it was concluded in this chapter that further research is needed to determine the specific needs of remote EM practitioners for groupware and develop a solution that meets those needs. This is the primary aim of this dissertation.

This chapter begins in Section 3.2 with a reiteration of the aims and objectives of this thesis. Section 3.3 then provides an overview of Design Science, which will be used as the primary methodology guiding the research. Design Science is a technique that aims to solve organisational problems through the development and evaluation of research artefacts. Next, two additional research methodologies are outlined in Sections 3.4 and 3.5, which include User-centred Design and Evolutionary development respectively. These research methodologies are complementary to the Design Science approach. Finally, Section 3.6 summarises the chapter and frames the methodological plan for the research project.

3.2. Research Aims and Objectives

From reviewing the literature in groupware for EM, the following aim and objectives of this thesis we developed (see Table 3.1). To achieve these aims and objectives, the research methodologies described in Section 3.3, 3.4, and 3.5 were utilised. These methods are described throughout the rest of this chapter to set the scene for the work presented in this thesis.

Table 3.1: Research aims and objectives for this research.

Research Aim:	To develop a solution for collaborative report writing that meets the needs of remote EM practitioners
<i>Objective 1:</i>	Determine the requirements for an improved collaborative report writing system to support remote EM practitioners
<i>Objective 2:</i>	Develop a research artefact that meets the needs of remote EM practitioners for collaborative report writing

<i>Objective 3:</i>	Evaluate the research artefact for performance, robustness, usability and acceptance by the target EM community
---------------------	---

3.3. Design Science

The work presented in this thesis followed the design science methodology for research. Design science is a research paradigm that seeks to create and evaluate research artefacts that are intended to solve real-world organisational problems [30]. Design science research includes iterative development cycles and user-centred design principles, which can help the project to achieve an artefact that is likely to be accepted by the target users. Artefacts can be manifested in many ways, including constructs, concepts, models, methods, or instantiations [57]. Research artefacts do not need to comprise of complete systems that are ready to roll out; rather the research artefact should demonstrate the design theory or construct and its ability to solve the relevant problem. Additionally, such artefacts can evolve over the course of the research as feedback is gained from evaluation. Design science research artefacts can be immediately applied to the research domain to gain efficient insights into the impact on the target research area.

Design science provides an applied information systems research approach suitable for the problem domain of remote EM collaborative report-writing. Traditionally, information systems research has used natural science methods such as behavioural or explanatory science to describe interactions between people, organisations and technology [30], [58]. Natural science methods can provide descriptions of phenomena, develop theories of human behaviour, and make predictions from theory. However, the natural science approach is often not applicable to problems encountered in research and practice [58]. In contrast, design science provides an applied method of research that sees “design” as a research contribution in itself. Research artefacts can be developed addressing practical problems, artefacts can be incrementally improved by both researchers and practitioners to develop an effective solution, and artefacts can be evaluated for the solution’s quality, resulting in a robust information systems solution.

The design science process involves three key steps, as outlined in Figure 3.1. First, *problem identification*, which includes finding a problem, defining it and doing background research, and specifying the requirements for a solutions; *Solution Design*, which includes brainstorming, evaluating the potential solution for its potential to solve the problem, and designing and developing a solution in the form of a research artefact; and *Evaluation*, which involves testing the solution using an appropriate method, determining the extent to which the artefact solves the organisational problem that is being investigated, and communicating the results. Based on the results gained during the evaluation, the design of the artefact may be altered or a new artefact developed in an iterative process until an artefact is developed that meets all of the initial requirements for the system.

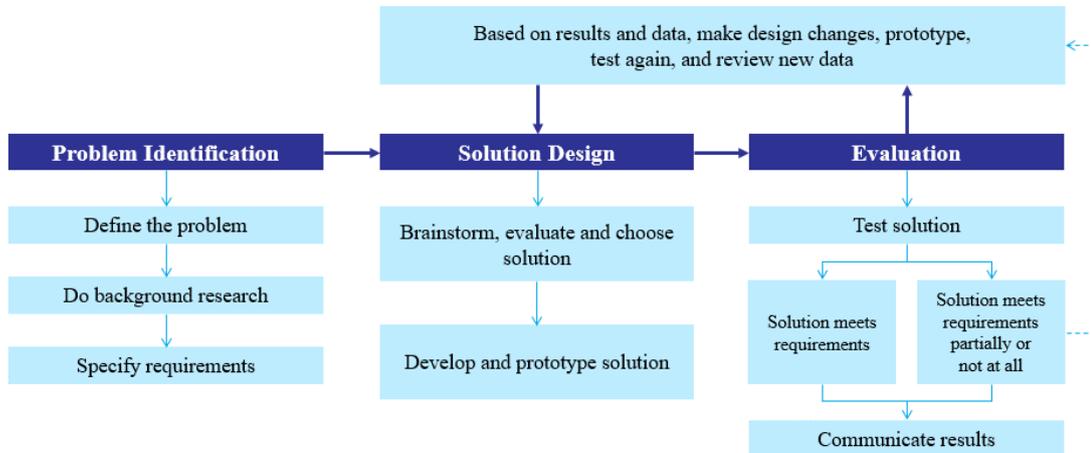


Figure 3.1: The Design Science research framework.

Hevner et al. [30] state seven guidelines for Design Science research:

- *Design as an artefact:* The research must result in the production of a viable artefact in the form of a model, method, framework or implementation of a system;
- *Problem relevance:* The research should develop technology-based solutions to important and relevant business problems;
- *Design evaluation:* Well-executed evaluations should be conducted to demonstrate the utility, quality and efficacy of the design artefact that is produced;
- *Research contributions:* The research must provide clear contributions in terms of the developed artefact, the foundations of the artefact’s design, and/or the methodologies used to develop the artefact;
- *Research rigor:* Rigorous methods should be applied throughout the process of research, including in the design and evaluation of the artefact;
- *Design as a search process:* Available means should be utilised to reach the desired goal of the research while ensuring that the rules and laws of the problem environment are adhered to; and
- *Communication of research:* The research should be presented effectively in a manner that is accessible to both technology and management-oriented audiences.

According to Hevner et al. [30], artefacts resulting from design science research should be evaluated in terms of utility, quality, and efficacy using appropriate methods. Within these categories, artefacts can be evaluated on metrics including functionality, completeness, consistency, accuracy, performance, reliability, usability, fit with the organisation, and any other relevant quality attributes [30]. Accepted evaluation techniques can include: *observational* methods (e.g. case studies and field observations); *analytical* methods (e.g. static analysis,

architecture analysis, optimisation, and dynamic analysis); *experimental* methods (e.g. controlled experiments and simulations); *testing* methods (e.g. functional testing and structural testing); and *descriptive* methods (e.g. informed arguments and scenarios) [30]. Further discussion on evaluations in Design Science is provided in Chapter 10.

3.4. User-centred design

User-centred design is a multidimensional research and software development methodology that aims to incorporate the target user’s needs and perspectives into the design and development process to ensure a usable and acceptable system is achieved [59], [60]. User-centred design fits within the Design Science methodology, as highlighted in Figure 3.2. For example, users can be involved in the *Problem Identification* process, specifically in specifying the requirements of the systems. Previous research has identified that user-centred design can increase user uptake and ensure that artefacts meet the contextual needs of the targeted users [61]. In the case of this thesis, remote EM practitioners were surveyed to determine their current practices for collaborative report writing and to generate a list of requirements for an improved system. Further, users can also be included in the *Evaluation* process. In this thesis several user-centred evaluation techniques were considered, including a comparison study and technology acceptance study in the form of Perceived Usability and Perceived Ease of Use [62].

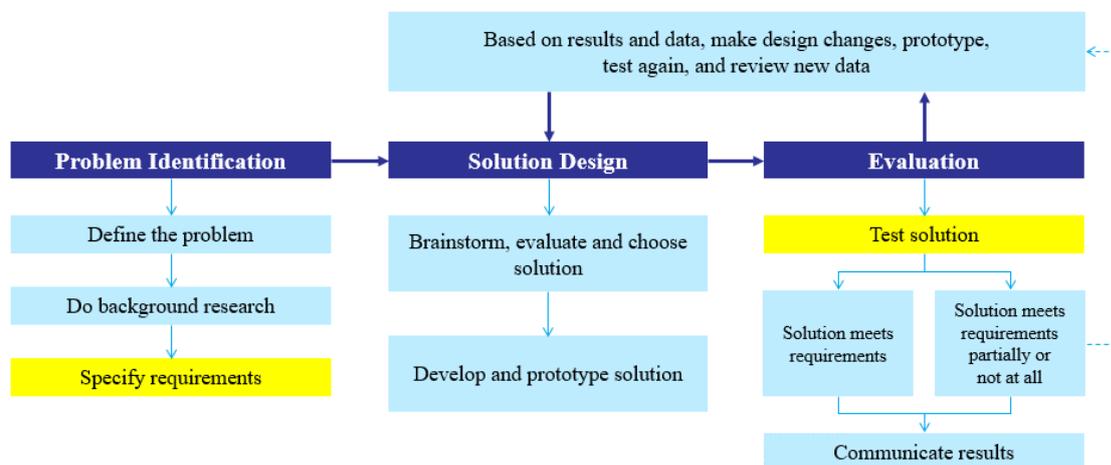


Figure 3.2: User-centred design highlighted within the Design Science framework.

In software development it is important to consider not only the technical and functional requirements of a system, but also that the system meets the requirements of the user to ensure the benefits of the software are realised [59]. According to Maguire [59] there are four key principles of user-centred design that ensure user requirements are met in the development of a system:

1. *The active involvement of users and clear understanding of user and task requirements:* Active involvement of the end-users in the development process can enhance the acceptance and commitment of users to the new software, as well as ensure that the designed system fits the organisational needs due to the expert knowledge and context provided by the end-users.
2. *An appropriate allocation of function between user and system:* The jobs and tasks related to the system should be appropriately divided between system and end-user, focusing on the strengths and limitations of human capabilities and software/hardware automation.
3. *Iteration of design solutions:* It is important that designs and artefacts receive iterative feedback throughout the development process in order to guide further development.
4. *Multi-disciplinary design teams:* Typically, user-centred design is a collaborative process which involves the skills and expertise of different experts, including managers, usability specialists, end-users, software engineers, graphic designers, interaction designers, training and support staff and task experts.

This research integrated the principles of user-centred design into its methodology. End-users were involved heavily in the requirements gathering process to determine the background of the problem and generate a list of requirements for an improved EM collaborative report writing system. Further, based on the technologies utilised and the demands of collaborative report writing, tasks were adequately divided between the system and the end-user. Iterative design and development was used throughout to seek feedback and improve on the design of the system. Finally, the artefact developed for this thesis involved a multi-disciplinary team; the end-users targeted for evaluation span a wide range of EM organisations and experience, including executive officers, field officers, social support, emergency services, marine rescue, and engineers.

3.5. Evolutionary development

Evolutionary development, also known as incremental or iterative development, is a software development process that involves more than one iteration of development. Evolutionary development fits within the Design Science methodology, as highlighted in Figure 3.3. For example, the tasks of *Solution Design* and *Evaluation* require an iterative process. Firstly, a conceptual solution is brainstormed and designed based on the requirements of the system and background research. This solution is transformed into a research artefact that is subjected to appropriate evaluation techniques, and the results of these evaluations guide subsequent development. This process continues iteratively until it is determined that the artefact meets the requirements of the target users.

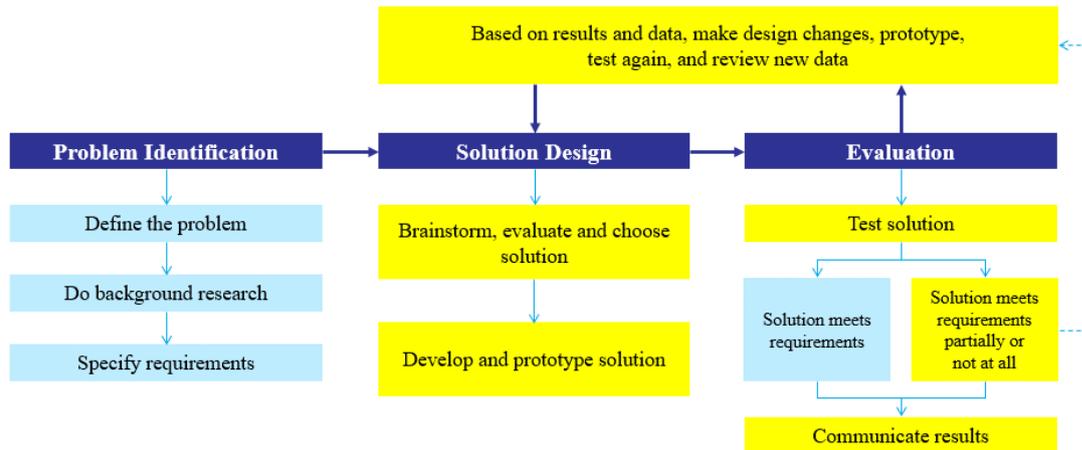


Figure 3.3: Evolutionary development highlighted within the Design Science framework.

Typically, the iterative stages of development are small, involving single features or subsets of the larger system. Future iterations build on the work developed in earlier iterations and the design specifications can evolve throughout the process to include new functional and operational capabilities. These iterations of development continue until the final system is implemented [63]. This process is depicted in Figure 3.4.

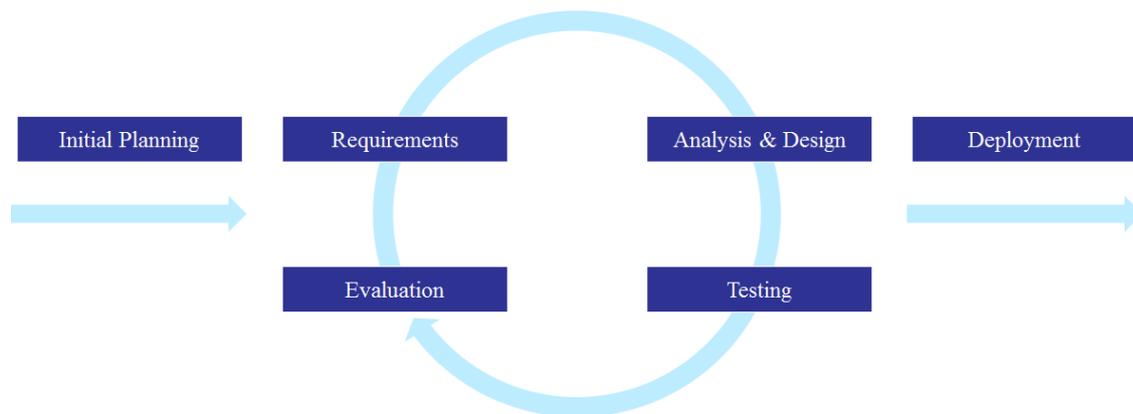


Figure 3.4: The evolutionary development process.

The following chapters in this thesis describe the iterative nature of this research. First, the requirements gathering phase is completed by surveying a community of remote EM practitioners. Based on these requirements, several key features are determined for development and the iterative nature of this development is conveyed in Chapters 5, 6, 7 and 8. These features build on the strengths and weaknesses of each other, and the results were communicated in conference papers to the Information Systems community during the course of the research.

3.6. Thesis structure relating to research methodologies

A design science research methodology was selected for this thesis, along with the principles of user-centred design and evolutionary development. Design science provided a framework for the research project. First, Chapters 2 and 4 cover the *Problem Identification* stage, covering

background research and requirements specification with the target user group. User-centred design aspects are used in the requirements specification phase as a remote community of EM practitioners is surveyed about their current report writing practices and needs for an enhanced collaborative report writing system.

Chapters 5, 6, 7, 8 and 9 cover the *Solution Design* phase of research. These chapters describe the brainstorming process of the artefact design (Chapter 5) and the development of conceptual solutions to meet the needs of the remote EM practitioners surveyed in the requirements gathering phase. Evolutionary development is also covered throughout these chapters, as each subsequent chapter builds on the work presented in the previous chapter to iteratively improve and develop new features into the research artefact.

The *Evaluation* phase of the research is presented in Chapter 10. The research artefact developed throughout the thesis is subjected to a rigorous evaluation using several techniques including case studies, scenarios, benchmarking, simulations, heuristics, and enhanced cognitive walkthrough. Additionally, some of these techniques involve the target end-users to determine whether the artefact meets their needs, as specified by user-centred design principles.

3.7. Summary

This chapter presented an overview of the research methodologies used to guide this thesis. The primary research methodology used is design science, with user-centred design and evolutionary development acting as secondary research methodologies within design science itself. The purpose of this chapter was to set the scene for the research described in the remainder of the thesis, including the requirements gathering, development of the research artefact, and evaluation of the research artefact. These three aspects of research match up with the three key phases of Design Science research and the primary objectives of the thesis, as listed in Table 3.1.

4. Report writing for emergency management

This chapter provides a background on the current state of collaborative report writing in remote North QLD councils. Firstly, the results from a preliminary survey of EM practitioners are presented. This information provides the basis for the development of the research artefact that is developed throughout this thesis. In addition, personas are created to consider the typical users of the systems and their needs, as well as a list of requirements to be supported by the research artefact.

4.1. Overview

In Chapter 3 the research methodologies guiding this research were described, including design science, user-centred design, and evolutionary development. In all of these research methods, the first step is to gather the requirements for the artefact that is to be developed. This can be achieved through a literature search to explain the current practices and design requirements. However, there are no design requirements in the literature for systems to support collaborative EM report writing. For this reason, the target community itself was involved in the requirements gathering process.

This chapter presents the results of a survey conducted with EM practitioners from remote North Queensland. The full set of survey questions are listed in Appendix B. The aim of this survey was to gather a set of requirements for collaborative EM report writing. The overall goal of this thesis is to develop and evaluate a research artefact to support collaborative EM report writing. Thus, the results gained from this survey guide the development and evaluation of the artefact that is presented in the following chapters. Section 4.2 provides an overview of the survey that was conducted and summarises the results. Section 4.3 considers several personas that were developed from the survey results. The chapter ends with a summary of the results in Section 4.4 that includes a list of requirements necessary for developing the research artefact.

4.2. Emergency Management Survey

A survey was conducted to determine current software use amongst EM organisations. The aim of this survey was to determine the current state of collaboration, report writing, and major challenges of report writing for EM. The results of this survey are also provided to guide the development of the prototype described later in this chapter. It was hypothesised that current EM report-writing tools used by the remote EM practitioners would not meet their desired requirements for collaborative work.

4.2.1. Recruitment methods

Participants were recruited through the EM networks in local governments and councils from remote Northwest Queensland (see Figure 4.1). Local government practitioners were specifically targeted due to their leading role in coordinating and responding to disaster relief efforts. These

contacts were emailed with an information sheet about the study and an invitation to participate in an online survey hosted through Google Forms¹. The survey contained a link to the information sheet, and reiterated to participants that no personally identifying information would be recorded and that participation was voluntary, such that participation could be revoked at any time by closing the survey. At the end of the survey, participants were invited to share the link with other EM practitioners who might be interested in completing the survey.



Figure 4.1: The remote Northwest Queensland region targeted for this project [64].

4.2.2. Participant demographics

There were 22 respondents to the survey in total. Participants had an average of 11.5 years of EM experience ($SD=9.59$). The roles of the participants varied in level of expertise and job description, including senior decision makers of fire services and engineering, and first responders in areas such as marine rescue, police, and SES (see Table 4.1). Participants often reported having multiple roles spanning council, disaster management groups, and volunteer roles, depicting the challenges of managing the multidisciplinary skills required in disasters and the limited resources available in remote communities. Participants were experienced in all four phases of EM, with nearly all participants involved in emergency response efforts (95.5% of sample). While the type of emergencies that respondents dealt with were varied, the most common response was Meteorological type emergencies, such as tropical cyclones, bushfires, storms, and drought. Finally, participants reported frequent use or completion of EM reports like

¹ <https://www.google.com.au/forms/about/>

sitreps ($M=6.6$, $SD=2.8$, 10 point Likert Scale where 0=not at all, 10=always), indicating expertise in collaborative EM report writing.

Table 4.1: EM experience of participants.

	<i>Frequency</i>	<i>% of Sample</i>
<i>EM Position</i>		
Local council	9	40.9%
Local Disaster Management Group	5	22.7%
District Disaster Management Group	3	13.6%
Social support	1	4.5%
Police	1	4.5%
Marine rescue/Coast guard	4	18.2%
SES	2	9.1%
Fire services	3	13.6%
Emergency medical officer	1	4.5%
<i>Disaster Experience</i>		
Meteorological	19	86.4%
Geological	4	18.2%
Conflict	4	18.2%
Health	6	27.3%
Accidents	12	54.5%
Search and rescue	4	18.2%
Other (unspecified)	1	4.5%
<i>Disaster Phase</i>		

Mitigation	14	63.6%
Preparedness	16	72.7%
Response	21	95.5%
Recovery	17	77.3%

4.2.3. Collaboration and report writing organisational practices

The current collaborative practices used by EM practitioners for report-writing was investigated. First, participants were asked to specify the primary methods used to communicate with members of their own EM team, as well as methods for communication with external organisations during an emergency situation. Mobile phones, face-to-face meetings, and email were consistently identified as common means of communicating both within and between organisations (see Table 4.2).

Table 4.2: Intra and inter-organisational communication.

	Intra-organisational communication		Inter-organisational communication	
	<i>Frequency</i>	<i>% of Sample</i>	<i>Frequency</i>	<i>% of Sample</i>
Desk phone	12	54.5%	8	36.4%
Video conferencing	7	31.8%	6	27.3%
Email	18	81.8%	15	68.2%
Face-to-face communication	20	90.9%	13	59.1%
Chat	1	4.5%	0	0%
Mobile	20	90.9%	19	86.4%
Written reports	9	40.9%	10	45.5%
Social media	1	4.5%	0	0%
Teleconferencing	12	54.5%	12	54.5%

Radio	7	31.8%	6	27.3%
Other (unspecified)	1	4.5%	0	0%

Participants were then asked to report their experience with reporting for EM, including the processes involved in report writing within their organisation (as per the taxonomy proposed by Lowry [4]) and collaborative work models employed (e.g. potential location and time zone differences between collaborators). It was found that reports were written by a single author on behalf of the organisation with assistance from other collaborators in real-time (see Table 4.3). Reports are usually written in real-time, with both onsite and offsite collaborators making contributions.

Table 4.3: Report-writing collaboration style.

	<i>Frequency</i>	<i>% of Sample</i>
<i>Group writing style (Lowry [4])</i>		
Single author	13	59.1%
Sequential	5	22.7%
Parallel	3	13.6%
Reactive	4	18.2%
<i>Type of collaboration</i>		
Same time and place	8	36.4%
Same place, different time	3	13.6%
Different place, same time	8	36.4%
Different place and time	6	27.3%

As well as collaborative practices, participants were asked about the type of software and hardware used in their organisation for report writing. Participants were asked to describe the category of software (e.g. synchronous or asynchronous) and the name of the software product. Participants were also asked to identify the strengths and weaknesses of the current software being used for EM report writing in their organisation. This was an open-ended response. It was

found that participants complete reports using a single-author, asynchronous software program ($n=14$, 63.6% of sample), in which information is gathered but compiled into a report by a single author on behalf of the organisation. The most common type of software used was Microsoft Word. Participants reported a wide variety of other software with low adoption rates, including other Microsoft Office tools (see Table 4.4).

Table 4.4: Software used in EM report writing.

	<i>Frequency</i>	<i>% of Sample</i>
Microsoft Word	10	45.5%
Microsoft One Note	2	9.1%
Microsoft Excel	2	9.1%
DIEMS	2	9.1%
Guardian	4	18.2%
Asset Edge	1	4.5%
SkyTrust	1	4.5%
QIT Plus	1	4.5%
Hand-written	1	4.5%
eIAP	1	4.5%
Unsure	5	22.7%

Participants were asked to discuss the key benefits and weaknesses of the current software used for report writing (see Table 4.5). The main benefits of the software includes the familiarity with current software (e.g. Microsoft Word), easy to make the report legible using in-built styling features of Microsoft Office products, and can file copies of the reports onto a shared drive for later access. The main weaknesses identified were that document version control is only basic, double handling of information is required due to the lack of collaborative features in the software, and additional effort is required to share the report with other agencies.

Table 4.5: Strengths and weaknesses of current report-writing software.

<i>Strengths</i>	<i>Weaknesses</i>
“Easy to use”	“Basic document control”
“Familiar software”	“It is slow and cumbersome and requires double handling of information”
“Document can be filed on a shared drive”	“Not a live document, i.e. requires human action to publish to other agencies via email”

Finally, participants were asked to identify the primary types of hardware used to complete reports. It was found that most report writing for EM is completed on desktop computers, followed by laptops and smartphones (see Table 4.6). Overall, participants were satisfied with their hardware support and as such did not list many weaknesses, apart from the challenges of losing power in a disaster event. Strengths of current hardware included participants’ familiarity with each platform, the ability for reports to be used across platforms easily (e.g. Microsoft Word), the portability of mobile and laptop devices, and the ability for mobile and laptop devices to use battery power. Most participants reported that there were no weaknesses to the hardware they use in report-writing ($n=12$, 54.5% of sample). Of the weaknesses reported, the most common was the challenges associated with network disruptions ($n=7$, 31.8% of sample) and power outages ($n=4$, 18.2% of sample).

Table 4.6: Hardware used in EM report-writing.

	<i>Frequency</i>	<i>% of Sample</i>
Desktop computer	21	95.5%
Laptop computer	13	59.1%
Smartphone	12	54.5%
Tablet	7	31.8%
PDA	1	4.5%

Wearable technology	0	0%
Radio	4	18.2%
Other (unspecified)	0	0%

4.2.4. Report writing challenges and recommendations for improvements

Participants were asked to describe any challenges they faced in report writing for EM using an open-ended question. The results are summarised in Table 4.6. Participants consistently reported challenges associated with the time constraints of an emergency, maintaining accurate and timely information in a report, as well as the difficulty of sharing information both within the organisation and with external organisations. Specifically, some of the key challenges identified by respondents included:

- The increasing frequency of reports required by the organisation, as well as increasing detail within the report consumes more officer time with no real gains as a result (when compared to less frequent reporting and fewer details in each report);
- It takes a long time to gather and record the information into the Sitrep, which is made more difficult because of the need to check the information for truth and currency;
- There is a challenge in coordinating deadlines and obtaining information from the various local government agencies who use different systems.

Table 4.7: Challenges in completing situation reports.

<i>Time</i>
“It takes a long time to gather and record the information into the SITREP and the need to check the truth and currency of the information.”
“Meeting time lines and obtaining information from various local governments who use different systems i.e. some use the Guardian system and some don't have anything.”
“Taking time away to complete reporting while operations still require completion”
<i>Information accuracy and currency</i>
“As the reports may be used in future legal actions (e.g. in the case of a serious injury or death), they must be completed with accuracy and contain all relevant information.”
“Compiling conflicting information from different users.”

“The sitreps are out of date by the time that they are submitted. The situation in disaster events is dynamic and evolving.”

Participants were asked to rate the potential benefits and usefulness of several features, including synchronous collaborative report writing, convergence of information, and support for mobile devices. Participants were also asked to identify any additional features that would be useful in a new web-based report writing system for EM. According to the responses, participants indicated that report writing systems could be improved by enabling multiple authors to work on the same report at distributed locations. Similarly, the majority of participants indicated that report writing systems would improve in usefulness if multiple authors could work on the same document from different locations. Further, participants indicated that robustness to poor or high latency network environments would be a useful feature in a collaborative report writing system, based on the potential for network systems to be strained or affected by a disaster event. Similarly, the majority of participants rated the need for long battery life as an important feature for a collaborative report writing system. Finally, the majority of participants rated highly the need for technology that supports both desktop and mobile hardware for collaborative report writing. These results are summarised in Figure 4.2 in terms of average response and standard deviation (SD).

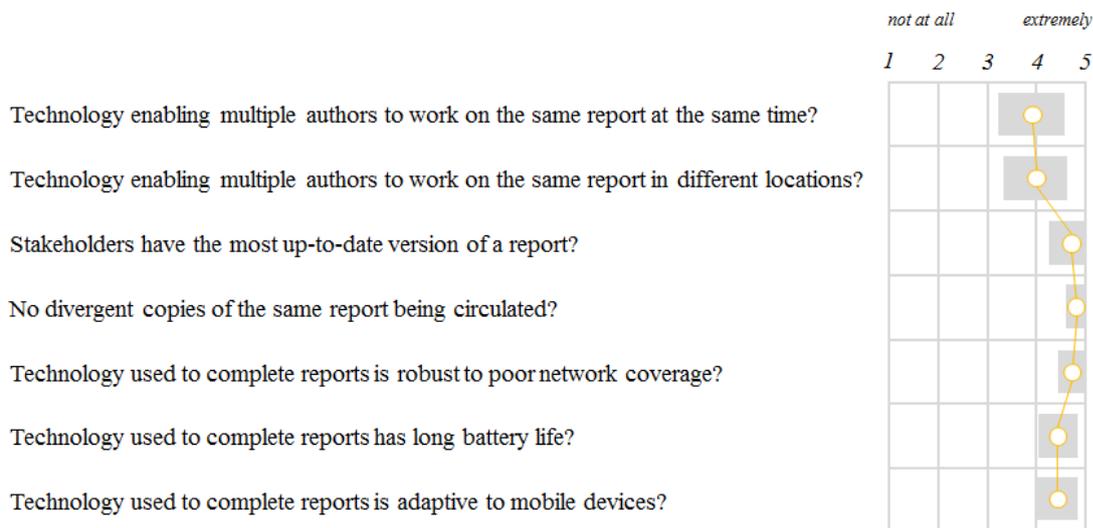


Figure 4.2: Mean and SD of participants’ usefulness rating of key collaborative features.

In addition to the participants rating the benefits of the features shown in Figure 4.2, participants were asked to suggest additional features that would benefit a new collaborative report writing system. The results are presented in Table 4.7, with a description of the suggested

features and the number of participants that suggested this type of feature (out of 22 total participants). The results are presented in order of the number of votes from most to least.

Table 4.8: Suggested features for collaborative report writing system.

<i>#</i>	<i>Description of feature</i>	<i>Frequency</i>	<i>% of Sample</i>
1	Highlighting where each author is currently interacting with the report	16	72.7%
2	Authors have the ability to lock sections of a report, so only they can edit that section while locked (prevent conflicting changes)	15	68.2%
3	Groups of authors have the ability to lock sections of a report, so only members of that group can edit a section while locked	12	54.5%
4	An administrator who oversees the report	14	63.6%
5	Ability to review changes to the report over time (history of edits)	16	72.7%
6	A map depicting where each author is located when they are editing the report	4	18.2%
7	Support for contributing to the report using a mobile device (e.g. phone or tablet)	14	63.6%
8	Real-time, read-only access for other stakeholders that need to read the report (updates in real-time)	16	72.7%
9	Reporting feature, where a selected version of the report is pushed to stakeholders as a read-only copy (does not update in real-time)	10	45.5%
10	An informal chat feature to allow collaborators to discuss the report while it is being written	8	36.4%
11	Information on the identity of each user contributing to the report	11	50.0%
12	Information on when content was added to the report	13	59.1%
13	Ability to see who contributed what content to the report	16	72.7%

14	Ability to see which areas of the report are being edited the most frequently	7	31.8%
15	Ability to see which areas of the report are being edited the least frequently	3	13.6%
16	Ability to contribute photos or other media	16	72.7%
17	Automated tools that extract important content from the report and display as a summary	15	68.2%
18	Integration with social media (e.g. Twitter, Facebook)	5	22.7%
19	Other (unspecified)	0	0%

4.3. Personas

Based on the results of the survey, several personas were created to determine the primary types of end-user for the collaborative report writing artefact developed in this thesis. Personas are a user-centred design method whereby fictional archetypes are written that represent the distinct behaviours, goals and motivations of a system's target users [63]. The personas are not required to be entirely accurate, only to determine broadly the types of users who might engage with the system. Table 4.8 presents the four main personas that were generated based on the survey results, including field officer, administrative support, senior positions in EM management, and executive level. These personas differ significantly on their primary use of the system, report writing style, challenges, and primary use of report writing systems. Field officers report difficult in sharing information from their offsite location. Admin support and mid to senior management instead report challenges with compiling information and the time pressures involved in report dissemination to key stakeholders. Finally, senior management reports challenges with the time constraints that are involved in decision-making during a disaster. These personas highlight the need for the artefact to support diverse user types with competing priorities.

Table 4.9: Personas for the collaborative report writing system based on survey results.

<i>Persona #1</i>	<i>Field officer / first responder</i>
EM experience	< 5 years
Previous roles	-

Primary communication methods	Mobile, radio, face-to-face meetings
Primary use of system	Provide reports and status updates to senior officers from the field
Report-writing style	Single author, same place same time
Report-writing challenge/s	Difficulty sharing information
<i>Persona #2</i>	<i>Administrative support</i>
EM experience	< 5 years
Previous roles	-
Primary communication methods	Mobile, email, teleconferencing face-to-face meetings
Primary use of system	Support senior management decision-making and assist information sharing
Report-writing style	Single author, different places same time
Report-writing challenge/s	Time constraints, sharing information, version control of the document
<i>Persona #3</i>	<i>Mid to senior positions in EM organisations</i>
EM experience	> 5 years, < 20 years
Previous roles	Field officers, response crew
Primary communication methods	Mobile, email, face-to-face meetings
Primary use of system	Compile information in report, in collaboration with team
Report-writing style	Sequential or parallel, different places same time
Report-writing challenge/s	Real-time collaboration, time constraints of completing the report, collecting information from the field
<i>Persona #4</i>	<i>Director of EM organisation</i>

EM experience	> 20 years
Previous roles	Field officers, response crew, social support, engineers
Primary communication methods	Mobile, email, face-to-face meetings
Primary use of system	Manage team of experts and officers to compile and disseminate timely and accurate report
Report-writing style	Single author, different place same time
Report-writing challenge/s	Time constraints required for effective decision-making

4.4. Summary and list of requirements

In this chapter, the results of a survey of 22 EM practitioners from remote Queensland, Australia were presented. These participants were asked questions about their experience with EM report writing, current collaborative practices, challenges, and potential improvements that could be afforded by a specialised collaborative report writing system for EM. The results indicate that a range of diverse methods are currently used to communicate within and between organisations, but a single author approach dominates the task of report writing with asynchronous tools such as Microsoft Word. Participants suggested that this arrangement is easy to use and familiar, but requires double-handling of information and means that information must be manually pushed out to stakeholders because the document is not live.

The biggest challenges suggested by the respondents of the survey for report writing include time constraints, information accuracy and currency, as well as challenges in sharing the information. Collaborative software with real-time synchronisation can overcome these challenges, so the remaining responses were used to develop a list of features that should be included in a report writing framework. These features can be categorised as follows:

- *Real-time synchronisation features*: Participants rated highly features such as having multiple authors working at the same time in different locations on an up-to-date and convergent report. These features can be provided by modern synchronisation techniques [11]. Chapter 5 provides an overview of synchronisation techniques to determine which method is best suited for the case of collaborative report writing for EM.
- *Locking features*: Participants rated single-author locking as important, as well as the ability for stakeholders to view a read-only version of a report that is updated in real-time as changes are made. Additionally, some users suggested that locking should be

flexible so that predefined groups of users can make edits to locked sections of the document. These features are explored in Chapter 6.

- *User attribution features*: Participants suggested features that track the identity of authors, their contributions, and history of edits as being important features. These features are explored in Chapter 7.
- *Multisynchronous features*: Features including support for mobile devices (e.g. smartphones and tablets), robustness to poor network environments and high latency networks, as well as improvements to reduce strain on battery life can be classified under multisynchronous as these features aim to allow users to collaborate under different conditions. These features are explored in Chapter 8.
- *Other features*: Additional features that do not fit under the previous categories include techniques to extract keywords and provide automated summarisation, communication features such as informal chat, incorporation with social media, and the ability to add photos and other media. As the scope of this thesis is limited primarily to text synchronisation methods to support collaborative report writing for EM, some of these features are not explored thoroughly in this thesis, though some are explored in Chapter 9.

Thus, based on the results of the survey presented in this chapter, it is clear that the current state of report writing for EM can be improved by using ICT. Specifically, collaborative technologies based on both web and mobile platforms may have inherent benefits for EM reporting by providing up-to-date and accurate information to all stakeholders in a central location. Thus, the remainder of this thesis focuses on the development of a collaborative report writing framework to support EM practitioners in compiling and sharing reports on disasters.

5. Selecting the System Architecture

This chapter aims to select an appropriate synchronisation architecture on which to develop the collaborative EM report writing artefact. The chapter provides an overview of the state of research into text synchronisation techniques, with a specific focus on supporting web-based collaborative text-editing. The techniques were evaluated on their support for the required features determined by the users surveyed in Chapter 4. The chapter concludes by selecting the diffsync technique and specifying the additional features that need to be developed to achieve the research artefact.

5.1. Overview

This chapter provides a review of the literature on text synchronisation techniques with the goal of determining the best technology on which to build the collaborative EM report writing framework. In Chapter 4, a list of requirements for a remote EM collaborative report writing system was identified based on a survey of EM practitioners. To reiterate, these requirements were categorised as *real-time synchronisation* features, *locking* features, *user attribution* features, *multisynchronous editing* and *crowdsourcing* features. In order to develop a research artefact to support the research objectives, the first step is to select a synchronisation architecture on which to build the artefact.

This chapter is structured as follows. Section 5.2 provides a background on the challenges in implementing text synchronisation. This is followed by an overview of the main text synchronisation techniques in the literature, including pessimistic methods, wikis, merging techniques, state-based techniques, and edit-based techniques. The benefits and limitations of each technique are considered in detail. Finally, Section 5.3 concludes with a summary of the features supported by each synchronisation technique, resulting in a recommendation that the diffsync technique is most suited for developing the research artefact.

5.2. Text synchronisation

Text synchronisation describes any technology that keeps text synchronised in a collaborative system. It is important for the development of a collaborative EM report writing system as such reports need to be synchronised between the key stakeholders responding to the disaster. When developing a text synchronisation technique, developers need to ensure that the technique maintains consistency and convergence of the shared information between users, as well as handling any synchronisation conflicts (e.g. mutually exclusive edits to the same area of a report). Further, it is important to consider workspace awareness factors to ensure that users are identified within the shared workspace in terms of their intentions and actions. These factors can help to maintain timeliness and accuracy of shared information, which is important for collaborative EM report writing.

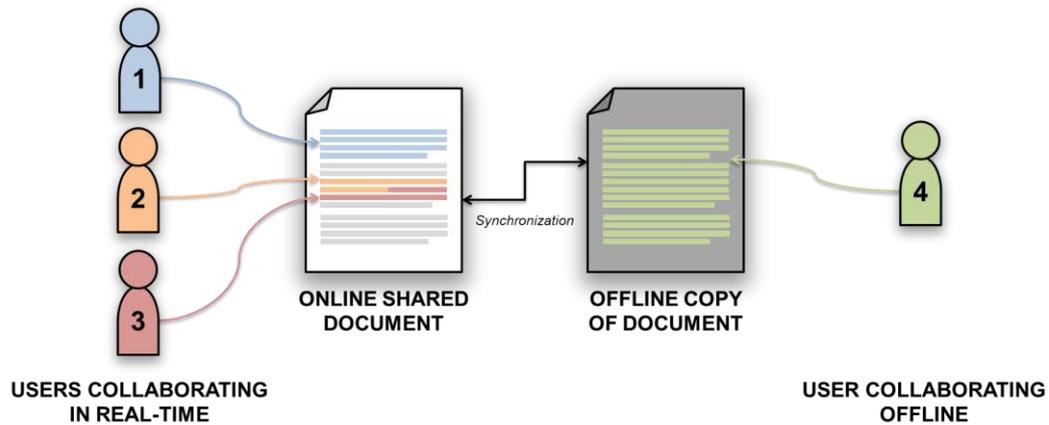


Figure 5.1: A typical collaborative editing scenario involving four users.

Synchronisation techniques must account for the different types of users and collaboration styles that can exist within a collaborative report writing scenario. Figure 5.1 illustrates a collaborative text-editing scenario that involves four users: three of these users are collaborating in real-time, while a fourth user is editing a copy of the same document while offline (which implies that this user is not editing in real-time). The users are working on a shared text document that is being synchronised by a central server. Each user's contributions to the document are denoted by the colour, for example User 1 contributes the blue changes, whereas User 3 contributes the red changes.

The scenario illustrates the core challenges inherent with collaborative editing systems: consistency, conflict management, convergence, user attribution, and support for offline or multisynchronous editing. Firstly, consider the collaboration taking part between users 2 and 3. Both of these users are simultaneously editing the same section of the document and it is unlikely that their contributions will be identical. This means that conflicts will arise when the server attempts to synchronise these edits to produce a converged copy of the document. One solution to this problem is to implement locking a technique that ensures data integrity by prohibiting concurrent conflicting updates on a shared document [66]. Another solution is to implement concurrency control algorithms [31] or offload conflict resolution to the user [67].

The next major challenge illustrated by this scenario is user attribution. User attribution refers to the implementation of methods used to distinguish contributions by user. In this scenario, the contributions made by each user are indicated clearly by a unique colour. However, consider a situation in which every user's contribution was denoted by a single colour. In a small scenario with only two collaborators, this is not a major problem as each user can easily identify who wrote each section. However, in larger scenarios (e.g. 100 collaborators) user attribution can improve the nature of the collaboration by providing each user with awareness of other user's actions [68], and can allow for additional features such as rolling back a certain user's edits [24].

Another important consideration for modern collaborative editing applications is providing support for users to continue collaborating whilst in multisynchronous, offline or unreliable network situations. In scenario depicted by Figure 5.1, User 4 is working on a copy of the shared document whilst experiencing network disconnection, and it is clear that this user has made changes to the entire document. This could pose a challenge at a later time when synchronisation occurs due to the subsequent edits made by the real-time users. While it is not possible to support *real-time* collaboration without a stable network connection, users should still be able to make changes to a document on a local device and synchronise those changes at a later time, either when a stable internet connection becomes available, or in the case of multisynchronous editing, when the user is ready to synchronise. However, the primary challenge facing such scenarios is that individual copies of a document shared amongst offline users can diverge rapidly, increasing the complexity of synchronisation when it does occur.

A primary example of a scenario in which offline and disconnected collaboration becomes an issue is field workers during an EM situation, due to the high possibility of intermittent network connectivity or high latency on cellular networks during a disaster. Users should be able to collaborate on work at anytime, anywhere, with a pocket-sized device, however high latency or poor infrastructure can make this difficult. Researchers have noted the benefits of allowing multisynchronous editing on mobile devices [69]. For example, providing support for mobile collaboration improves convenience and flexibility for users, such that group work can be completed at anytime, anywhere with a convenient device [70]. However, it is important to acknowledge the potential limited network connectivity that can occur on the mobile platform, as well as the wide range of devices with different memory constraints and processing power.

When a document is changed during a disconnected period, its content will become divergent from the copy of the document stored on a server being edited by other users in real-time. This means that as the number of offline collaborators increases, so too does the potential number of divergent copies of the document. For this reason, applications that support asynchronous collaboration must be well equipped to handle conflict resolution at synchronisation time while maintaining semantic consistency and user intention [71]. Typically, real-time collaborative applications are powered by optimistic algorithms (such as OT). There are challenges and problems that can arise when attempting to convert real-time applications to support synchronisation in multisynchronous settings. The leading techniques that are used to power text synchronisation and collaborative editing are discussed in the next section focusing on strengths and weaknesses.

5.3. Current text synchronisation techniques

5.3.1 Pessimistic methods

The simplest method for avoiding divergent outcomes in a collaborative editing scenario is to

only allow one user to make changes to a shared document at a time, with other users blocked from participating until that user is finished. Pessimistic concurrency control mechanisms are named as such because they may prevent transactions from executing concurrently, even in situations where there is no risk of a synchronisation conflict or anomaly [22]. In pessimistic synchronisation, a conflict refers to the concurrent execution of two transactions that may generate an inconsistent state for the shared text [22]. Locking is often used when critical data is involved, such that if edit conflicts pose a threat to the consistency of the data (e.g. data could easily be lost), it is better to avoid such conflicts altogether by locking edits to a single user at once. A simple example of document locking can be encountered when two users attempt to edit a Microsoft Word document that is shared on a local server – Microsoft Word locks the document for editing to the first user who accessed it, and any subsequent users who open the document are provided with read-only access until the first user is finished with the document and closes it.

Pessimistic concurrency control mechanisms are the simplest to implement, as content is only ever edited by one user at a time (e.g. the first user to open the document on a network) [24]. This means that there are never any synchronisation conflicts and additional techniques to handle convergence and maintain consistency are not required. However, the major limitation of pessimistic controls is that it is not an efficient method of collaboration, due to the extra time required to facilitate “turn taking”. While pessimistic systems can be set up to allow multiple users to edit different sections of a single document in real-time (by dynamically locking and releasing subsections of a shared document [24], such as paragraphs), this is not suitable for situations that require more granular collaboration.

Another limitation of pessimistic concurrency methods for achieving real-time collaboration is that such systems may not be robust on unreliable network connections. Typically, the transactions made by a user within a locked document or locked section of a document are not updated until the user is finished and that section or document becomes unlocked. Thus, if a user is working on a large section of the document (hence locking that section), the edits may not be propagated for a significant length of time and other users will not be able to participate.

5.3.2. Shared Content Management Systems and Wikis

Another category of collaborative editing is shared content management systems. The term “content management system” (CMS) [72] refers to a computer program that allows for the publishing, editing, and updating of content from a centralised interface, with this content being distributed through a public platform (such as a website or mobile application). The benefits of a CMS include support for dynamic content and also allow for non-skilled users to make changes to the content displayed by a technical system. A CMS can be utilised by single-user applications as well as multi-user collaboration.

One example of a CMS for collaborative use is a shared wiki. A wiki, first introduced in

2002 by Cunningham [73], is defined as an application that allows for users to add, modify, and delete content in collaboration with other users [74]. Typically wikis are used to implement knowledge management systems, community websites, and intranets. The methods used to edit content on a wiki page can be compared to a revision-control system, as some wikis store each version of the content to allow for future rollbacks (in the case of vandalism or mistakes). However, this does not indicate that user attribution is necessary in wikis. Some wiki implementations require user authorisation for editing privileges, whereas other implementations allow for any visitor to contribute changes. A large-scale example of a shared wiki system is Wikipedia [75]. Wikipedia allows for outside editing, which means that any user with access to the website can contribute content, except in the case of sensitive or vandalism-prone pages that enforce protection (locked to registered users or administrators). According to Cohen [76], Wikipedia has 500 million unique visitors each month, with roughly 75,000 of those visitors contributing edits to the articles.

Due to the large-scale, open nature of collaboration and the freedom afforded to users of Wikipedia, there have been concerns raised about the quality of writing, the amount of vandalism, and the accuracy of information. While these challenges are not inherently algorithmic-based, researchers have investigated the use of data mining practises to automate processes and predict the nature of edits. For example, researchers have proposed solutions to detect vandalism [77], [78], predict the quality of edits made [79], and automate the outcome of page deletion requests by predicting the outcome (e.g. delete or not delete) based on the content of the page [80].

5.3.3. Merging

Merging refers to another category of synchronisation techniques that are used to reconcile changes made to a shared collection of files. Unlike the previous two categories of synchronisation techniques (pessimistic and wikis), merging algorithms are divergent in that multiple copies of a shared document can exist simultaneously where they are edited locally and synchronised at a later time. For this reason, merging algorithms are required to implement manual or automatic conflict management to ensure that the results of a merge are accurate. Examples of applications that use merging techniques for synchronisation are version control systems such as GitHub.

There are two primary types of merging algorithms that are commonly used to develop groupware: two-way and three-way merge algorithms. First, two-way merging attempts to merge two versions of a document without relying on the common ancestor from which both versions of the document originated. In contrast, three-way merging does utilise the information derived from the common ancestor. Three-way merging can detect more conflicts than in two-way merging because more information is available. For this reason, three-way merge is used by more applications [81]. Figure 5.2 illustrates three-way merging, which consists of three steps [24]:

1. Clients send the contents of the local document to the server;
2. The server performs a three-way merge, extracting the changes (differences) from the user's document (revised document v1a) and merging them with changes made by other users (revised document v1b), based on the common ancestor (v1);
3. A new copy of the document (merged document) is generated and sent back to the client.

The differences between two-way merge and three-way merge can be demonstrated with Figure 5.2. First, two-way merge would only compare the differences between the two document revisions (v1a and v1b). Depending on the differences made to each document, it could be difficult for the algorithm to resolve conflicts. For example, if revision 1a made a change on line 4 of the document, and line 4 of revision 1b had contained completely different content, the system would be unable to determine the cause, i.e. a new line could have been added earlier in the document, or the existing line could have been replaced [81]. Clearly this challenge is overcome by using three-way merging, because the algorithm can compare each line against the common ancestor to determine if a new line has been added or the existing line has been replaced. This additional information can then be used to produce a document merge that incorporates all of the changes, insertions, and deletions from both version 1a and 1b. It should be noted that differences between text files are determined by using a *diff* algorithm (similar to the methods used by *diffsync*, described later in this chapter). Typically the *Unix diff* algorithm [82], [83] is used to support two-way merging while the *Unix diff3* algorithm is used for three-way merging [81].

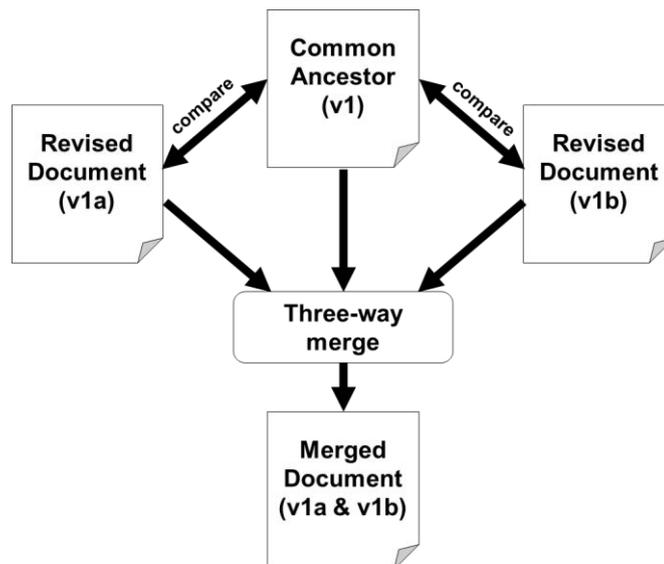


Figure 5.2: Three-way merge (adapted from [24], [84]).

Despite the fact that merging algorithms have been in development since the 1990s, conflict resolution remains a primary challenge. There are two methods for resolving conflicts that occur during a merge: automatic merging and manual merging. Automatic merging is the least disruptive to users, as changes are reconciled automatically without the requirement of additional user input. However, this can be a relatively complex problem to solve, and achieving the correct output is dependent on the number of changes made to a divergent copy before synchronisation occurs. For this reason, many implementations resort to manual conflict management. That is, when a conflict is detected by the system it is flagged for manual review. The major limitation of this method is that it can be time consuming and may require the user to have expert knowledge of the data.

In the scope of the mobile platform, server-based three-way merges have poor scalability for real-time collaboration across high latency networks [24]. This is because three-way merge is a half-duplex system. If a user makes new changes to a document at any time during the synchronisation process, the client must discard the newly received version of the document and begin the synchronisation process again when user input stops [24]. For this reason, other algorithms have been devised that are able to handle real-time collaboration, including state-based techniques such as diffsync.

5.3.4. Edit-based techniques

Edit-based techniques refers to any text synchronisation algorithm that represents changes in the form of single operations. This is in direct contrast to the merging algorithms discussed in the previous section which rely on finding only the differences between two copies of a shared document. This functionality means that edit-based algorithms can be used to store a fine-grain history of every change made to a document and attribute each of these operations to a specific user.

Edit-based algorithms attempt to capture all of the user actions made to shared data such as inserting and deleting text in a shared document, and mirror those changes across a network to collaborators [24], [85], [86]. Figure 5.3 displays the progression of an edit-based collaborative editing task between three independent sites (Site 1, Site 2, and Site 3) over time. As displayed, a shared document is replicated locally from the server at each of the three sites. Collaborators at each site perform “operations” on document and these operations are propagated to all other sites. When each site receives an update from another site via a server, the local document merges the new updates with the current state of the local document. While all sites receive all updates in this example, the order of operations received and merged locally is different for each site. This means that all users will end up with a divergent document.

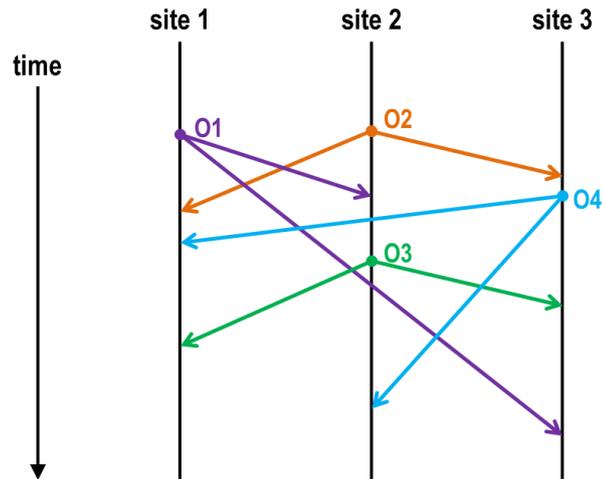


Figure 5.3: A common scenario of distributed document editing (adapted from [71]).

Consider the example in Figure 5.4, which is the same collaborative editing scenario as Figure 5.3 with specific operations and document states defined. Note that the index refers to a position in the text content, and that indexing begins at zero. All three sites begin the process with identical replicas of a document, in this case the string “cat”, which is edited by each site throughout the duration of the collaborative editing session. Site 1 executes operation 1 (**O1**) which inserts an ‘h’ character at index 1 of the current document. Site 2 first executes operation 2 (**O2**) which deletes the character at index 0, and then later executes operation 3 (**O3**) which inserts the string “dog” at index 1. Site 3 executes operation 4 (**O4**) which deletes the character at index 1.

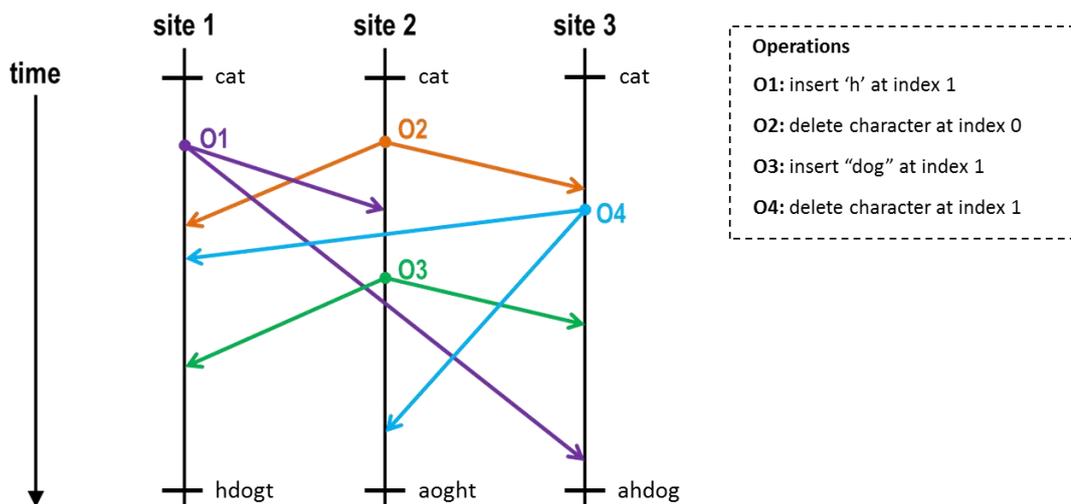


Figure 5.4: Initial state and final document state for 3 sites based on four operations.

Note that each site executes its own operations locally first, and then propagates the changes out to all other sites. However, these updates are not always received immediately by the other sites, and are sometimes preceded by local updates. This means that each site executes both local

and remote updates in a different sequence to other sites, resulting in divergent results at all three sites by the end of the process. For example, site 1 executes **O1** first and the result is “chat”. Next, operation **O2** is received and the first character is deleted, resulting in “hat”. Then **O4** is received deletes the character at index 1, resulting in “ht”. Finally, **O3** is executed at site 1 and the result is “hdogt”. This result is not consistent with the final outcomes at the other sites, so clearly this is not an ideal situation.

This scenario illustrates the challenge of consistency maintenance in edit-based synchronisation techniques. Not only is it important for all collaborators to receive and propagate updates between other sites, but those updates must also preserve the intentions of the user and ensure correctness is maintained at all sites. This is referred to as *consistency maintenance*, which is the ability of a collaborative editing algorithm to maintain consistency across all shared replicas despite differences in the order of operation.

A popular edit-based text synchronisation is Operational Transformation (OT) [87]. OT can handle both unstructured and structured documents including source code, text files, and XML [33], [71]. OT has been a popular choice for supporting consistency maintenance in many collaborative text editing applications since the algorithm was first introduced, including GROVE [31], Jupiter [88], and Google Drive [7]. The first OT system was proposed in 1989 by Ellis and Gibbs [31] to support concurrency control in real-time collaborative editing of plain text documents [89]. In the following decades, OT has been improved and its capabilities expanded to support real-time collaborative editing in many problem domains, including 3D design tools [90]–[92], spreadsheets [93], word processing [94], and graphics editing systems [95], [96].

The most differentiating feature of OT from other synchronisation techniques is the use of primitive insert and delete actions in place of regular write actions [87], [97]. While other update actions may appear more complex (e.g. setting text to bold, cut-paste, and so on), they can often be reduced to a combination of the insert action (e.g. insert HTML or style elements to set text to bold) and the delete action (e.g. cut is a delete action, paste is an insert action). This avoids the need for overwriting data and allows OT to deviate from the order of execution of updates, without losing consistency. In recent years, researchers have proposed adding an “update” action to implementations of OT to support the richer data types and comprehensive nature of modern text editors [94]. Another differentiating property of OT is that objects in the data structure are uniquely identified by a position within the data structure (for example a linear array with indexing) [87]. This feature, coupled with the primitive insert and delete operations, allows OT techniques to handle fine-grained updates, merging simultaneous edits, and handling conflicts in real-time.

Figure 5.5 represents the typical operation of an OT implementation when two independent users (Site 1 and Site 2) are working on a shared document [87]. In the beginning of this scenario,

a file containing the simple string “cat” is replicated at both sites. First, the user at Site 1 performs an insert operation [$O_1 = \text{ins}(1,h)$] which inserts an ‘h’ character at position 1. This changes the content of the local document at Site 1 to “chat”. At the same time, the user at Site 2 performs a delete operation [$O_2 = \text{del}(0)$], which will delete the character ‘c’ at position 0, resulting in the content “at” for Site 2. Both of these operations are propagated to all of the other sites.

The next step is synchronisation. First, Site 1 is required to integrate the remote operation O_2 with its local operation O_1 . Upon inspecting the positions of the updates, it is determined that O_1 inserts on the right side of where O_2 deletes, so the updates do not affect each other and can be executed as-is resulting in content “hat”. In contrast, Site 2 cannot execute the operations as-is, because the execution of O_2 will invalidate the position of O_1 . Therefore, operation O_1 needs to be transformed by shifting its position to $O_1' = \text{insert}(0,h)$, which will correctly execute according to the intention of Site 1 resulting in content of “cat” at Site 2. While the operations at both sites followed a different order of execution, transformation helps to ensure that both sites correctly converge the operations and result in the same outcome.

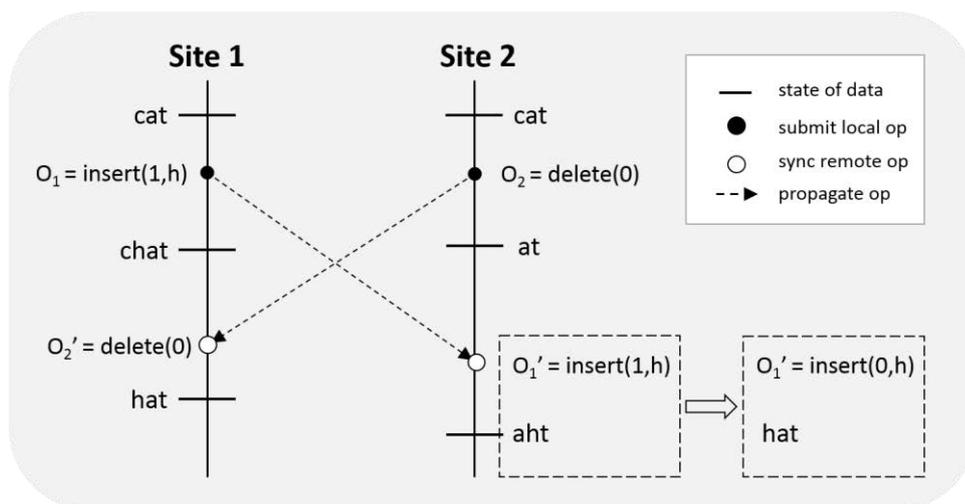


Figure 5.5: Demonstration of the OT algorithm (adapted from [87]).

Many text synchronisation techniques enforce the same total order of operations at all sites to ensure that all shared data is converged in the same state. However, it is clear that enforcing the same order operation would introduce additional challenges. If the total order is that O_1 precedes O_2 , then Site 2 must first undo the O_2 operation and then execute O_1 and O_2 in order. On the other hand, if the total order is that O_2 precedes O_1 , Site 1 must first undo O_1 and then execute O_2 followed by O_1 . While the first example produces the correct result of “hat”, the second example will violate user intention and incorrectly result in “aht”. OT differs in that it requires the execution of operations to preserve intentions or admissibility, which means that the order of operations at each site is not important, as all sites will transform each operation according to the extra information to ensure that the correct result is achieved.

An abundance of OT variations have been proposed over the past two decades, e.g., [28], [33], [98]–[100]. Throughout this review it has been noted that there are many challenges for OT (and also general synchronisation techniques) to handle, including consistency maintenance, group undo, and conflict resolution. Some of the implementations below handle several of these challenges, while others only handle a single challenge. See Table 5.1 below for several examples.

Table 5.1: Implementations of OT and key features.

Implementation	Key feature/s	Reference/s
dOPT	consistency maintenance only	[31]
GOT	consistency maintenance only	[33]
GOTO	consistency maintenance only	[71]
Selective Undo	group undo only	[101]
AnyUndo	group undo only	[102]
adOPTed	consistency maintenance and group undo	[103]
COT	consistency maintenance and group undo	[28], [29]
SCOP	operation compression only	[104]

There are also several examples of applications that are powered by OT for tasks including collaborative programming (e.g. Subethaedit², ACE Editor³), graphics and 3D modelling (e.g. CoMaya⁴), text collaboration (e.g. Google Drive [7], EtherPad⁵), and design (e.g. Mockingbird⁶).

Research has demonstrated that OT can support a variety of features including optional locking [26], [66]. In the scope of multisynchronous editing, there has been research into optimising OT for synchronising large batches of edits that have accrued during an offline period [87]. However, as OT was primarily designed to handle real-time editing, merging of updates at a later point in time can cause semantic or synchronisation problems for users in certain scenarios.

² <http://www.codingmonkeys.de/subethaedit/>

³ <http://ace.c9.io/>

⁴ <http://cooffice.ntu.edu.sg/comaya/>

⁵ <http://etherpad.org/>

⁶ <https://gomockingbird.com/>

These problems may include [16]:

- Semantic inconsistencies as the result of two concurrent updates on the same sentence or segment of data - for example when Google Drive attempts to merge these updates the sentence can easily lose meaning and violate the intentions of both users;
- Typographical errors, such that two or more users attempt to fix the same error (e.g. a spelling mistake) by inserting a new character - for example when Google Drive attempts to merge these edits, it will insert multiple instances of the same character which is a violation of the user intention;
- The cursor position of a user can suddenly change upon the merging of another user's edits - while not a semantic error, it presents a usability problem and can result in further user errors;
- Loss of updates, which can be caused when two or more users are editing a block of data offline (e.g. a paragraph) - insertions from one user can be deleted unintentionally by another user if the second user has deleted the lines in which the former user inserted the text.

Ahmed-Nacer et al. [16] argue that these problems are inherent in multisynchronous editing environments (those that support both online and offline collaborative merging), and Google Drive does not adequately inform users about potential conflict issues or convergence results. While Drive does support for recovering previous revisions of the document, this feature does not always accurately reproduce the executed operation, which means that some user's updates are lost entirely.

The major challenge facing developers of real-time collaborative text editing systems built on OT is that the technique was primarily designed to handle real-time synchronisation on a live connection (i.e. frequent changes are merged and propagated in real-time). For this reason, most OT algorithms focus on transforming and merging single operations, rather than focusing on longer sequences [87]. On the mobile platform, there can be arbitrary lengths of time where a network connection is unavailable, resulting in larger sequences of updates that must be merged at a later time. This problem compounds as the number of users and activity increases, and in extreme scenarios the algorithm may be required to converge thousands of edits while attempting to manage conflicts and preserve user intention.

Shao, Li and Gu [87], [97] conducted a study on the efficiency of OT running in "offline mode", and found that most existing algorithms are not suitable for merging long sequences of updates, similar to those that will occur after a long period of editing in "offline mode". For example, they found that the state-of-the-art OT algorithm takes $O(n^2)$ to merge one remote update and $O(n^3)$ to merge a longer sequence, where n is the size of the local operation (update) log. This is clearly suboptimal for the mobile platform, or any situation in which a large sequence

of operations must be merged. The authors of [87], [97] present an optimised OT-based technique that supports both mobile and asynchronous collaboration, which improves the time complexity of the state-of-the-art from $O(n^3)$ to $O(n)$, with n representing the size of both sequences (which are comparable). For example, the optimised algorithm was able to integrate two sequences of 3,000 operations each in ~ 1.5 seconds. In contrast, an OT algorithm developed in 2005 [105] took 59 minutes to complete the same task. This efficiency was achieved by maintaining a special order on operations in each sequence and exploiting it to merge any two concurrent sequences in linear time. The algorithm is also well grounded on correctness as it is based on admissibility theory (see [106]).

However, the research conducted by Shao, Li and Gu [87], [97] is limited in that it only focuses on the efficiency aspect of the proposed technique and does not research its effectiveness in supporting practical applications, such as collaborative EM report writing. In addition, the proposed technique does not currently support selective undo of any operation or sequence, which means that error recovery and conflict resolution is limited. The ability to resolve conflicts is one of the most important features for collaborative editing applications. This would be especially important in an asynchronous environment (e.g. mobile) where thousands of edits are being merged simultaneously, as the chance for conflicts increases with the size of edits.

Additional techniques that have been developed to improve on the limitations of OT include Conflict-free Replicated Data Types (CRDT) approaches [107]. These approaches define abstract data types that provide a commutative set of operations to update the data, replacing the need to preserve the order of individual operations as in OT [108]. For example, André et al. [108] propose a CDRT that dynamically adjusts the granularity of the data type to provide efficient conflict resolution and remove conflicting updates. However, it should be noted that these algorithms face issues in enforcing features such as flexible locking and preserving user intention [109].

While edit-based techniques have been demonstrated to support several features that are suited to collaborative EM report writing, there are also several limitations. For example, this section outlined the fact that edit-based techniques can have convergence issues [24]. Most frameworks based on OT techniques are also subject to scalability issues due to the use of vector clocks [110].

5.3.5. Differential Synchronisation

Differential Synchronisation (diffsync) is a proposed alternative synchronisation technique to edit-based methods. Diffsync was first introduced in 2009 [24]. The technique is simple, robust, convergent, fault-tolerant, and efficient, making it a worthwhile alternative to other synchronisation techniques. It must be noted that the amount of published research on diffsync is small when compared to other techniques such as OT, thus the definition will refer extensively

to the original paper by Neil Fraser [24].

Diffsync aims to overcome the limitations and challenges facing other collaboration solutions, e.g. edit-based algorithms, three-way merges, and pessimistic methods. The premise of diffsync is that a server stores a shared document, and each collaborator (client) stores a separate, shadow copy of that document on both the server and client side, along with a local copy for editing [24], [111].

Diffsync offers the following attributes that allow it to be used in real-time collaboration applications:

- Symmetry - the algorithms runs nearly identical code on both the client-side and the server-side;
- State-based - which removes the need for the system to save history;
- it is asynchronous;
- it is suitable for unreliable or high-latency networks;
- Convergent;
- Scalable;
- Compatible with multiple types of data.

Additionally, diffsync is designed so that it can be appended to existing applications, thus introducing real-time collaboration to previously static systems [24].

There are two main operations performed by diffsync to maintain document convergence: diff calculations, to calculate the edits made by a client in comparison to the local shadow, and patch calculations to merge the diffs to the copy of the document on the server (as discussed in Section 3.3) [24], [111]. When a client makes changes to its document, the differences between the newly edited document and the latest document shadow are calculated using a diff algorithm, for example Myer's $O(nd)$ difference algorithm [112]. Once the diff is calculated, a patch containing the differences is sent to the server, which updates both the server-side shadow and the shared document that is stored on the server [111]. See [113] for examples of matching algorithms that could be used to generate patch information.

The diffsync algorithm is symmetrical in nature, as the process of communicating changes from the client to the server is identical to the process of communicating changes from the server to the client [24]. Figure 5.6 illustrates the symmetrical nature of the algorithm. The only major difference between the client and the server is the Backup Shadow stored by the server. The reason for this is that diffsync was designed to target web-based client-server configurations where the client is the only entity that can initiate a connection (thereby making it an asymmetrical connection).

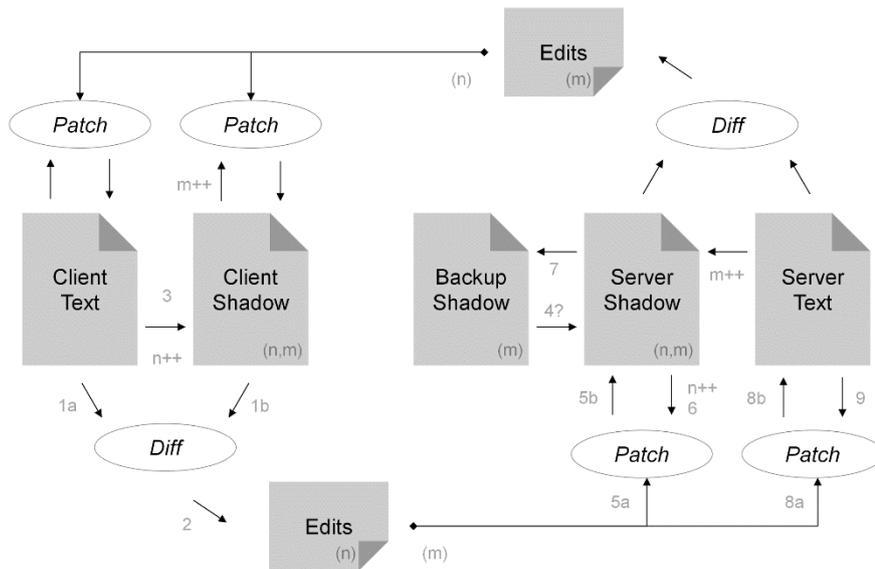


Figure 5.6: Guaranteed delivery implementation of diffsync (adapted from [24]).

There are only three possible scenarios for a connection between a client and server in the diffsync architecture. According to Fraser [24], these are: (1) client sends data but it is lost before it reaches the server; (2) client sends data to server and it is received, but server's response is lost before reaching the client; and (3) client and server complete a successful round-trip. It is not possible for the client's data to be lost and the server's data to be received in a single iteration. This is because every time the server sends data to the client, it implies that a successful connection was established from the client to the server. Otherwise, the server could enter a situation whereby it repeatedly sends packets to the client that never arrive, while also never receiving any new packets from the client [24].

In terms of scalability, diffsync can be expanded to handle a large number of clients, with each client requiring an additional server-side shadow of the document. Figure 5.7 illustrates an example of a diffsync network topology for six clients. However, this topology is not robust in terms of latency for a much larger number of clients, though Fraser [24] suggests that this can be overcome easily by storing the shared document across multiple servers and dividing the load. In the case of packet loss on the network, the edits made by the client are stored in a stack on the client-side and retransmitted on the next attempt [24]. The stack of edits (see Figure 3.6) is stored until the client receives an acknowledgement of receipt from the server.

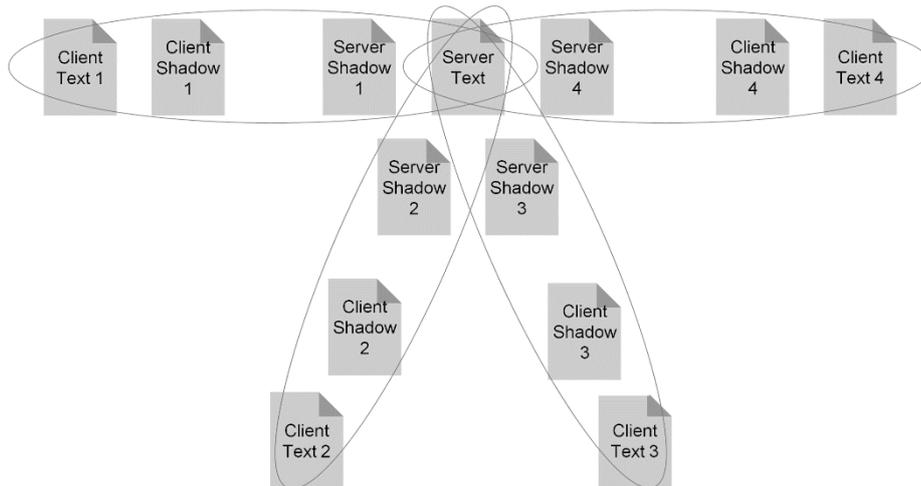


Figure 5.7: Network topology of four clients using diffsync (adapted from [24]).

Diffsync has advantages over other algorithms for collaboration. First, diffsync is relatively simple to implement when compared to other algorithms [111], and it meets the important requirements for collaborative editing (see [33]). For example, diffsync is highly responsive as the edits can be made locally and only the differences are transmitted to the server, thus making local actions in diffsync systems as efficient as single-user editors. Additionally, no locking mechanisms are required and multiple users can edit any part of the document simultaneously while still maintaining high concurrency. Finally, diffsync is able to hide the effects of communication latency to an acceptable extent, however increased latency will increase the conflict rate [111].

There are only a minimal number of examples in the literature of groupware systems based on the Differential Synchronisation algorithm. This could be attributed to its inception occurring only within the last five years, whereas other concurrency control algorithms (specifically edit-based algorithms) have been considered by researchers for several decades. Nonetheless, there are a few implementations of groupware systems based on diffsync that are worth noting. Table 5.2 lists these implementations.

Table 5.2: Implementations of collaborative editing applications based on diffsync.

<i>System Name</i>	<i>Description</i>	<i>References</i>
MobWrite	A real-time synchronisation and collaboration service used to convert single user systems into multi-user systems.	[114], [115]

Web-Based Simulation (WBS)	Based on MobWrite [114], this platform supports Advanced Distance Learning for collaborative simulation building and execution.	[116]
CoRED	A browser-based collaborative real-time code editor for Java applications, with error checking, automatic code generation and social media type features.	[111]

As everyday computing tasks continue the trend towards smartphones and other mobile devices, developers of collaborative editing programs must consider the suitability of synchronisation techniques, including diffsync. Recall that there are different challenges presented by volatile environments (such as those present in EM scenarios), including unreliable network connectivity caused by poor network coverage, roaming, and weather. The ubiquity of smartphones means that existing networks can suffer from high latency as the number of devices increases. The large number of different handsets available also means that memory and processing power can be a concern.

Further, diffsync has many benefits for supporting synchronisation on mobile devices. Firstly, diffsync is a state-based technique [24], [117] that uses a tree topology for synchronisation. Changes made to a shared document are converged on the inner nodes of the tree, allowing diffsync to capture causality without the need to use complex version modelling. In contrast to the OT algorithm, each diffsync participant is not required to store a long history of edits or operations, thus saving time in the synchronisation task, and memory resources on mobile devices. Instead, changes are detected by using the diff algorithm and propagated to peers for patching.

Diffsync is also non-blocking [24], meaning that users can continue to input new data while the system is waiting for a network response from previous update. Another benefit of diffsync for the mobile platform is that it is symmetrical on both the client-side and server-side. Fraser [24] explains this symmetry by citing the three several outcomes that are possible in a web-based client-server configuration. First, a client can send data to the server which is dropped before reaching the server. Second, the client can send data to the server, which reaches its destination, but the server's response is dropped and never reaches the client. Third, a successful round-trip will occur where the client sends a packet to the server and receives a response. This means that if a packet is lost somewhere along the way, either on the way to the server or on the way to the client, divergence of documents do not occur. The client will not delete a set of edits from its outbound stack until it receives a response from the server to indicate that the diff was received. Conversely, the server will use the Backup Shadow to temporarily store the latest version of a

document it has received from a user. If packets are lost from server to client, a new set of updates will be sent to the server (with an increased version), and the server can discard the previous backup and update the document accordingly. As such, diffsync is suitable to handle the problems posed by high-latency and unreliable networks. If a packet does not make it to the server, it is retained in a “stack of edits” for later transmission. Conversely, the server will not patch updates from a user until it receives another response from the client to confirm that the updates have propagated. It should be noted that to date, there has been no research conducted on the suitability of a diffsync-based technique for running collaborative programs specifically in a multisynchronous environment. However, it is clear that the aforementioned features of the diffsync technique [24] can offer benefits for collaborative editing applications that require support for multisynchronous collaboration, such as report writing for EM.

5.4 Selecting the system architecture

After analysing the strengths and weaknesses of different text synchronisation techniques, each technique was evaluated for its support of the collaborative EM report writing features derived in Chapter 4. This evaluation would result in the system architecture and development plan for the research artefact. Table 5.3 provides a comparison between each technique for each collaborative EM report writing features. Importantly, no single technique was found to support all of the features required by remote EM practitioners. A technique had to be selected based on its current features for supporting the needs of EM practitioners and a plan developed for implementing the remaining features.

Table 5.3: Overview of synchronisation techniques based on support for required features.

	<i>Pessimistic</i>	<i>Wikis</i>	<i>Merging</i>	<i>Edit-based</i>	<i>Diffsync</i>
<i>Real-time synchronisation</i>	✗	✗	✗	✓	✓
<i>Convergence</i>	✓	✓	✓	✗	✓
<i>Scalability</i>	✗	✓	✗	✓	✓
<i>Robust to poor network</i>	✗	✗	✗	✗	✓
<i>Locking</i>	✓	✗	✗	✓	✗
<i>User attribution</i>	✗	✓	✓	✓	✗

<i>Multisynchronous</i>	×	✓	×	✓	×
-------------------------	---	---	---	---	---

Pessimistic, wikis, merging and OT were assessed as unsuitable for remote EM collaborative report writing. Pessimistic methods are not suitable due to the lack of support for real-time synchronisation, poor scalability, and inability to support poor network environments. Similarly, wikis and merging techniques cannot provide real-time synchronisation and are robust to poor network environments. While OT has support for real-time synchronisation, locking, user attribution and multisynchronous editing, its inability to provide convergence makes it unsuitable for collaborative EM report writing. This is because information in EM reports must be accurate and a lack of convergence would introduce errors and confusion amongst EM practitioners.

Based on the weaknesses of the other synchronisation, it was decided that diffsync was the most appropriate to build the collaborative report writing framework with. Diffsync has many inherent benefits that can be utilised for real-time EM reporting, including its high scalability, robustness to poor and high latency network environments (e.g. increased load on cellular networks during an emergency or damaged infrastructure), and natural convergence (so all collaborators have an accurate view of the report at all times with no divergence) [24]. These features align with several of the key features suggested by EM practitioners (based on the survey results presented in Chapter 4 and presented in Table 5.3).

However, it should be noted that diffsync does not currently provide support for other key features suggested by the EM practitioners, based on the results of the survey presented in Chapter 4. For example, diffsync does not currently provide support for locking to provide collaborators with the choice to place temporary locks on subsections of text to claim exclusive editing privileges. Further, diffsync does not currently differentiate the edits made by different users, instead losing authorship information on the server. Finally, there are several improvements that could be made to improve the performance of diffsync for mobile users, as well as to support field workers in collaborating on a shared document in real-time alongside staff on desktop computers. Thus, chapters 6 - 9 of this thesis explore techniques for implementing these features into diffsync with the overall goal of developing a framework to support collaborative report writing for EM.

5.5. Summary

In this chapter, five techniques for text synchronisation were evaluated to determine an appropriate architecture for building a collaborative EM report writing artefact. These techniques include pessimistic methods, wikis, three-way merge, edit-based techniques, and diffsync. The techniques were evaluated based on the needs identified by the EM practitioners surveyed in Chapter 4, including real-time synchronisation, convergence, scalability, robustness to poor

network environments, support locking, user attribution, and multisynchronous editing.

The evaluation concluded that diffsync was the most appropriate text synchronisation technique for developing a collaborative EM report writing artefact. Diffsync supports real-time synchronisation, convergence, scalability, and is robust to poor network environments. These are all important features for EM. However, it was also noted that diffsync lacks support for locking, user attribution, multisynchronous editing, and support for diverse user types. As such, Chapters 6, 7, 8, and 9 continue on from this chapter by developing new methods and frameworks within diffsync to support these features.

6. Flexible Locking Techniques for Diffsync

Chapter 6 describes the flexible locking framework that was implemented in diffsync. Flexible locking was determined to be a key requirement for the collaborative EM report writing framework in Chapter 4. As diffsync does not currently support flexible locking, a new framework is implemented. This chapter discusses the design issues surrounding flexible locking, alternative approaches for flexible locking that exist to inspire the development of the current technique, as well as a consideration on how the proposed framework meets the challenges of locking.

6.1. Overview

This chapter provides an overview of the development of a flexible locking framework within the diffsync technique. Locking is a technique originally used by distributed systems and database environments to maintain the integrity of shared data [26]. As specified in Chapter 4, flexible locking is a key requirement to support collaborative report writing for remote EM practitioners. Chapter 5 selected diffsync as the system architecture for developing the research artefact, however it was noted that diffsync does not support flexible locking. Thus, this chapter aims to demonstrate a new framework that supports flexible locking in diffsync.

This chapter begins by providing a background on flexible locking and the challenges involved in its implementation. Next, the benefits of flexible locking for collaborative EM report writing are considered. This is followed by a description of the flexible locking framework and implementation of a web-based system to demonstrate the features of the framework. This implementation is evaluated on its support for the key challenges of locking. Finally, the chapter concludes with a summary of the flexible locking framework.

6.2. Flexible locking

Locking is a technique used to maintain integrity of shared content in a collaborative system [12]. Traditional locking systems maintain data consistency by permitting only a single user to edit a shared document at any one time, making it read-only for other users until editing is completed [22]. This form of locking can be considered extreme since it results in disjointed collaboration. Hence flexible methods of locking are required to enable continuity and flexibility in collaborative software [66], as described in this chapter.

Flexible locking defines a locking framework that is both optional and dynamic in nature. Optional locking allows users to retain control over when and where locks are used in a shared environment [26]. Existing research identifies optional locking as an important feature to assist in maintaining data integrity in collaborative editing scenarios by enforcing semantic consistency [66], [111]. Dynamic locking is a similar technique that allows for locks to work in synchronous editing environments [13]. The nature of real-time editing with multiple users means that the content of a shared document will constantly change. To handle this, dynamic locks automatically adjust in size (e.g. increase or decrease) to compensate for changes made within the lock or in

other areas of the document.

A simple example of document locking can be encountered when two users attempt to edit a Microsoft Word document that is shared on a local server – Microsoft Word locks the document for editing to the first user who accessed it, and any subsequent users who open the document are provided with read-only access until the first user is finished with the document and closes it [24]. While this approach may achieve consistency, it inhibits usability and does not scale well to real-time collaboration on the web. As such, researchers have developed techniques to support synchronous editing with optional and dynamic locking.

The majority of the work on optional and dynamic locking for collaborative editing comes from the field of edit-based synchronisation techniques, specifically the OT technique [118]. For example, Sun [26] proposes a scheme that is optional, non-blocking, responsive, and allows for multiple locks to be used simultaneously inside a single document editing session. Similarly, Xu et al. [13] propose Customisable and Dynamic Locking in which the locking mechanism can be adjusted with separate locking policies, and the system can “pre-lock” sections automatically based on the enforced policies and recent activity in the shared workspace.

6.3. Benefits of flexible locking for collaborative EM report writing

There are several key benefits can be derived for collaborative EM report writing. These benefits are as follows:

- Flexible locking restricts access to certain sections of a document so that only suitable experts can edit that information. For example, a senior engineer may be required to present information on the state of a bridge in the affected disaster area. By locking the appropriate section of the report, only the senior engineer can update and change the content. For other users this content will become read-only for the duration of the lock.
- Further, flexible locking ensures that important information is not missed, omitted or overwritten within a collaborative interface. When a user locks an area of text for exclusive editing that sections becomes read-only for the other users. Other users can see the locked section of text change in real-time but cannot contribute. This means that other users maintain awareness about the contents of the document but cannot mistakenly overwrite the data.
- Finally, flexible locking provides fine-grained access control to the document while still maintaining workspace awareness. Users maintain the ability to lock text at the level of document, paragraph, sentence, word or character. For all levels of locking the area still maintains a read-only state for all other users. An administrator or user with higher powers can unlock any section of the document when necessary to ensure that locks do not inhibit collaboration.

However there are many technical and usability issues that should be considered when

implementing locking functionality in collaborative applications. These issues include syntactic consistency, semantic consistency, non-blocking input, dynamic region adjustment, lock ownership, and representation of locks to users. Section 6.4 provides a brief description of these issues.

6.4. Challenges in supporting collaborative locking

6.4.1. Syntactic consistency

Syntactic consistency is achieved in collaborative editing scenarios if all collaborators have an identical view of the shared document (regardless of whether this shared view is semantically correct) [71]. There are three major problems relating to syntactic inconsistency in synchronisation, which are *divergence*, *causality-violation*, and *intention-violation* [33]. These problems are caused by the generation of different operations at each site before synchronisation occurs, and relate to the fact that operations are propagated in an arbitrary order to other sites [26]. As such, research has determined that locking in edit-based synchronisation cannot be used to maintain syntactic consistency [26].

6.4.2. Semantic consistency

Another problem that can occur in collaborative-editing scenarios is semantic inconsistency of shared content [26]. Semantic inconsistency problems relate to errors in the semantics of the shared data (e.g. grammar, programming language rules, or any other rule relating to the language of the shared text). This means that synchronisation techniques such as OT or diffsync cannot overcome these problems by simply enforcing syntactic consistency. Optional locking can be used in a complementary manner to support this functionality.

To understand the role of optional locking in maintaining semantic consistency, consider a shared document with the following text (example adapted from [26]):

The storm pass quickly.

The text has an English grammar error represented by the underlined text, such that it should read “will pass”, “passed”, “passes” or similar. The problem of semantic inconsistency will occur if two users attempt to fix this grammar error at the same time by adding mutually exclusive fixes. For example:

The storm will passed quickly.

While this fix is syntactically correct (i.e. both edits are synchronised to all users), the resulting text does not achieve the semantic intention of the application (i.e. a grammatically correct English document). However, if one user was able to lock this section of the document before editing, it would ensure that only a single semantic fix was allowed and the resulting inconsistency would be avoided.

6.4.3. Non-blocking input

Another issue to be considered is the procedure used by collaborative editing in granting lock access. Systems can be categorised as being either blocking or non-blocking. In blocking systems, the user is blocked from editing a requested lock region until the system returns a decision based on any currently active locks. While there are benefits to blocking, such as simplified handling of conflicting lock requests and edits, there can be negative implications for user experience.

In contrast, a non-blocking system will allow a user to continue editing the requested region while the system makes a decision to allow or deny the lock request. In this case, a tentative lock is granted immediately after a lock request is generated. The tentative lock will become approved or denied once the system processes it, at which point the user will either be allowed to continue editing or the user will be blocked from editing the region [26].

While non-blocking systems can provide higher responsiveness, the system must also handle conflicting lock requests from multiple users at the same time. Further, these lock requests may contain overlapping regions, for example one lock request ranging from index 0 to 16 and another lock request ranging from index 4 to 25 (see Figure 6.1). The way in which a collaborative editing system handles such a scenario depends on the design of the system and the application context.



Figure 6.1: Conflicting lock requests on a sentence.

6.4.4. Dynamic region adjustment

The nature of shared editing implies that the content of a document is frequently appended and deleted. As such, it can be assumed that when a user locks a section of the document the captured area of content may not remain static. This provides another challenge to locking - the locking system needs a mechanism to grow and shrink the locked region when a user makes changes. Previous research has provided OT with ability for dynamic lock adjustment [26].

6.4.5. Lock ownership

Lock ownership refers to the coupling of a locked region of shared text with an entity, most commonly a human user. The locking system will track the ownership of each lock and ensure that only permitted entities are allowed to change locked regions, whereas non-permitted entities are denied access to those regions. In an optional and flexible locking system, users must have

the ability to request a lock on a section of the shared data, and in turn relinquish ownership of that lock when it is no longer required.

While single-user lock ownership is a common use case, there may exist scenarios in which other lock ownership schemes are required. Alternative lock ownership schemes may include:

- Multiple-user ownership: each locked region can be owned by one or many users - this may be useful when enforcing permissions on sections of a shared document;
- Single-lock ownership: each user is strictly allowed to own one lock at a time; and
- Multiple-lock ownership: users can own one or many locks at a time.

Another consideration in lock ownership is the issue of user disconnection. As collaborative editing systems are network-based, a user may disconnect from the application without relinquishing ownership of a lock. Depending on the nature of the application and the sensitivity of the shared data there are two ways in which this scenario can be handled. Firstly, in the event that a user leaves the document, the system may automatically release ownership of any locks owned by that user. Alternatively, the system may retain the lock indefinitely, allowing a user to return to the collaborative editing environment at a later time to continue working on a locked region. Such a system would require user identification (e.g. a login system) to ensure the persistence of locks across editing sessions.

6.4.6. Representation of lock to user

While locking systems are required to maintain a logical reference to lock ownership to ensure that locking is enforced, another consideration is the representation of lock ownership to end users. This is typically a usability issue and would be considered unique to each application. Support for factors such as learnability, efficiency, memorability, error recovery, and user satisfaction can depend on the application context. Some methods that could be used to identify lock ownership to end users include disabling edit abilities on locked content, highlighting locked content with different colours, or including visual cues such as symbols.

6.5. Implementing Flexible Locking in Diffsync

After carefully considering the benefits of the diffsync technique and the issues surrounding the implementation of locking in collaborative editing applications, several techniques were developed to incorporate flexible locking within diffsync. The section of this chapter describes these approaches.

6.5.1. Optional locking methods

A conceptual framework for locking in diffsync is presented in Figure 6.2. The framework is similar to diffsync as presented by Fraser [24], however *Lock Tables* have been added to both the client-side and server-side of the synchronisation technique. These features are described in the following sections of the paper.

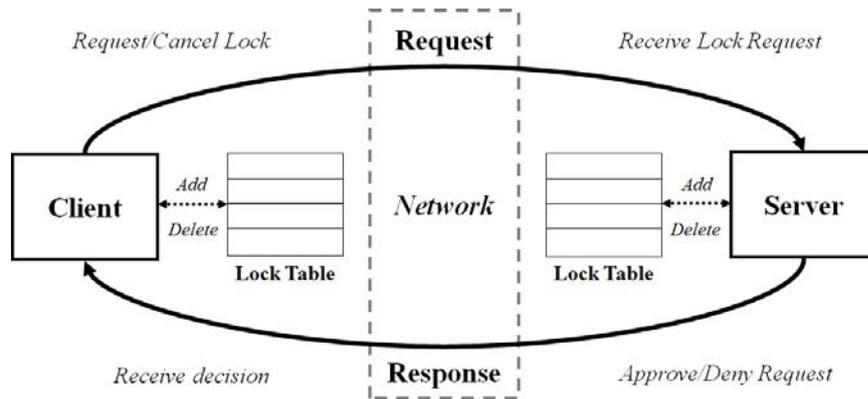


Figure 6.2: Overview of client and server-side locking methods in the implementation.

6.5.2. Locking Tables

The first and most important addition to diffsync for locking is the concept of a locking table. The locking table is a data structure that maintains a record of the current locks that exist on the shared document. In the proposed implementation the locking tables have the following structure (Figure 6.3):

<u>Global Locking Table</u>		
<u>Lock 1</u>	<u>Lock 2</u>	<u>Lock n</u>
ownerID = KxV64vhBXD groupID = A startIndex = 10 endIndex = 22	ownerID = cVpfUaBmSa groupID = A startIndex = 40 endIndex = 43	ownerID = 28FXsX3Va2 groupID = B startIndex = 60 endIndex = 100

Figure 6.3: Conceptual structure of Global Locking Table (GLT) and Lock objects.

The *ownerID* denotes a unique identifier for the entity that created the lock, e.g. each client socket receives a unique ID upon connecting to the server. In other implementations this could be tied to a login system (e.g. username) or IP address. The *groupID* denotes an identifier for arbitrary user groups and provides the system with the ability to share a single lock between multiple users. Our implementation of diffsync conceptualises plaintext as having an index-based structure, so the *startIndex* and *endIndex* indicate the locked region of text. The server maintains

a Global Locking Table (GLT) and each client maintains Local Locking Table (LLT). When a client connects to the server for the first time, a copy of the GLT is sent to the client and is copied into a LLT.

To ensure that locks are synchronised between users the existing synchronisation loop of `diffsync` has been extended by augmenting it with lock data. Thus, when a user requests a lock, this lock request is sent to the server with the normal synchronisation data including diffs. Upon receiving a lock request, the server will compare this request with the contents of the GLT. If the requested lock area does not conflict with an existing lock the lock is granted and added to the GLT.

Subsequently, the updated locking information is included in the server reply to all clients so that all clients can update their LLT. The LLT also serves a validation role in that all local edits are compared against the LLT locks. When a client makes a change, the diffs are checked against the locked areas. If a collision is detected and that lock is not owned by the current user, the edits are removed from the edit stack and the changes are reverted. Otherwise, if there are no conflicts or the edits exist within a lock owned by the user then synchronisation proceeds as normal.

When a client revokes ownership of a single lock, an unlock notification is sent to the server containing information about the relevant lock. The server then removes the lock from the GLT. Further, if a client relinquishes ownership of all locks that they own, an unlock all request is sent to the server in the next synchronisation loop and all locks tied to that particular ownerID are removed from the locking tables.

6.5.3. Dynamic lock region adjustment

The GLT mentioned in the previous section is also useful in providing support for dynamic lock region adjustment (Figure 6.4). When the edit stack is received by the server, it checks if changes are being made within a locked region. If true, a function runs to adjust the locked region by updating the values of the *startIndex* and *endIndex*, e.g. increasing or decreasing to expand or decrease a lock region respectively. This function also adjusts the positions for other affected locks, e.g. a lock exists on line 1 and another lock on line 2; the first lock inserts a new line prompting the second lock to adjust its lock to now occur on line 3.

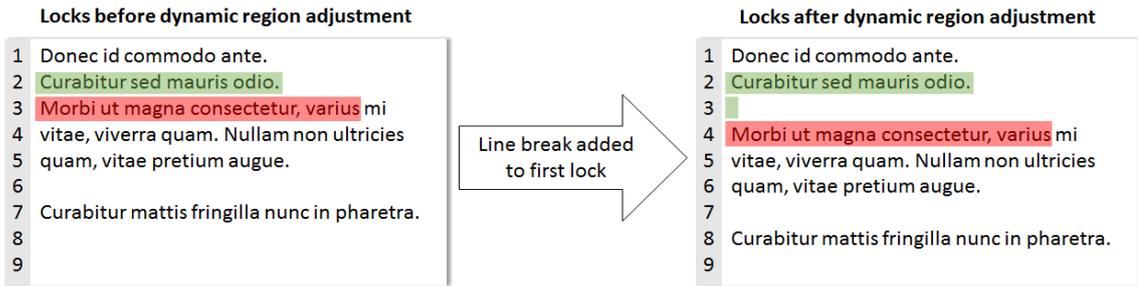


Figure 6.4: Example of dynamic region adjustment after one user adds a line break.

6.5.4. Visual representation of locks

The data stored in the LLT is also useful for providing a visual representation of locks on the client-side of the application. As the LLT stores data including an ID of the lock’s owner and the region which the lock covers, the client-side can easily use this data to render a visual representation of the lock. In the implementation described in this thesis, a range of text is highlighted according to the owner ID. If the lock is owned by the current client, the lock is highlighted green; if the lock is not owned by the current client, the lock is highlighted red (Figure 6.5). As the lock table is updated with each synchronisation loop, it is trivial to also update the visual appearance of locks after every server reply is received by a client. It should be noted that this colour choice is arbitrary for the purposes of this implementation - additional techniques are described in Chapter 7 to provide better user attribution.

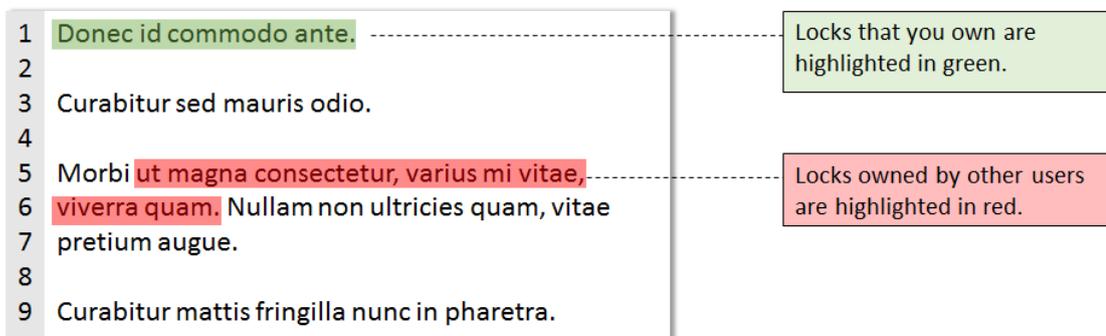


Figure 6.5: Visual representation of locks using two colours.

6.6. Support for derived features of flexible locking in diffsync

In Section 6.3 of this chapter, several key issues were presented that should be considered when developing locking systems for collaborative editing applications. These issues include syntactic consistency, semantic inconsistency, non-blocking input, dynamic region adjustment, lock ownership, and representation of locks to the user. These features are considered to be derived features of our system resulting from either the design of the locking functionality or the design of diffsync. In this section of the paper, the framework for flexible locking is discussed based on

how each issue is addressed. This includes a discussion on the benefits of diffsync when addressing these issues, as well as any limitations of the current locking framework.

6.6.1. Syntactic consistency

The first locking issue is *syntactic consistency*. Typically, edit-based techniques can encounter issues surrounding syntactic consistency because the order in which operations are received may differ between clients. This means that locking cannot be used to enforce syntactic integrity in collaborative editing systems. However, the client-server nature of diffsync means that all edits (i.e. diffs) are propagated and received in the intended order, with all edits being compiled with existing content on the server before being propagated to other collaborators. Therefore, if an edit isn't received by the server, the client will not receive a response and the edit will be retained on the stack to be sent out in proceeding synchronisation loops until it finally arrives at the server.

6.6.2. Semantic consistency

The second locking issue is *semantic consistency*. This refers to the ability of a locking system to enforce the rules of the shared content (e.g. English grammar) by allowing a user to take exclusive ownership over a section to avoid mutually exclusive (but semantically correct) edits from being made at the same time. Semantic consistency can be considered a derived feature of locking; by supporting flexible locking, semantic consistency can be enforced if used correctly.

The real benefits for semantic consistency result from the time saved in error checking. In collaboration scenarios where users are simultaneously focusing on separate areas of a document, there is no risk of semantic inconsistency arising. However, in scenarios where there are multiple users making changes to the same region of text, the risk increases. If a high traffic area of a shared document is locked only a single user will be able to make semantic changes to the text.

The locking implementation presented in this chapter also supports many users owning a single lock. In this scenario there is still a risk for semantic inconsistency to arise because several users may be working within the same locked area simultaneously. However, depending on the number of other users in the document who do not have access to a shared lock the risk of semantic inconsistency would still be reduced overall.

6.6.3. Non-blocking input

The third locking issue considered is the nature of *blocking*, referring to the process in which the locking system handles lock requests. The presented framework's approach is non-blocking because users are able to continue editing a requested lock region while the server determines the validity of the lock request. If a lock request is denied (due to another user having granted a lock in that region) the user will then be blocked from editing that region further. Otherwise, the lock will be granted and the user can continue editing without interruption.

To determine how the proposed approach handled simultaneous lock requests from different users on the same region of text, a small simulation was conducted. Three different users simultaneously sent multiple lock requests for the same region of text to the server. It was found that the first lock request received by the server was granted the lock, but repeated tests showed that it was not always the same client who was successful. There are often minor latency effects on any given synchronisation loop which means that the first user to request the lock may not always be the overall winner. For some applications this might be sufficient, though it could be avoided in other implementations by sending timestamps when requesting a lock or allocating a priority status to lock requests (e.g. urgent vs. non-urgent).

6.6.4. Dynamic region adjustment

The fourth locking issue considered is *dynamic region adjustment*. The implementation presented in this chapter provides a method to dynamically adjust the size of locks based on edits made to the document. Through simulations the ability of the proposed framework to handle edits made within a lock, before a lock, and after a lock was tested. It was found that this locking technique is robust to adjust locks around such edits. The technique is also complemented by diffsync: since locking data is sent to the server with each synchronisation loop and adjustments are made server-side, conflicts are managed and adjustments are propagated to all users. This means that consistency of locks and their adjustments is maintained for all users.

6.6.5. Lock ownership

Another issue that was considered is *lock ownership*. Currently, the literature does not report a method for supporting complete user attribution of edits in diffsync as all diffs are merged on the server-side. While the presented implementation does not untangle these edits, identification numbers are used at the algorithmic level to keep track of each connection. This identification number is then used to track lock ownership in the Global Locking Table and Local Locking Tables. While additional work is needed to provide user attribution for individual edits in diffsync (presented in Chapter 7), simpler user attribution such as a unique identification number could provide further flexibility in the diffsync locking scheme. For example, locking policies may be established to allow multiple users to own a single lock (permissions system). Another potential use may be to pre-lock sections of a document so that certain users can only manipulate specified sections of shared data.

The presented framework also supports locks that can be owned by multiple users at the same time. This is achieved by storing a group identifier within the lock object itself and checking this against a user's group affiliation. Considering the scalability and efficiency data presented in sections 4.5.2 and 4.5.3, it should be noted that group locks do not increase the number of

locks utilised. For example, a group lock owned by ten users requires only a single lock as opposed to ten locks. Thus, this feature does not have a negative effect on scalability, efficiency, or integrity.

6.6.6. Visual representation of locks

The final locking issue considered is the *visual representation of locks*. In the presented test framework this was achieved through the use of colour highlighting, i.e. locks owned by a user appear in a green highlight whereas locks not owned by a user appear in a red highlight. This provides users with immediate awareness as to which sections of locked text they can edit. However, one limitation could be that the identity of individual users is difficult to determine with only two colours. Additional formats of visual representation would be considered a usability issue specific to each application, but the same data could be used to present alternative visual representation of locks. Additional work towards achieving this is presented in Chapter 7 of this thesis.

6.7. Summary

In this chapter a framework for flexible locking was introduced that is built on the diffsync technique. Flexible locking is an important feature required by remote EM practitioners within a collaborative report writing system, as specified by the practitioners surveyed in Chapter 4. Chapter 5 identified that flexible locking was not currently supported by diffsync, thus the new framework for flexible locking was developed in this chapter. The new diffsync-based locking framework supports the key challenges of flexible locking, including syntactic consistency, semantic consistency, non-blocking input, dynamic region adjustment, lock ownership and representation of lock to the user. The framework achieves the necessary locking functionality as specified by the EM practitioners surveyed in Chapter 4 by allowing diffsync to maintain exclusive locks for both single authors and groups.

In the next chapter, the research artefact is further developed with a conceptual framework that supports user attribution within diffsync. The flexible locking and user attribution frameworks will be combined to produce technical support for multisynchronous and crowdsourcing work paradigms, discussed in Chapters 8 and 9. These features combined will comprise the research artefact to support collaborative report writing for remote EM practitioners.

7. User Attribution Techniques for Diffsync

Chapter 7 describes the user attribution framework that was implemented in diffsync. User attribution was determined to be a key requirement for the collaborative EM report writing framework in Chapter 4. As diffsync does not currently support user attribution, a new framework is implemented. This chapter discusses the design issues surrounding user attribution and describes methods for supporting workspace awareness, relative contribution of authors, and maintaining a history of edits within diffsync.

7.1. Overview

This chapter provides an overview of the development of a user attribution framework within the diffsync technique. User attribution refers to techniques that support workspace and change awareness within a collaborative system [27]. As specified in Chapter 4, user attribution is a key requirement to support collaborative report writing for remote EM practitioners. Chapter 5 selected diffsync as the system architecture for developing the research artefact, however it was noted that diffsync does not support user attribution. Thus, this chapter aims to demonstrate a new framework that supports user attribution in diffsync.

This chapter begins by providing an overview of user attribution with specific focus on techniques that support workspace awareness techniques, relative contribution of authors, and maintaining a history of edits. Next, the benefits of user attribution for collaborative EM report writing are stated. This is followed by a description of the user attribution framework and implementation of a web-based system to demonstrate the features of the framework. Finally, the chapter concludes with a discussion and summary of the user attribution feature described in this chapter.

7.2. User attribution in collaborative editing

7.2.1. Workspace awareness

Maintaining awareness of user actions and intention are crucial for successful collaboration [119]. Workspace awareness embodies knowledge of user identity, user edits (type, position, time), and user intention [44], [120]. Mechanisms for communicating this knowledge to users helps coordinate activity, simplify communication, provide relevant assistance, and manage the dynamism between individual work and shared work [120].

Many different methods enable user attribution in shared workspaces. One method draws attention to changes with visual cues such as highlighting and colouration [27], [44]. Another involves tracking and recording a user's eyes to infer which sections of the workspace are gaining attention and which parts are being ignored [121], [122]. Similarly, mouse and cursor tracking are an effective awareness cue in real-time environments [123]. Additionally, these features should be tied to user profiles to include additional cues such as user name, email address, and affiliations.

While visual cues promote awareness, social factors should also be taken into account when implementing user attribution features. Individual user behaviour is impacted by understanding the behaviours of all participants [124]. Furthermore, the rationale behind edits is not always obvious in a shared workspace, which leads to misunderstandings that negatively affect collaboration. User attribution mitigates these risks [27].

7.2.2. Relative contribution of authors

Another useful user attribution feature is calculating the relative contribution of authors. *Relative contribution* measures the edits in a document in terms of total number of edits per user and the quality of those edits. Much of the work in this area relates to wikis [125]. A wiki is a collaboratively created set of web pages that are iteratively improved by many participants, typically using an asynchronous updates [25].

Several techniques exist for automatically calculating user contributions on wiki systems and displaying that information to participants. One such technique compares the current and previous versions of a document with the total contribution of all revisions made over time [126]. This technique was extended by assigning a ‘rating’ based on user contributions, which feed into a reputation system [127]. A second technique combines visual user attribution with relative contribution to display the total number of user edits made to a document [128]. A third technique provides a weighting system to measure the importance of revisions for calculating relative contribution [129].

Wikis were originally designed with the intent of hiding the association between author and content as an attempt to remove social bias and promote group goals over individual benefits [125]. However, this caused problems in terms of document vandalism, malicious edits, and low quality edits by anonymous users [79]. Nowadays, wikis mitigate these problems with a document revision process where edits by authors verified as experts are given greater weighting than non-expert authors [129].

Real-time editing systems might benefit from relative user contribution features by providing additional awareness to collaborators engaging in a shared activity. Within the scope of EM, it is important to identify the author of critical information if further verification is needed. Relative user attribution allows collaborators to identify the main contributors to a document based on the number of edits made.

7.2.3. History of edits

Apart from the user attribution techniques already mentioned, collaborative applications support change awareness by maintaining a history of revisions [44]. *Change awareness* refers mechanisms used for detecting and tracking changes made to a document over time [44]. One method of change awareness is to compare two versions of a document to find differences by using *diff* operations [82]. Some applications display this information statically to the user as a

tool for tracking changes (e.g. GitHub⁷). Other applications utilise real-time *diff* operations (e.g. MobWrite [115]) to maintain shared consistency, however additional mechanisms may be required to visually display information about changes to the end user.

Version control systems automate the process of tracking changes of shared documents stored in a centralised repository [130]. Many benefits of version control systems have been identified such as increased awareness, mutual understanding, and future insights on the potential success of projects based on the revision history [131]. Additionally, users are able to provide *commit messages* that annotate the changes. Commit messages also inform other users about the context and rationale of edits which improves understanding [130].

Moreover, version control systems store all incremental document changes and hence support the future review of edit history [44]. Additionally, it is possible to revert changes back to an earlier version of a document. Techniques such as operational transformation, which support consistency between shared documents, provide rollback features such as ‘group undo’ that allow users to revert changes [71]. Google Docs [132] is a popular application for real-time collaborative writing that stores the change history of its documents. These changes can be viewed and restored by users at any time.

Synchronisation techniques can benefit from maintaining a history of past edits. With such information, users might better understand the evolution of a document during collaboration and plan future edits by reviewing existing versions. For example, a user could view changes made to sections of a document in which they did not contribute. A single user could also revisit earlier edits that may have since been deleted.

7.3. Benefits of user attribution for collaborative EM report writing

Based on the background information provided on user attribution in this chapter, several key benefits can be derived for collaborative EM report writing. These benefits are as follows:

- User attribution can attribute information to an author which can assist in verification and accuracy of report content. If information is added to the report by a civil engineer about the state of roads in the disaster area, it would be considered accurate and trustworthy. Additionally, with the authorship of information identifiable, it becomes easier to locate that author for further information if necessary.
- Another benefit provided by user attribution for collaborative EM report writing is that it maintains a history of changes. This history can provide context for the information currently in the report. Users can quickly view how the report has evolved over time at a fine-grained level to determine how the disaster has unfolded. Further, as the history is

⁷ <https://github.com/>

stored alongside the user attribution information it can become useful in future review to determine the effectiveness of the current response as well as plan better ways to respond to disasters in the future.

- Finally, user attribution features within the underlying text synchronisation technique provide general improvements to collaboration by providing additional workspace awareness. As software-based collaboration loses many of the benefits of face-to-face collaboration, it is important to convey both user actions and intentions within the interface to ensure collaboration is effective. User attribution displays the current location and actions of a user in the document, as well as information about their relative contribution and a history of actions.

7.4. Developing a user attribution framework for diffsync

While diffsync provides a robust mechanism for synchronising content between collaborators, it does not provide a mechanism for user attribution [24]. This section of the chapter describes additional techniques that can be used to achieve user attribution in diffsync.

7.4.1. Current state of user attribution in diffsync

In its existing form, diffsync does not support user attribution features as all changes are merged on the server [24]. This poses a challenge to developers and users alike in untangling and attributing edits made by users in collaborative workspaces. Several applications have been developed using diffsync for real-time collaboration [12], [111], [115], but there has been no reported work on developing techniques for diffsync to support visual attribution, relative contribution of authors, and maintaining document history. In the next section, several methods are presented for enabling these features in diffsync and implement them in a prototype web-based collaborative report-writing application as a proof of concept.

7.4.2. Overview of the technique (attributed diffsync)

There are five additional methods utilised to achieve user attribution in diffsync. These are: 1) exploiting the results of the *diff* operation to determine authorship of content; 2) merging adjacent attribution items; 3) dynamically adjusting the size of an attribution when content is added or removed; 4) dynamically adjusting the position of attributions to account for edits made within the surrounding document; and 5) maintaining a history of the edits for future use. All of these additional methods run on the server-side of the synchronisation. The details about each of these functions are provided below.

7.4.3. Exploiting diffs to track authorship

A *diff* operation returns the differences found between two versions of some shared text, which will be referred to as the *diff result*. The *diff result* contains the *type* of change detected

(represented as a number) and the *affected text* (represented as a string), where *type* is either **-1** (text removed), **0** (text identical), or **1** (text added), and *affected text* is the string that was added, removed, or remained the same between two versions of a document. For example, consider a *diff* operation between the following two texts:

There is a flood in Brisbane.

There was a fire in Cairns.

The result of the *diff* yields the following diff result:

0, There	<i>text identical</i>
-1, i	<i>text removed</i>
1, wa	<i>text added</i>
0, s a f	<i>text identical</i>
-1, lood in Brisbane	<i>text removed</i>
1, ire in Cairns	<i>text added</i>

The user added one character and removed two characters. If this data is grouped with a user ID, and the position of the affected text is calculated in terms of index and range, then it is possible to provide a simple structure to begin attributing contributions to each user. This object is referred to as the *attribution object*. For example:

```
attribution { userID: 1, range: 10->11, text: a }
```

The underlying structure of some shared text can be represented with an index-based data structure (e.g. 0 to $n-1$) as illustrated in Figure 7.1. Using this structure it can be determined where in the document the user made the change.

<i>text</i>	r	a	n	f	a	
<i>index</i>	9	10	11	12	13	14

Figure 7.1: A document has an index-based structure, e.g. 'a' is at index 10.

During each synchronisation loop, the client propagates a stack of edits to the server. Each edit contains a series of *diff result* objects. Our user attribution technique works primarily on the server-side: when a stack of edits is received from a client, the server loops through each diff result object and handles it according to the following rules:

1. If the diff result states that the client added a region of text, a new attribution object is added to a server attribution stack. The attribution object contains the user ID, start index of the edit, end index of the edit, and the text contained in the edit;
2. If the diff result states that the client removed a region of text, another method runs to decrease the size of the affected attribution objects. Additionally, the start and end index of other attribution objects are adjusted to reflect the new state of the shared text;
3. If the diff result states that a region of text remained unchanged, the server does nothing with that information.

Immediately after adding, removing, or adjusting attributions based on the incoming diff result objects, the server runs an additional method to accommodate for failed *patch* operations to ensure accuracy of attribution is maintained.

7.4.4. Merging adjacent attribution items

Depending on the time taken for a synchronisation loop to complete, the addition of larger groups of edits is accounted for over several diffs. For example, if a user takes 4 seconds to type large word, but synchronisation occurs once per second, then the server might only receive partial attribution objects that span the single word. It is inefficient to store four attribution objects that cover adjacent characters in the main text, so the following technique is proposed to merge attribution objects where possible. For example:

```
{ userID: 1, range: 31 -> 35, text: good }
{ userID: 1, range: 35 -> 38, text: bye  }
      is merged to become
{ userID: 1, range: 31 -> 38, text: goodbye }
```

This function resides on the server and is performed whenever edits are received that include diff results that account for text added or text removed.

7.4.5. Adjusting range of attributions

Since a shared document is constantly changing due edits made by many users, attribution objects need to adjust based on those edits. For example, if User A owns an attribution on the text “Cat”, but User B edits that area of the text to become “Chat”, there are a few possible approaches. First, the system could remove the user attribution of the word “Cat” from User A, and add a new attribution for User B to own the newly added “h” only. However, this approach does not capture the semantic meaning of the edit and results in many disjointed attribution objects scattered throughout the document.

Another approach is to transfer the attribution of “Cat” from User A to User B, also adding in the new “h” for User B so that now there is a single attribution for “Chat” owned by User B

only. As User B has changed the semantic meaning of the word in a way that may not align with the intention of User A, it is fair to transfer the ownership of this contribution to User B.

A third approach is to retain the original user attribution but split it into two objects so that User A now owns “C” and “at”. A third attribution would be added for User B to contain the “h”. Depending on the granularity of the changes being captured by an application, this may be a good approach. This is the approach used in the new user attribution technique to cover general authorship. For example:

```
{ userID: 1, range: 0->1, text: C }  
{ userID: 2, range: 1->2, text: h }  
{ userID: 1, range: 2->4, text: at }
```

Another consideration is that a user could can erase text that exists within the attribution object. The system must also have a method for reducing the size of an attribution object based on such edits. Depending on the latency of the network, these edits may span multiple attribution areas. To handle this, the affected text is split and handled on a per character basis. The server iterates through these characters and adjusts relevant attribution areas as necessary.

7.4.6. Dynamic adjustment of user attribution position

Another consideration related to the constantly changing nature of a shared document is the position of conceptual ownership regions. When text is inserted into the document before an attribution object, that attribution object must adjust its location in the positive direction to remain accurate. Conversely, when text is removed the attribution object must adjust its location in the negative direction to remain accurate. For example, if User A owns some text spanning index 3->7, and User B inputs some text at index 0, the position of User A’s text will move forward. In this instance, nobody has modified the content of User A’s text so there is no need for transfer of attribution, but it still requires adjustment. This method runs on the server any time text is added or removed.

7.4.7. Maintaining document history

The final feature supported using these techniques is maintaining an edit history. Diffsync utilises version numbers on both the client and server-side. Thus, saving the state of the shared document is a simple matter of storing a *history object* on the server. A history object includes the state of the server text at the time, the current version of the server text, the state of the attribution objects at the time, and a timestamp. This information is propagated to all clients for rendering on request.

7.5. Demonstration of new user attribution framework for diffsync

To demonstrate the applicability and flexibility of the user attribution techniques presented in this chapter, a web-based report writing prototype was developed that incorporates user

attribution features within difsync. This section describes how user attribution was achieved on the client-side of the application.

7.5.1. Highlighting user contributions

Utilising the stack of attribution items, the client can render each area based on its region. In the demonstrated implementation, each user is assigned a random colour upon connecting to the server. This colour is stored with the attribution objects, along with the ID and username of the user. At the end of each synchronisation loop (i.e. when the client receives a server response), the client runs a function to highlight each attribution with its colour. The username and colour for each attribution area are also added to a visible panel (see Figure 7.2).

As a secondary user attribution characteristic, tooltips were implemented that display the name of the author when hovering over any section of the text. This is achieved by detecting the position of the mouse, and finding which attribution region it falls in. The username for that region is displayed in a small pop up that follows the mouse cursor (see Figure 5.2), which means that immediate attribution can be determined in any region of the text.



Figure 7.2: Attributions are colour highlighted and associated with a username.

7.5.2. Calculating relative user contribution

Another consideration for user attribution mentioned in this chapter is relative contribution of authors in a shared workspace. Utilising the same data that is stored on the server, the application can determine the percentage of contribution for each author. The client side inspects the attribution objects and finds the total number of characters owned by each user, and then calculates the percentage of contribution compared to the length of the shared text. This is updated during each synchronisation loop, and is displayed on the screen next to the user's name (see Figure 7.2).

7.5.3. History

The final user attribution feature that was implemented is storing a history of edits. This was achieved by storing a collection of *History objects* on the server-side. Each history object contains a version number, a copy of the server text, a copy of the attribution stack, and a timestamp. On each synchronisation loop a new *History* object is added to the collection, unless there have been no changes since the last time. This ensures that the server does not store duplicate versions.

Using this information the client can request a separate history page. When this page is opened a request is sent to the server and the history stack is returned. The client renders each version in a similar interface to the collaborative environment with editing disabled (see Figure 7.3). Using a slider, the user can browse through each version in a chronological order to view how the document has changed over time.

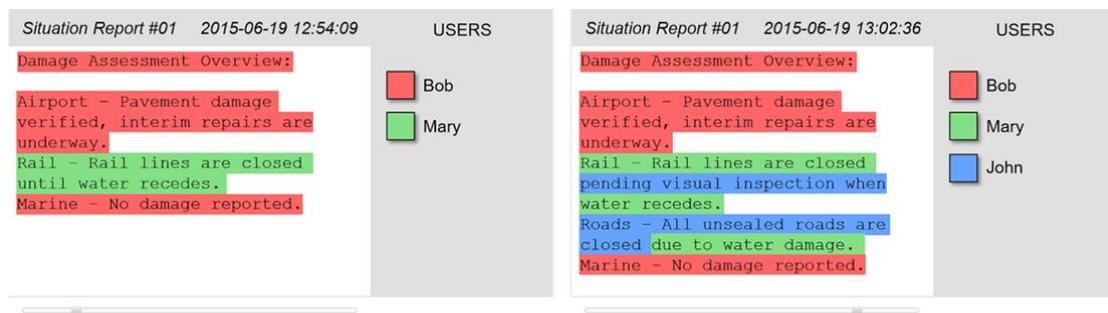


Figure 7.3: Two versions of the document saved by the history technique.

7.6. Summary

In this chapter a framework for user attribution was introduced that is built on the diffsync technique. User attribution is an important feature required by remote EM practitioners within a collaborative report writing system, as specified by the practitioners surveyed in Chapter 4. Chapter 5 identified that user attribution was not currently supported by diffsync, thus the new framework for user attribution was developed in this chapter. The new diffsync-based user attribution framework supports three key features, including workspace awareness, relative contribution of authors, and maintaining a history of edits. These features achieve the necessary user attribution functionality as specified by the EM practitioners surveyed in Chapter 4 by allowing collaborators to maintain awareness of user actions and intentions within the shared workspace. In the next chapter, the research artefact is further developed with a conceptual framework that supports multisynchronous editing within diffsync. This is achieved by combining the flexible locking and user attribution frameworks. These features combined with the crowdsourcing framework presented in Chapter 9 will comprise the research artefact to support collaborative report writing for remote EM practitioners.

8. Developing a multisynchronous framework for diffsync

Chapter 8 describes the multisynchronous editing framework that was implemented in diffsync. Support for diverse user types was determined to be a key requirement for the collaborative EM report writing framework in Chapter 4. Multisynchronous editing aims to support both control-centre staff and field officers in collaborative writing during an emergency. As diffsync does not currently support multisynchronous editing, a new framework was implemented based on the flexible locking and user attribution frameworks presented in Chapters 6 and 7. This chapter discusses the design issues surrounding multisynchronous editing and describes methods for supporting multisynchronous editing in diffsync.

8.1. Overview

This chapter provides an overview of the development of a multisynchronous framework within the diffsync technique. Multisynchronous collaboration is a process in which some users work in real-time (e.g. desktop-based users) while other users work in isolation and commit updates when necessary (e.g. mobile users) [15]. As specified in Chapter 4, multisynchronous editing is a key requirement to support collaborative report writing for remote EM practitioners. Chapter 5 selected diffsync as the system architecture for developing the research artefact, however it was noted that diffsync does not support multisynchronous editing. Thus, this chapter aims to demonstrate a new framework that supports multisynchronous editing in diffsync.

This chapter begins in Section 8.2 by describing multisynchronous editing for collaborative EM report writing. Next, the benefits multisynchronous collaboration for EM report writing are identified in Section 8.3. Following on, Section 8.4 provides an overview of the new framework for supporting multisynchronous editing within diffsync. Finally, Section 8.6 provides a summary of the chapter.

8.2. Multisynchronous collaboration

Multisynchronous collaboration is a process in which some users work in real-time (e.g. desktop-based users) while other users work in isolation and commit updates when necessary (e.g. mobile users) [15]. The nature of disaster management in terms of time pressure, uncertainty, complexity, and the need for collaboration between multiple organisations means that emerging decision support systems are required to be real-time, multisynchronous, and capable of supporting distributed users on different devices. Collaborative report writing is an important task conducted by emergency service agencies during a disaster to collate and distribute the facts of a disaster situation [133]. Multisynchronous collaboration is an emerging area of research that can facilitate report writing for EM.

Supporting real-time collaboration between control centre staff and field officers using mobile devices has several challenges. Primarily, network availability may be severely disrupted during an emergency due to infrastructure damage and increased network traffic. Such

disconnections have an effect on the ability for people to use ICT for communication during an emergency. According to Gutwin et al. [134], there are three types of disconnection that can affect online collaborative applications, which are known as *delay-based interruptions*, *network outages*, and *explicit departures*.

Delay-based interruptions are short-term gaps in message delivery caused by network congestion or high CPU load, resulting in the “piling up” of messages such that they are delivered all at once when the interruption subsides. While most *delay-based interruptions* go unnoticed due to the small delay times, they typically affect only the receiver of the message, as the underlying network connection is still valid and there is only the illusion of short-term disconnection [134]. *Network outages* occur when a node within the network unintentionally loses its link to another node (e.g. a client loses its connection to the server in diffsync). The cause of such outages can include network failure (either hardware or software), mobile devices moving between cellular tower ranges, or devices being powered off or unplugged [134]. In contrast, *explicit departures* occur when a user intentionally quits the collaborative workspace or disconnects from the network. Unlike the previous two types of disconnection, *explicit departures* are planned which means other users are usually notified that the user has left the collaborative session.

It should be noted that all three of these disconnection types are relevant to consider within the scope of the multisynchronous framework presented in this chapter. For example, there can be increased latency on mobile networks during a disaster, which means that *delay-based interruptions* can become more frequent. Further, depending on the type of disaster, network infrastructure can be damaged causing widespread *network outages*. Finally, EM practitioners working in the disaster area may need to make frequent *explicit departures* from the collaborative application to maintain safety as they move through the environment. In fields such as EM where accurate and timely information is imperative, diffsync can provide robust synchronisation, but additional techniques should be investigated to facilitate collaborative multisynchronous report writing with input from first responders in the disaster area.

8.3. Benefits of multisynchronous editing for collaborative EM report writing

One of the key benefits of using a multisynchronous approach to collaborative sitrep writing is that trusted workers in the field can provide accurate information about the situation. When accuracy is important, information provided by the public needs to be verified before it can be used to inform disaster response. However, recent research indicates that large groups may perform as well or better than individual experts in producing useful information [135]. The multisynchronous framework presented in this chapter could easily be extended to engage public users through mobile devices for situation reporting.

Another benefit is that mobile users can move throughout a disaster environment, whereas

workers in a control centre are at a fixed location. This means that as the conditions change in the disaster, mobile users can move around. According to Guerro et al. [136], handheld devices are considered more appropriate than laptops or tablets in unsafe environments (e.g. a disaster site). This is because they are easy to deploy and carry, require lower user attention, and have a short start-up time. These features combine to allow for quick reactions from a user, which could be necessary in volatile environments.

A mobile user wandering around an environment will likely engage in short and simple interactions with the system, thus only basic communication support is required (in terms of network availability and bandwidth). In contrast, a user in a fixed position requires larger amounts of bandwidth and greater network availability, as interaction with the system is typically longer and more complex [136]. This is the intended use case for the multisynchronous framework, as control centre staff are considered as information providers, consumers, and decision-makers, whereas mobile users are only required to contribute small sections of information about a single incident.

Despite the benefits, there are also potential limitations of the framework. For instance, as mobile users are given single, isolated tasks, there is a lack of situation awareness and their responses cannot be tailored to the context of the wider document. This is mitigated using a task description, but additional mechanisms such as radio or chat could be used to clarify details [2]. Finally, some aspects of the communication are one-sided. Staff within a control centre can push notifications to mobile users for information, but there is no method within our system for mobile users to initiate communication or submit content without a lock first being granted. This problem is investigated further in Chapter 9.

It should also be noted that the current system has a heavy dependency on a stable network connection. While diffsync is robust to network faults such as packet loss or data corruption [9], complete loss of Internet connection would render the system not operational. For this reason, it is important to consider alternative network solutions that could support our system in an emergency. Nilsson and Stølen distinguish four main network solutions for emergency response: 1) wireless ad hoc networks; 2) cellular networks; 3) special emergency networks, and 4) router-based networks that are deployed for the emergency operation [41]. They conclude that wireless ad hoc networks are well-suited for emergency response as they are quick to deploy, provide local communication within the disaster area, and can be used alongside an Internet connection (if it is available) [41].

8.4. Framework for multisynchronous collaborative report writing

Based on the need for multisynchronous collaborative report writing in EM, a prototype was developed to support distributed collaboration between browser-based and mobile-based users. The implementation is described in this section of the paper. First the web technologies that were

used in the development of the prototype are outlined. This is followed by an overview of the prototype with specific focus on its major features and how these features intend to support multisynchronous collaboration between web and mobile users.

Figure 8.1 displays the framework for multisynchronous collaboration between desktop-based users and mobile users for collaborative report writing. Multiple desktop-based clients can be connected to the system for real-time editing. These clients utilise the diffsync synchronisation mechanism to maintain consistency of shared text as changes are being made in real-time. Meanwhile, mobile clients are provided with small tasks via a locking mechanism. The mobile clients do not collaborate in real-time, instead pushing changes back to the main document as needed or when a network connection is available.

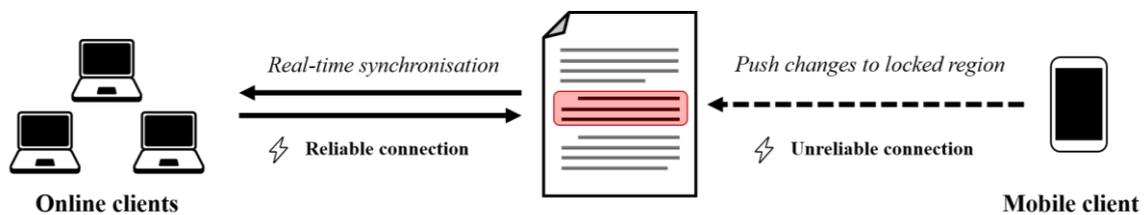


Figure 8.1: The multisynchronous framework.

The intended workflow of the multisynchronous framework is as follows:

- Emergency staff are working on a report at a control centre. The web-based, real-time collaborative nature of the system allows for several individuals and agencies to collaborate in a single shared workspace.
- Meanwhile, remote emergency staff are moving throughout the disaster area equipped with mobile devices (e.g. smartphones or tablets). Control centre staff can view the location of these users on a map within the report writing system. When information about the situation is required, tasks can be assigned to mobile users in which information is requested about their immediate surroundings such as condition of roads or flooding.
- The mobile users are notified when a new task is received. Notes can be added to the task and synchronised with the server when ready.
- Meanwhile on the server, online users can continue to edit the document in real-time. They will see that certain sections are locked for remote users, and they won't be able to make any changes to those areas to avoid conflicts. It does not matter what changes are made within other sections of the document as the underlying synchronisation method ensures that locked sections adjust position automatically when necessary.
- When a remote worker has network access and is ready to synchronise information, synchronisation can be triggered within the smartphone application. This

synchronisation is different to normal diffsync as only the locked content is pushed to the server and replaces the current content within the relevant lock.

- In this way, both desktop-based users and mobile users with limited resources can work together in a multisynchronous environment while maintaining consistency.

An example web-based interface based on the framework is displayed in Figure 8.2. The interface has several key features. First, there is a large section used for collaborative report writing. Templates can be used so that a new sitrep can be started efficiently. All collaborators using the web-based interface will see the same text, due to synchronisation happening in real-time. Second, there are three sub features used to provide tasks to mobile users in the disaster area:

- (1) A user can select an area of text and use the “Lock Mobile” button at the top of the interface to push a new task to a mobile user. The user is prompted to enter a brief description of the task which is sent to the mobile user.
- (2) Current mobile tasks are displayed in a list to the right of the text-editing screen. Each task is identified with both a colour and a username. The system provides each mobile user with a unique colour so that each task and the related section of the document can be highlighted as necessary. The task also shows an indication of when the mobile user last updated their text (e.g. *3 mins ago*). A task can be cancelled from this view as well.
- (3) A map displays the last known location of a mobile user in the field. This information is updated whenever a mobile device synchronises with the web-based system, which occurs automatically every five minutes or whenever text is manually synchronised by the mobile user. This provides web-based users with awareness of the mobile users’ locations, aiding in decision-making when assigning tasks to mobile users.

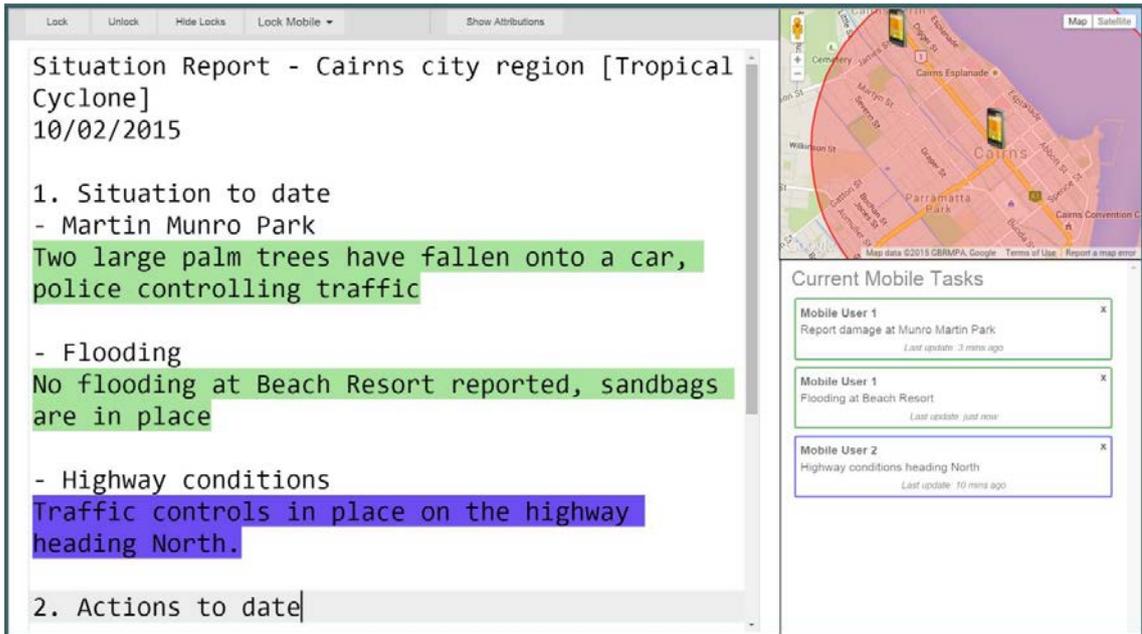


Figure 8.2: The web-based interface showing several tasks locked to mobile users.

The second component of the framework is support for mobile users. This allows mobile users to receive requests from centralised users to provide information about a context-specific incident or situation within the disaster area. Screenshots of a mobile interface based on the multisynchronous framework are displayed in Figure 8.3a, Figure 8.3b, and Figure 8.3c.

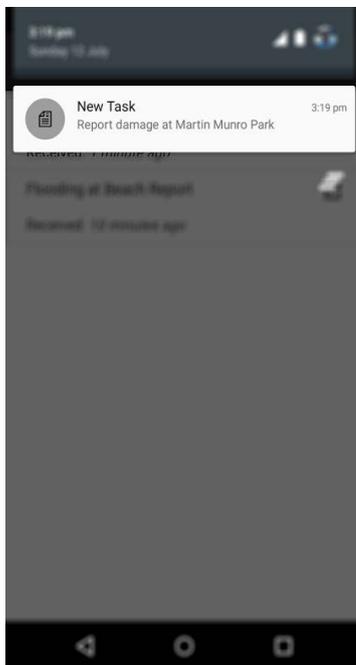


Figure 8.3a: Notifications are received when a new task is available.

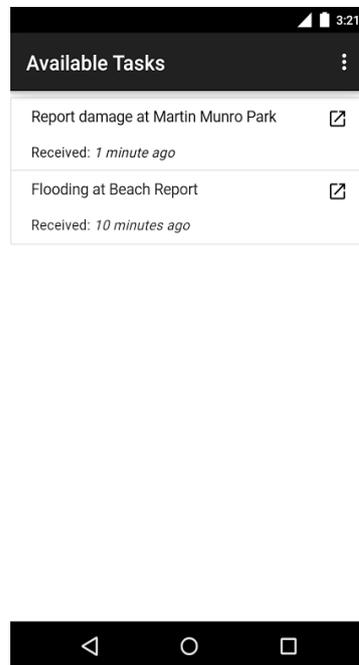


Figure 8.3b: The user can view a list of all active tasks they are assigned.

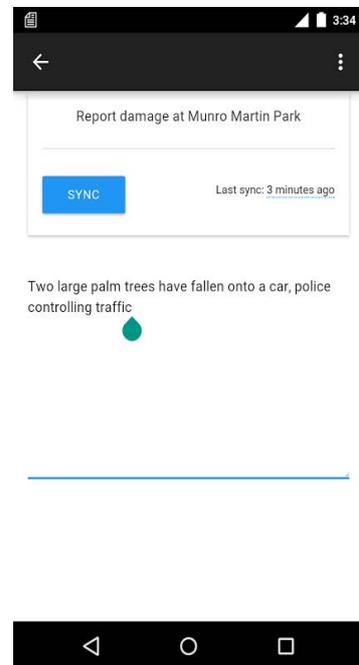


Figure 8.3c: Notes can be added about the task and synchronised.

The mobile device receives notifications when a new task is locked to that particular user from the web interface (see Figure 8.3a). A user can click directly on the notification to enter immediately into the note taking screen (Figure 8.3c). Notifications are also received when a web user revokes the mobile lock (e.g. if no further information is required from the mobile user). This ensures that the mobile user is aware that no further information is required and will not spend any more time on this task.

All tasks that are currently assigned to the mobile user are displayed in the main screen of the application (Figure 8.3b). For each task, a brief description of the task and the time it was assigned is displayed. A user can click on a specific task to enter the note taking screen (Figure 8.3c). There is also the option to refresh the list of current tasks (triggered in the submenu).

The note taking screen (Figure 8.3c) displays the brief description of the information required at the top of the screen. The synchronisation button is used to push information back to the main document, and an indication of when the last synchronisation occurred is also displayed. The main text area below these controls is where the mobile user adds information about the disaster situation as necessary. The mobile user can also dismiss a task from the submenu of this screen to notify the web users that no more information will be provided about this task.

As a final consideration for mobile users, there may be situations where additional situation awareness is required by the agent on the field to adequately provide information about an incident. One of the primary goals of this framework is to minimise the need to synchronise large amounts of data between mobile and desktop users. This problem can be solved by using automatic text summarisation techniques to send a summarised version of the current report to mobile users. This has several benefits for mobile users in EM, including less data being transferred, avoidance of information overload in a hostile environment, and increased situation awareness.

To achieve this in the demonstrated framework, the Rapid Automatic Keyword Extraction (RAKE) technique is used. The unpredictable nature of a disaster means that relying on a corpus of previous reports for keyword extraction could produce unreliable results for the current situation. RAKE can extract keywords from a single document which means that a corpus of documents is not required, and it has been demonstrated to work as effectively as corpus extraction techniques [137].

An example of the RAKE function within the system is demonstrated in Figure 8.4. Using the top terms extracted from the report, the system can extract the top sentences by calculating a *sentence score*, which is the sum of the weights of all words in each sentence. The 30% rule used by RAKE to determine important keywords is also used to select the number of sentences to display for the auto summary. Further, by sorting the extracted sentences from a document and assigning *sentence scores*, the system can also provide a summary of the document with ranked sentences based on that keyword. Users are able to input a keyword using a search input above

the summary.

The screenshot shows a web interface with a top navigation bar containing 'Lock', 'Unlock', 'Hide Locks', and 'Show Attributions'. Below this is a text area with the title 'Londor' and a news article snippet: 'Severe Tropical Cyclone Pam struck Vanuatu, affecting the capital of Port Vila, as an extremely destructive category 5 cyclone on the evening of 13 March at around 11pm. local time. The cyclone's eye passed close to Efate Island, where the capital is located, and winds are estimated to have reached 250kmph with gusts peaking at around 320kmph. Information from colleagues and partners indicates that the cyclone was stronger than expected, and Port Vila has experienced widespread damage with debris strewn in the streets. There are six confirmed fatalities, although the death toll is expected to rise as communication is reestablished with outer islands. The entire country has likely been affected, to some extent, by the extremely damaging winds, heavy rainfall, storm surges and flooding. There is concern for the southern-most islands of Tafea Province, total population 32540, which was directly struck by the eye wall and is without communication. The northern islands of...'. To the right of the text area is a search bar with the placeholder 'Summarise by keyword: enter keyword...' and a link 'OR View Automatic Summary'. Below the search bar is an 'Auto Summary' section with a list of bullet points: '- Severe Tropical Cyclone Pam struck Vanuatu, affecting the capital of Port Vila, as an extremely destructive category 5 cyclone on the evening of 13 March at around 11pm.', '- The northern islands of Sanma, Penama and Torba Provinces (population 86000) are also expected to have been heavily impacted as the cyclone headed south-southwest towards capital.', '- There are a further 430 people seeking emergency shelter in Torba and Penama.', '- A state of emergency was officially declared today for Shefa Province, which includes the capital Port Vila, and will be expanded to other provinces following aerial assessments in coming days.' Below the 'Auto Summary' is a section 'Top terms in sitrep:' followed by a list of keywords: 'Severe Tropical Cyclone Pam struck Vanuatu', 'extremely destructive category 5 cyclone', 'cyclone headed south-southwest towards capital', '430 people seeking emergency shelter', 'provinces following aerial assessments', 'Torba Provinces (population 86000)', '25 evacuation centres', 'total population 32540', 'extremely damaging winds', and 'six confirmed fatalities'.

Figure 8.4: Automatic summarisation and keyword extraction using RAKE.

8.5. Summary

In this chapter a framework for multisynchronous collaboration within the diffsync technique was introduced. This framework utilises the new diffsync-based flexible locking and user attribution frameworks developed in Chapters 6 and 7. Multisynchronous collaboration is an important feature required by remote EM practitioners within a collaborative report writing system, as specified by the practitioners surveyed in Chapter 4. Chapter 5 identified that multisynchronous collaboration was not currently supported by diffsync, thus the new framework for multisynchronous collaboration was developed in this chapter.

The new diffsync-based multisynchronous framework supports the two key user types involved in collaborative report writing: desktop-based users working in emergency control centres, and mobile-based users working in the field. These features achieve the necessary multisynchronous functionality as specified by the EM practitioners surveyed in Chapter 4 by allowing these two user types to contribute to a shared report based on their different technological needs. In the next chapter, the research artefact is further developed with a conceptual framework that supports crowdsourcing within diffsync. This is an extension of the multisynchronous framework presented in this chapter. These features combined with the flexible locking and user attribution frameworks presented in Chapters 6 and 7 will comprise the research artefact to support collaborative report writing for remote EM practitioners.

9. Crowdsourcing framework for diffsync

Chapter 9 describes the crowdsourcing framework that was implemented in diffsync. Support for diverse user types was determined to be a key requirement for the collaborative EM report writing framework in Chapter 4. Crowdsourcing aims to provide additional support for field officers beyond multisynchronous editing in contributing to a shared report. As diffsync does not currently support crowdsourcing, a new framework was implemented as an extension of the multisynchronous framework presented in Chapter 8. This chapter discusses the design issues surrounding crowdsourcing and describes methods for supporting crowdsourcing in diffsync.

9.1. Overview

This chapter provides an overview of the development of a crowdsourcing framework within the diffsync technique. Crowdsourcing is a method used to obtain information from a large group via the Internet [138]. As specified in Chapter 4, support for diverse types of users including control centre officers and field officers is a key requirement to support collaborative report writing for EM practitioners. Chapter 5 selected diffsync as the system architecture for developing the research artefact, however it was noted that diffsync does not support features such as multisynchronous editing. This chapter aims to extend the multisynchronous framework presented in Chapter 8 to support crowdsourcing within diffsync. This framework will allow field-based users to contribute information and initiate communication, as the multisynchronous framework presented in Chapter 8 only allowed field officers to communicate when determined via the control centre-based users.

This chapter begins in Section 9.2 by defining crowdsourcing for EM. Next, the benefits of crowdsourcing for collaborative EM report writing are identified in Section 9.3. Following this, a framework is presented to support crowdsourcing within diffsync. Finally, Section 9.4 summarises the chapter.

9.2. Crowdsourcing for emergency management

Crowdsourcing has two distinct definitions: (1) *Crowdsourcing* refers to the idea that a group can solve a problem more efficiently than a single expert, despite the group lacking the required expertise in the subject domain; and (2) *Crowdsourcing* implies that information collected from a group of observers is likely to be closer to the actual truth than information obtained from a single observer [138]. Within the scope of EM, the second definition is relevant to the accuracy of reports provided by members of the public, such that the more reports received about the same incident, the more likely it is to require an official response.

With the rise of social media, people tend to use social networks to communicate with each other and provide status updates during an emergency situation [135], [139]. While reports on social media may be accurate, there can be additional noise caused by reposts that can inflate the importance of a post. Existing solutions such as *Emergency Situation Awareness* (ESA) [5]

employ data mining techniques to automatically detect potential disaster-related incidents from Twitter posts while taking into account such noise. Other forms of social media such as wikis allow responders to connect directly and collaboratively share knowledge about the disaster, effectively bypassing the more traditional “chain of command” [1].

According to Gao et al. [140], crowdsourcing has three main benefits for emergency response. Firstly, crowdsourced data in the form of user requests and status reports can be collected from social media during and immediately after a disaster, allowing relief organisations to identify and respond to urgent situations in a timely manner. Second, crowdsourcing tools can collect unstructured data from diverse repositories (e.g. social media sites such as Twitter and Facebook, emails, web forms) and utilise automated summary techniques to categorise the data and identify trends. Third, crowdsourced information can be linked to geo-tag data which can assist responders in visualising the status of an unfolding disaster.

Despite the benefits of crowdsourcing for EM, there exist several challenges that can complicate its effectiveness in a real disaster. For example, any information gained from the public during a disaster must be subjected to verification from official sources before it is propagated as fact [138], [140]. Crowdsourcing software such as Ushahidi⁸ place the responsibility of verification on the crowd via collective feedback, but there is still room for improvement. Another limitation of crowdsourcing is the potential for information overload caused by large volumes of incoming reports from the public [140]. Techniques from the field of text mining (such as automated tagging and text summarisation [137]) may provide solutions to these problems. Table 9.1 presents several existing applications that utilise crowdsourcing for EM.

Table 9.1: Existing solutions that support crowdsourcing for EM.

<i>System</i>	<i>Features</i>	<i>Reference/s</i>
Emergency Situation Awareness (ESA)	Automatically detect potential disaster events by analysing real-time social media activity.	[5]
Ushahidi	Detailed incident reporting for the public, information sharing, geolocation, and support for mobile users to both submit and access reports.	[141]
Social Media (e.g. Twitter, Facebook)	Public users can upload information about situations (including media) and categorise using keywords, hashtags and geolocation.	[140], [142]

⁸ <https://www.ushahidi.com/>

9.3. Benefits of crowdsourcing for EM report writing

Large groups working on the same problem (whether individually or in groups) have the potential to generate useful knowledge [135]. Research in the field of EM has shown that individuals respond quickly to a disaster, are willing to share information, and often converge in the impacted locations of a disaster [135], [143], [144]. However, a common problem faced by EM practitioners when dealing with information provided by the public is its quality and accuracy in contrast to data from official sources [138].

Typically, emergency control staff have limited resources available to investigate and respond to the impacted areas of a disaster. This challenge can be compounded when the EM control centre is swamped with non-urgent reports from the public (e.g. fallen trees on property that have not caused significant damage, temporary loss of power to a residential property, and so on). In handling these non-urgent reports, it can be difficult to allocate official resources to respond to more urgent matters such as life-threatening damage or injury caused by a disaster. Thus, it is proposed that crowdsourcing information from members of the public located within the disaster impact area could assist in meeting this challenge.

One of the primary benefits of crowdsourcing information from the public via mobile devices is that official EM practitioners can gain a better understanding of the disaster and its impact, without the need to employ additional staff. While there may still be an overhead required in reviewing reports for spam and urgency, this overhead can be significantly reduced by offloading verification tasks to members of the public or trusted users in the field. The information collected via crowdsourced micro-reports can also form a basis for resource allocation in the future. Researchers and EM practitioners could use this information to not only assist in the recovery of the related disaster, but also in the preparation and mitigation of future disasters in the area. In addition, this information could potentially be used to predict the impact of a disaster and the support needs of similar areas in future disasters. More research is needed to investigate this possibility.

When standard web-based technologies are used to implement a crowdsourcing system, it may provide additional benefits for EM. If the crowdsourcing aspect of the system relies on members of the public to submit reports and perform verification tasks using their own mobile devices, there will be a wide range of devices required to access the crowdsourcing system. Standard web technologies ensure that the wider population can access the system without additional hardware for individual devices. A current limitation that is related to this is the need for users to download a specialised application on their mobile devices. This could be mitigated by further development to allow the system to run completely within the web browser of a mobile device, differentiating users by device ID and communicating via push notifications.

9.4. Implementing the crowdsourcing framework in diffsync

9.4.1. Framework

The proposed framework for microtask crowdsourcing is displayed in Figure 9.1 and Figure 9.2. Figure 9.1 illustrates the process of a field agent submitting a report to the disaster control centre. Using a mobile interface, a field agent (e.g. member of the public or an EM practitioner) submits a report, for example “*A tree has fallen on Sturt Street.*” The location of the report can be added automatically (using phone GPS) or manually, depending on whether the report is being submitted from the location of the incident. Media such as photographs can also be included with the report. Once submitted, it can be reviewed by an EM practitioner at the local disaster control centre through a web-based interface. If the report warrants further investigation but is not urgent or life-threatening, it can be flagged for verification by other members of the public or trusted EM practitioners in the field.

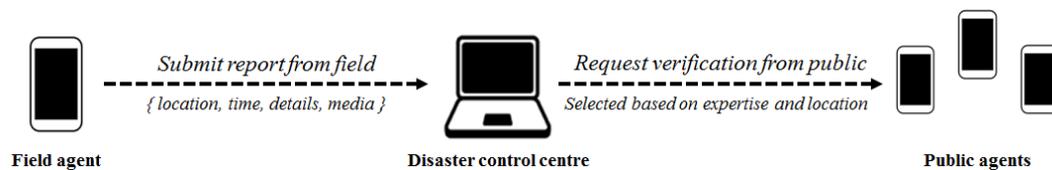


Figure 9.1: The crowdsourcing framework.

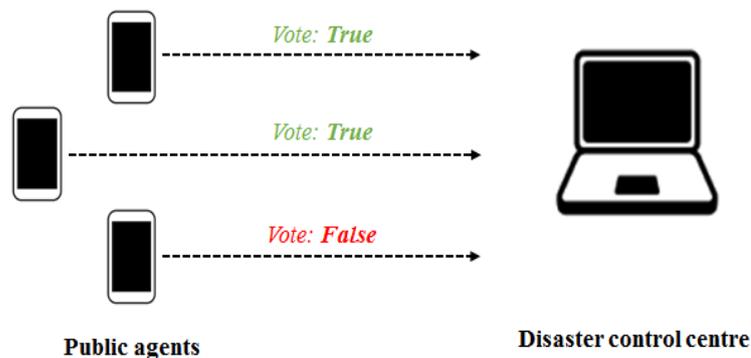


Figure 9.2: Public agents vote on report accuracy and results are aggregated at disaster control centre.

If the disaster control centre flags the report for further investigation, a set of parameters can be entered to determine which field agents will receive the verification microtask. These parameters include the expertise of the public agent as well as the location. Based on the resources available, the disaster control centre can also specify the number of public agents the task will be assigned to. When the public agents vote on the accuracy of the report, they can also make edits

to the original report to further clarify details. Once complete, this information is sent back to the disaster control centre and aggregated by the online system. Disaster control centre staff can view which reports have been verified as accurate and add that information to the Situation Report as necessary. Further, the system automatically takes into account the accuracy of the report (as voted by public agents) and the number of edits made to the original report to calculate a score, which is added to the profile of the original field agent who submitted the status.

In addition, a history of all micro-reports that have been submitted by public agents is recorded by the system (including the verification information). While the framework presented in this chapter does not perform any analysis of this data, it could be used to provide useful insights for mitigation and preparation of future disasters as well as better allocation of resources within a specific area.

9.4.2. Example mobile and desktop interfaces

To demonstrate the crowdsourcing features demonstrated in this chapter, the following interfaces were developed. The first interfaces to be demonstrated are the mobile features for field officers or the public. There are three main features:

1. Field agents can submit a report at any time through a mobile interface (see Figure 9.3a). The report automatically takes the location of the user using the mobile device's inbuilt Global Positioning System (GPS). However, if GPS is unavailable on the device, inaccurate, or the user is submitting the report from a different location at a later time, the user can also manually select their location using the map. Further, the user can write a brief description of the incident that is being reported. Finally, the user can also add photos to the report using the inbuilt camera of the mobile device.
2. Mobile users can complete verification tasks - a new type of microtask. These tasks are assigned to users from the emergency control centre via the web-based interface. Typically, a user will receive a task based on their location relative to the incident needing verification or their expertise in the type of incident that needs investigating. To engage in a verification task, the user touches the task from the list and is taken to the next screen (see Figure 9.3b).
3. The third interface displays the information contained in the report needing verification, including the location, the description of the incident, and any photographs that were included. To verify the task the user is presented with two options: True, which is selected if the information contained in the report is accurate, and False, which is selected if the information contained in the report is inaccurate. Upon completion of the verification task this information is sent back to the control centre and the task is removed from the user's assigned list.

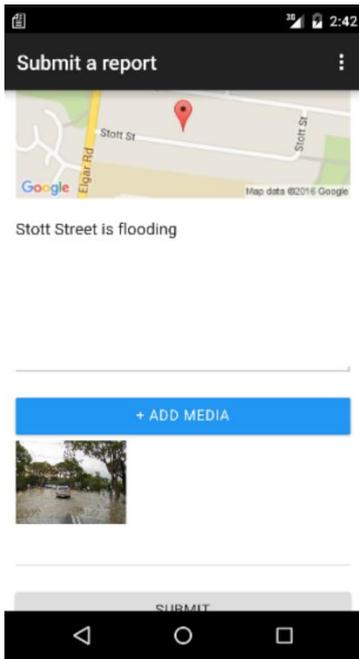


Figure 9.3a: User can submit a report through a mobile interface.

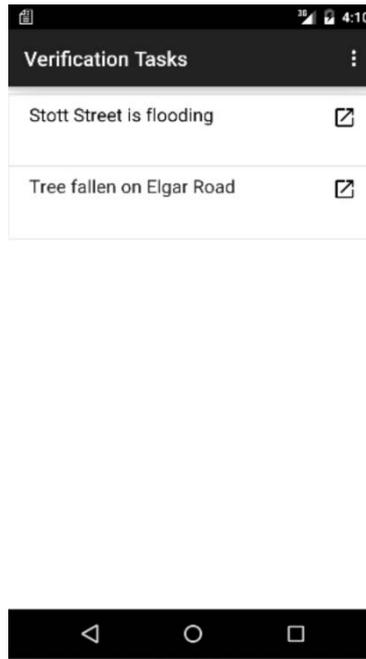


Figure 9.3b: User can access a list of verification tasks that have been assigned to them.

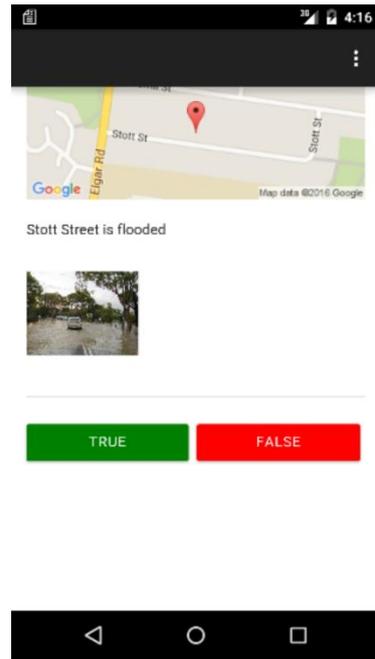


Figure 9.3c: The verification task requires the user to vote True/False.

An example of the control-centre staff web-based interface is displayed in Figure 9.4 and Figures 9.5a, 9.5b and 9.5c. While the collaborative report writing component of the web-based interface remains the same, there are additional controls added to handle the new microtasks for crowdsourcing. Firstly, there is a feed to display incoming micro-reports. Users can view the details of the micro-report, the identity of the author, the location of the incident, as well as any photographs submitted. There are two options to handle the micro-report: *Add to Report* inserts the details of the incident directly into the sitrep; *Verify* triggers the crowdsourcing verification task where the micro-report is sent back to other field agents to vote on the accuracy of the report (as shown in Figure 3c).

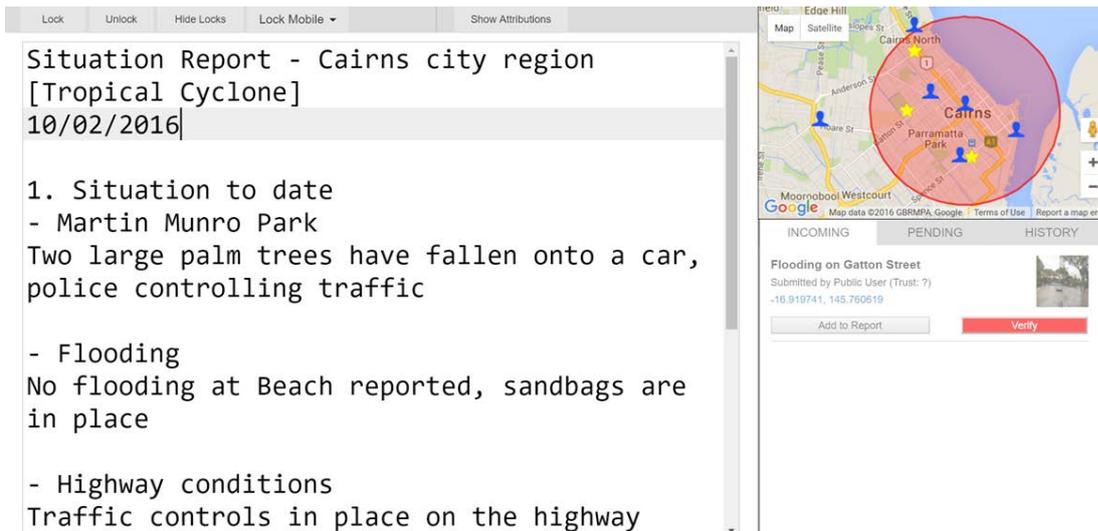


Figure 9.4: An example of the web-based crowdsourcing interface.

When a microreport is sent out for verification from mobile users, a form is presented to the desktop user for inputting options about the verification task (see Figure 9.6). In the developed prototype, the options are number of users to send the task to, and the choice to include public and/or trusted users in the verification task. Once submitted, the system automatically determines which mobile users will receive the verification task based on their GPS location and the options selected by the desktop user. As mobile users complete the verification task (using the interface in Figure 9.3c), the task is updated to display the results (as shown in Figure 9.5b). For example, if a user votes negatively their response will appear in red with a minus symbol (-), whereas a user voting positively will display in green with a plus symbol (+). If a user has not yet voted, the response will remain grey with a question mark (?). Desktop users can hover over the user information to display their identity and trust score (e.g. “Council | Trust: 1”).

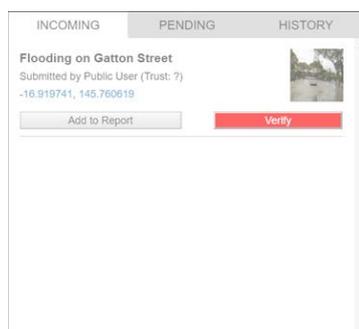


Figure 9.5a: Incoming microreports can be added to report or verified.

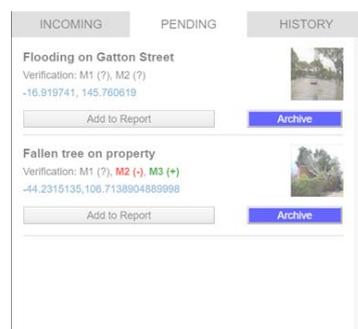


Figure 9.5b: Microreports awaiting verification from other users.

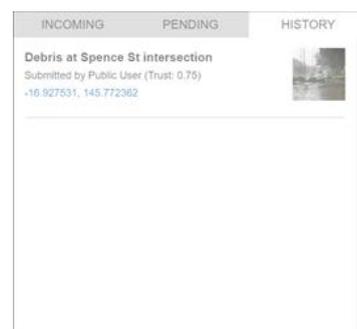


Figure 9.5c: Archived microreports are stored for future reference.

Verify information

No. Users: 1

Include: Trusted Public

Send Cancel

Figure 9.6: Options for submitting a verification task to mobile users.

9.5. Summary

In this chapter a framework for crowdsourcing within the diffsync technique was introduced. This framework extends the multisynchronous framework that was developed in Chapter 8. Crowdsourcing is an important feature required by remote EM practitioners within a collaborative report writing system because it provides additional support for the existing work roles within their EM teams. Chapter 5 identified that crowdsourcing was not currently supported by diffsync, thus the new framework for crowdsourcing was developed in this chapter.

The new crowdsourcing framework provides additional support for field officers in emergency response, who are typically working on mobile devices. This framework is an extension of the multisynchronous framework presented in Chapter 8 and allows field officers to initiate communication and submit reports. Further, it supports information verification by allowing trusted users or the public to vote on the accuracy of information when requested. Information accuracy was a key requirement for collaborative EM report writing systems as specified by the practitioners surveyed in Chapter 4.

The new crowdsourcing framework presented in this chapter, combined with the flexible locking, user attribution, and multisynchronous frameworks presented in Chapters 6, 7, and 8, comprise the research artefact for supporting collaborative EM report writing for remote practitioners. Chapter 5 identified that diffsync lacked support for these features so they were developed, thus making the research artefact complete to meet the needs of the EM practitioners for collaborative report writing. In the next chapter, this research artefact is evaluated using several design science techniques including benchmarking, a static analysis based on groupware heuristics, development of a prototype, comparison study, Perceived Usability and Ease of Use, and an enhanced cognitive walkthrough.

10. Evaluation

This chapter describes the evaluation of the research artefact developed in this thesis. The aim of this chapter is to determine how well the research artefact meets the requirements specified by the EM practitioners in Chapter 4. Several evaluation techniques are utilised, including benchmarking, static analysis based on groupware heuristics, the development of a prototype, a comparison study, a technology acceptance survey, and an enhanced cognitive walkthrough.

10.1. Overview

In the previous chapters of this thesis, several features were iteratively developed to make up a framework to support collaborative report writing for EM. These features include flexible locking, user attribution, support for multisynchronous editing via mobile microtasks, and support for mobile crowdsourcing to gain and verify new information from the disaster environment. Now that all of these features have been demonstrated, this chapter discusses the development of a report writing prototype that incorporates all of the features into a single system. This artefact is then subjected to several additional evaluation methods based on the *Design Science* research methodology.

Specifically, the aim of these evaluation methods is to determine whether the prototype meets the needs of real-world EM practitioners. In Chapter 4 of this thesis, the results of a survey were presented in which several EM practitioners from North Queensland provided information about their current experiences with report writing for disaster response. The results gained through this survey indicated that there are several categories of software-based improvements that would benefit collaborative report writing.

The structure of this chapter is as follows: Section 10.2 provides an overview of the evaluation methods used to evaluate the artefact developed in this thesis and the hypotheses for each study. These evaluation methods are described in detail and results provided in the following sections, including benchmarking in Section 10.3, static analysis in Section 10.4, prototype development in Section 10.5, comparison study in Section 10.6, Perceived Usability Perceived Ease of Use (PUEU) in Section 10.7, and Enhanced Cognitive Walkthrough (ECW) in Section 10.8. Finally, the results are summarised in Section 10.9.

10.2. Evaluation Methods

According to Hevner et al. [30], artefacts resulting from design science research should be evaluated in terms of utility, quality, efficacy using appropriate methods. Within these categories, artefacts can be evaluated on metrics including functionality, completeness, consistency, accuracy, performance, reliability, usability, fit with the organisation, and any other relevant quality attributes [30]. Accepted evaluation techniques can include: *observational* methods (e.g. case studies and field observations); *analytical* methods (e.g. static analysis, architecture analysis, optimisation, and dynamic analysis); *experimental* methods (e.g. controlled experiments and

simulations); *testing* methods (e.g. functional testing and structural testing); and *descriptive* methods (e.g. informed arguments and scenarios) [30].

The evaluation methods used for the artefact produced in this study are presented in Figure 10.1. The evaluation methods selected were based on the type of artefact being tested [145]. In this research, the primary artefact generated is a *framework* for supporting the needs of collaborative EM report writing in remote North Queensland. To evaluate the framework the following three methods were selected:

- *Benchmarking*: To test the performance of the framework, several experimental simulations were conducted that compare the improved diffsync technique against a non-improved diffsync implementation. The aim of the benchmarking evaluation is to determine whether the improved diffsync framework has any performance limitations in terms of efficiency, scalability, correctness and integrity. The methodology and results are described in Section 10.3.
- *Static Analysis*: A static analysis is a technique in which an artefact is examined in terms of static qualities. In this study, the aim of the static analysis is to determine the extent to which the framework meets the functional design requirements for groupware. The evaluation criteria selected are based on the Groupware Heuristic Analysis [146] which focuses on the manner in which aspects of face-to-face collaboration are translated into software interfaces. The methodology and results are described in Section 10.4.
- *Prototype*: The aim of developing a prototype is to determine the feasibility of the framework artefact for supporting a collaborative EM report writing application. The development of the prototype and its ability to support the needs of EM practitioners is described in Section 10.5.

The development of a prototype results in an additional artefact, known as an *Instantiation* type artefact [145], which can be subjected to further evaluation. To evaluate the Instantiation artefact, an additional three methods were selected:

- *Comparison*: The aim of the comparison study is to compare the artefact against an existing report writing system. Specifically, the artefact is compared against asynchronous Microsoft Word which was the most common report writing methodology identified in the survey of EM practitioners in Chapter 2. The methodology and results are described in Section 10.6.
- *Perceived Usability and Ease of Use (PUEU)*: To evaluate the system for acceptance and perceived usability and ease of use, the PUEU was conducted with a sample from the target population of EM practitioners. The methodology and results of the PUEU are described in Section 10.7.

- *Enhanced Cognitive Walkthrough (ECW)*: The aim of the ECW is to investigate usability issues that affect the functional and operational utility of the key features developed for EM collaborative report writing. The methodology and results of the ECW are described in Section 10.8.

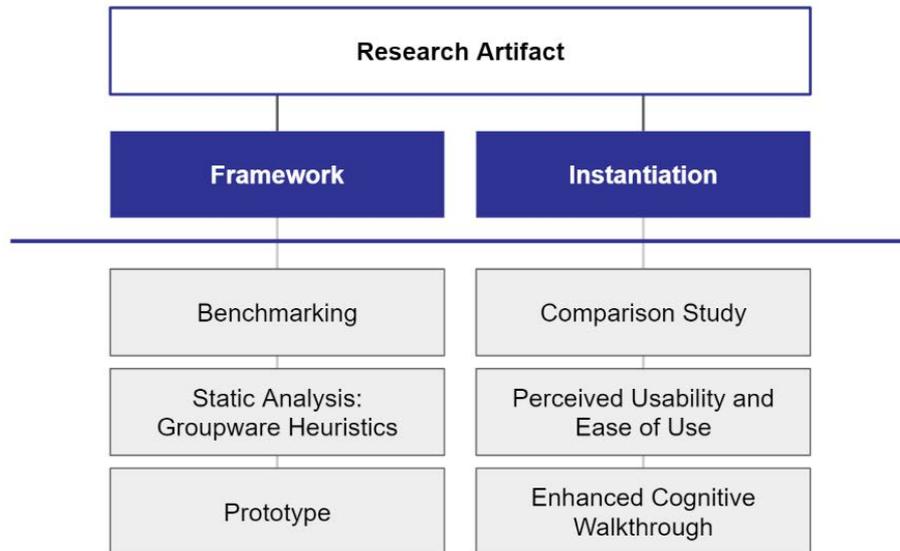


Figure 10.1: The evaluation process for the collaborative EM report writing artefact.

The hypotheses for the evaluation techniques are presented in Table 10.1:

Table 10.1: Hypotheses for the artefact evaluations.

<i>Evaluation Technique</i>	<i>Hypotheses</i>
<i>Benchmarking</i>	<p>H1: There will be no significant reductions in the efficiency of diffsync with the locking technique implemented.</p> <p>H2: There will be no significant reductions in the scalability of diffsync with the locking technique implemented.</p> <p>H3: The locking technique will maintain integrity in locked conditions.</p> <p>H4: There will be no significant reductions in the efficiency of diffsync with the user attribution technique implemented.</p> <p>H5: There will be no significant reductions in the scalability of diffsync with the user attribution technique implemented.</p> <p>H6: The user attribution technique will maintain accuracy/correctness.</p>
<i>Static Analysis (Groupware Heuristic Evaluation)</i>	<p>H7: The framework will satisfy all Groupware Heuristics (with a rating of 1 or above).</p>

<i>Prototype</i>	H8: A prototype will be successfully developed from the framework that meets the requirements of EM practitioners surveyed in Chapter 2.
<i>Comparison Study</i>	H9: Participants will prefer the instantiation artefact over asynchronous Microsoft Word in all evaluated features.
<i>PUEU</i>	H10: Participants will rate the instantiation positively for perceived usefulness, ease of use, and overall acceptance of the technology.
<i>ECW</i>	H11: No serious usability issues will be identified in the instantiation artefact (all ratings will be 3 or higher).

10.3. Benchmarking

Previous studies have determined that *diffsync* is highly scalable [24]. As the framework for collaborative EM report writing developed in this thesis has introduced several additional methods into the implementation of *diffsync*, performance of synchronisation may be affected as the number of collaborators increases. Thus, several benchmarking simulations were conducted to verify that the scalability and efficiency of synchronisation was maintained when compared to an implementation of *diffsync* without the new features. The two features that were tested are flexible locking and user attribution. In addition, flexible locking was also tested for locking integrity and user attribution was tested for correctness. Only locking and user attribution were tested specifically because the additional features were derived from these two features. The results of these simulations are reported in Sections 10.3.2 and 10.3.3.

10.3.1. Simulation tools

The following software tools were used to develop a prototype implementation of the locking and user attributions framework as well as run the evaluation benchmarking:

google-diff-match-patch library:

The *google-diff-match-patch* library (code.google.com/p/google-diff-match-patch/) offers a robust collection of functions to perform the three main operations required when synchronising plain text: *diff*, *match*, and *patch*. The library provides APIs for Java, JavaScript, Dart, C++, C#, Objective C, Lua and Python. *Diffsync* relies on such operations perform synchronisation, while attempting to maintain correctness and convergence [24].

Node.js

Node.js is a highly scalable, server-side JavaScript environment based on Google's V8 runtime engine [147]. It is commonly used to develop real-time applications (such as web servers and

networking tools) due to its single-threaded, asynchronous nature. Node.js is also non-blocking. All of these features make useful for developing real-time, collaborative text-editing systems.

Ace editor

Ace editor is an embeddable text editor that is written in JavaScript. Thus, it can be easily embedded into existing applications that run on web technologies, and is compatible with Node.js. Ace editor supports many features including syntax highlighting, automatic indentation, line wrapping, and custom highlighting. We selected Ace editor due to its compatibility with the other technologies used in our prototype, and to utilise its highlighting functionality for the visual representation of locks. The client-side of the application is written in JavaScript, so it is simple to embed an Ace editor into the page. Further, diffsync supports synchronisation of plaintext, which means that Ace editor can be synchronised without any modifications.

Chance

Chance is an open source JavaScript library that can generate random strings and numbers for use in automated tests. While *Chance* provides only pseudo-random generation of strings and numbers, this is considered suitable for our purposes of simulating random user input. More specifically our test implementation utilises the random word functionality of *Chance*, which returns a semi-pronounceable nonsense word.

10.3.2. Locking simulations

To measure scalability, a custom script was developed in JavaScript which ran in the browser alongside the web-based implementation of diffsync with optional locking. The script provided random input and deletion of text to simulate the behaviour of users in a collaborative text editing session. Random text was generated using *Chance.js*⁹. During each synchronisation loop, clients had an equal chance of adding text, deleting text, or making no changes at all. Additionally, clients had an equal chance of requesting a lock on a random subsection of the shared text or removing locks.

Five conditions were tested with an increasing number of simulated users: one user (1), two users (2), five users (5), ten users (10), and twenty-five users (25). Each condition was run over 250 synchronisation loops. Two timestamps were recorded during each synchronisation loop: one at the beginning of the cycle, and one at the end of the cycle. The difference between these two timestamps was recorded and the average synchronisation time was calculated based on this data.

The results of this scalability experiment are shown in Figure 10.2. These results indicate that the locking implementation follows a linear trend in scalability within the parameters tested.

⁹ <http://chancejs.com/>

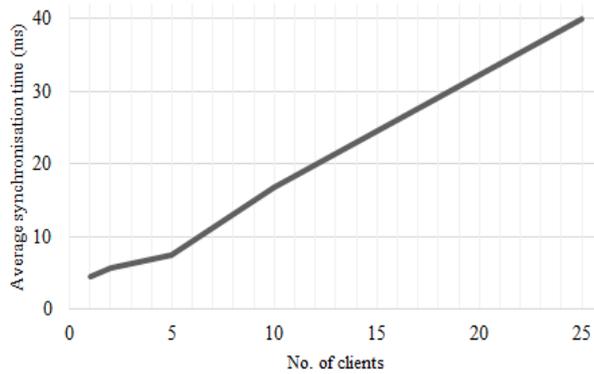


Figure 10.2: The average synchronisation time followed a linear trend.

To determine the effect of the locking implementation on the efficiency of diffsync compared with no locking, the following benchmarking experiment was conducted. Ten clients were simultaneously connected to the same server running the flexible locking implementation. A custom script running on all ten clients generated random user input including adding new text and erasing existing text. In addition, each client also requested random lock and unlock requests on the shared text.

The average time for a synchronisation loop to complete successfully was recorded over 250 trials. This process was repeated with locking and without locking. The results of these trials are represented in Figure 10.3. On average, there was a difference of 1ms between locking and no locking for synchronisation. This is considered to be an acceptable cost that would not noticeably affect usability of the system. Interestingly, some trials indicated that locking was more efficient whereas other trials indicated that no locking was more efficient. This is attributed to the random nature of the input generated in the test script - if a client request a lock over most of the text for a long duration, less text was added to the locked region and the system performed an increased number of empty synchronisation loops.

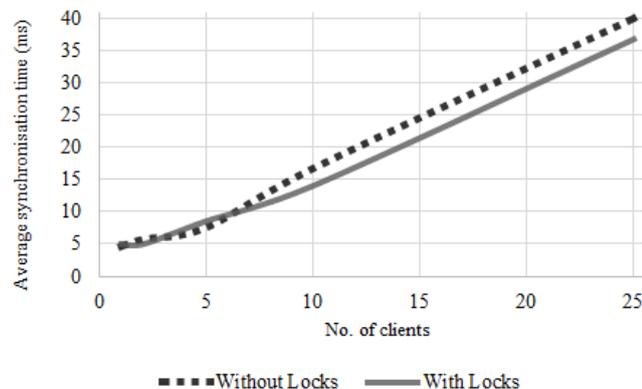


Figure 10.3: Average synchronisation time of non-locking vs. locking.

To demonstrate the integrity of the locking framework another simulation was conducted. In this simulation, a server was initialised with one paragraph of text. One client was initialised with a lock on the whole paragraph. Five additional clients were connected to the server, and each client ran a script that randomly added and deleted text from the locked area (using *Chance.js*). This simulation ran for ten minutes. After the simulation was complete, the initial state of the shared text was compared to the final state of the shared text across all clients and the server (using a diff tool). Based on the client-side implementation of the prototype and its methods for checking lock privileges before outgoing synchronisation is triggered, it was expected that the original state of the text would be maintained across all clients. After running the simulation, it was found that there were no differences between the texts for all clients, and that the value of the text was identical to its initial state, indicating that the lock maintained integrity.

10.3.3. User attribution simulations

To measure scalability, a simulation ran with the following parameters. Ten (10) clients were connected to the server. Each client generated random input (using *chance.js*) and randomly erasing content to simulate a collaboration session. The time taken to complete a full synchronisation loop of our *diffsync* implementation was calculated by logging a timestamp at the beginning of a loop, and a timestamp at the end of a loop, and subtracting the difference. Each condition was run across 600 synchronisation loops. This same simulation ran for both the user attribution version and the standard *diffsync* version.

After running the simulations, the data was cleaned using the Tukey method to remove outliers. This resulted in $n = 509$ (user attribution condition) and $n = 500$ (no user attribution condition) valid data points. An independent-samples t-test was conducted to compare the synchronisation time between the two applications. There was no significant difference found between the synchronisation time for both groups ($t = 8.81$, $p > 0.05$ *n.s.*). This indicates that the addition of user attribution techniques did not have a significant effect on synchronisation speed in the conditions tested.

To verify that scalability is maintained as the number of clients increases, the following benchmarking test was conducted. A custom script was developed which ran in the browser alongside the web-based implementation of *diffsync* with user attribution. The script provided random input (using *chance.js*) and deletion to simulate the behaviour of typical users in a collaborative text editing session. For each synchronisation loop, clients had an equal chance of adding text, deleting text, or making no changes at all. All clients were provided with a random identification number upon connecting to the server which was used in place of a user name.

There were five conditions tested in terms of number of simulated users: one user (1), two users (2), five users (5), ten users (10), and twenty-five users (25). Each condition was run over 250 synchronisation loops. Two timestamps were recorded: one at the beginning of a synchronisation loop, and one at the end of a synchronisation loop. These timestamps were recorded for further analysis. The difference between these two timestamps was also recorded and the average synchronisation time was calculated based on this data.

The results of this scalability experiment are shown in Figure 10.4. Based on these results, it can be concluded that the diffsync with user attribution implementation followed a linear trend in scalability.

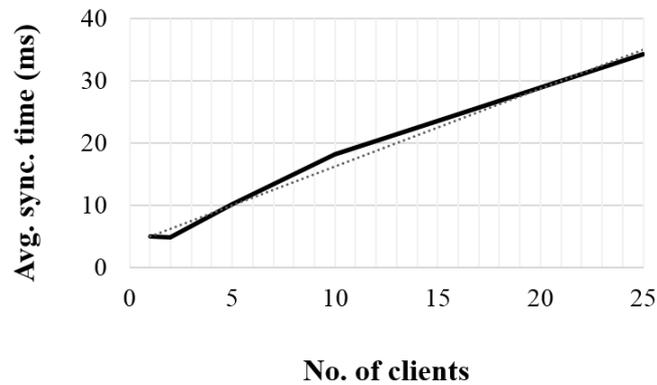


Figure 10.4: The synchronisation time followed a linear trend for scalability.

To evaluate the accuracy of the technique at correctly tracking user contributions, another simulation was conducted with three (3) clients. Using a modified version of the custom script from the previous benchmarking studies, each client was given a probability of adding text. Client 1, 2 and 3 had 50%, 30%, and 20% probability of adding text on a synchronisation loop respectively. The simulation was run over one-thousand (1000) synchronisation loops. At the end of the simulation, the percentage calculated for each user was recorded. It was expected that the total contributions for all three users would be 100%, and that individual contribution should be close to the expected values due to the random nature of the script. The results are shown in Figure 10.5 and indicate that the technique maintains correctness.

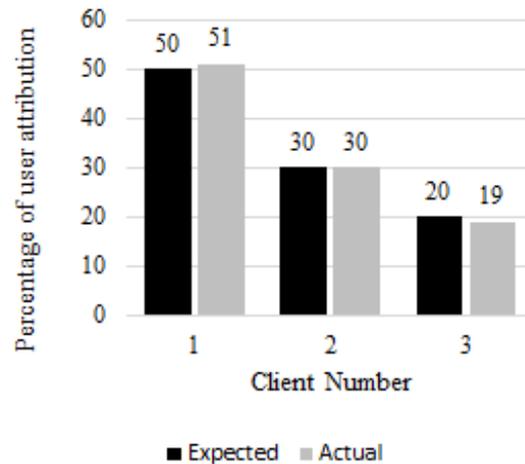


Figure 10.5: The user attribution mechanism maintained accuracy (with random variation).

10.4. Static Analysis: Groupware Heuristic Evaluation

The static analysis is an analytical evaluation technique used in *Design Science* research to evaluate a developed artefact for static qualities [30]. Static analysis has been used to evaluate similar systems for EM [148]. To evaluate the prototype artefact presented in this chapter, the static qualities refer to the groupware heuristics defined in the Groupware Heuristic Evaluation [146]. The *Heuristic Evaluation* (HE) is a widely accepted evaluation method for diagnosing potential usability problems in user interfaces [146]. The HE is popular due to its high success at detecting the most common usability problems, without requiring extensive resources in terms of time and end users. Instead, a small set of recognised usability principles are applied to the system. HE has been demonstrated to be effective when used by both usability experts and non-usability experts, due to the heuristics being well documented and easy to learn. Baker, Greenberg and Gutwin [146] propose a modified set of heuristics that apply directly to groupware systems and teamwork. The authors compared the performance of these groupware-specific heuristics with Nielsen’s usability heuristics to determine whether the same benefits applied, and found that the groupware-specific heuristics were a useful metric for determining usability issues in groupware interfaces [149].

10.4.1. Heuristic 1: Provide means for intentional and appropriate verbal communication

Heuristic 1 refers to the mechanisms for which users are supported in direct conversation alongside the collaborative task. In most groupware systems, direct verbal communication is not supported. Instead, chat systems can be used (whether it is type-and-submit or real-time text synchronisation). The developed artefact in this research uses the advantages of the client-server model of the framework to provide both types of communication. That is, it has an explicit chat system (type-and-submit) with messages attributed to the user. In addition, the synchronised document could be used for small interactions as it is synchronised in real-time. However, part

of the reasoning for having the additional chat feature was to avoid creating additional noise in the main document that is not part of the actual report. Interestingly, EM practitioners who were surveyed (results displayed in Chapter 2) did not rate a chat feature highly on the list of requirements for a collaborative report writing system, perhaps indicating that email, face-to-face, phone, and radio-based communication is preferred in this community.

10.4.2. Heuristic 2: Provide means for intentional and appropriate gestural communication

Heuristic 2 refers to support for the physical gestures that are typically used in face-to-face collaboration. Explicit gestures are used alongside verbal communication during collaboration to convey intentional communication. In groupware systems, explicit gestures are not typically supported unless there are direct methods such as live video feeds, avatars, or telepointers. The developed artefact utilises a visual text cursor that is personalised with colour for each user via the user attribution framework. This provides a simple method for gestural communication, but it is restricted in that it only provides a subtle cue as to the user's current actions. Improvements could be made by sharing the actual mouse cursor movements of the user to allow for direct gestural communication.

Further, another form of gestural communication could be achieved by sharing any currently selected text with the other users. In the current artefact, this may cause usability issues due to the similarity of the locking visual attribution with the text highlighting that occurs when text is selected. However, this could be overcome by changing the visual attribution of locks (e.g. using icons or textures) or reserving a specific highlight colour that is only used for intentional and gestural communication. Based on the results of the survey in Chapter 2, it is evident that much collaboration occurs outside of the reporting system before the information is recorded. Improvements could be made by integrating video and audio feeds into the system, though collaborators could use existing technologies such as Skype or Google Hangouts alongside this system to achieve the same result.

10.4.3. Heuristic 3: Provide consequential communication of an individual's embodiment

Heuristic 3 refers to the bodily actions such as position, gesture, and movement of head, arms and eyes that give off secondary information in face-to-face collaboration. Similar to Heuristic 2, avatars and live video sharing can provide explicit means to display these actions. However, this can be much more difficult to achieve in a groupware system. Telepointers and shared mouse cursors can only provide subtle suggestions as to how a person is interacting within a shared workspace. In the presented framework, the real-time nature of synchronisation coupled with the user attribution features means that as changes are being made, it is easy to identify who is working in the document, where they are working in the document, and what changes are being made. Further, the shared text caret allows other participants to view the current location of a collaborator in the report. However, this does not guarantee that the collaborator is actually

viewing the document at that time. Thus, improvements could be made to display a status for each user, for example after a period of inactivity the user could be display as *Away*.

10.4.4. Heuristic 4: Provide consequential communication of shared artefacts

Heuristic 4 suggests that users should be informed about who is manipulating a virtual artefact within the shared workspace, and suggests that feedback should be provided as the artefact changes. Further, an artefact's appearance should also show traces of its history to show how it came to be in its current state. In groupware applications this means that collaborators should see not only the final state of an artefact after an action (for example dragging text to another paragraph), but also all the intermediate steps of this interaction (for example selecting the text, dragging it, and dropping into another paragraph). This provides context to the actions made by users in the workspace and can be used by other collaborators to influence future changes and actions.

In the developed artefact, the text input area displays all actions made by users in real-time. Further, the user attribution framework ties each action with a user by visually highlighting the author of an action and also recording the history of changes made. The history of edits can be viewed at any time by any collaborator to gain context about how the current report has changed over time. However, there is room for improvement in that other types of actions could be recorded and displayed in the history section, such as the intermediate steps involved in making the changes instead of simply text added and text deleted.

10.4.5. Heuristic 5: Provide protection

In physical collaboration settings, there are typically social constraints such as turn-taking which prevent collaborators from concurrently working on a shared artefact when conflicts may arise. In contrast, web-based groupware systems typically provide equal access to all collaborators to make changes to shared artefacts at any time, with limited knowledge of how these actions will affect other collaborators. While predefined roles and policies can influence the behaviour of collaboration, there are no guarantees that these rules will be followed within the system unless specific protection is provided. Human error or misuse can also contribute to additional conflicts that may or may not be intentional.

The demonstrated artefact provides direct protection to collaborators in the form of flexible locking. Any user can request an exclusive lock on any subsection of the shared text at any time, preventing other collaborators from making changes while that lock is active. The fact that the locks are optional and dynamic means that multiple locks can exist simultaneously without necessarily having a negative impact on other collaborators as they can still edit non-locked section at the same time. Further, the user attribution couple with flexible locks means that users

are made aware of who owns the lock. One of the highly requested features by EM practitioners (see Chapter 2) was an administrator to oversee the document. While the framework will automatically release a lock if the owner leaves the report, an administrator could also release locks when necessary if the situation arises.

10.4.6. Heuristic 6: Management of tightly and loosely-coupled collaboration

In Heuristic 6, coupling refers to the degree to which people are working together. According to research, people typically move between individual and group-based tasks in a collaborative scenario, otherwise known as loosely and tightly coupled collaboration [146]. To this end, it is suggested that collaborators who are working individually should still maintain awareness of what other collaborators are doing in the workspace, for example an additional interface could display this. This information allows a user to infer without direct request when they may be required to assist another user with their current task.

In the demonstrated artefact, the locking feature currently allows a user to claim an individual section of the shared text which can be edited in isolation from other users. However, as the lock is enforced within the shared text, the changes being made by other users are still visible and the lock changes are also visible to the other users. Users can maintain control over whether or not this information is visible by toggling the visibility of the user attribution and locking layers. Another features that supports this heuristic directly is the chat feature in which a user could request direct assistance from other users if necessary.

However, improvements could be made by providing support for private interfaces that allow smaller groups of collaborators and individuals to work on a smaller section of the report in isolation. Such a feature should allow these users to easily move between private and public group views to support the natural flow of work. Additionally, a visual overview could be added to the document that displays the current location of users within the document alongside additional user attribution information.

10.4.7. Heuristic 7: Allow people to coordinate their actions

Heuristic 7 specifies that people need to be supported in turn-taking and negotiating the use of a shared workspace. This means that some tasks need to occur at the right time in the right order within defined constraints, while avoiding duplication of actions, and preventing multiple individuals from manipulating the same shared object with conflicting intentions. In face-to-face collaboration, people can easily see what other collaborators are doing and plan their actions accordingly to ensure that resources are shared. Groupware applications need to provide additional methods due to the separation of collaborators by technology.

In the demonstrated artefact, the flexible locking and user attribution features provide explicit methods for users to avoid concurrent editing of protected sections of the document. These features also provide a cue to other users that another user is working on a specific section of the report. Further, the user attribution features such as colour highlighting and displaying the text caret allows other users to determine who authored each change and in which section of the report. The history feature can provide further context to other users as to how each user contributed to the report over time, and this information can be used to plan future actions. The chat feature provides another informal mechanism for people to make others aware of their actions, or to suggest additional constraints. Within the EM community (such as those practitioners surveyed in Chapter 2) there are already existing protocols and methods when compiling reports for a disaster, and these protocols would follow through to a web-based collaborative system to further influence how each collaborator acts during a scenario.

10.4.8. Heuristic 8: Facilitate finding collaborators and establishing contact

Heuristic 8 refers to the mechanisms through which collaborative situations arise, for example how people determine if an individual is available to participate in a collaborative task, however to initiate contact with potential collaborators, and how to ensure that potential collaborators are equipped with the right tools to engage in the collaborative task. Face-to-face meetings can be spontaneous, casual and unplanned, such as two people passing each other in the hall and having a discussion about a shared project. On the other hand, web-based systems face challenges in supporting spontaneity due to the distance that separates collaborators.

In the scope of report writing for EM, it is much less likely that collaborative meetings will be spontaneous as the tasks revolve around an ongoing disaster situation. Collaborators typically have predefined roles within their disaster management organisation, and will participate in collaborative tasks as appropriately for their role. This means that it may be less important to support spontaneous collaboration for EM report writing, though there is still a need to support users in connecting to the shared workspace if they are working from different locations.

The artefact demonstrated in this chapter provides mechanisms to determine who is currently available in the shared workspace through the list of users, which provides the user's name, their role, and an identifying colour that is tied to their edits. Users could infer which collaborators are not currently in the shared workspace by using this information and contact them to join if necessary. Further, the geographic location of mobile users in the field can be viewed on a map. This allows control centre staff to determine which field agents are located in ideal locations to provide micro-reports from the field. Similarly, the crowdsourcing aspect of the framework provides mechanisms for determining which field agents are available to confirm information in the field.

10.5. Prototype Instantiation

Using the framework developed in Chapters 6 - 9, the following prototype system was developed to support EM practitioners in collaborative report writing. Figure 10.6 displays an overview of the main interface of the system. There are several key features in the main interface:

- *Report writing interface*: The left section of the interface contains the text input area of the system in which EM practitioners compile the report. The users can input the *Sitrep Number* (e.g. a chronological representation of which sitrep is being written), a *Version Number* (e.g. the current version of this sitrep), and a date and timestamp which is applied automatically based on the system time. There are additional items in this menu, which are described in detail in the next section.
- *Menu*: There are several options available in submenus above the text input area of the system. Firstly, the *Edit* menu provides options to insert formatted text in Markdown (e.g. Header, Sub header, List Item, Link, and Horizontal Rule). The *Lock* menu presents options to utilize the flexible locking features as presented in Chapter 6 of this thesis, for example *Lock Region*, *Lock Region (Group)*, *Unlock Region*, or push a microtask to a mobile user to provide information (using the methodologies presented in Chapters 6 and 7). The *View* menu presents options to toggle the display of the locking and user attribution overlays, and also an option to open the *History* which displays the version history of the document (using the methodologies presented in Chapter 7). The *Analytics* menu presents options to view Keyword Analysis and Relative User Contribution, as demonstrated in Chapters 7 and 8 respectively. Finally, the *+Share* menu provides the link to a static version of the document that is updated in real-time, which can be shared with stakeholders who need access to real-time information but do not require editing privileges.
- *List of users*: The right-hand side of the interface contains a section that displays the current authors who are accessing the collaborative report. This includes the name of the user, their affiliation (e.g. *Roads department*), their user type (e.g. *Control centre* or *Mobile user*), and a colour to denote user attribution for that author (using methods from Chapter 7). If the user is a *Mobile user*, an option appears to view that user on a map, which displays their location.
- *Chat*: Below the list of users there is also a chat interface that can be used for informal communications between collaborators currently sharing the workspace. The chat messages are attributed to each author using the user attribution techniques identified in Chapter 7, including user name, colour and timestamp.
- *Analytics*: Using the RAKE method presented in Chapter 8 for automatic summarisation for mobile users, the artefact contains a web-based view that displays the automatic

summarisation and a list of keywords from the current report. This information is updated in real-time as changes are made to the document, allowing collaborators to gain a quick overview on the contents of the report and the key terms being used.

- *Static real-time document view*: The system supports an additional view that can be shared to stakeholders that require real-time access to the report without editing privileges. This page updates in real-time using the diffsync technique. Additionally, the text is rendered based on the markdown formatting that can be used in the report. This means that using stylesheets the appearance of the report can be customised based on the needs of a specific organisation.
- *Mobile features*: These aspects of the artefact remain similar to their presentation in Chapters 8 and 9.

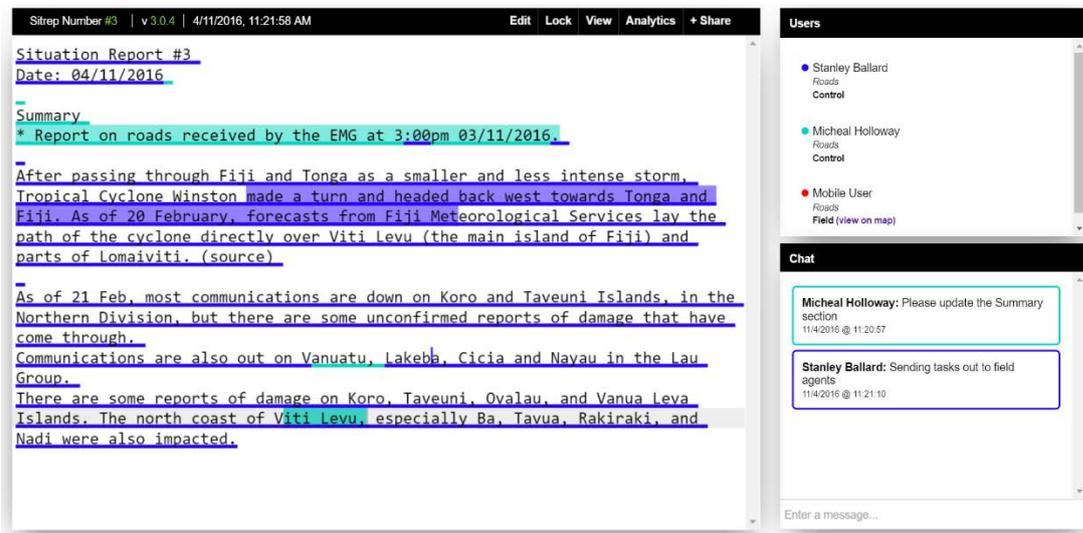


Figure 10.6: The main interface of the prototype developed using the EM framework.

The prototype artefact handles the requirements of the EM practitioners as follows. First, real-time synchronisation features are provided by the diffsync technique that was used to handle synchronisation in the prototype. Using web-based technologies, multiple users are able to connect to the system through any web browser and the server automatically handles synchronisation of text as changes are made. As diffsync is naturally convergent by design [24], users will always view the most up-to-date version of the shared text and any errors will be handled automatically by the system.

In terms of locking features, the system provides support for both single-author locking and group-based locking. This locking is flexible in the sense that any author can trigger a lock on a section of the report at any time, providing there are no prior conflicting locks on that section. Further, the framework provides support for group-based locking which is managed by using the metadata of a user that is connected to the system (e.g. their department or organisation). Further,

as the server synchronises the shared text in real-time, other stakeholders can view the document without editing privileges, allowing them to obtain information from the report as it changes but not contribute.

User attribution features are integrated throughout the system using the framework demonstrated in Chapter 7. For example, the interface has an explicit list of users currently in the collaborative session, including details of identity, EM affiliation, and a unique colour to identify that user within the system. As each user makes changes to the shared report, explicit attribution of authorship is provided through a visual representation (the text is underlined with that author's colour and a tooltip appears when the cursor is hovered over that text). Each user's text caret is shared with all other collaborators to promote workspace awareness by indicating their current position in the document.

Multisynchronous features are provided using the frameworks demonstrated in both Chapter 8 and 9. In the unified prototype demonstrated in this chapter, the location of mobile users can be viewed by clicking a link displayed in the user list. Utilising the locking functionality of the system, authors can lock a section of the report and push it out to that mobile user as a microtask. In addition, the mobile interface is designed to support the separate needs of the field agent, providing only the necessary information needed to complete the task (e.g. automated summary of the report to provide context if necessary, short tasks with brief descriptions, and limited input options). This means that both desktop and mobile users can contribute to the same report using different methods. While *diffsync* already provides support for poor network environments by design [24], these improvements provide further support for this as users can synchronise when the network becomes available. Battery life improvements are supported by using a push-based mechanism for synchronisation when on a mobile device, as demonstrated in Chapter 8.

Finally, the additional features suggested by the surveyed EM practitioners are also included. For example, the automated summarisation and keyword extraction techniques were supported for mobile users to provide context, and this was extended for the artefact as a web-based display using RAKE [137]. Informal chat is included by exploiting the client-server nature of the system and integration with the user attribution features to assign identity to chat messages. Finally, while the scope of this research was limited primarily to text synchronisation, the crowdsourcing framework (as implemented in Chapter 9) included the ability to submit photos with microreports, providing additional context to information when needed.

10.6. Comparison Study

To determine whether the target population of remote EM practitioners accepted the *Instantiation* artefact over existing techniques for report writing, a comparison study was conducted. A scenario-based comparison was selected based on the common use of scenario-based training in the target population, as recommended by the EM contacts that were approached. Based on the

improvements made to the framework that were based on requirements from EM practitioners, it was hypothesised that participants would prefer the instantiation over current techniques for the scenario presented in this study.

10.6.1. Recruitment methods

Similar to the requirements gathering survey reported in Chapter 4, participants were recruited through contacts in local governments and councils from remote North Queensland. These contact were emailed with an information sheet about the study and an invitation to participate in an online survey hosted on Google Forms. The survey contained a link to the information sheet, and reiterated to participants that no personally identifying information would be recorded and that participation was voluntary, such that participation could be revoked at any time by closing the survey. At the end of the survey, participants were invited to share the link with other EM practitioners who might be interested in completing the survey.

10.6.2. Participants

Seventeen EM practitioners participated in the comparison survey. The roles of the participants varied in level of expertise and field of EM experience, including executive officers, field officers, social support, engineers, utility workers, police, fire fighters, and marine rescue. The average EM work experience in this group of participants was 16 years. Experience was roughly even across the four phases of EM, with a slightly larger percentage of respondents having experience in emergency response and recovery. The most common type of disaster that the participants had experience in dealing with was meteorological (94% of respondents), followed by accidents (41% of respondents) and geological emergencies (29% of respondents).

10.6.3. Comparison Task

After answering basic demographics questions and providing information about EM experience, participants were presented with the scenario presented below:

For the purposes of this questionnaire, assume that you are a coordinator of a local disaster management group during a cyclone event. You and your team are required to compile a situation report (sitrep) for dissemination. To complete the report, you and your team will gather information on the state of local assets and record this information in a single report. Your team is distributed between the local area and the state capital city. Some team members are in the field providing status updates via mobile devices and radio.

Participants were presented with a series of writing software features and asked to vote which software they would prefer for report writing in the presented scenario. The two versions of software that were presented included the research artefact developed in this study, as well as Microsoft Word which was identified in the requirements gathering survey as the most commonly

used software within the EM community to generate sitreps. Participants were blind to which software was being compared and were only asked to specify a preference of either *Software A* or *Software B*. Both software instances were de-identified (e.g. the interfaces were adjusted to focus on the specific features) and counterbalanced. The features that were compared are presented in Table 10.2.

Table 10.2: Software features compared in the evaluation survey.

<i>Feature</i>	<i>Software A (artefact)</i>	<i>Software B (Microsoft Word)</i>
<i>Collaboration</i>	Software A can have multiple users add information to the document at the same time, regardless of their location or device. Users access the document from a web browser and can add information and see changes in real-time.	Software B can have a single user work on the document and share the document with another user when they are finished. Information from other users must be manually added into the document.
<i>User attribution</i>	Software A displays a visual representation of author contributions using colour and usernames displayed in tooltips. This information can be toggled on or off.	Software B does not automatically identify who added information to the document.
<i>Locking</i>	Software A allows multiple users to work on the same document at the same time. To avoid conflicting edits, users can choose to lock sections of the document to prevent other users from editing that section for the duration of the lock. Locked information becomes read-only to other users while locked.	Software B allows only a single user to work on the entire document at one time so flexible locking is not supported. Other users cannot view the content of the document while it is being edited, and it can only be shared after editing is complete.
<i>Content Review</i>	Software A automatically saves versions of the document. Users can view how the document has changed over time using a slider and easily see who made each change.	Software B allows users to view changes to the document using "track changes", where changes are formatted with highlighting to show new text and strikethrough to show removed text.

<i>Mobile Support</i>	Software A allows users to submit information to the report using a mobile app. Administrators working in a control centre can review this information. Alternatively, mobile users can be requested to provide updates and verify reports by control centre staff.	Software B does not provide additional support for mobile users.
<i>Feedback</i>	Software A has a group chat tool to support informal group discussion. Users can send specific questions regarding sections of the document to another user's mobile.	Software B allows document text to be annotated with user comments.
<i>Sharing</i>	Software A can be shared simultaneously with multiple collaborators as an active, editable document, and optionally shared with other stakeholders as a read-only active document or read-only file.	Software B can be accessed by only one user at a time as an editable file, but can be shared with other stakeholders as a read-only file.
<i>Automatic summarisation</i>	Software A has an automatic summarisation feature, where sections of text can be summarized to give a brief overview of the key topics in the document.	Software B does not support automatic summarisation.
<i>Formatting</i>	Software A Users can write text in a simple format called Markdown, which is automatically styled using predefined style sheets.	Software B has rich formatting features, allowing multiple fonts and styles to be changed in the document while the content is being written.

10.6.4. Results

Figure 10.7 displays the results of the software comparison section of the survey that was completed by the EM practitioners. To reiterate, *Software A* represented the research artefact developed in this research, whereas *Software B* represented Microsoft Word. The highest feature preferences toward the research artefact were automatic summarisation, mobile support, user attribution, and sharing, with 90% of participants supporting this preference. Other features such as the informal chat, history of edits, locking, and real-time collaboration received over 80% preference for the research artefact. Interestingly, only 60% of participants preferred the Markdown formatting style of the research artefact over the more flexible options provided by Microsoft Word. Overall, the results indicated a higher preference for the features supported by the research artefact than Microsoft Word in all categories.

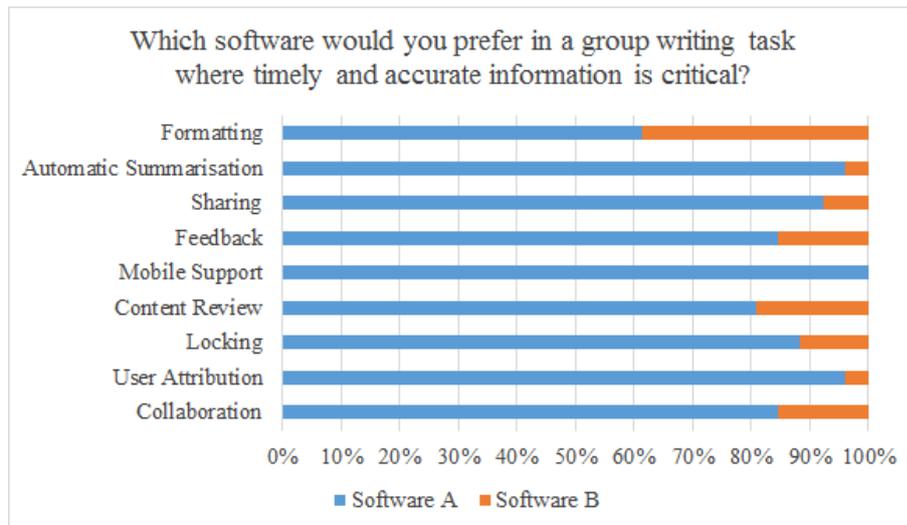


Figure 10.7: Results of the software comparison section of the questionnaire.

The results of the comparison study indicate that the EM practitioners preferred the feature support provided by the artefact when compared to asynchronous Microsoft Word in the scope of a flood scenario. This result supports the hypothesis of the study and provides evidence that the artefact will meet the needs of remote EM practitioners for collaborative report writing.

10.7. Perceived Usability and Ease of Use (PUEU)

The PUEU study involved a qualitative survey that was filled out by participants. This survey was adapted from the Perceived Usefulness Perceive Ease of Use (PUEU) questionnaire [62].

10.7.1. Recruitment methods and participants

The recruitment method used in this study was the same as described in Section 10.6.1. In addition to the EM practitioners, a second group of novice participants was recruited from a convenience sample. These participants were considered novice users such that they did not have prior experience working in EM or conducting report writing tasks in the field of EM. According to previous usability research [150], evaluations should consider the perspectives of both expert and novice users to ensure that usability is not simply a product of expertise in the considered field. In the EM group there were 17 participants (average EM experience 16 years) whereas the novice group included 10 participants (EM experience less than 1 year).

10.7.2. PUEU method and results

Participants were provided a complete overview of the features supported by our research artefact, referred to in this study as *Software A*. These features included real-time collaboration, user attribution, locking, content review (also known as history of edits), mobile support, real-time chat and feedback, sharing of read-only reports, automated summarisation, and automatic formatting via markdown. Participants were asked to complete the Perceived Usefulness and

Perceived Ease of Use (PUEU) questionnaire, as devised by Davis [62]. The data obtained from the PUEU questionnaire consists of ratings on a scale of 1 - 7 (unlikely to likely) in terms of how useful a system is perceived to be for completing a task, and other factors such as ease of learning. The second set of data collected was open-ended opinions on what participants perceived to be the most positive and negative aspects of the proposed artefact.

Figure 10.8 displays each question in the PUEU questionnaire, alongside the results from the evaluation in terms of mean and standard deviation for each question. On average, participants rated all twelve questions a score of 5 or higher as shown in Figure 10.8. This indicates that the users perceived the system to be easy to use and did not foresee any major issues with its use for EM collaborative report writing.

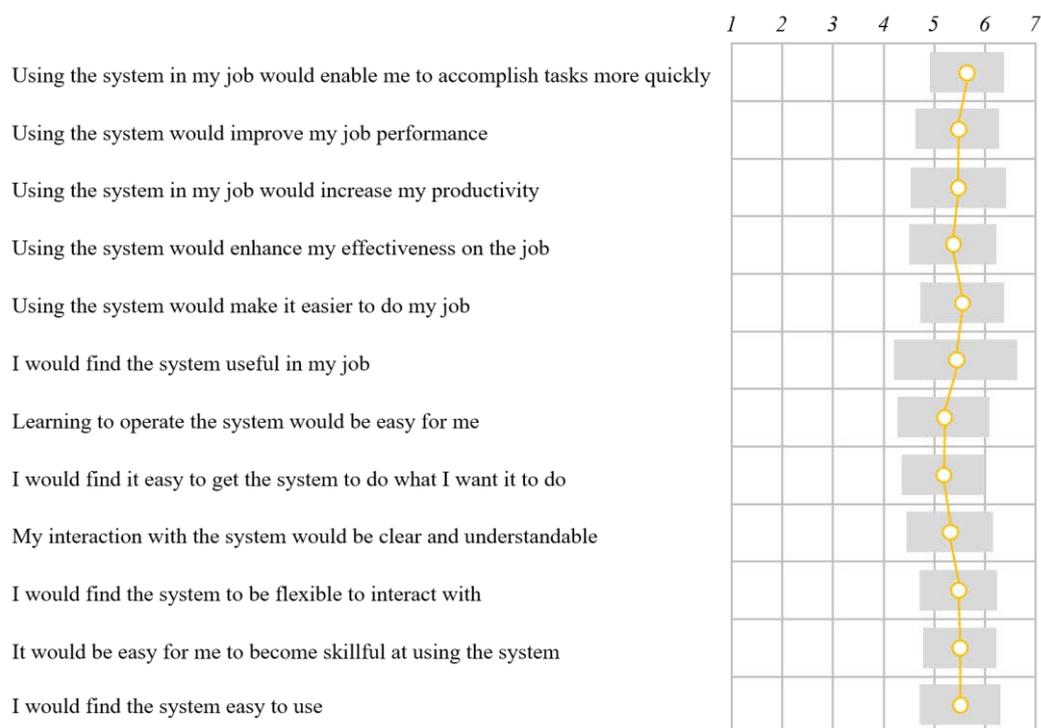


Figure 10.8: Results of the PUEU section of the questionnaire.

The second type of information gained from this section of the questionnaire was the opinions on perceived positive and negative features of the research artefact. Table 10.3 displays a list of both the positive and negative aspects of the artefact as listed by survey participants. Beside each feature is the frequency of participants that indicated a similar response.

Table 10.3: Positive and negative aspects listed by participants with frequency of response.

	<i>Frequency</i>	<i>% of Sample</i>
<i>Positive aspects</i>		
Real-time synchronisation	18	66.6%
User attribution	7	25.9%
Sharing	7	25.9%
Mobile support	7	25.9%
Chat	7	25.9%
Faster collaboration	4	14.8%
Locking	3	11.1%
Automatic summarisation	3	11.1%
Formatting	2	7.4%
Automatic saving	1	3.7%
Reliability	1	3.7%
<i>Negative aspects</i>		
None	8	29.6%
Formatting	4	14.8%
Authenticity	3	11.1%
Rollout costs	3	11.1%
Complexity	2	7.4%
System security	1	3.7%

Locking	1	3.7%
Information integrity	1	3.7%
Automatic summarisation	1	3.7%

Participants rated the real-time synchronisation of the artefact to be the most positive aspect, followed by user attribution, sharing and mobile support. This indicates that the participants foresee a benefit in having shared reports automatically synchronised with other stakeholders. In terms of negative aspects, the majority of participants did not list any. However, of the negative aspects that were listed, there were concerns about formatting limitations, the potential authenticity of data and security of the system. Finally, some participants reported concerns with the associated costs of rolling out new software in terms of financial, infrastructure, and training costs. Overall, participants were satisfied with the actual features supported by the artefact.

10.8. Enhanced Cognitive Walkthrough (ECW)

The enhanced cognitive walkthrough (ECW) [151] is a method used for evaluating the usability of a system. ECW uses a detailed procedure to simulate a user's problem-solving process during each step of interaction with the interface of the system. The ECW method is run by an evaluator (who can be a designer, software developer, presumptive user, and so on) who is either an intended user of the system or has knowledge about the use of and intended users of the system. In this study, the researchers conducted the evaluation in correspondence with the most experienced EM practitioner (taken from the pool of participants from the requirements survey presented in Chapter 4). ECW is comprised of three major steps: preparation, analysis, and presentation of the results in matrices. The first step is to identify the intended use and intended users of the system [151].

10.8.1. Intended use and users of system

(1) Analysed Artefact. The prototype built on the framework presented in this chapter is used to support collaborative report writing during an emergency (described in Section 10.5). The system provides users with a network-supported, centralised interface to write the report, despite being physically separated (whether it is within the same office or across different regions). Additionally, it has a framework for multisynchronous interaction from practitioners in the field using micro-tasks. Other features include flexible locking, user attribution, chat, map, automatic keyword and document summary, and mobile features such as multisynchronous microtasks and crowdsourcing.

(2) *Intended Use*. Report writing is an important task conducted during an emergency. All relevant stakeholders connect to the centralised interface through a web browser and compile the report. This ECW deals with the use of the system for writing a report during an emergency.

(3) *Intended Users*. The EM practitioners that contribute to report writing vary in role and experience (see the survey results presented in Chapter 4). However, they share the same goal of ensuring that timely and accurate information about the current situation of the emergency is reflected in the report, so that it can be disseminated to other relevant stakeholders in a timely manner (e.g. the public, support agencies, and so on).

10.8.2. Selection and grading of tasks for evaluation

To select the tasks for evaluation, the responses from the survey of EM practitioners were utilised based on the results reported in Chapter 4. Specifically, the respondents' suggestions of useful features for collaborative report writing systems formed the list of tasks and the grade of these tasks were assigned based on the number of participants who identified the feature as essential for a collaborative report writing system. As a further step, this list of features was reduced by selecting only those tasks that are relevant to text synchronisation, as is the scope of this thesis. The final list of tasks is presented in Table 10.4.

Table 10.4: Grading of key tasks in the collaborative EM report writing prototype (1 = most important, 5 = least important).

#	Task	Grade
1	Lock a section of text (single-user)	1
2	Unlock a section of text that was previously locked	1
3	View history of edits made to the report	1
4	Share dynamic, read-only report to stakeholders	1
5	View automated summary of report	1
6	Request information from a mobile user	2
7	Lock a section of text (group)	3

10.8.3. Specification of the tasks

The next step in the ECW is to specify the exact steps required by the user to complete these tasks within the system. This is achieved using a Hierarchical Task Analysis (HTA), with the

bottom level of the tree indicating interactions between the user and the interface [151]. The HTA for the first task is displayed in Figure 10.9. The HTAs for the remaining tasks are listed in Appendix C.

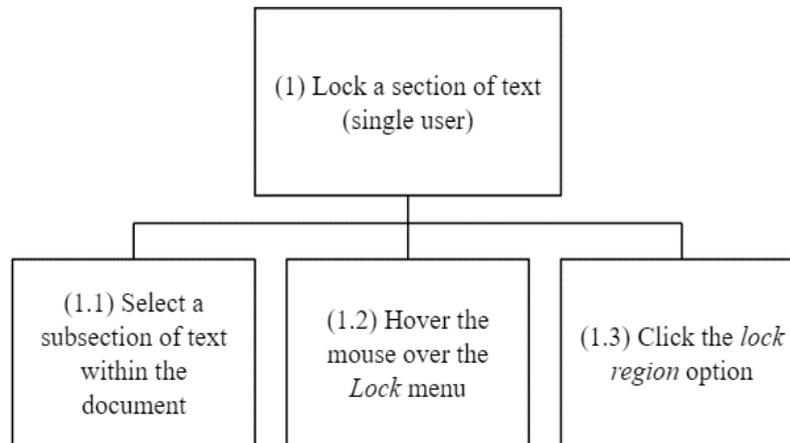
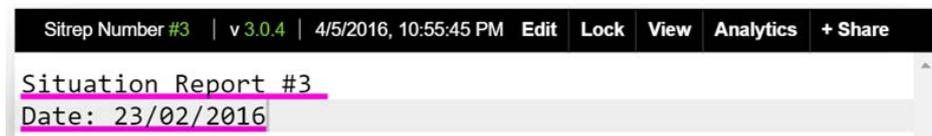


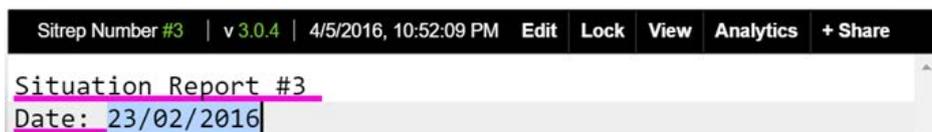
Figure 10.9. HTA for single-user locking task.

10.8.4. Specification of the user interface

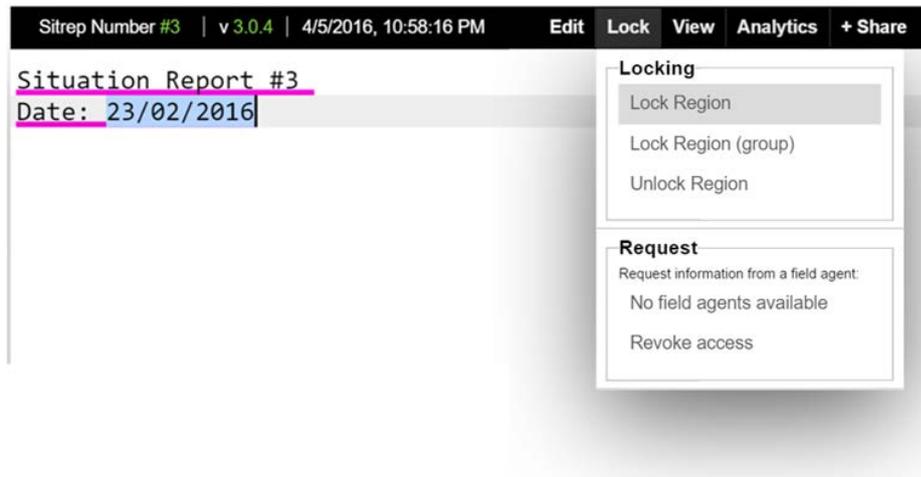
The next step in the ECW is to outline the specifications of the user interfaces for each task. The HTA diagrams developed previously describe the correct order of operations required to perform a task within the system. The specification of the user interface demonstrates how the interface appears for each different operation, including both the user’s actions and the feedback presented by the interface to those actions. The user interface specification for the single-user locking task is displayed in Figure 10.10. The user interface specifications for the remaining tasks are listed in Appendix D.



(a) Task/function 1.0: Lock a section of text (single-user).



(b) User action to operation 1.1: Select a subsection of text within the document.



(c) User action to operation 1.2: Hover the mouse over the *Lock* menu.



(d) User action to operation 1.3: Click the *Lock Region* option

Figure 10.10. Specification of the user interface for single user locking task.

10.8.5. Specification of the user and use

Specification of the user and use is the last step involved in preparation for the ECW. This step aims to define the user's knowledge and experience of the artefact and its use in terms of its physical, organisational, and psychosocial context [151]. In the context of report writing for EM, the expected users of this artefact do not possess explicit knowledge of this system as it is newly developed in this research. However, its design was based on responses provided by expected users in a survey, according to current use of report writing software in organisations and considering the benefits and limitations of these systems.

It is expected that users of this system will be familiar with report writing protocols for EM in their field or organisation. As such, the use of the artefact will aim to support these existing protocols and provide a collaborative workspace to compile information and write a report. Users are expected to connect to the collaborative report through a web browser at an EM control centre or the premises of their organisation. The users will only need to access this artefact during an emergency, which may include the mitigation, preparation, response and recovery phases of disaster management depending on the type of disaster and the protocols of the organisation.

10.8.6. Analysis

With the preparation complete, the next step is to conduct the analysis of the functions and operations identified in the preparation stage of the ECW. Analysis involves working through a series of questions for each defined task, determining potential problems that may occur during this task, grading the severity of these problems, and categorising the problems. There are two levels of questions that are used in analysis: the first level considers the ability of nodes from the HTA (e.g. the top level of the tree) in capturing the user's attention; the second level considers the ability of the steps in the HTA (e.g. the bottom level of the tree) to lead the user to perform the correct function. The questions used for both functional and operational evaluation are displayed in Tables 10.5 and 10.6 alongside the results of the analyses.

The answer to each question is graded with a number between 1 and 5 alongside a justification known as a *failure/success story*. This grade is referred to as the Problem Seriousness (PS). A grade of 5 indicates that the user has a very good chance of success in understanding the function or operation within the system, whereas a grade of 1 indicates a very small chance of success. A grade of 3 indicates that it is impossible to know the outcome, for example the outcome may depend on external knowledge that is not contained within the system, for example if a user reads the manual for the system prior to its use. Finally, if a problem is scored between 1 and 4 it indicates that a potential usability problem exists. These problems are categorised according to Problem Type (PT), for example: User (U) if the problem is due to the user's experience or knowledge; Hidden (H) if the interface gives no indication that a function is available or instructions on how to use that function; Text and icon (T) if the representation of a function within the system is confusing; Sequence (S) if the order in which subtasks must be completed is suboptimal; Physical demands (P) if the interface requires too much in terms of physical speed, motor skills, or force; and Feedback (F) if the system gives unclear information about the outcomes of any actions made within the system [151].

The ECW function analysis for the single-user locking task is presented in Table 10.5 and the operational analysis in Table 10.6. The analyses for the remaining tasks are listed in Appendix E.

Table 10.5: Analysis of functions for the single-user locking task.

Lock a section of text (single-user)		Lock a section of text (single-user)	
<i>1.0</i>	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS PT</i>
(1)	<p><i>Will the user know that the evaluated function is available?</i></p> <p>Do not know. It depends on whether the user has the expectation of being able to execute this function in the system.</p>	User does not expect functionality	3 U
(2)	<p><i>Will the user be able to notice that the function is available?</i></p> <p>Yes, there is a menu and buttons to indicate locking functionality.</p>	No usability problem	5 -
(3)	<p><i>Will the user associate the clues with the function?</i></p> <p>Yes, probably. There is a menu titled "Lock" within the text editing area.</p>	Unclear text	4 T
(4)	<p><i>Will the user get sufficient feedback when using the function?</i></p> <p>No, uncertain. There are no clear instructions that a user needs to select text prior to using the locking function.</p>	Instructions are not displayed	2 H
(5)	<p><i>Will the user get sufficient feedback to understand that the function has been fully performed?</i></p> <p>Yes, probably, the locked text will change colour to indicate that a lock has been activated.</p>	Feedback is not sufficient	4 F

Table 10.6: Analysis of operations for the single-user locking task.

Select a subsection of text within the document		Select a subsection of text within the document	
<i>1.1</i>	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS PT</i>
(1)	<p><i>Will the user try to achieve the right goals of the operation?</i></p> <p>No, unlikely. It depends if the user expects that the sequence involves selecting text first or using the lock menu first.</p>	Unclear indication of correct action	2 S

(2) <i>Will the user be able to notice that the action of the operation is available?</i> Yes, probably, as selecting text within a document is a standard feature of similar applications.	User does not relate the action of selecting text with the locking outcome	4	H
(3) <i>Will the user associate the action of the operation with the right goal of the operation?</i> Do not know, it depends if the user expects to select text before selecting the lock option or after.	User does not expect the sequence of events required by the system	3	S
(4) <i>Will the user be able to perform the correct action?</i> Yes, there are no restrictions in the text input area that would stop a user from selecting text.	No usability problem	5	-
(5) <i>Will the user get sufficient feedback to understand that the action is performed and the goal is achieved?</i> Yes, like similar applications the selected text will be highlighted when selected.	No usability problem	5	-
1.2 Hover the mouse over the Lock menu <i>Failure/success story</i>	Hover the mouse over the Lock menu <i>Usability problem</i>		<i>PS PT</i>
(1) Yes, the user has the goal of locking some text and will see the Lock menu and associate it with this task.	No usability problem	5	-
(2) Yes, the menu is clearly displayed.	No usability problem	5	-
(3) Yes, the user will associate hovering the mouse over the lock menu with displaying options for locking	No usability problem	5	-
(4) Yes, there are no restrictions on the user moving the mouse cursor to the menu	No usability problem	5	-
(5) Yes, when the menu title is hovered over a submenu will immediately appear	No usability problem	5	-
1.3 Click the lock region option <i>Failure success story</i>	Click the lock region option <i>Usability problem</i>		<i>PS PT</i>
(1) Yes, the user wants to lock a region so will click the lock region option	No usability problem	5	-

(2)	Yes, the operation is clearly displayed in the menu	No usability problem	5	-
(3)	Yes, probably, as the user intends to lock a region of text and that is what the menu item conveys	Unclear instructions	4	T
(4)	Yes, the user can easily click the option with the mouse	No usability problem	5	-
(5)	Yes, probably, the selected text will change to be highlighted with the user's colour to indicate that the lock is now active	Feedback is not sufficient	4	F

10.8.7. Compilation of results in matrices

The final step in the ECW after preparation and analysis is to present the results in matrices. Several matrices can be used to display the data gathered via the ECW analysis [151]. To display the results of this ECW, two matrices were selected: *Problem Seriousness versus Task Number* to reveal how serious the problems are that occur in each task (see Figure 10.11), and *Problem Type versus Task Number* to reveal the most severe problem types that exist for each category (see Figure 10.12).

Problem seriousness

<i>Task Number</i>	1	2	3	4
1	0	2	2	5
2	0	0	2	5
3	0	0	1	9
4	0	0	5	4
5	0	0	2	6
6	1	2	4	6
7	0	2	2	5
Total	1	6	8	0

Figure 10.11. Problem seriousness versus task number.

Problem type

<i>Task Number</i>	<i>U</i>	<i>H</i>	<i>S</i>	<i>T</i>	<i>F</i>
1	1	2	2	2	2
2	1	0	1	2	3
3	2	0	0	7	1
4	3	0	1	3	2
5	1	0	0	7	0
6	1	5	2	2	3
7	1	2	2	2	2
Total	10	9	8	25	13

Figure 10.12. Problem type versus task number.

Based on the results of the conducted ECW as presented in the matrices, there are several implications for the artefact going forward. Firstly, it is a positive sign that the majority of the problems were given a score of 4, as this indicates that these are not major problems. However, it is clear that there are several aspects of the interface that require improvement. The main problems in the interface are related to *Text and Icon* and *Feedback*, suggesting that while the functionality of the system exists to conduct important tasks, there are design aspects that make it difficult to achieve these tasks. The tasks with the biggest issues in this scope are *View history of edits*, *View automated summary of report*, and *Request information from mobile user*.

Within the scope of EM, it is not ideal to have important functions hidden or misrepresented by interface design. For example, the steps involved in achieving the *Request information from mobile user* goal are presented in a confusing sequence. In the evaluated artefact, this design was because the multisynchronous framework was an extension of the locking functionality so the feature was added to the Lock menu. However, this is clearly not a logical assumption that can be made by an EM practitioner. Such problems could be remedied by better organisation of the

functions within the menus, better labelling, more instructions, and ensuring that adequate feedback is provided after each operation is performed in the system.

Additionally, the problems labelled as *User* type problems in this system related primarily to the fact that an end user may not expect some of the functionality to be present in the system. This is because several of these functions are not commonly included in the report writing solutions currently used by EM practitioners. Such problems could be remedied by providing adequate training for end users, as well as detailed user manuals and help features within the system itself.

10.9. Summary

In this chapter, the results of several evaluation techniques were presented. The purpose of these evaluation techniques was to determine the utility and applicability of the designed artefact for supporting EM practitioners in collaborative report writing. Six evaluation techniques were utilised, including benchmarking, static analysis based on Groupware Heuristic Analysis, development of a prototype, comparison study, PUEU, and ECW. Table 10.7 presents the hypotheses for each technique and the extent to which each hypothesis was supported. It was concluded that all of the hypotheses were supported, except for H7 and H11, which were only partially supported due to the need for improvements in groupware functionality and the usability issues uncovered by the ECW.

Table 10.7: Evaluation results based on supported hypotheses.

Evaluation Technique	Hypothesis	Supported
<i>Benchmarking</i>	H1, H2, H3, H4, H5, H6	Yes, benchmarking results supported these hypotheses
<i>Static Analysis (Groupware Heuristic Evaluation)</i>	H7	Partially, the static analysis identified several key areas for improvement
<i>Prototype</i>	H8	Yes, the prototype developed supports the requirements of the EM practitioners surveyed in Chapter 2
<i>Comparison Study</i>	H9	Yes, the EM practitioners rated a higher preference for the artefact over Microsoft Word
<i>PUEU</i>	H10	Yes, the participants rated the artefact positively on the PUEU, and feedback was positive

<i>ECW</i>	H11	Partially, the majority of the usability issues identified are rated 3 and 4, but there were was one issue rated 1 and 6 issues rated 2 that need addressing in future iterations
------------	------------	---

The results of these evaluation techniques indicate that there are both benefits and limitations of the artefact for supporting EM practitioners in collaborative report writing. Firstly, the results of the benchmarking indicate that the framework for locking and user attribution did not have significant performance impacts on the functionality of difsync in terms of scalability, efficiency, and correctness. Next, the results of the static analysis indicate that the developed artefact partially meets the requirements of groupware, though several improvements could be made in this area. For example, improvements can be made in supporting additional gestural techniques for collaboration and embodiment of the user’s activity to indicate how a user is interacting within the workspace (e.g. is the user active or inactive?). Another major area for improvement is to provide spaces for users to break into smaller groups or individual sessions within the larger system to better replicate the flow of real-world collaboration. Next, the development of the instantiation prototype and the comparison and PUEU indicate the utility and acceptance of the artefact amongst the target community of EM practitioners. Lastly, the ECW indicated that while the system includes the functionality that is required by EM practitioners, better organisation of the interface and more feedback is required to ensure that these features can be used effectively.

While there have been several evaluation methods used to test the individual frameworks (throughout the previous chapters of this thesis) and the prototype (in this chapter), there are still additional methods that could be employed in future studies to further improve the system. For example, case studies and field studies would provide direct observation into how such tools are used by real-world EM practitioners. In the current study, it was not feasible to conduct real-world field studies due to limited resources and difficulty in coordinating the members of regional and remote councils who participated in the online survey. Further, the unpredictable nature of disasters means it is difficult to replicate a real-world scenario. Finally, employing a new system immediately into the field would be premature in a high-risk field like disaster management. Future work is required to conduct more comprehensive user testing in the field.

11. Discussion and Conclusion

This chapter summarises the results of this thesis and outlines the main contributions of the research. Further, the limitations of the research are considered and recommendations for future research provided. Finally, the chapter provides a conclusion to the research presented in this thesis.

11.1. Summary of results

This project aimed to develop a framework to support remote EM practitioners in collaborative report writing. A survey of remote EM practitioners demonstrated that current collaboration techniques did not meet the requirements for collaborating in remote disasters. A list of requirements was derived, including real-time synchronisation features, locking features, user attribution features, and multisynchronous features. After reviewing text synchronisation techniques, difsync was selected as the system architecture. However, difsync required the implementation of locking, user attribution, multisynchronous editing and crowdsourcing to meet the user requirements. The framework was evaluated using benchmarking, a groupware heuristic static analysis, and the development of a prototype system. The prototype was then subjected to further evaluation, including a comparison study with users against an existing system, a usability evaluation at the system-level using the PUEU and at the task-level using an enhanced cognitive walkthrough. Overall, the performance, robustness, usability, and acceptance of the system by the target EM community was demonstrated.

11.2. Research contributions and implications

The primary aim of this thesis was to define a conceptual framework to support remote EM practitioners in collaborative report writing. Table 11.1 presents the research contributions of this thesis. This includes a list of design requirements for EM collaborative technologies with a specific focus on remote North Queensland (NQ). Prior to this thesis, such a list of requirements did not exist in the literature as concluded in Chapter 2. Further, the list of requirements was used to guide the development of a framework to support collaborative EM report writing for EM practitioners. This framework resulted in the instantiation of a collaborative report writing system that met the design requirements and was accepted by the EM practitioners. The prototype can be further developed and used by the EM practitioners.

There were also several technological contributions developed in this thesis. Chapter 5 revealed that the difsync technique did not provide support for flexible locking and user attribution. New conceptual frameworks to support these features were demonstrated in this thesis and evaluated in terms of their performance. Using the flexible locking and user attribution frameworks, new features were developed including support for multisynchronous editing and crowdsourcing. Prior to this thesis difsync did not support any of these features.

Finally, the research methodology employed in this thesis is a contribution to the design science literature. As design science is an emerging research paradigm, it is important that more research is conducted that explores methods for requirements gathering, artefact development, and robust evaluation techniques. A challenge in employing user-centred design for this project was the remote location of users separated across a large geographical area. The use of an online requirements gathering survey ensured that the artefact met the needs of a diverse cohort of remote EM practitioners. Further, the remote nature of the EM practitioners made a field study difficult, though the multi-faceted evaluation framework ensured that the artefact was evaluated thoroughly against the needs of the target EM practitioners. This methodology will assist future design science researchers in conducting similar studies.

Table 11.1: Research contributions of this thesis.

Primary aim: To define a conceptual framework to support remote EM practitioners in collaborative report writing
1. A list of design requirements for EM collaborative technologies, as identified by a NQ EM practitioner user group;
2. A real-time collaborative report writing artefact (i.e. instantiation) for the EM domain that meets design requirements and is usable and acceptable to EM practitioners.
Technological contributions: To provide improvements on text synchronisation methods within the field of collaborative EM report writing.
3. A conceptual framework for implementing flexible locking into the diffsync technique;
4. A conceptual framework for implementing user attribution into the diffsync technique;
5. A toolkit for implementing multi-synchronous editing and crowdsourcing using diffsync, based on the flexible locking and user attribution features.
Design Science contributions
6. User-centred design requirement gathering framework and evaluation framework for artefact targeting a remote user group.

11.3. Generalisability of results

There are several implications of the research presented in this thesis that can be generalised to other domains. Firstly, the comprehensive list of design requirements for supporting remote EM practitioners in collaborative report writing could be generalised to help develop other EM systems. Researchers and EM organisations can use this list to improve their existing report-writing systems or to create new collaborative report-writing systems suitable for remote emergencies. While this thesis focused specifically on the task of collaborative report writing,

there may also be other groupware systems that could be developed based on this list of requirements, such as meeting and decision-support systems. Combined with the theoretical design requirements for EM systems as specified in Chapter 2, this will ensure that future EM systems are more robust to the needs of different EM communities.

Another implication of this research is the framework's support for emerging work paradigms such as multisynchronous editing and crowdsourcing. While the EM practitioners specified that it is common practice to engage in collaboration between control-centre staff and field officers, this collaboration was achieved primarily using phone and radio. The new framework formalises this practice and allows field officers to make contributions directly into a report, rather than relying on another user to relay the information to another author. This framework may also enable new work practices to begin, such as crowdsourcing information from the public and utilising the crowd to verify information accuracy. This would be invaluable in remote communities during an emergency due to the vast areas to be managed by limited skilled workers.

The resulting framework may also be generalised to other domains that face similar challenges in sharing timely and accurate information. While the current study focused primarily on North QLD, the nature of remote work and the need for collaboration amongst remote workers is applicable to other fields. For example, domains such as defence, mining exploration, or agriculture are areas that require time-critical collaboration between diverse user types. These jobs are often conducted in geographically remote areas but rely on accurate information being shared from experts (e.g. meteorologists). There are also similar challenges in network connectivity and maintaining information accuracy. Future researchers can utilise the framework developed in this thesis in similar fields to determine the benefits and make additional contributions to the framework.

While the EM practitioners involved in this survey were from remote North QLD, the challenges they face can be generalised to EM practitioners worldwide. There are many areas in the world that face similar types of disasters including drought, cyclones (or hurricanes), and bushfires and such disasters do not discriminate based on location or population. Though the example of remote North QLD may be considered extreme in its geographical isolation, lack of reliable network infrastructure, and limited emergency management resources, these challenges are not exclusive to remote North QLD so the solutions can contribute to emergency management in other regions.

Finally, the technical improvements to diffsync provide a more robust tool for text synchronisation that can be used in new research directions. Currently research in the field of text synchronisation is dominated by the OT approach. However, the benefits of diffsync including the ease of appending it to single-user applications means that it could be applied to existing systems to introduce collaboration. With additional support for flexible locking, user attribution,

multisynchronous editing and crowdsourcing means existing applications could gain a rich set of collaborative features with limited development overhead.

11.4. Limitations and recommendations for future work

While the research had many benefits, there were some limitations identified that future research can address. These limitations are described in this section in terms of impact on the current work and the steps needed to address the limitations in future research.

The first limitation that was identified was the lack of comprehensive field testing of the research artefact with the remote EM practitioners. This was due to the difficulty and logistics involved with having practitioners engage simultaneously for a collaborative exercise, which is a known challenge in groupware evaluation [146]. Further, the remote location of users posed additional logistical challenges in coordinating a field evaluation. Future work may attempt to use the framework presented to develop and deploy a remote EM collaborative report-writing system, which this work has evaluated comprehensively using lab-based techniques. It would also be worthwhile to investigate the scalability of the system in the field to determine the effects of synchronisation delay on user groups, as considered by Ignat et al. [152].

The next limitation that was identified is that the framework was developed to specifically meet the needs of a single target practitioner group from remote North QLD. It is likely that the needs of this emergency management community differ to the needs of emergency management practitioners in other environments. For example, the primary type of disasters facing these communities are meteorological (e.g. tropical cyclones and bushfires). In contrast, urban environments may use such a system to collaboratively report on the situation of large protests, traffic incidents, or even terrorist attacks. Future research can address this by investigating the needs of different emergency management communities to determine whether the proposed techniques are suitable and whether changes are needed to support different types of emergencies.

Another limitation that was identified is the lack of support for additional groupware features, including support for user permissions, visualisation of data, and support for fine-grained collaboration (e.g. small groups). These areas were not explored because the current work focused specifically on the task of collaborative report writing. Future research should explore additional features to support other aspects of collaboration in the remote EM community. For example, flexible locking could be extended to include user permissions and automatic editing policies based on the expertise of each collaborator in terms of emergency management. Further, there are several additional techniques that could be implemented into the user attribution framework, including support for *undo* and *redo*. Finally, there are several additional improvements that could be investigated to provide a more flexible collaboration experience, including better visualisation of each user within the workspace to indicate gestural intention, and allowing users to move between individual and small group collaboration within the larger

interface. Future work will also investigate further theoretical analysis of the locking and user attribution techniques developed to determine the trustworthiness of these protocols.

Finally, the research presented in this thesis focused primarily on report writing for emergency management. However, there may be other areas that have similar situations that could benefit from the use of a synchronised system for managing information. For example, the current system focused primarily on emergency response, but it could also be used for emergency planning in the sense that diverse stakeholders could converge information in a central location to determine the needs of a community. Beyond emergency management there may also be other fields that can benefit from the techniques proposed in this thesis. Future research could determine what these fields are and investigate how the proposed techniques may benefit such areas.

11.5. Conclusion

In conclusion, this thesis presented a framework to meet the requirements of remote EM practitioners for collaborative report writing. This research primarily followed the design science approach to determine requirements, develop a solution, and evaluate it. Firstly, the needs of the EM community were established through a survey to derive design principles for a collaborative report writing system. Next, the diffsync technique for synchronisation of text was further developed to introduce support for flexible locking, user attribution, multi-synchronous editing and crowdsourcing to support the features requested by the EM community. Finally, these features were implemented into a research artefact and evaluated through several techniques including benchmarking, static analysis on groupware heuristics, prototype development, comparison study, technology acceptance survey, and enhanced cognitive walkthrough. Based on these evaluations it was determined that the research artefact was able to support the needs of the EM community for collaborative report writing. Directions for future research were also suggested to allow researchers to continue this work.

References

- [1] D. Yates and S. Paquette, "Emergency knowledge management and social media technologies: A case study of the 2010 Haitian earthquake," *International Journal of Information Management*, vol. 31, no. 1, pp. 6–13, Feb. 2011.
- [2] T. Ludwig, C. Reuter, and V. Pipek, "What You See is What I Need: Mobile Reporting Practices in Emergencies," in *Proceedings of the 13th European Conference on Computer Supported Cooperative Work (ECSCW)*, 2013, pp. 181–206.
- [3] Emergency Management Australia, "Multi-Agency Incident Management," 2015. [Online]. Available: <https://www.em.gov.au/Documents/Manual17-Multi-AgencyIncidentManagement.pdf> [Accessed: 04-Jun-2015].
- [4] P. B. Lowry, A. Curtis, and M. R. Lowry, "Building a Taxonomy and Nomenclature of Collaborative Writing to Improve Interdisciplinary Research and Practice," *Journal of Business Communication*, vol. 41, no. 1, pp. 66–99, Jan. 2004.
- [5] M. A. Cameron, R. Power, B. Robinson, and J. Yin, "Emergency situation awareness from twitter for crisis management," in *Proceedings of the 21st international conference companion on World Wide Web*, 2012, pp. 695–698.
- [6] Á. Monares, S. F. Ochoa, J. A. Pino, V. Herskovic, J. Rodriguez-Covili, and A. Neyem, "Mobile computing in urban emergency situations: Improving the support to firefighters in the field," *Expert Systems with Applications*, vol. 38, no. 2, pp. 1255–1267, Feb. 2011.
- [7] Google Inc., "Google Drive," 2015. [Online]. Available: <https://drive.google.com/>. [Accessed: 01-Jan-2014].
- [8] P. Bhattacharya, M. Guo, L. Tao, Y. Fu, and K. Qian, "A Collaborative Interactive Cyber-learning Platform for Anywhere Anytime Java Programming Learning," in *11th IEEE International Conference on Advanced Learning Technologies (ICALT)*, 2011, pp. 14–16.
- [9] H. Natsu, J. Favela, A. L. Morán, D. Decouchant, and A. M. Martinez-Enriquez, "Distributed pair programming on the Web," in *Proceedings of the Fourth Mexican International Conference on Computer Science*, 2003, pp. 81–88.
- [10] M. Heinrich, "Enriching Web Applications Efficiently with Real-Time Collaboration Capabilities," PhD, Chemnitz University of Technology, 2014.
- [11] I. Koren, A. Guth, and R. Klamma, "Shared editing on the web: A classification of developer support libraries," in *9th International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom)*, 2013, pp. 468–477.
- [12] A. Shatte, J. Holdsworth, and I. Lee, "The impact of dynamic locking on collaborative programming," in *Proceedings of the 25th Australasian Conference on Information Systems*, 2014.
- [13] X. Xu, J. Bu, C. Chen, and Y. Li, "Distributed Dynamic-Locking in Real-Time Collaborative Editing Systems," in *Groupware: Design, Implementation, and Use*, Springer Berlin Heidelberg, 2004, pp. 271–279.
- [14] H. Shen and C. Sun, "Improving real-time collaboration with highlighting," *Future Generation Computer Systems*, vol. 20, no. 4, pp. 605–625, May 2004.
- [15] C. Rahhal, H. Skaf-Molli, P. Molli, and S. Weiss, "Multi-synchronous Collaborative Semantic Wikis," in *Web Information Systems Engineering - WISE 2009.*, Springer Berlin Heidelberg, 2009, pp. 115–129.

- [16] M. Ahmed-Nacer, P. Urso, V. Balegas, and N. Pregoica, “Concurrency control and awareness support for multi-synchronous collaborative editing,” in *9th International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom)*, 2013, pp. 148–157.
- [17] J. Teevan, S. Iqbal, and C. von Veh, “Supporting Collaborative Writing with Microtasks,” in *Proceedings of CHI*, 2016.
- [18] Y. Qin, J. Liu, C. Wu, and Y. Shi, “uEmergency: A Collaborative System for Emergency Management on Very Large Tabletop,” in *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces*, 2012, pp. 399–402.
- [19] J. Li, Q. Li, C. Liu, S. Ullah Khan, and N. Ghani, “Community-based collaborative information system for emergency management,” *Computers & Operations Research*, vol. 42, no. 0, pp. 116–124, Feb. 2014.
- [20] CSIRO, “ERIC: Improving disaster response effectiveness,” 20-Mar-2015. [Online]. Available: <http://www.csiro.au/en/Research/DPF/Areas/The-digital-economy/Disaster-management/ERIC>. [Accessed: 04-Jun-2015].
- [21] CSIRO, “ESA: Software for Emergency Situation Awareness,” 20-Mar-2015. [Online]. Available: <http://www.csiro.au/en/Research/DPF/Areas/The-digital-economy/Disaster-management/ESA>. [Accessed: 04-Jun-2015].
- [22] D. A. Menascé and T. Nakanishi, “Optimistic versus pessimistic concurrency control mechanisms in database management systems,” *Information Systems*, vol. 7, no. 1, pp. 13–27, 1982.
- [23] D. Sun, S. Xia, C. Sun, and D. Chen, “Operational transformation for collaborative word processing,” in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 2004, pp. 437–446.
- [24] N. Fraser, “Differential synchronization,” in *Proceedings of the 9th ACM Symposium on Document Engineering*, 2009, pp. 13–20.
- [25] C. Wagner, “Wiki: A technology for conversational knowledge management and group collaboration,” *The Communications of the Association for Information Systems*, vol. 13, no. 1, p. 58, 2004.
- [26] C. Sun, “Optional and responsive fine-grain locking in Internet-based collaborative systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 9, pp. 994–1008, Sep. 2002.
- [27] J. Birnholtz and S. Ibara, “Tracking Changes in Collaborative Writing: Edits, Visibility and Group Maintenance,” in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, 2012, pp. 809–818.
- [28] D. Sun and C. Sun, “Context-Based Operational Transformation in Distributed Collaborative Editing Systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 10, pp. 1454–1470, Oct. 2009.
- [29] D. Sun and C. Sun, “Operation context and context-based operational transformation,” in *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW)*, 2006, pp. 279–288.
- [30] A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design science in information systems research,” *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.
- [31] C. A. Ellis and S. J. Gibbs, “Concurrency Control in Groupware Systems,” *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, vol. 18, no. 2,

- pp. 399–407, Jun. 1989.
- [32] Cloud9 IDE Inc., “Cloud9IDE.” [Online]. Available: <https://c9.io/>. [Accessed: 12-Feb-2014].
- [33] C. Sun, X. Jia, Y. Zhang, Y. Yang, and D. Chen, “Achieving Convergence, Causality Preservation, and Intention Preservation in Real-time Cooperative Editing Systems,” *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 5, no. 1, pp. 63–108, Mar. 1998.
- [34] M. Laal and S. M. Ghodsi, “Benefits of collaborative learning,” *Procedia - Social and Behavioral Sciences*, vol. 31, pp. 486–490, 2012.
- [35] D. M. Woods and K.-C. Chen, “Evaluation Techniques For Cooperative Learning,” *International Journal of Management & Information Systems (IJMIS)*, vol. 14, no. 1, Jan. 2011.
- [36] M. Franceschet and A. Costantini, “The effect of scholar collaboration on impact and quality of academic papers,” *Journal of Informetrics*, vol. 4, no. 4, pp. 540–553, Oct. 2010.
- [37] J. T. Langton, T. J. Hickey, and R. Alterman, “Integrating Tools and Resources: A Case Study in Building Educational Groupware for Collaborative Programming,” *Journal of Computing Sciences in Colleges*, vol. 19, no. 5, pp. 140–153, May 2004.
- [38] Á. Santos, A. Gomes, and A. J. Mendes, “Integrating New Technologies and Existing Tools to Promote Programming Learning,” *Algorithms*, vol. 3, no. 2, pp. 183–196, Apr. 2010.
- [39] L.-T. Cheng, C. R. B. de Souza, S. Hupfer, J. Patterson, and S. Ross, “Building Collaboration into IDEs,” *Queueing Syst.*, vol. 1, no. 9, pp. 40–50, Dec. 2003.
- [40] J. T. Nosek, “The Case for Collaborative Programming,” *Communications of the ACM*, vol. 41, no. 3, pp. 105–108, Mar. 1998.
- [41] A. Cockburn and L. Williams, “The costs and benefits of pair programming,” *Extreme Programming Examined*, pp. 223–247, 2000.
- [42] J. Galegher, R. E. Kraut, and C. Egido, *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*. Taylor & Francis, 2014.
- [43] S. Salinger, C. Oezbek, K. Beecher, and J. Schenk, “Saros: an eclipse plug-in for distributed party programming,” in *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, 2010, pp. 48–55.
- [44] J. Tam and S. Greenberg, “A framework for asynchronous change awareness in collaborative documents and workspaces,” *International Journal of Human-Computer Studies*, vol. 64, no. 7, pp. 583–598, Jul. 2006.
- [45] W. L. Waugh and G. Streib, “Collaboration and Leadership for Effective Emergency Management,” *Public Administration Review*, vol. 66, pp. 131–140, Dec. 2006.
- [46] R. Chen, R. Sharman, H. Raghav Rao, and S. Upadhyaya, “Design principles for critical incident response systems,” *Information Systems and E-Business Management*, vol. 5, no. 3, pp. 201–227, Mar. 2007.
- [47] I. McLean, D. Oughton, S. Ellis, B. Wakelin, and C. Rubin, “Review of the Civil Defence Emergency Management Response to the 22 February Christchurch Earthquake.” 29-Jun-2012.
- [48] B. Teague, R. Mcleod, and S. Pascoe, “2009 Victorian Bushfires Royal Commission - Final Report Summary.” Jul-2010.

- [49] E. W. Stein, "Improvisation as Model for Real-Time Decision Making," in *Supporting Real Time Decision-Making*, Springer US, 2011, pp. 13–32.
- [50] D. Mendonça, "Decision support for improvisation in response to extreme events: Learning from the response to the 2001 World Trade Center attack," *Decision Support Systems*, vol. 43, no. 3, pp. 952–967, Apr. 2007.
- [51] S. R. Hiltz, P. Diaz, and G. Mark, "Introduction: Social Media and Collaborative Systems for Crisis Management," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 18, no. 4, pp. 18:1–18:6, Dec. 2011.
- [52] B. Van de Walle, M. Turoff, and S. R. Hiltz, *Information systems for emergency management*. ME Sharpe, 2009.
- [53] V. Hristidis, S.-C. Chen, T. Li, S. Luis, and Y. Deng, "Survey of data management and analysis in disaster situations," *Journal of Systems and Software*, vol. 83, no. 10, pp. 1701–1714, Oct. 2010.
- [54] Intermedix, "WebEOC," 2015. [Online]. Available: <https://www.intermedix.com/product/product-webeoc/index.php>. [Accessed: 04-Jun-2015].
- [55] G. Singh and D. Ableiter, "TwiddleNet: Smartphones as Personal Content Servers for First Responders," in *Mobile Response*, Springer Berlin Heidelberg, 2009, pp. 130–137.
- [56] E. G. Nilsson and K. Stølen, "Ad Hoc Networks and Mobile Devices in Emergency Response - A Perfect Match?," in *Ad Hoc Networks*, Springer, 2010, pp. 17–33.
- [57] N. Prat, I. Comyn-Wattiau, and J. Akoka, "Artifact Evaluation in Information Systems Design-Science Research-a Holistic View," in *Proceedings of the 18th Pacific Asia Conference on Information Systems*, 2014, p. 23.
- [58] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007.
- [59] M. Maguire, "Methods to support human-centred design," *International Journal of Human-Computer Studies*, vol. 55, no. 4, pp. 587–634, Oct. 2001.
- [60] J. Iivari and N. Iivari, "Varieties of user-centredness: An analysis of four systems development methods," *Information Systems Journal*, vol. 21, no. 2, pp. 125–153, 2011.
- [61] S. J. Miah, D. Kerr, and L. von Hellens, "A collective artefact design of decision support systems: design science research perspective," *Information Technology & People*, vol. 27, no. 3, pp. 259–279, 2014.
- [62] F. D. Davis, "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology," *MIS Quarterly*, vol. 13, no. 3, pp. 319–340, 1989.
- [63] C. Larman and R. Victor, "Iterative and incremental development: A brief history," *IEEE Computer Society*, vol. 36, no. 6, pp. 47–56, 2003.
- [64] Gulf Savannah Development, "Our Shires," *Gulf Savannah Development*, 2015. [Online]. Available: <http://www.gulf-savannah.com.au/information/our-shires>. [Accessed: 27-Jun-2016].
- [65] J. Gulliksen, B. Göransson, I. Boivie, S. Blomkvist, J. Persson, and Å. Cajander, "Key principles for user-centred systems design," *Behaviour and Information Technology*, vol. 22, no. 6, pp. 397–409, 2003.
- [66] C. Sun and R. Sosič, "Optimal locking integrated with operational transformation in

- distributed real-time group editors,” in *Proceedings of the 18th Annual ACM Symposium on Principles of Distributed Computing*, 1999, pp. 43–52.
- [67] P. Brosch, M. Seidl, K. Wieland, M. Wimmer, and P. Langer, “We can work it out: Collaborative Conflict Resolution in Model Versioning,” in *Proceedings of the Eleventh European Conference on Computer-Supported Cooperative Work*, 2009, pp. 207–214.
- [68] R. E. Kraut, S. R. Fussell, and J. Siegel, “Visual Information as a Conversational Resource in Collaborative Physical Tasks,” *Human–Computer Interaction*, vol. 18, no. 1–2, pp. 13–49, 2003.
- [69] K. P. N. Puttaswamy, C. C. Marshall, V. Ramasubramanian, P. Stuedi, D. B. Terry, and T. Wobber, “Docx2Go: collaborative editing of fidelity reduced documents on mobile devices,” in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, 2010, pp. 345–356.
- [70] N. Palmer, E. Miron, R. Kemp, T. Kielmann, and H. Bal, “Towards Collaborative Editing of Structured Data on Mobile Devices,” in *Proceedings of the 12th IEEE International Conference on Mobile Data Management (MDM)*, 2011, vol. 1, pp. 194–199.
- [71] C. Sun and C. Ellis, “Operational transformation in real-time group editors: issues, algorithms, and achievements,” in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 1998, pp. 59–68.
- [72] M. White, *The content management handbook*. Facet Publishing London, UK, 2005.
- [73] W. Cunningham, “What is Wiki,” 27-Jun-2002. [Online]. Available: <http://www.wiki.org/wiki.cgi?WhatIsWiki>. [Accessed: 07-Mar-2014].
- [74] TechTerms, “Wiki.” [Online]. Available: <http://www.techterms.com/definition/wiki>. [Accessed: 07-Mar-2014].
- [75] Wikimedia Foundation, Inc., “Wikipedia.” [Online]. Available: <http://en.wikipedia.org/>. [Accessed: 03-Mar-2014].
- [76] N. Cohen, “Wikipedia vs. the Small Screen,” *The New York Times*, 09-Feb-2014.
- [77] K. Smets, B. Goethals, and B. Verdonk, “Automatic vandalism detection in Wikipedia: Towards a machine learning approach,” in *Proceedings of the AAAI Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy*, 2008, pp. 43–48.
- [78] M. Potthast, B. Stein, and R. Gerling, “Automatic Vandalism Detection in Wikipedia,” in *Advances in Information Retrieval*, Springer Berlin Heidelberg, 2008, pp. 663–668.
- [79] G. Druck, G. Miklau, and A. McCallum, “Learning to Predict the Quality of Contributions to Wikipedia,” in *Proceedings of the AAAI Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy*, 2008, pp. 7–12.
- [80] A. K. Ashok, “Predictive Data Mining in a Collaborative Editing System: The Wikipedia Articles for Deletion Process,” PhD, Kansas State University, 2011.
- [81] T. Mens, “A state-of-the-art survey on software merging,” *IEEE Transactions on Software Engineering*, vol. 28, no. 5, pp. 449–462, May 2002.
- [82] J. W. Hunt and M. D. McIlroy, *An algorithm for differential file comparison*. Bell Laboratories, 1976.
- [83] J. W. Hunt and T. G. Szymanski, “A fast algorithm for computing longest common subsequences,” *Communications of the ACM*, vol. 20, no. 5, pp. 350–353, May 1977.
- [84] J. Cruz, “Branching Out with Git.” [Online]. Available:

<http://www.mactech.com/articles/mactech/Vol.26/26.04/BranchingOutWithGit/index.html>. [Accessed: 13-Feb-2014].

- [85] E. Lippe and N. van Oosterom, "Operation-based merging," in *Proceedings of the Fifth ACM Symposium on Software Development Environments (SIGSOFT)*, 1992, vol. 17, pp. 78–87.
- [86] C.-L. Ignat and M. C. Norrie, "Operation-based versus state-based merging in asynchronous graphical collaborative editing," in *Proceedings of the 6th International Workshop on Collaborative Editing Systems*, 2004.
- [87] B. Shao, D. Li, and N. Gu, "A Fast Operational Transformation Algorithm for Mobile and Asynchronous Collaboration," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 12, pp. 1707–1720, Dec. 2010.
- [88] D. A. Nichols, P. Curtis, M. Dixon, and J. Lamping, "High-latency, low-bandwidth windowing in the Jupiter collaboration system," in *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*, 1995, pp. 111–120.
- [89] C. Sun, "OT FAQ." [Online]. Available: <http://cooffice.ntu.edu.sg/otfaq/>. [Accessed: 31-Jan-2014].
- [90] F. Liu, S. Xia, H. Shen, and C. Sun, "CoMaya: incorporating advanced collaboration capabilities into 3d digital media design tools," in *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work (CSCW)*, 2008, pp. 5–8.
- [91] C. Sun, "Televiewpointer: an integrated workspace awareness widget for real-time collaborative 3d design systems," in *Proceedings of the 16th ACM International Conference on Supporting Group Work*, 2010, pp. 21–30.
- [92] C. Sun and D. Xu, "Operational transformation for dependency conflict resolution in real-time collaborative 3D design systems," in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW)*, 2012, pp. 1401–1410.
- [93] C. R. Palmer and G. V. Cormack, "Operation transforms for a distributed shared spreadsheet," in *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work (CSCW)*, 1998, pp. 69–78.
- [94] D. Sun, S. Xia, C. Sun, and D. Chen, "Operational transformation for collaborative word processing," in *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, 2004, pp. 437–446.
- [95] S. Xia, D. Sun, C. Sun, and D. Chen, "Collaborative object grouping in graphics editing systems," in *Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2005, pp. 10–19.
- [96] C. Sun and D. Chen, "Consistency maintenance in real-time collaborative graphics editing systems," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 9, no. 1, pp. 1–41, Mar. 2002.
- [97] B. Shao, D. Li, and N. Gu, "A sequence transformation algorithm for supporting cooperative work on mobile devices," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 2010, pp. 159–168.
- [98] M. Suleiman, M. Cart, and J. Ferrie, "Concurrent operations in a distributed and mobile collaborative environment," in *Proceedings of the 14th International Conference on Data Engineering*, 1998, pp. 36–45.
- [99] C. Sun, S. Xia, D. Sun, D. Chen, H. Shen, and W. Cai, "Transparent adaptation of single-user applications for multi-user real-time collaboration," *ACM Transactions on Computer-*

Human Interaction (TOCHI), vol. 13, no. 4, pp. 531–582, Dec. 2006.

- [100] D. Wang, A. Mah, and S. Lassen, “Google Wave Operational Transformation,” Google, 1.1, Jul. 2010.
- [101] A. Prakash and M. J. Knister, “A framework for undoing actions in collaborative systems,” *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 1, no. 4, pp. 295–330, Dec. 1994.
- [102] C. Sun, “Undo as concurrent inverse in group editors,” *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 9, no. 4, pp. 309–361, Dec. 2002.
- [103] M. Ressel, D. Nitsche-Ruhland, and R. Gunzenhäuser, “An integrating, transformation-oriented approach to concurrency control and undo in group editors,” in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 1996, pp. 288–297.
- [104] H. Shen and C. Sun, “Flexible notification for collaborative systems,” in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 2002, pp. 77–86.
- [105] R. Li and D. Li, “Commutativity-based concurrency control in groupware,” in *Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2005.
- [106] D. Li and R. Li, “An Admissibility-Based Operational Transformation Framework for Collaborative Editing Systems,” *Computer Supported Cooperative Work (CSCW)*, vol. 19, no. 1, pp. 1–43, Oct. 2009.
- [107] W. Yu, L. André, and C.-L. Ignat, “A CRDT Supporting Selective Undo for Collaborative Text Editing,” in *Distributed Applications and Interoperable Systems*, 2015, pp. 193–206.
- [108] L. André, S. Martin, G. Oster, and C. L. Ignat, “Supporting adaptable granularity of changes for massive-scale collaborative editing,” in *Proceedings of the 9th International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom)*, 2013, pp. 50–59.
- [109] M. Ahmed-Nacer, P. Urso, and F. Charoy, “Improving textual merge result,” in *Proceedings of the 9th International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom)*, 2013, pp. 390–399.
- [110] S. Weiss, P. Urso, and P. Molli, “Logoot: A Scalable Optimistic Replication Algorithm for Collaborative Editing on P2P Networks,” in *Proceedings of the 29th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2009, pp. 404–412.
- [111] J. Lautamäki, A. Nieminen, J. Koskinen, T. Aho, T. Mikkonen, and M. Englund, “CoRED: browser-based Collaborative Real-time Editor for Java web applications,” in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, 2012, pp. 1307–1316.
- [112] E. W. Myers, “AnO(ND) difference algorithm and its variations,” *Algorithmica*, vol. 1, no. 1–4, pp. 251–266, 1986.
- [113] S. Wu and U. Manber, “Fast Text Searching: Allowing Errors,” *Communications of the ACM*, vol. 35, no. 10, pp. 83–91, Oct. 1992.
- [114] N. Fraser, “MobWrite,” 2010. [Online]. Available: <https://code.google.com/p/google-mobwrite/>.
- [115] MobWrite, “MobWrite,” 2010. [Online]. Available: <http://www.mobwrite.net/>.

- [116] H. Rajaei, “A shared-edit & view web-based simulation framework,” in *Proceedings of the 11th Symposium on Communications and Networking Simulation*, 2008, pp. 147–152.
- [117] Y. Saito and M. Shapiro, “Optimistic replication,” *ACM Computing Surveys (CSUR)*, vol. 37, no. 1, pp. 42–81, Mar. 2005.
- [118] S. Kumawat and A. Khunteta, “A Survey on Operational Transformation Algorithms: Challenges, Issues and Achievements,” *International Journal of Computer Applications*, vol. 3, no. 12, pp. 30–38, Jul. 2010.
- [119] J. M. Carroll, D. C. Neale, P. L. Isenhour, M. B. Rosson, and D. S. McCrickard, “Notification and awareness: synchronizing task-oriented collaborative activity,” *International Journal of Human-Computer Studies*, vol. 58, no. 5, pp. 605–632, May 2003.
- [120] C. Gutwin and S. Greenberg, “Effects of awareness support on groupware usability,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1998, pp. 511–518.
- [121] P. Jermann, D. Gergle, R. Bednarik, and S. Brennan, “Duet 2012: Dual Eye Tracking in CSCW,” in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 2012, pp. 23–24.
- [122] V. Navalpakkam and E. F. Churchill, “Eye Tracking: A Brief Introduction,” in *Ways of Knowing in HCI*, Springer New York, 2014, pp. 323–348.
- [123] C. A. Gutwin, M. Lippold, and T. C. N. Graham, “Real-time Groupware in the Browser: Testing the Performance of Web-based Networking,” in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 2011, pp. 167–176.
- [124] M. J. Bietz, “Effects of Communication Media on the Interpretation of Critical Feedback,” in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 2008, pp. 467–476.
- [125] O. Arazy and E. Stroulia, “A Utility for Estimating the Relative Contributions of Wiki Authors,” in *Proceedings of the Third International AAAI Conference on Weblogs and Social Media*, 2009.
- [126] M. Hess, B. Kerr, and L. Rickards, “Wiki user statistics for regulating behaviour,” Citeseer, 2006.
- [127] M. Sabel, “Structuring Wiki Revision History,” in *Proceedings of the 2007 International Symposium on Wikis*, 2007, pp. 125–130.
- [128] X. Ding, C. Danis, T. Erickson, and W. A. Kellogg, “Visualizing an Enterprise Wiki,” in *CHI '07 Extended Abstracts on Human Factors in Computing Systems*, 2007, pp. 2189–2194.
- [129] B. Hoisl, W. Aigner, and S. Miksch, “Social Rewarding in Wiki Systems – Motivating the Community,” in *Online Communities and Social Computing*, Springer Berlin Heidelberg, 2007, pp. 362–371.
- [130] D. Spinellis, “Version control systems,” *IEEE Software*, vol. 22, no. 5, pp. 108–109, 2005.
- [131] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, “Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository,” in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 2012, pp. 1277–1286.
- [132] Google Inc., “See the history of changes made to a file,” *Docs editors Help*, 2015. [Online]. Available: <https://support.google.com/docs/answer/190843?hl=en>. [Accessed: 07-May-2015].

- [133] A. Schulz, H. Paulheim, and F. Probst, "Crisis information management in the web 3.0 age," *Proceedings of the 9th International Conference on Integrative and Analytical Approaches to Crisis Response and Emergency Management Information Systems*, 2012.
- [134] C. Gutwin, T. C. N. Graham, C. Wolfe, N. Wong, and B. de Alwis, "Gone but Not Forgotten: Designing for Disconnection in Synchronous Groupware," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 2010, pp. 179–188.
- [135] A. S. Vivacqua and M. R. S. Borges, "Taking advantage of collective knowledge in emergency response systems," *Journal of Network and Computer Applications*, vol. 35, no. 1, pp. 189–198, Jan. 2012.
- [136] L. A. Guerrero, S. F. Ochoa, J. A. Pino, and C. A. Collazos, "Selecting Computing Devices to Support Mobile Collaboration," *Group Decision and Negotiation*, vol. 15, no. 3, pp. 243–271, Jun. 2006.
- [137] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic keyword extraction from individual documents," *Text Mining*, pp. 1–20, 2010.
- [138] M. F. Goodchild and J. A. Glennon, "Crowdsourcing geographic information for disaster response: a research frontier," *International Journal of Digital Earth*, vol. 3, no. 3, pp. 231–241, 2010.
- [139] S. Vieweg, L. Palen, S. B. Liu, A. L. Hughes, and J. Sutton, "Collective intelligence in disaster: An examination of the phenomenon in the aftermath of the 2007 Virginia Tech shootings," in *Proceedings of the Information Systems for Crisis Response and Management Conference (ISCRAM)*, 2008.
- [140] H. Gao, G. Barbier, R. Goolsby, and D. Zeng, "Harnessing the crowdsourcing power of social media for disaster relief," Arizona State University, 2011.
- [141] O. Okolloh, "Ushahidi, or 'testimony': Web 2.0 tools for crowdsourcing crisis information," *Participatory Learning and Action*, vol. 59, no. 1, pp. 65–70, 2009.
- [142] J. Yin, A. Lampert, M. Cameron, B. Robinson, and R. Power, "Using social media to enhance emergency situation awareness," *IEEE Intelligent Systems*, vol. 27, no. 6, pp. 52–59, 2012.
- [143] A. L. Hughes, L. Palen, J. Sutton, S. B. Liu, and S. Vieweg, "Site-seeing in disaster: An examination of on-line social convergence," in *Proceedings of the 5th International ISCRAM Conference*, 2008.
- [144] M. Zook, M. Graham, T. Shelton, and S. Gorman, "Volunteered Geographic Information and Crowdsourcing Disaster Relief: A Case Study of the Haitian Earthquake," *World Medical & Health Policy*, vol. 2, no. 2, pp. 7–33, Jul. 2010.
- [145] K. Peffers, M. Rothenberger, T. Tuunanen, and R. Vaezi, "Design Science Research Evaluation," in *Design Science Research in Information Systems. Advances in Theory and Practice*, 2012, pp. 398–410.
- [146] K. Baker, S. Greenberg, and C. Gutwin, "Heuristic Evaluation of Groupware Based on the Mechanics of Collaboration," in *Engineering for Human-Computer Interaction*, Springer Berlin Heidelberg, 2001, pp. 123–139.
- [147] S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to build high-performance network programs," *IEEE Internet Computing*, vol. 14, no. 6, pp. 80–83, 2010.
- [148] D. Díez, S. Tena, R. Romero-Gomez, P. Díaz, and I. Aedo, "Sharing your view: A distributed user interface approach for reviewing emergency plans," *International Journal*

of Human-Computer Studies, vol. 72, no. 1, pp. 126–139, Jan. 2014.

- [149] K. Baker, S. Greenberg, and C. Gutwin, “Empirical Development of a Heuristic Evaluation Methodology for Shared Workspace Groupware,” in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 2002, pp. 96–105.
- [150] K. F. MacDorman, T. J. Whalen, C.-C. Ho, and H. Patel, “An Improved Usability Measure Based on Novice and Expert Performance,” *International Journal of Human-Computer Interaction*, vol. 27, no. 3, pp. 280–302, 2011.
- [151] L.-O. Bligård and A.-L. Osvalder, “Enhanced Cognitive Walkthrough: Development of the Cognitive Walkthrough Method to Better Predict, Identify, and Present Usability Problems,” *Advances in Human-Computer Interaction*, vol. 2013, Jan. 2013.
- [152] C.-L. Ignat, G. Oster, O. Fox, V. L. Shalin, and F. Charoy, “How Do User Groups Cope with Delay in Real-Time Collaborative Note Taking,” in *Proceedings of the 14th European Conference on Computer Supported Cooperative Work (ECSCW)*, 2015, pp. 223–242.

Appendix A: Ethics Approval H6412

This administrative form
has been removed

Appendix B: Requirements Gathering Survey (Chapter 4)

Table B1: Questions contained in the requirements gathering survey.

Section of Survey	Question	Response type
Demographics	What stages of emergency management are you involved in?	Select all that apply {Mitigation, Preparedness, Response, Recovery}
	How many years have you been involved in emergency management?	Open-ended short answer text
	What role do you have in emergency management?	Open-ended short answer text
	What types of emergencies do you typically assist in?	Select all that apply {Meteorological, Geological, Health, Accidents, Conflict, Other}
	How do you usually communicate with members of your team or organisation during emergency management activities?	Select all that apply {Radio, Mobile, Desk phone, Email, Chat, Written Reports, Face-to-face meetings, Teleconferencing, Video conferencing, Social media, Other}
	How do you usually communicate with professionals from other organisation during emergency management activities?	Select all that apply {Radio, Mobile, Desk phone, Email, Chat, Written reports, Face-to-face meetings, Teleconferencing, Video conferencing, Social media, Other}
Use of reports for emergency management	How often do you use or complete situation reports (sitreps), incident reports, or other forms of reporting for emergency management?	10-point scale {1=Never, 10=Frequently}
	How are these reports typically written in your organisation?	Select all that apply {Single author, Sequential, Parallel, Reactive}
	What work modes are used to complete sitreps?	Select all that apply {All contributors working in same place/same time, All contributors working in same place/different times, Contributors working in different places/same time, Contributors working in different places/different times}
Software tools	What software tools do you use to write reports for emergency management?	Select all that apply {Synchronous writing programs, Asynchronous writing programs}
	Please list the software you use	Open-ended long answer text

	What are the strengths of this software for report writing in emergency management?	Open-ended long answer text
	What are the weaknesses of this software for report writing in emergency management?	Open-ended long answer text
Hardware tools	What hardware do you use to support report writing for emergency management?	Select all that apply {Desktop computer, Portable computer, Smartphone, Tablet, PDA, Wearable technology, Radio, Other}
	What are the strengths of this hardware for report writing in emergency management?	Open-ended long answer text
	What are the weaknesses of this hardware for report writing in emergency management?	Open-ended long answer text
Challenges	What challenges do you experience in completing reports for emergency management?	Open-ended long answer text
Improving report writing through collaborative technologies	How useful would it be to have technology enabling multiple authors to work on the same report at the same time?	5-point scale {1=Not at all, 5=Very}
	How useful would it be to have technology enabling multiple authors to work on the same report in different locations?	5-point scale {1=Not at all, 5=Very}
	How important is it in emergency management that stakeholders have the most up-to-date version of a report?	5-point scale {1=Not at all, 5=Very}
	How important is it in emergency management that there are no divergent copies of the same report being circulated?	5-point scale {1=Not at all, 5=Very}
	How important is it in emergency management that technology used to complete reports is robust to poor network coverage?	5-point scale {1=Not at all, 5=Very}
	How important is it in emergency management that technology used to complete reports has long battery life?	5-point scale {1=Not at all, 5=Very}
	How important is it in emergency management that technology used to complete reports is adaptive to mobile devices (e.g. smartphones, iPad)?	5-point scale {1=Not at all, 5=Very}

	<p>What features would make a computer-based, collaborative report writing system effective?</p>	<p>Select all that apply {Highlighting where each author is currently interacting within the report, Authors have the ability to lock sections of a report, Groups of authors have the ability to lock sections of a report, An administrator who oversees the report, Ability to review changes to the report over time, A map depicting where each author is located when they are editing the report, Support for contributing to the report using a mobile device, Real-time read-only access for other stakeholders that need to read the report, Reporting feature where a selected version of the report is pushed to stakeholders as a read-only copy, Informal chat, Information on the identity of each user contributing to the report, Information on when content was added to the report, Ability to see who contributed what to the report, Ability to see which areas of the report are being edited the most, Ability to see which areas of the report are being edited the least, Ability to contribute photos or other media, Automated tools that extract important content from the report and display as a summary, Integration with social media, Other}</p>
--	--	--

Appendix C: Hierarchical Task Analysis for ECW (Chapter 10)

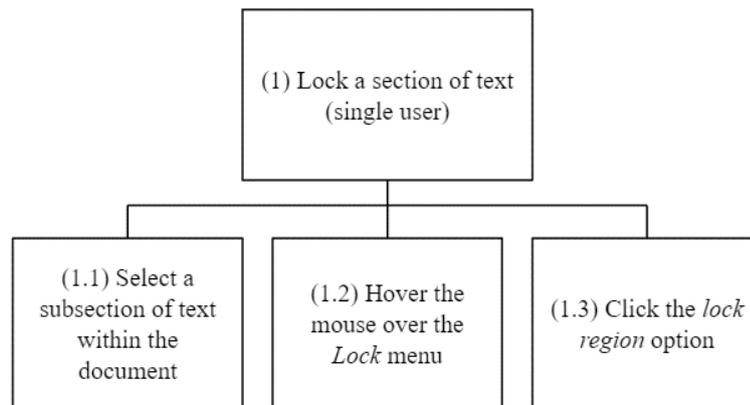


Figure C1: HTA for single user locking task.

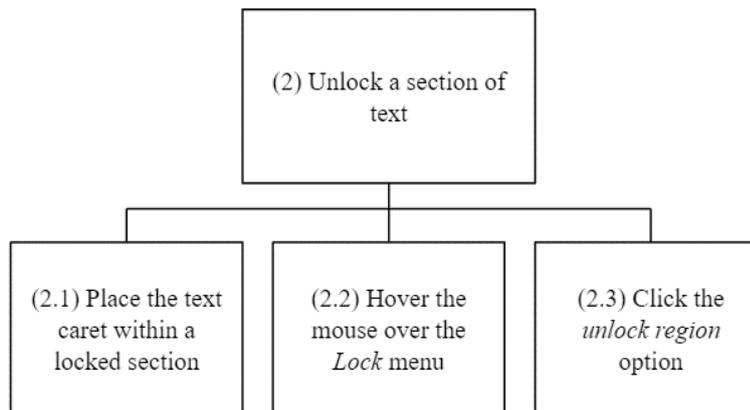


Figure C2: HTA for unlocking task.

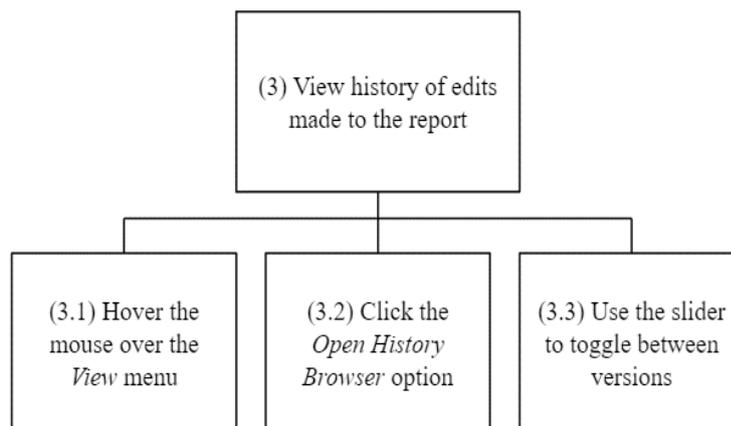


Figure C3: HTA for view history of edits task.

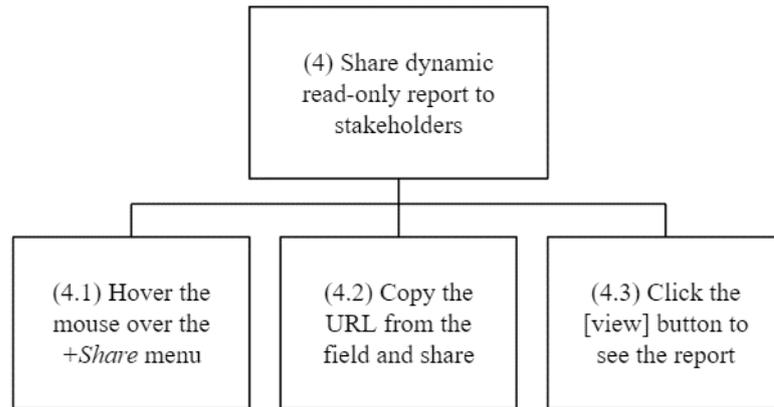


Figure C4: HTA for share dynamic read-only report to stakeholders task.

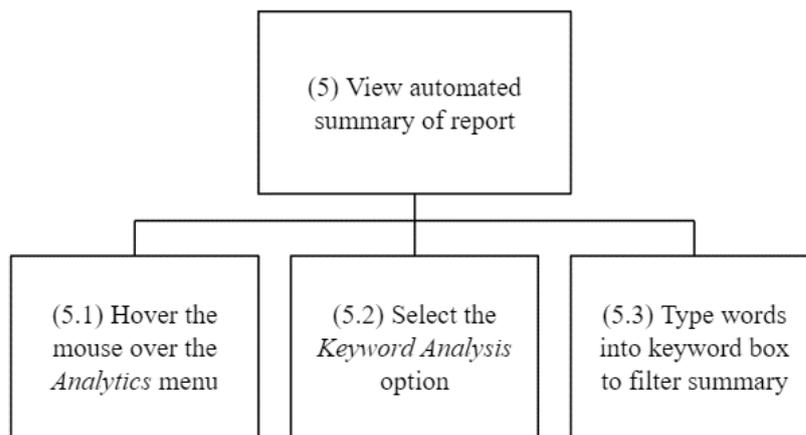


Figure C5: HTA for view automated summary of report task.

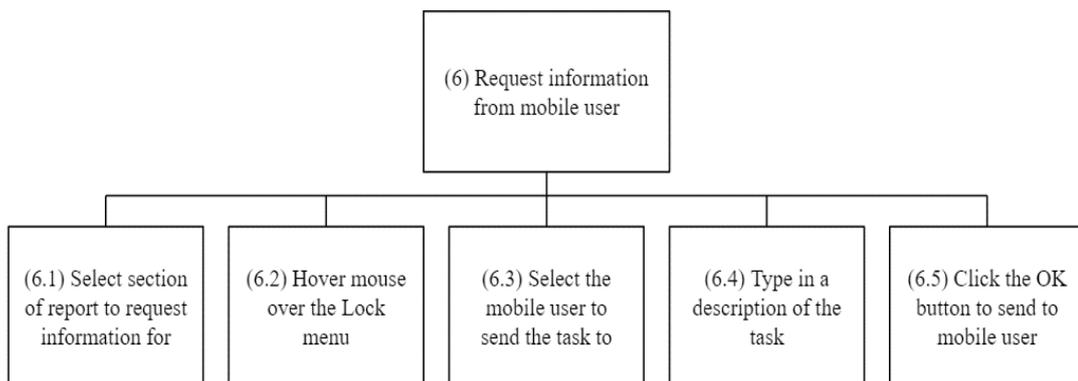


Figure C6: HTA for request information from mobile user task.

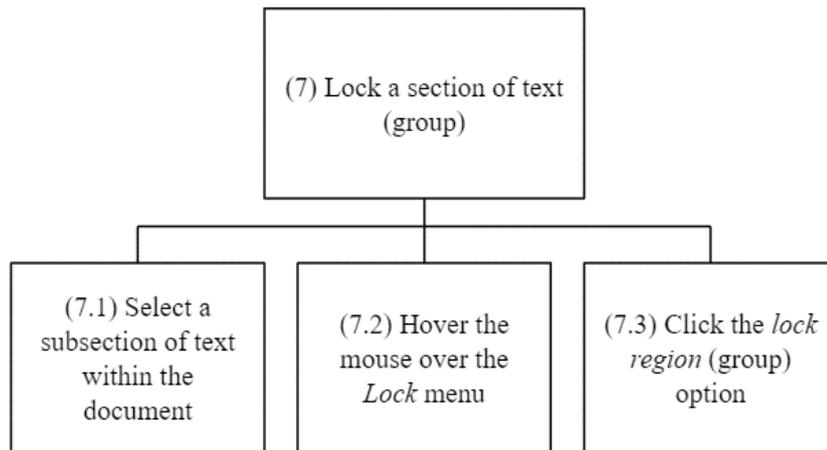


Figure C7: HTA for group user locking task.

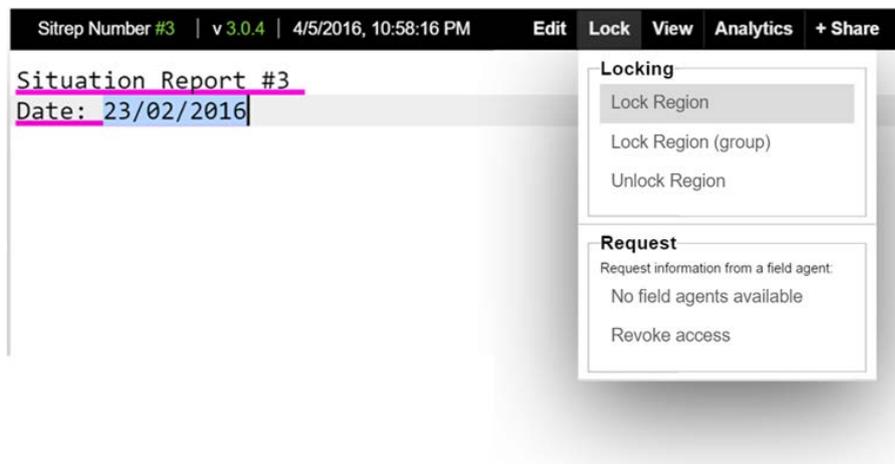
Appendix D: Specification of user interface for ECW (Chapter 10)



(a) Task/function 1.0: Lock a section of text (single-user).



(b) User action to operation 1.1: Select a subsection of text within the document.



(c) User action to operation 1.2: Hover the mouse over the *Lock* menu.



(d) User action to operation 1.3: Click the *Lock Region* option

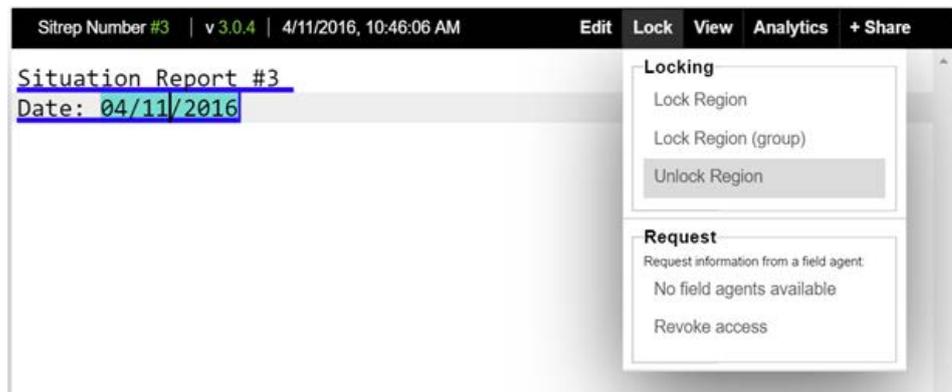
Figure D.1: Specification of the user interface for single user locking task.



(a) Task/function 2.0: Unlock a section of text.



(b) User action to operation 2.1: Place the text caret within a locked section.

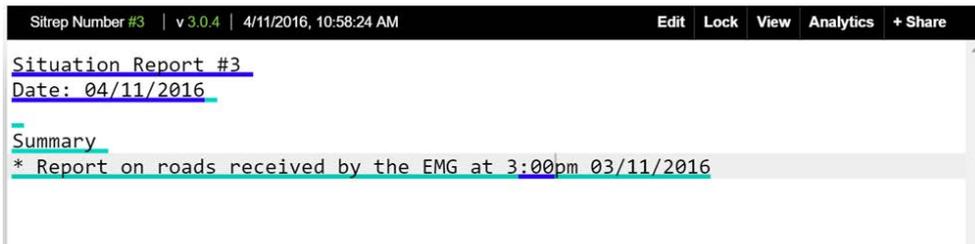


(c) User action to operation 2.2: Hover the mouse over the Lock menu.

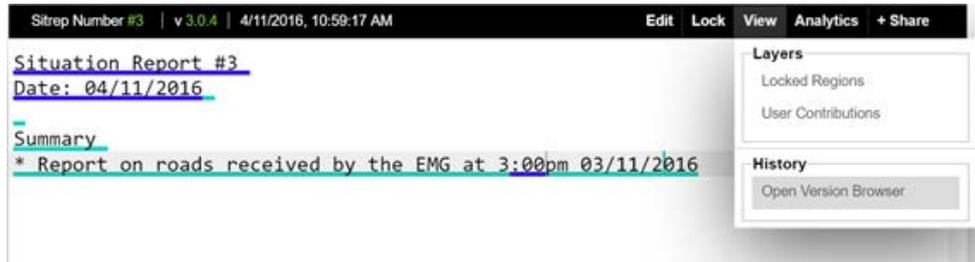


(d) User action to operation 2.3: Click the unlock region option.

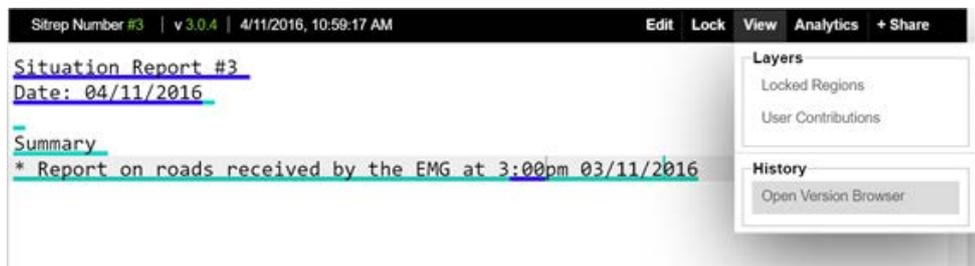
Figure D.2: Specification of the user interface for the unlocking task.



(a) Task/function 3.0: View history of edits.



(b) User action to operation 3.1: Hover the mouse over the view menu.

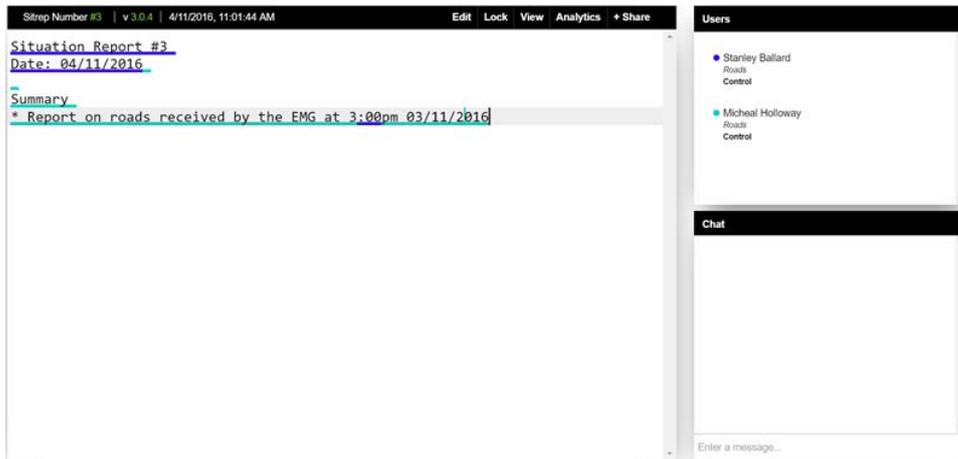


(c) User action to operation 3.2: Click the Open History Browser option.

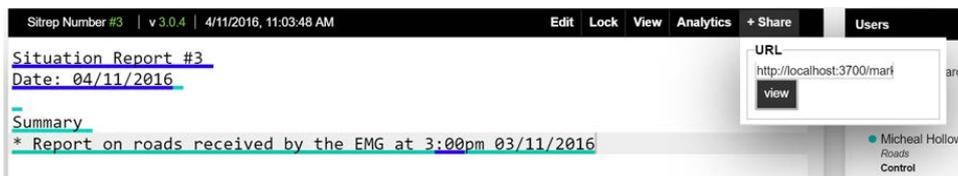


(d) User action to operation 3.3: Use the slider to move between report versions.

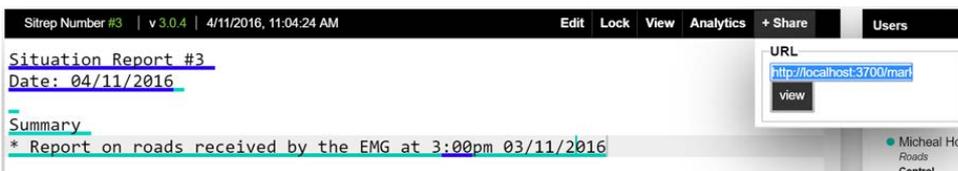
Figure D.3: Specification of the user interface for the view history task.



(a) Task/function 4.0: Share dynamic read-only report.

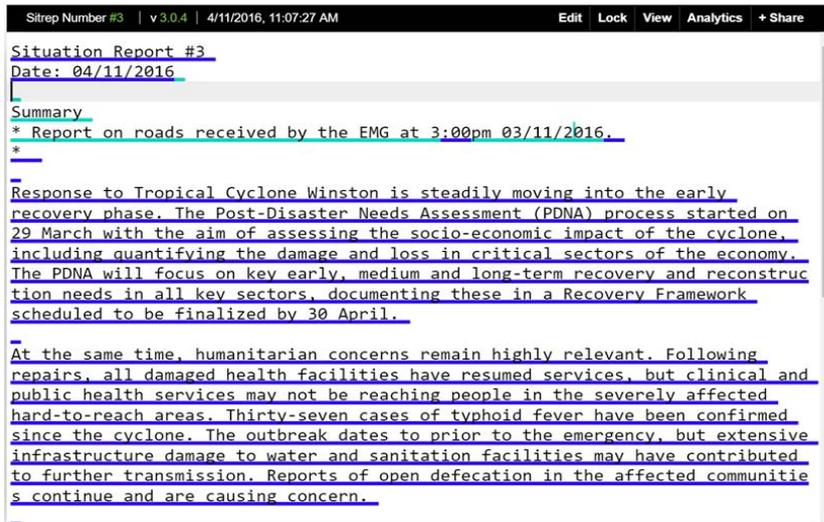


(b) User action to operation 4.1: Hover the mouse over the +Share menu.

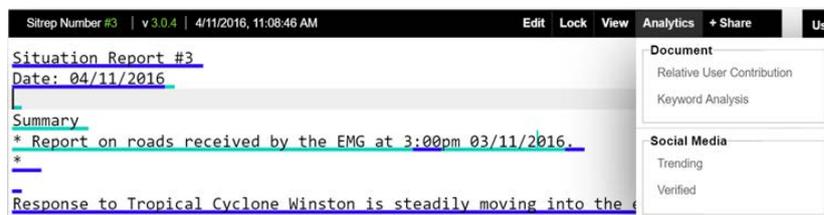


(c) User action to operation 4.2: Copy the URL from the field and share.

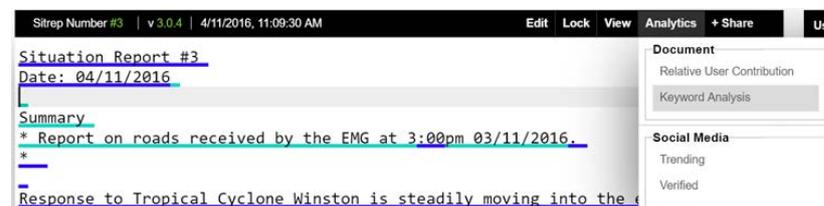
Figure D.4: Specification of the user interface for the share dynamic report task.



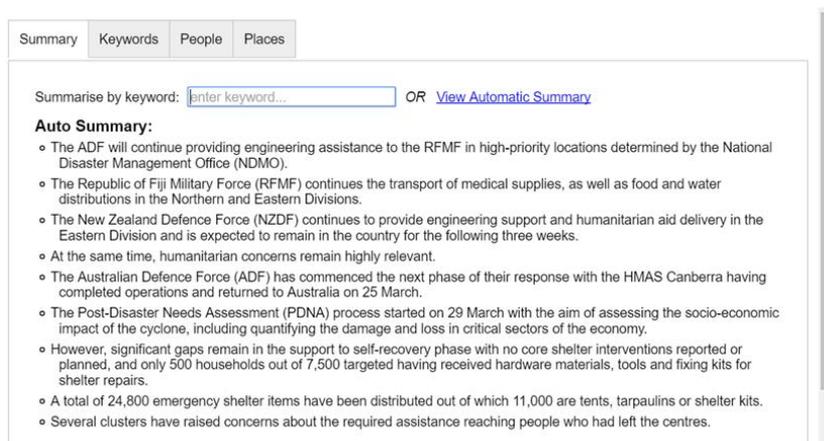
(a) Task/function 5.0: View automated summary of report.



(b) User action to operation 5.1: Hover the mouse over the analytics menu.

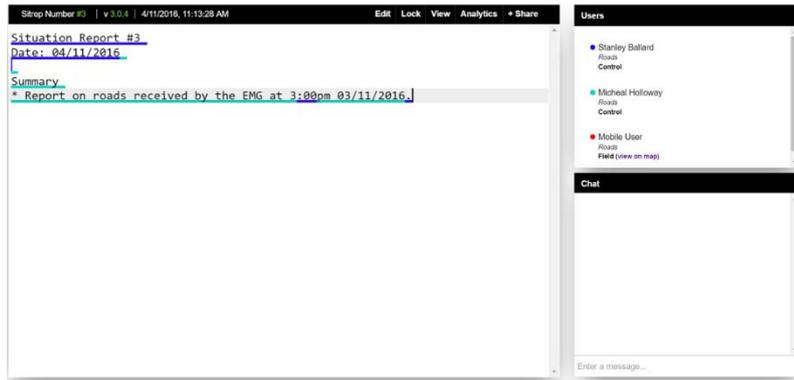


(c) User action to operation 5.2: Click the Keyword Analysis option.

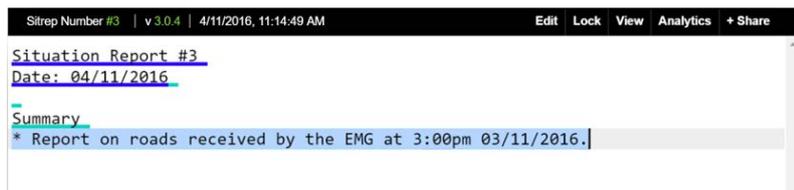


(d) User action to operation 5.3: Type keywords into the search box to filter.

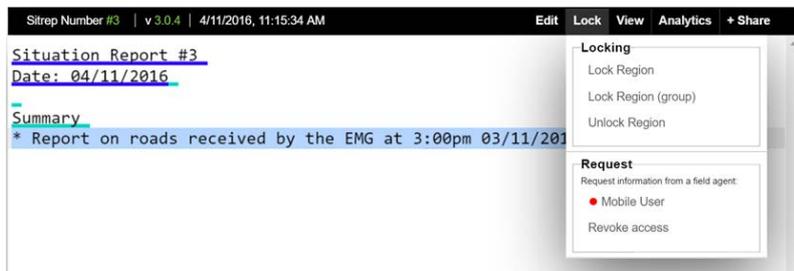
Figure D.5: Specification of the user interface for the show analytics task.



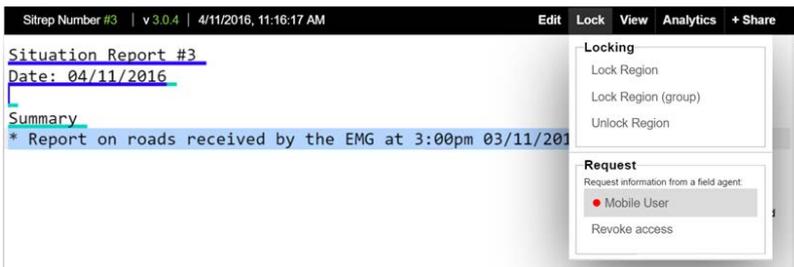
(a) Task/function 6.0: Request information from mobile user.



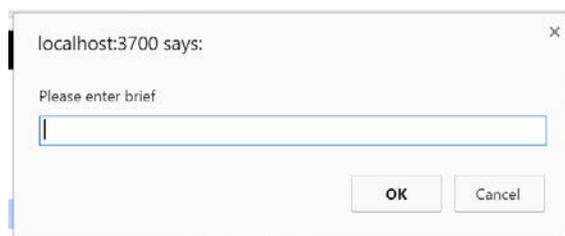
(b) User action to operation 6.1: Select section of report to request information for.



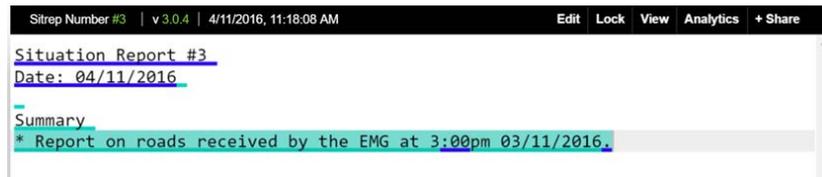
(c) User action to operation 6.2: Hover mouse over the Lock menu.



(d) User action to operation 6.3: Click the mobile user to send the task to.



(e) User action to operation 6.4: Type in a description of the task.

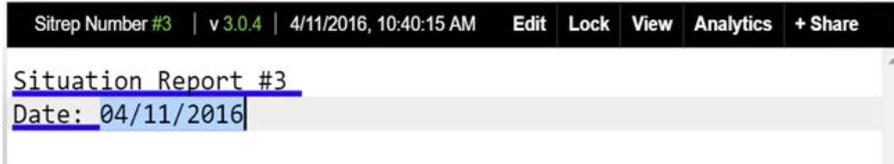


(f) User action to operation 6.5: Click OK to send the task to the mobile user.

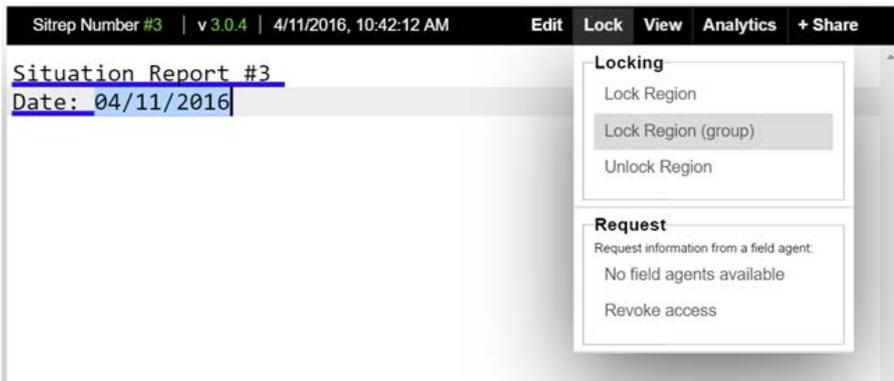
Figure D.6: Specification of the user interface for the request information from mobile user task.



(a) Task/function 7.0: Lock a section of text (group).



(b) User action to operation 7.1: Select a subsection of text in the report.



(c) User action to operation 7.2: Hover the mouse over the Lock menu.



(d) User action to operation 7.3: Click the lock region (group) option.

Figure D.7: Specification of the user interface for the group locking task.

Appendix E: ECW Analysis Tables (Chapter 10)

Table E1: Analysis of functions for the single-user locking task.

1.0	Lock a section of text (single-user)	Lock a section of text (single-user)		
	Failure/success story	Usability problem	PS	PT
(1)	<p>Will the user know that the evaluated function is available?</p> <p>Do not know. It depends on whether the user has the expectation of being able to execute this function in the system.</p>	User does not expect functionality	3	U
(2)	<p>Will the user be able to notice that the function is available?</p> <p>Yes, there is a menu and buttons to indicate locking functionality.</p>	No usability problem	5	-
(3)	<p>Will the user associate the clues with the function?</p> <p>Yes, probably. There is a menu titled "Lock" within the text editing area.</p>	Unclear text	4	T
(4)	<p>Will the user get sufficient feedback when using the function?</p> <p>No, uncertain. There are no clear instructions that a user needs to select text prior to using the locking function.</p>	Instructions are not displayed	2	H
(5)	<p>Will the user get sufficient feedback to understand that the function has been fully performed?</p> <p>Yes, probably, the locked text will change colour to indicate that a lock has been activated.</p>	Feedback is not sufficient	4	F

Table E2: Analysis of operations for the single-user locking task.

1.1	Select a subsection of text within the document	Select a subsection of text within the document		
	Failure/success story	Usability problem	PS	PT
(1)	<p>Will the user try to achieve the right goals of the operation?</p> <p>No, unlikely. It depends if the user expects that the sequence involves selecting text first or using the lock menu first.</p>	Unclear indication of correct action	2	S
(2)	<p>Will the user be able to notice that the action of the operation is available?</p> <p>Yes, probably, as selecting text within a document is a standard feature of similar applications.</p>	User does not relate the action of selecting text with the locking outcome	4	H
(3)	<p>Will the user associate the action of the operation with the right goal of the operation?</p> <p>Do not know, it depends if the user expects to select text before selecting the lock option or after.</p>	User does not expect the sequence of events required by the system	3	S
(4)	<p>Will the user be able to perform the correct action?</p>	No usability problem	5	-

	Yes, there are no restrictions in the text input area that would stop a user from selecting text.			
(5)	<i>Will the user get sufficient feedback to understand that the action is performed and the goal is achieved?</i> Yes, like similar applications the selected text will be highlighted when selected.	No usability problem	5	-
1.2 Hover the mouse over the Lock menu				
1.2	Hover the mouse over the Lock menu	Hover the mouse over the Lock menu		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Yes, the user has the goal of locking some text and will see the Lock menu and associate it with this task.	No usability problem	5	-
(2)	Yes, the menu is clearly displayed.	No usability problem	5	-
(3)	Yes, the user will associate hovering the mouse over the lock menu with displaying options for locking	No usability problem	5	-
(4)	Yes, there are no restrictions on the user moving the mouse cursor to the menu	No usability problem	5	-
(5)	Yes, when the menu title is hovered over a submenu will immediately appear	No usability problem	5	-
1.3 Click the lock region option				
1.3	Click the lock region option	Click the lock region option		
	<i>Failure success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Yes, the user wants to lock a region so will click the lock region option	No usability problem	5	-
(2)	Yes, the operation is clearly displayed in the menu	No usability problem	5	-
(3)	Yes, probably, as the user intends to lock a region of text and that is what the menu item conveys	Unclear instructions	4	T
(4)	Yes, the user can easily click the option with the mouse	No usability problem	5	-
(5)	Yes, probably, the selected text will change to be highlighted with the user's colour to indicate that the lock is now active	Feedback is not sufficient	4	F

Table E3: Analysis of functions for the unlock a section of text task.

2.0	Unlock a section of text.	Unlock a section of text.		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Do not know. It depends on whether the user has the expectation of being able to execute this function in the system.	User does not expect functionality	3	U
(2)	Yes, there is a menu titled Lock that has a clearly marked Unlock option.	No usability problem	5	-
(3)	Yes, probably. The menu and options are titled appropriately.	Unclear text	4	T
(4)	Yes, probably. Upon using the function the menu will disappear to indicate that the option has been selected.	Feedback is not sufficient	4	F
(5)	Yes, probably. The selected lock will disappear from the document to indicate that the lock has been removed.	Feedback is not sufficient	4	F

Table E4: Analysis of operations for the unlock a section of text task.

2.1	Place the text caret within a locked section	Place the text caret within a locked section		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Do not know. It depends on whether the user has the expectation that the caret should be placed within a lock before unlocking occurs.	Sequence of events is unclear	3	S
(2)	Yes, to click on text within a text input area is common within many applications that already exist.	No usability problem	5	-
(3)	Yes, that action of clicking within a text area is associated with placing the text caret inside a locked section	No usability problem	5	-
(4)	Yes, there are no restrictions to stop the user from clicking inside a locked section of text	No usability problem	5	-
(5)	Yes, the visual caret will instantly appear in the locked section when it is clicked	No usability problem	5	-
2.2	Hover the mouse over the Lock menu	Hover the mouse over the Lock menu		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Yes, the user has the goal of unlocking some text and will see the Lock menu and associate it with this task.	No usability problem	5	-
(2)	Yes, the menu is clearly displayed.	No usability problem	5	-
(3)	Yes, the user will associate hovering the mouse over the lock	No usability problem	5	-

	menu with displaying options for locking			
(4)	Yes, there are no restrictions on the user moving the mouse cursor to the menu	No usability problem	5	-
(5)	Yes, when the menu title is hovered over a submenu will immediately appear	No usability problem	5	-
2.3	Click the unlock region option	Click the unlock region option		
	<i>Failure success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Yes, the user wants to unlock a region so will click the unlock region option	No usability problem	5	-
(2)	Yes, the operation is clearly displayed in the menu	No usability problem	5	-
(3)	Yes, probably, as the user intends to unlock a region of text and that is what the menu item conveys	Unclear instructions	4	T
(4)	Yes, the user can easily click the option with the mouse	No usability problem	5	-
(5)	Yes, probably, the selected text will be unhighlighted to indicate that the lock has been removed	Feedback is not sufficient	4	F

Table E5: Analysis of functions for the view history of edits task.

3.0	View history of edits	View history of edits		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Do not know. It depends on whether the user has the expectation of being able to execute this function in the system.	User does not expect functionality	3	U
(2)	Yes, probably. There is a View menu that has a history option.	Unclear text	4	T
(3)	Yes, probably. The history option is labelled to suggest is functionality.	Unclear text	4	T
(4)	Yes, probably. When the history view is open the slider will show all changes made over time	Feedback is not sufficient	4	F
(5)	Yes, the history view will appear when it is selected and feedback is presented immediately when interacting with this view.	No usability problem	5	-

Table E6: Analysis of operations for the view history of edits task.

3.1	Hover the mouse over the View menu	Hover the mouse over the View menu		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Yes, probably. The user has the goal of viewing the history of edits and will seek the option from the menu..	Unclear text	4	T
(2)	Yes, probably. The view menu is displayed but may not be obvious that history is in this menu.	Unclear text	4	T
(3)	Yes, probably. The user may associate hovering the mouse over the view menu with displaying options to view the history	Unclear text	4	T
(4)	Yes, there are no restrictions on the user moving the mouse cursor to the menu	No usability problem	5	-
(5)	Yes, when the menu title is hovered over a submenu will immediately appear	No usability problem	5	-
3.2	Click the Open History Browser option	Click the Open History Browser option		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Yes, the user wants to view this history so will select this option	No usability problem	5	-
(2)	Yes, the operation is clearly displayed in the menu	No usability problem	5	-

(3)	Yes, as the user intends to view the history of edits and that is what the menu item conveys	No usability problem	5	-
(4)	Yes, the user can easily click the option with the mouse	No usability problem	5	-
(5)	Yes, the history view will appear upon clicking this option	No usability problem	5	-
3.3	Use the slider to move between report versions	Use the slider to move between report versions		
	<i>Failure success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Yes, probably. The slider is presented below the document to indicate a chronological sequence of events.	Unclear instructions	4	T
(2)	Yes, probably. The slider is visible and quite large, positioned below the history window.	Unclear representation of slider	4	T
(3)	Yes, probably. As the context of this view is to see the history of reports, a slider presents a similar use case to other systems that show a chronological timeline	User does not expect slider to be used this way	4	U
(4)	Yes, there are no restrictions on the user interacting with the slider	No usability problem	5	-
(5)	Yes, when the slider is dragged all elements of the history view will change to reflect the different versions of the document	No usability problem	5	-

Table E7: Analysis of functions for the share dynamic read-only report task.

4.0	Share dynamic read-only report	Share dynamic read-only report		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Do not know. It depends on whether the user has the expectation of being able to execute this function in the system.	User does not expect functionality	3	U
(2)	Yes, probably. The share menu is clearly displayed in the upper section of the interface	Unclear text	4	T
(3)	Yes, probably. The goal of the user in this function is to share a copy of the report and the menu is titled appropriately.	Unclear text	4	T
(4)	Yes, probably. When the menu is accessed the user can see a link, but may not understand that it should be copied and shared outside the system.	User does not expect sequence of events	4	S
(5)	Do not know. The user will get feedback within the system while accessing the URL and copying it, but it is impossible to know how they will use this outside of the system.	User does not understand the process of copying the link and sharing outside the system	3	U

Table E8: Analysis of operations for the share dynamic read-only report task.

4.1	Hover the mouse over the +Share menu	Hover the mouse over the +Share menu		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Yes, the user has the goal of sharing the report and will see the Share menu and associate it with this task.	No usability problem	5	-
(2)	Yes, the menu is clearly displayed.	No usability problem	5	-
(3)	Yes, the user will associate hovering the mouse over the share menu with displaying options for sharing	No usability problem	5	-
(4)	Yes, there are no restrictions on the user moving the mouse cursor to the menu	No usability problem	5	-
(5)	Yes, when the menu title is hovered over a submenu will immediately appear	No usability problem	5	-
4.2	Copy the URL from the field and share	Copy the URL from the field and share		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Do not know. The URL is available to be copied and shared but there are no instructions that the user should do this.	Unclear text	3	T
(2)	Yes, probably. The URL is presented in a field that allows for text selection similar to the report.	Unclear instructions	4	F

(3)	Do not know. The goal of this operation is to share the report with other stakeholders, but since it relies on the user copying the link outside the system it is difficult to know.		User does not expect functionality	3	U
(4)	Yes, there is no restriction on the user copying the URL		No usability problem	5	--
(5)	Do not know. The interface will display standard feedback for selecting text and copying, but it depends on whether the user expects more feedback from the system than this.		Insufficient feedback presented	3	F

Table E9: Analysis of functions for the view automated summary of report task.

5.0	View automated summary of report	View automated summary of report		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Do not know. It depends on whether the user has the expectation of being able to execute this function in the system.	User does not expect functionality	3	U
(2)	Yes, probably. There is an option for viewing keyword analysis within an Analytics menu, but it could be clearer	Unclear text	4	T
(3)	Yes, probably. There are only two options available in the Analytics menu so the user would expect it to occur within these functions.	Unclear text	4	T
(4)	Yes, when the user selects this option a sub-window will appear that has the automated summary of the report visible and other options clearly labelled	No usability problem	5	-
(5)	Yes, the new window will appear and display the automated summary	No usability problem	5	-

Table E10: Analysis of operations for the view automated summary of report task.

5.1	Hover the mouse over the Analytics menu	Hover the mouse over the Analytics menu		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Yes, probably. The user has the goal of viewing an automated summary and will see the Analytics menu and probably associate it with this task.	Unclear text	4	T
(2)	Yes, probably. The menu is clearly displayed but may not be immediately associated with automatic summarisation.	Unclear text	4	T
(3)	Yes, probably. The user will likely associate hovering the mouse over the analytics menu with displaying options for this task.	Unclear text	4	T
(4)	Yes, there are no restrictions on the user moving the mouse cursor to the menu	No usability problem	5	-
(5)	Yes, when the menu title is hovered over a submenu will immediately appear	No usability problem	5	-
5.2	Click the Keyword Analysis option	Click the Keyword Analysis option		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Do not know. It depends if the user associates Keyword Analysis with the goal of viewing an automated summary of the report, based on previous experience with other systems.	User does not associate the option with the goal	3	T

(2)	Yes, similar to other menu options in the system it needs to be clicked on to activate the operation	No usability problem	5	-
(3)	Yes, probably. The user intends to view an automatic summary of the report and would reasonably assume that this falls under the category of keyword analysis	Unclear text	4	T
(4)	Yes, the user can easily click the option with the mouse	No usability problem	5	-
(5)	Yes, the keyword analysis/automatic summarisation view will appear upon clicking this option	No usability problem	5	-
5.3	Type keywords into search box to filter	Type keywords into search box to filter		
	<i>Failure success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Yes, the user has the goal of filtering by a keyword and will follow the instructions to achieve this	No usability problem	5	-
(2)	Yes, the action is clearly labelled with instructions at the top of the view	No usability problem	5	-
(3)	Yes, the instructions and prior experience with search boxes will indicate that the action is related to the goal	No usability problem	5	-
(4)	Yes, there are no restrictions on the user typing in the search box and hitting the Enter key to initiate the action	No usability problem	5	-
(5)	Yes, when the action has been initiated the interface will display feedback to indicate success	No usability problem	5	-

Table E11: Analysis of functions for the request information from mobile user task.

6.0	Request information from mobile user	Request information from mobile user		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Do not know. It depends on whether the user has the expectation of being able to execute this function in the system.	User does not expect functionality	3	U
(2)	Yes, probably. It is well labelled but hidden inside the lock menu which may confuse the user.	Option is hidden	4	H
(3)	Yes, probably. The user may be confused that it is in the Lock menu but when the task is found it is clearly labelled.	Option is hidden	4	H
(4)	Yes, probably. The user will be presented with a sequence of instructions and can infer what step of the process is currently being completed.	Unclear feedback	4	F
(5)	Do not know. The user would expect that the task has been completed based on the sequence of events but may not receive enough feedback to indicate that the task was successfully received by a mobile user.	Not enough feedback provided	3	F

Table E12: Analysis of operations for the request information from mobile user task.

6.1	Select section of report to request information for	Select section of report to request information for		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	No. There are no instructions or feedback to indicate to the user that this is the correct step to take.	Unclear indication of correct action	1	H
(2)	Yes, probably, as selecting text within a document is a standard feature of similar applications.	User does not relate the action of selecting text with the locking outcome	4	H
(3)	No, unlikely. The user would not expect to select text prior to requesting information from the mobile user as there are no instructions	User does not expect the sequence of events required by the system	2	S
(4)	Yes, there are no restrictions in the text input area that would stop a user from selecting text.	No usability problem	5	-
(5)	Yes, like similar applications the selected text will be highlighted when selected.	No usability problem	5	-
6.2	Hover mouse over the Lock menu	Hover mouse over the Lock menu		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	No, unlikely. The user wishes to gain information from a mobile user and would likely not associate this task with the	Option is hidden	2	H

	Lock menu			
(2)	Do not know, the menu is clearly displayed but may not be associated with the sequence of events	Sequence of events is unclear	3	S
(3)	Do not know, the goal of hovering over the menu is clear but it may not be titled appropriately to indicate that this operation matches the goal of the task	Unclear text	3	T
(4)	Yes, there are no restrictions on the user moving the mouse cursor to the menu	No usability problem	5	-
(5)	Yes, when the menu title is hovered over a submenu will immediately appear	No usability problem	5	-
6.3	Click the mobile user to send the task to	Click the mobile user to send the task to		
	<i>Failure success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Yes, probably. There are small instructions to indicate that the mobile user should be clicked on to send them the task	Unclear instructions	4	T
(2)	Yes, the option is clearly marked in the menu	No usability problem	5	-
(3)	Yes, the user has the goal of requesting information from a mobile user and this is labelled as such	No usability problem	5	-
(4)	Yes, there are no restrictions to stop the user from clicking on a mobile user	No usability problem	5	-
(5)	Yes, upon clicking on the mobile user the next operation is displayed as a popup	No usability problem	5	-
6.4	Type in a description of the task	Type in a description of the task		
	<i>Failure success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Yes, there are adequate instructions to indicate that the user should type in the task description	No usability problem	5	-
(2)	Yes, the window will pop up with clear instructions and a text input field	No usability problem	5	-
(3)	Yes, the text input field is standard as used in all web applications	No usability problem	5	-
(4)	Yes, there are no restrictions on the user typing in the text field	No usability problem	5	-
(5)	Yes, as the user types into the text field the letters will appear	No usability problem	5	-
6.5	Click OK to send the task to the mobile user	Click OK to send the task to the mobile user		

	<i>Failure success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Yes, the option to click OK is standard as used in other web applications to indicate a positive response to a task	No usability problem	5	-
(2)	Yes, it is clearly displayed underneath the text input field	No usability problem	5	-
(3)	Yes, the OK button is displayed next to a Cancel button and the user can infer the positive response associated with the provided instructions	No usability problem	5	-
(4)	Yes, there are no restrictions on the user clicking the button	No usability problem	5	-
(5)	Yes, probably. The input window will disappear when the button is clicked but there may not be enough feedback to indicate that the mobile user received the task	Insufficient feedback	4	F

Table E13: Analysis of functions for the group locking task.

7.0	Lock a section of text (group)	Lock a section of text (group)		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Do not know. It depends on whether the user has the expectation of being able to execute this function in the system.	User does not expect functionality	3	U
(2)	Yes, there is a menu and buttons to indicate group locking functionality.	No usability problem	5	-
(3)	Yes, probably. There is a menu titled "Lock" within the text editing area.	Unclear text	4	T
(4)	No, uncertain. There are no clear instructions that a user needs to select text prior to using the locking function.	Instructions are not displayed	2	H
(5)	Yes, probably, the locked text will change colour to indicate that a lock has been activated.	Feedback is not sufficient	4	F

Table E14: Analysis of operations for the group locking task.

7.1	Select a subsection of text in the report	Select a subsection of text in the report		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	No, unlikely. It depends if the user expects that the sequence involves selecting text first or using the lock menu first.	Unclear indication of correct action	2	S
(2)	Yes, probably, as selecting text within a document is a standard feature of similar applications.	User does not relate the action of selecting text with the locking outcome	4	H
(3)	Do not know, it depends if the user expects to select text before selecting the lock option or after.	User does not expect the sequence of events required by the system	3	S
(4)	Yes, there are no restrictions in the text input area that would stop a user from selecting text.	No usability problem	5	-
(5)	Yes, like similar applications the selected text will be highlighted when selected.	No usability problem	5	-
7.2	Hover the mouse over the Lock menu	Hover the mouse over the Lock menu		
	<i>Failure/success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Yes, the user has the goal of unlocking some text and will see the Lock menu and associate it with this task.	No usability problem	5	-
(2)	Yes, the menu is clearly displayed.	No usability problem	5	-
(3)	Yes, the user will associate hovering the mouse over the lock menu with displaying options for locking	No usability problem	5	-

(4)	Yes, there are no restrictions on the user moving the mouse cursor to the menu	No usability problem	5	-
(5)	Yes, when the menu title is hovered over a submenu will immediately appear	No usability problem	5	-
7.3	Click the lock region (group) option	Click the lock region (group) option		
	<i>Failure success story</i>	<i>Usability problem</i>	<i>PS</i>	<i>PT</i>
(1)	Yes, the user wants to lock a region for a group so will click the group lock region option	No usability problem	5	-
(2)	Yes, the operation is clearly displayed in the menu	No usability problem	5	-
(3)	Yes, probably, as the user intends to lock a region of text for a group and that is what the menu item conveys	Unclear instructions	4	T
(4)	Yes, the user can easily click the option with the mouse	No usability problem	5	-
(5)	Yes, probably, the selected text will change to be highlighted with the user's colour to indicate that the lock is now active	Feedback is not sufficient	4	F