The Technology of Casually Connected Collaboration

by

Theodor Werner Danzfuss

January 2009

Submitted in partial fulfilment of the requirements for the degree Magister Scientiae (Computer Science) in the Faculty of Engineering, Built Environment and Information Technology.

University of Pretoria.

The Technology of Casually Connected Collaboration

By

Theodor Werner Danzfuss

Since the early eighties researchers have been studying the use of technology that supports collaboration amongst co-workers and group members. This field of computer science became known as Computer Supported Cooperative Work (CSCW). With the advent of wireless and mobile Internet communication technologies research in the CSCW field has been focused on providing “access, anytime and anywhere”. The main contribution of this study is to introduce and analyze the technology required to support casually connected collaboration. Firstly, we define casually connected collaboration as having “access, anytime and anywhere” to collaborators and resources without having explicit control or knowledge over the environment and its technical abilities. In order to distinguish between connected, mobile, and casually connected collaboration we introduce a conceptual model of collaboration that extrapolates the term “access, anytime and anywhere”. We then aim to prove the soundness of our model by using it to classify some well known collaboration scenarios. Furthermore, by evaluating the functional and non-functional requirements for a casually connected collaboration solution, we argue that current commercial and CSCW research implementations do not sufficiently meet these demands. We then present Nomad: a Peer-to-Peer framework specifically designed to overcome the challenges encountered in casually connected collaboration. We study the technology requirements and highlight the implementation details that enabled us to successfully conform to the requirements set by casually connected collaboration. Finally, we pave the road for future work by investigating new features introduced into the Microsoft .NET Framework version 4.0, Visual Studio 2010 and language enhancements made to C# version 4.0.

Keywords: CSCW, groupware, P2P, Microsoft .NET, information sharing, distributed applications, casual connectivity, collaboration

Supervisor: Prof. J. Bishop

Department: Department of Computer Science, University of Pretoria

Degree: Magister Scientiae

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

The Technology of Casually Connected Collaboration

by

Theodor Werner Danzfuss

January 2009

Submitted in partial fulfilment of the requirements for the degree Magister Scientiae (Computer Science) in the Faculty of Engineering, Built Environment and Information Technology.

University of Pretoria.

The Technology of Casually Connected Collaboration

By

Theodor Werner Danzfuss

Since the early eighties researchers have been studying the use of technology that supports collaboration amongst co-workers and group members. This field of computer science became known as Computer Supported Cooperative Work (CSCW). With the advent of wireless and mobile Internet communication technologies research in the CSCW field has been focused on providing "access, anytime and anywhere". The main contribution of this study is to introduce and analyze the technology required to support casually connected collaboration. Firstly, we define casually connected collaboration as having "access, anytime and anywhere" to collaborators and resources without having explicit control or knowledge over the environment and its technical abilities. In order to distinguish between connected, mobile, and casually connected collaboration we introduce a conceptual model of collaboration that extrapolates the term "access, anytime and anywhere". We then aim to prove the soundness of our model by using it to classify some well known collaboration scenarios. Furthermore, by evaluating the functional and non-functional requirements for a casually connected collaboration solution, we argue that current commercial and CSCW research implementations do not sufficiently meet these demands. We then present Nomad: a Peer-to-Peer framework specifically designed to overcome the challenges encountered in casually connected collaboration. We study the technology requirements and highlight the implementation details that enabled us to successfully conform to the requirements set by casually connected collaboration. Finally, we pave the road for future work by investigating new features introduced into the Microsoft .NET Framework version 4.0, Visual Studio 2010 and language enhancements made to C# version 4.0.

Keywords: CSCW, groupware, P2P, Microsoft .NET, information sharing, distributed applications, casual connectivity, collaboration

Supervisor: Prof. J. Bishop

Department: Department of Computer Science, University of Pretoria

Degree: Magister Scientiae

Acknowledgements

"*It's better to hang out with people better than you. Pick out associates whose behaviour is better than yours and you'll drift in that direction...*"

Warren Buffett

With these words of Warren Buffett in mind I would like to thank my family, friends and associates for the contributions they have made towards my life, and in turn towards this work piece. Recognizing the influence people around you have on your thoughts and actions empowers you to take advantage of it and grow in unexpected leaps and bounds.

Special recognition goes out towards my parents for giving me such remarkable opportunities in life, the quality upbringing, support and education I received from you is a critical piece of my life puzzle. Another special word of thanks goes out to all my colleagues and friends at Retro Rabbit, E-Logics, Polelo, and the University of Pretoria for their inspiration, guidance and critique. To Professor Bishop, my supervisor, many thanks for sharing your knowledge with me, the experience gained from working with you will stay with me for the rest of my career. Many thanks to all the sponsors of the Nomad project: THRIP, ELogics, and Microsoft Research - without your support this might not have been possible. Lastly, I would like to thank my Lord and Creator for giving me the talents and abilities to undertake this amazingly rewarding opportunity.

Table of Contents

Chapter 1. History and scope

This investigation into casually connected collaboration primarily builds on the fields of Computer Supported Collaborative Work (CSCW), Human Computer Interfacing (HCI), and distributed Internet computing. As the title suggests, we are investigating the use of appropriate technology that will contribute towards successful collaboration in a casually connected environment.

This dissertation will focus on technology in the services and application layer of the OSI reference model (Wetteroth, 2001), and will not steer into the realm of transport, network or data-link layer issues. We believe that successful collaboration can be achieved by building on top of the existing technology stack, one only has to look at all the successful CSCW projects already implemented using these technologies, to justify this belief (Lamming, et al, 2000) (Kleinrock, 2000) (Ma, et al, 2003).

Common problem experienced by collaborators is the notion of information spread and neglect. One only has to look at the lifecycle of an artefact that needs to be distributed amongst several peers to understand the problem. The artefact is created on a local machine which then gets copied onto removable media or a laptop to continue working at home. Now, the file is backed up onto file servers for safe-keeping before it is being attached in e-mail messages which are sent out to collaborators. When each collaborator receives his e-mail, the same process repeats itself. The main aim of this dissertation is to evaluate and design technology that will reduce the replication of artefacts and manage the distribution thereof, whilst collaborating in a casually connected environment.

1.1 Outline

Chapter 1: Introduction – Defines the scope of our research and discusses the concepts, history and technology behind CSCW systems.

Chapter 2: A conceptual model for casually connected collaboration – We elaborate on the phrase: “Access, anytime and anywhere” and introduce the concept of casual connectivity. We then present a conceptual model that will enable us to distinguish between casual, mobile and connected collaboration scenarios. By evaluating application scenarios against our model, we highlight the technology focus areas for each type of collaboration.

Chapter 3: Nomad – We introduce the user to Nomad, a Peer-to-Peer Framework that supports Casually Connected Collaboration. A comparison between commercial and other CSCW research projects is done on both functional and non-functional requirement level.

Chapter 4: Technology decisions – We present the user with a more in-depth view on the technology decision of a system that will support casually connected collaboration. Implementation details of Nomad are discussed in depth during this Chapter.

Chapter 5: Conclusion and future work – This Chapter concludes by highlighting the objectives that we have achieved, and it aligns itself with web technologies such as semantic web in order to pave the road for future work.

1.2 Computer supported collaborative work

Computer Supported Collaborative Work refers to the study of how people work in groups and how technology can support them (Grudin, 1994). For the purpose of this dissertation the focus will be on the technology spectrum of CSCW, and especially how technology can assist in:

- facilitating communication;
- coordinating tasks between collaborators;
- creating awareness amongst collaborators; and
- collaborating through shared resources.

Strong emphasis has been put on the Human Computer Interface aspects of CSCW and as many other researches have shown (York and Pendharkar, 2004), we believe that people should work in a familiar way even when collaborating. Collaboration should not force people into working with unfamiliar technology, just as it should not change the normal process of how they work.

This work starts off by familiarizing the reader with the history, concepts and technology that goes into computer supported collaborative work (CSCW).

1.3 Casual connectivity

In Layman's terms we can define casual connectivity as having "access, anytime and anywhere" to distributed resources, without having explicit control or knowledge over the technical resources available.

From the research done by Harper, et al (2001) it is clear that we have to elaborate on the phrase "Access, anytime and anywhere". As an example, the type of access, amount of time available, and reason for being mobile are three factors that can affect the technology and design of a successful collaboration system.

In this dissertation we present a model of distributed collaboration that elaborates on the multiple facets of "access, anytime and anywhere". We will investigate and classify the type of access; mode of communication used during the access; reason for initiating the access; and lastly, the resources being accessed. We argue that different technology solutions are applicable for each of the classifications.

1.4 Nomad: A Peer-to-Peer solution

Subsequently we introduce the reader to Nomad: A Peer-to-Peer (P2P) framework that supports collaboration in a casually connected environment. This dissertation builds on the Nomad protocol as defined by Rama and Bishop (2006) and aims to clarify the implementation and technology design of Nomad.

A functional and non-functional comparison is drawn up between Nomad and other projects found in the CSCW domain (both commercial and current research). The functional properties of CSCW systems as presented by Ellis, et al (1991) are used as the foundations for our comparison on a functional compliance level, whilst technical and non-functional compliance are derived from our model of casual connectivity.

1.5 Technology

We then delve into the implementation details of Nomad and highlight the system architecture, network organization, and enabling technologies of the Microsoft .NET Framework version 3.5 that we used during implementation.

We conclude by presenting our results and highlighting future work, such as linking adaptive systems and semantic web technologies with Casually Connected Collaboration.

1.6 CSCW History and scope

The term Computer Supported Cooperative Work was coined in 1984 at a workshop attended by not more than twenty people (Grudin, 1994). These attendees came from different backgrounds and fields of study, but they had a shared interest in investigating how people work and how technology can assist them. Prior to 1984, Office Automation (OA) was the research field that looked into the use of computers in the working environment, mostly to automate a previously manual task. Office Automation (OA) however, did not put any focus on the social/group aspect of the working environment. CSCW can be seen as Office Automation (OA) all grown up; CSCW shifted the focus of research from replacing/automating the human in the work place, towards assisting him and his colleagues in the work process. From its humble beginnings as an effort to learn how people work and interact in groups, CSCW has exploded into a multidisciplinary field covering aspects from social science, economics, and computer science to almost everything in between. Terms such as Groupware, Workgroup Computing, and Computer Supported Collaboration (CSC) are often used in studies that focus on the application or software aspects around CSCW.

To gain a better understanding of the scope of CSCW one has to look at where it fits in with other fields of computer science, in terms of its target audience and enabling technologies.

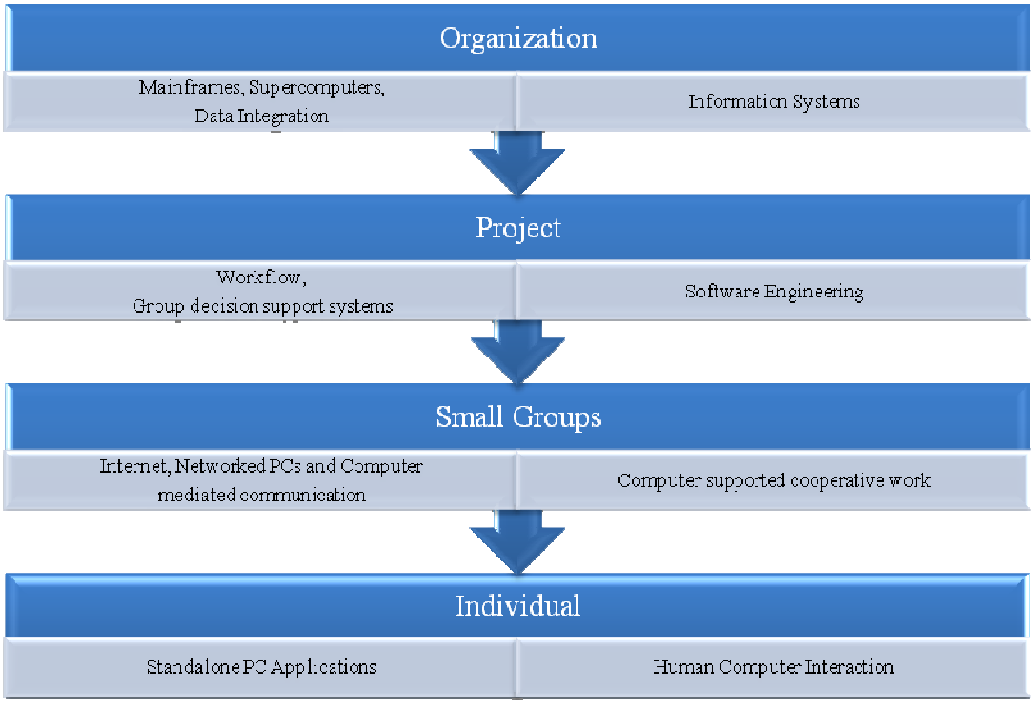

Figure 1 – Fields of computer science in relation to target audience and enabling technologies

Figure 1, as presented by Grudin (1994) clearly shows that the focus of CSCW research is on facilitating collaboration within small groups. The enabling technologies for CSCW are personal computing devices, networks and communication infrastructures. We can thus argue that any development in the field of networking or communication can open new possibilities in the field of CSCW, therefore, is CSCW considered by many as an emerging and extremely dynamic multi-disciplinary field of research (Fitzpatrick. 2003).

1.7 Concept classification

The key functional areas of focus in CSCW research are communication, collaboration, and coordination between members of a group. One can argue that any system that provides one or more of these functionalities can be classified as a CSCW system. Research has presented many classification schemes for groupware applications.

1.7.1 Functional Classification

The first classification system is a functional classification. This group's applications are using their main functional focus – some applications span many of the groups and therefore, a functional classification alone is not sufficient to group these systems.

(i) Electronic Mail and Messaging

These types of applications mainly focus on the communication aspects of group activities; there is typically no support for coordination and collaboration activities. E-mail, Internet Relay Chat (IRC), Instant Messaging (IM), and Wikis are the most prolific examples in this category.

(ii) Electronic Conferencing

Electronic Conferencing systems provide group members with a communication medium where they can simultaneously interact, and share information. These systems facilitate the exchange of information and provide limited collaboration functionalities such as sharing files, creating shared workspaces and whiteboards. Typically, these systems also coordinate group structure, security, and the information sent between collaborators.

Video Conferencing, Electronic Meeting Systems (EMS), Blogging, and Forums are some examples of Electronic Conferencing systems.

(iii) Group Decision Support

Group Decision Support systems are portal-like applications that ensure all collaborating members have access to the same, accurate and most up to date data on a given subject. The focus of Group Decision Support systems is to provide members of a group with a common set of data and information to support the decision making process. Decision Support systems tend to focus more on sharing of information than on the communication between members of the group.

Typical Group Decision Support systems include: shared calendars, time management, project management, and knowledge management systems.

(iv) Group Resource Management

The focus of Group Resource Management systems is to coordinate and manage the collaboration activities of shared resources. This includes indexing, searching, and distributing shared resources, as well as providing history, versioning, and changing management functionalities for these resources.

Version Control (CVS), source code management, change management, and collaborative authoring tools are common examples of shared resource management applications.

(v) Workflow

Workflow systems are tasked with the management of people, tasks, and artefacts within a business process. Workflow applications are a very particular type of applications which have successfully been implemented in many scenarios.

1.7.2 Time/Space Classification

To further categorize CSCW systems, Ellis, et al, (1991) presented the now famous time/ space taxonomy as can be seen in Figure 2. This taxonomy classifies collaboration in terms of time and space and presents four very distinct groups, or methods of collaboration.

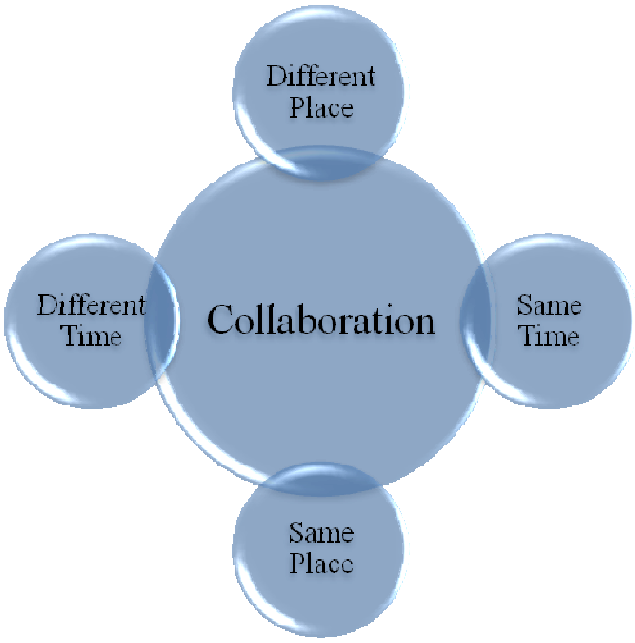

Figure 2 - The time space taxonomy classifies collaboration into 4 distinct groups

Technologies that focus on providing "Same Time" solutions are called *Synchronous*; this is opposed to technologies that support "Different Time" solutions called *Asynchronous*. "Same Place" technologies are often being referred to as being *Co-located*; this is in contrast with "Different Places" technologies which are called *Distributed.* This taxonomy enables us to group CSCW systems into four categories, each with their own set of unique technological challenges.

(i) Synchronous Co-located

This group of collaboration software assists users who are at the same place, and working together at the same time. Typical technical challenges faced during Synchronous co-located collaboration are ensuring safe access to shared resources and propagating data, information, or artefacts in real-time.

(ii) Asynchronous Co-Located

This type of systems enables 'same place – different time' collaboration. Artefact versioning, dependencies, and time related issues are some of the biggest technical challenges faced by this group of CSCW systems.

(iii) Synchronous Distributed

Synchronous distributed collaboration systems support 'different place – same time' collaboration. These systems share the same technical challenges as encountering during synchronous co-located collaboration, with all the added complexity of being distributed such as serialization and connection reliability.

(iv) Asynchronous Distributed

'Different place – different time' collaboration can also be referred to as asynchronous distributed collaboration. This group of CSCW systems shares the technical complexity of all the other groups.

Recent research (Marquès and Navarro, 2005) (Harper, et al, 2001) (Li, 2000) has shown that targeting a single group might cause a system to fail due to human behaviour and group dynamics. Therefore, CSCW research is moving towards creating systems that are more flexible and adaptable towards the time-space constraints of group members.

1.7.3 Architectural Classification

The architectural classification of CSCW systems classifies the systems on how the technology is organized to support the collaboration. We make use of a physical architectural view as defined by ISO/IEC 42010:2007: Systems and software engineering — Recommended practice for architectural description of software-intensive systems (2007). The physical view defines components in terms of their responsibility, physical distribution, and relationship with each other. Most CSCW systems can be classified in one of the following physical architectural classifications:

(i) Central Architectures

A Software system that makes use of a centralized architecture is built on the classic Client/Server network model. Multiple clients are served by a server/server farm.

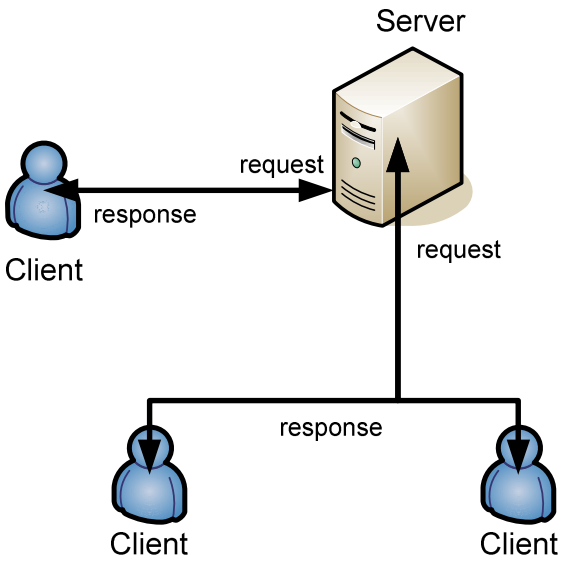

Figure 3 - The classic client/server network topology

Typically, all clients send requests through to a single central server, which then processes the requests and sends back a response.

(ii) Replicated Architectures

Replicated architectures point to a Peer-to-Peer network model where responsibilities and artefacts are replicated amongst peers/nodes in the network.

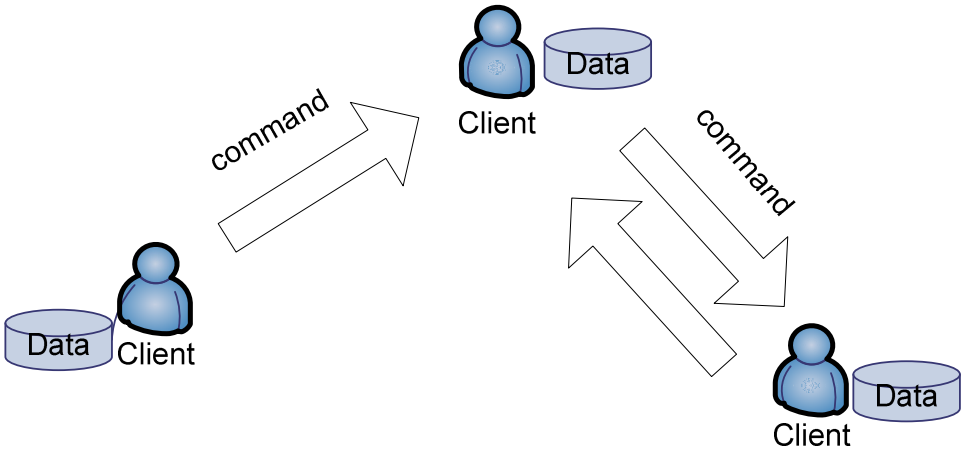

Figure 4 - A P2P network build on the replicated architecture model

Nodes in the network typically send commands to each other — each client is responsible to perform the commands it receives and then send the expected response back to the requesting node.

(iii) Hybrid Architectures

Hybrid architectures make use of both replicated and centralized concepts. Typically, a centralized server is responsible for the control and coordination of tasks and resources which are replicated amongst peers but also backed-up onto the centralized server.

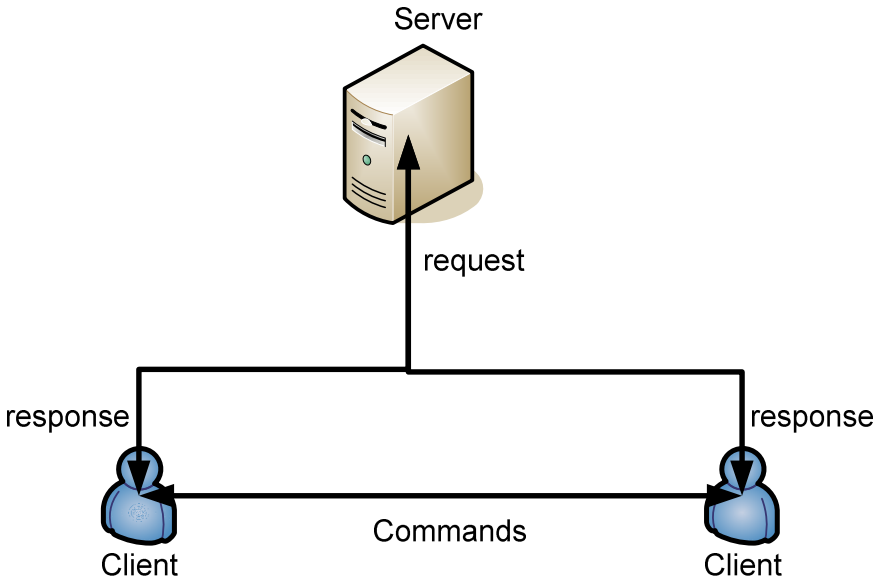

Figure 5 - A P2P network build on the Hybrid architecture model

In Hybrid architecture, communication exists between both the clients and the server. Hybrid architectures are discussed in more details in Chapter 6, where we cover P2P purity.

1.7.4 Focus Classification

Dyson (1990) introduced a classification model to classify CSCW systems depending on where the focus of control resides. The focus classification model splits collaboration into user, work, and processes. A User is the person responsible to perform a specific piece of work that is necessary to complete a process. By evaluating a system in relation to this model, it is easy to understand the functional requirements of the system.

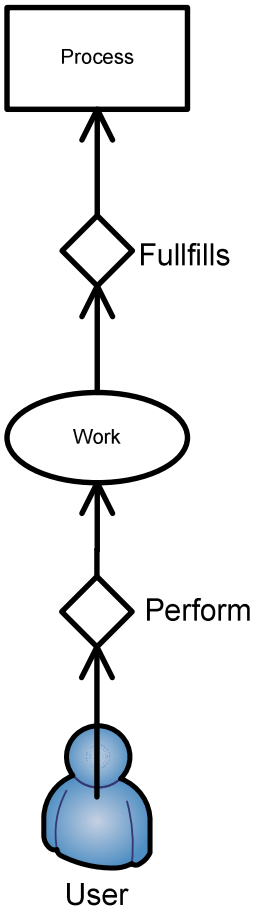

Figure 6 - Shows the relation between a user, work and process

(i) User-Centred

In a user-centred system, the user is seen as the centre of control which sends and receives data, requests, and responses. Collaboration is performed between a user and the outside world.

(ii) Work-Centred

In a work-centred system, the document of work itself is the focus of the collaboration effort. The system needs to control all aspects around the document including document delivery, change management, and version control. Work-centred classification is also known as artefact/object-centred classification.

(iii) Process-Centred

In a process-centred system, the focus of control is on the process itself. The system is tasked with synchronizing tasks between users and artefacts that have an effect on the process at hand.

Chapter 2. A conceptual model for Casually Connected Collaboration

2.1 Introduction

This Chapter provides a model which will enable us to differentiate between casually connected and other collaboration scenarios. The model will focus on the aspects of collaboration that guide the technical qualities and properties of a CSCW system, which will support the scenario. This model will prove invaluable when making technology decisions.

In order to distinguish casual connectivity from other forms of collaboration, we will define a conceptual model that highlights the properties needed to make a clear distinction between the forms of collaboration. Our model will look at the properties of the actors, operations, and environment in which the collaboration takes place. By elaborating on "access, anytime and anywhere" we deduce a model that is sufficient to distinguish between three collaboration models, namely connected, mobile, and Casually Connected Collaboration.

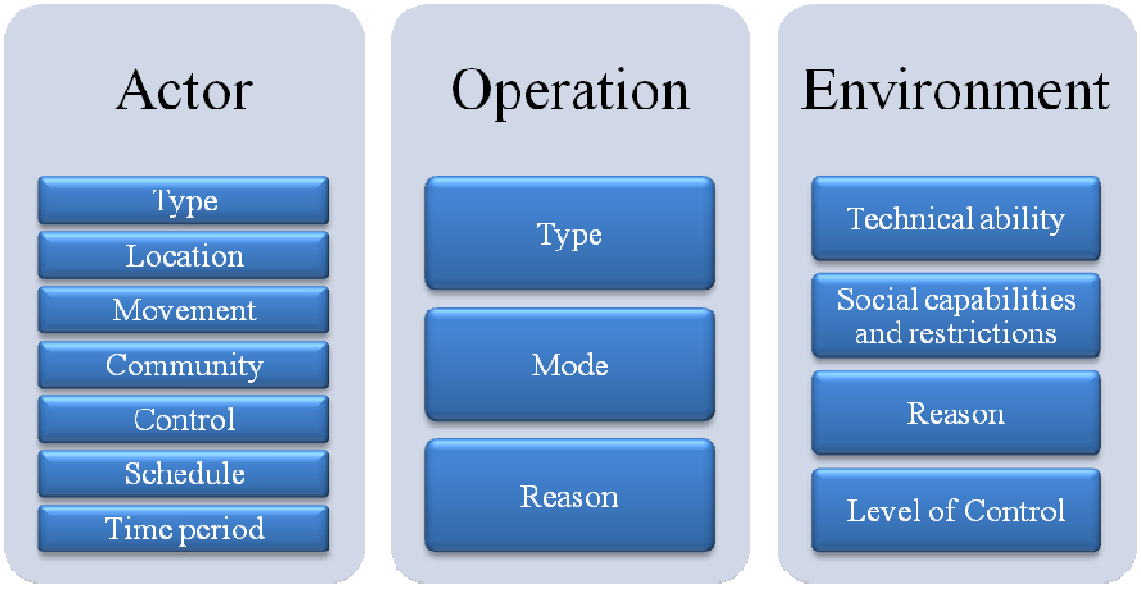

Figure 7 - The conceptual model used to classify methods of collaboration

2.1.1 Actors

Actors are the most important entity in our model of collaboration; it describes the person(s) and/or device(s) performing the collaboration task. In order to distinguish between the various types of collaboration, we have to distinguish between the following properties:

(i) Type

The type of actor defines whether the collaboration activity is performed between:

- People
- Devices
- Both

From a technology perspective it is important to know whether people or devices are involved in the collaboration process. Apart from solving the normal technology issues faced with CSCW system design, people as users, introduce problems highlighted by the field of HCI research (Shackel, 2000). This is in contrast to designing a system that needs to collaborate between devices which also have to be concerned with topical issues in the domain of computer intelligence and the semantic web (Berners-Lee, et al, 2001).

(ii) Location

Location describes the geographic relativity between the actors. They can be:

- Dispersed
- Co-located

In order to support collaboration, between actors, that is geographically dispersed, a CSCW system has to adhere to the non-functional requirements of distributed systems (Emmerich. 2000) – scalability, openness, heterogeneity, resource sharing, and fault tolerance. Support for co-located collaboration faces different obstacles, which have been highlighted in research done on augmented reality (Billinghurst and Kato, 2002) and the fluidity of co-located collaboration tools. (Scott, et al, 2003)

(iii) Movement

Movement describes the tendency of our actors to move from one location to another.

- Roam
- Static

Actors that roam constantly introduce a whole range of technical challenges for a CSCW system designer. Mobility and ubiquitous computing issues (Bardram. 2005) are just two of the many

challenges one has to overcome when dealing with roaming actors. In contrast: when dealing with actors that collaborate from a static or a fixed location, the focus of the CSCW system would be on improving stability or extending existing CSCW functionality.

(iv) Community

Community describes the size and structure of the group of actors.

- Small closed
- Open public

When working with a CSCW system that will be open to the public, one has to emphasize issues around security and scalability. In a closed community, you have the luxury of ignoring a lot of the issues commonly encountered in large open public systems.

(v) Control

Control describes the ability of our actors to control and manipulate the technical ability and status of the collaboration environment. These can be categorized as:

- None
- Limited
- Full

An actor with full control is typically an office worker working on his LAN, he will be able to contact a network administrator when the network goes down, and he will be able to manipulate the environment to give him his desired quality of service. Limited control is, typically, an actor collaborating over his 3G connection while at home, if the network goes down he can phone the service provider, but he is not guaranteed to be online again within desired time. An actor with no-control will have to use what is available. If the network drops, he will have to continue working without network access; this is typical when using publicly accessible WiFi hotspots at airports or cafés.

(vi) Schedule

The schedule of an actor refers to, whether the time he is using to perform a task was planned for or not.

- Scheduled
- Unscheduled

Scheduled time can also be seen as planned time. Working in scheduled time implies that the actor planned for the action, and knew in advance that he would be performing the specific task during a predefined time period. Proper planning can ensure that the equipment an actor needs is at a specific place when needed. A prime example of this is when actors collaborate in a meeting – they can make sure that a video projector, laptop, and a network cable is in the meeting room during the time of the meeting. Unscheduled or dead time is usually time that became available, for which you did not plan any specific task for. Therefore, the actor could not arrange for the correct equipment to be available during this time, and he has to make do with what he has on him. Good examples of dead time are time spent on an airplane, in a waiting room, or after a conference in your hotel room. When an actor performs a task during dead time, he has to be contempt with the equipment and environment on hand.

(vii) Time Period

Time period is used to describe how long an actor will be busy performing his collaboration task. When an actor works in scheduled time he is usually aware of the amount of time available to perform a specific task, and can coordinate his functions in such a way as to ensure that all tasks are performed during the given period. An actor that works in unscheduled time is usually interrupted during his work activity, he should be able to stop working and continue at any given time. The amount of time available to perform a task is not clearly known beforehand.

- Known
- Unknown

2.1.2 Type of operation

Operation is the term used to describe the ability to gain access to specific resources, or to perform a particular function. Having access to my bank account infers that I can withdraw money, make deposits, or view my bank balance. Three groups of operations that CSCW systems need to provide are identified by Ellis, et al (1991) as: communication, collaboration, and coordination.

(i) Communication

A very simplified definition of communication is: the ability to transfer messages or information through a communication medium from one actor to another.

The importance of communication in a collaboration environment was once again highlighted by Hua, et al. (2006) during their design of the Cooperative Program Understanding System (CPUS). The authors highlighted the fact that both, formal and informal communication plays a very important aspect when working in a group.

(ii) Collaboration

The functions of collaboration are dependent on the task at hand. But, a collaboration task that is common to most collaborative functions is the ability to have read-access to shared information.

(iii) Coordination

Coordination is an extremely important task in any group activity. Good coordination prevents replication of work and facilitates timely completion of the task at hand. Coordination encapsulates:

- Awareness;
- Event monitoring, logging and propagation;
- Version management; and
- Task progress.

Most CSCW systems provide users with all three operations, but they tend to specialize in just one of them. The type of operation that is supported by the system hugely impacts the technology one might choose. As an example, when the focus is on providing communication, one can choose between text, voice, and video technologies. Collaboration oriented systems move towards shared space and resource technologies, where as coordination oriented systems are more focused on workflow issues.

2.1.3 Mode of operation

We can split the mode of access into synchronous or asynchronous.

(i) Synchronous Access

Synchronous access refers to “same time” or “real time” access. During synchronous access, a user will send a request and wait until the request is fulfilled. If the request cannot be fulfilled at that specific time, the request will be terminated. One of the most common forms of synchronous access is a phone call, when you phone a person both parties have to be able to attend to the phone call at that specific time. If either party is unavailable, the phone call cannot be completed.

(ii) Asynchronous Access

Asynchronous access refers to “different time” access. During asynchronous access a user will send a request, and continue with what he was doing. The user will then be notified when the request has been completed, upon which the desired action will continue. E-mail is a well known method of asynchronous access. A user can send an e-mail any time of day/night and expect a response within reasonable time.

The mode of operation also guides technology decisions, synchronous access technologies are concerned with providing high reliability stable connection, while performing the operation. This is in

contrast to technologies that focus on providing asynchronous access, which might delay delivery of a message in order to find a better or cheaper route of delivery.

2.1.4 Reason for operation

The reason for access is another very important aspect to consider when distinguishing between Casual Connectivity and other forms of connectivity. Many reasons exist for trying to access a specific resource. But, we argue that for purpose of collaboration the reason is dependent on the importance of the access. We use a simple classification of importance:

- High
- Medium
- Low

The reason also guides technology decisions. From experience we have learned that it is not advisable to use bleeding edge technologies in a high risk environment, one tend to go for more mature technologies and leave the experimental technologies to the low importance scenarios, until they have proven themselves.

2.1.5 Environment

With the advent of mobile and wireless technologies, people have started to work at unconventional work places. This paradigm shift has introduced a lot of new opportunities and challenges for CSCW systems. We make a clear distinction between two types of environments, managed and unmanaged. This distinction is made on the ability of the actor to control both social and technical abilities of his environment.

(i) Managed environment

In some places the actor/worker has control over the technical and social capabilities of the environment; we call these *Managed Environments*. In a managed environment the actor can ensure that he has the necessary technical capability that is required to perform his action. Managed environments usually provide the actor with a known level of service; meaning if something goes wrong there is usually some form of technical support available to assist. Home and office spaces are good examples of managed environments.

(ii) Unmanaged environment

Environments where the actor has very little or no control over the social and technical capabilities are called *Unmanaged Environments*. In an unmanaged environment the actor has no guarantee as to what capabilities exist in the environment, and to the quality of service. If for example the Internet

connection at the airport becomes unserviceable, there is very little that the actor can do in order to improve the situation. Coffee shops and airport terminal buildings are good examples of unmanaged environments. The environment describes the physical location in which actors find themselves when they are working

(iii) Technical ability

The technical ability of the environment points to the availability and quality of technical resources at a location; this includes LAN, Backup infrastructure, Internet connectivity, power adapters, etc…

We can classify the technical ability of an environment as follows:

- Unknown
- Known
- Partially known

(iv) Social capabilities and restrictions

The social capabilities and restrictions of an environment determine whether it is socially acceptable to work at a specific location, with the device available to the actor. Whilst it might be socially acceptable for an actor to work on his Laptop computer in a coffee shop, he might find it impractical/unaccepted to work on his laptop while inspecting a building site.

- Unknown
- Known
- Partially known

(v) Reason

Reason describes why an actor is at the specific location.

- Task related
- Non task related

(vi) Level of control

The level of control describes whether the actors have knowledge and control over a location. If the location can be controlled, actors can change the properties of the location to adapt to their collaboration task, if actors cannot manage the properties of a location, then the collaboration task has to be adapted to adhere to the location.

- Managed locations
- Partially managed
- Unmanaged

2.2 Casually Connected Collaboration

2.2.1 Introduction

The aim of a Casually Connected Collaboration system is to provide actors with a best-effort approach towards collaboration operations in an uncertain environment. Actors will typically make use of a Casually Connected Collaboration system during dead-time or unscheduled time whilst travelling, or out of the office.

2.2.2 Actors

During collaboration, actors work together to perform a specific task at hand. Actors in a Casually Connected Collaboration have the following properties:

- ***Type:*** Both

The actors in a Casually Connected Collaboration environment can be either people or machines.

- ***Location:*** Dispersed

Actors are, typically, geographically dispersed from one another.

- ***Movement:*** Roam

The actors have a tendency to roam.

- ***Community:*** Small closed

Actors collaborate in a relatively small closed community.

- ***Control:*** None

The actor has no control over the connection between him and his collaborators. The actor has limited control over his own network connection.

- ***Schedule***: Unscheduled

The actor performs operations in unscheduled/dead time; we can make no assumption as to when the actor will perform a specific operation.

- ***Time Period***: Unknown

The actor has an unknown and limited amount of time available to perform his operations, we cannot assume that an actor will be online for long enough to complete a specific operation.

2.2.3 Operations

Operations can also be defined as the “access” that actors have during their collaboration effort. The operations that can be performed during Casually Connected Collaboration have the following properties:

- ***Type:*** Communication and Collaboration

All three types of operations; communication, collaboration, and coordination can be performed during Casually Connected Collaboration, but it tends to be more communication and collaboration functions.

- ***Mode:*** Asynchronous

Asynchronous operations are the only feasible mode of operation, supported by Casually Connected Collaboration.

- ***Reason:*** Low importance

The reason for performing an operation is usually of low importance, if the operation could not be performed immediately, a best-effort solution will suffice for the time being. Alternate arrangements can be made, or the actor can wait until the operation has completed successfully.

We can summarize, by saying, that casual connectivity only supports asynchronous low importance operations.

2.2.4 Environment

The environment describes the physical location in which actors find themselves when they are working.

- ***Technical ability:*** Unknown

No assumptions can be made with regards to the availability and quality of the technical abilities provided by the environment.

- ***Social capabilities and restrictions:*** Unknown

No assumptions can be made with regards to the social capabilities or restrictions of the environment of our actors.

- ***Reason:*** Non task related

The reason for our actors to be on a specific environment has nothing to do with the task at hand.

- ***Level of Control:*** Unmanaged

Due to the fact that actors roam, we cannot make any assumptions as to the whereabouts of actors. We can assume that our actors will make an effort to obtain the most reliable Internet connection available to them when, and if, they need to perform a collaborative operation.

2.2.5 Scenario: Authoring a Book

The classic example is that of multiple authors working on the same book/article/document. The idea here is that there are multiple “specialists” distributed all over the world, working on one deliverable.

These people are highly mobile, living mostly out of their suitcases and working in hotel rooms. Each worker is responsible for their own part of the deliverable, but they need to be able to view other people's work to ensure that the deliverable forms a unity. From research done on Cooperative document editing (Mendoza-Chapa, et al. 2000.), we known the following facts about people collaborating during document authoring:

- There is no guarantee as to the connection status of each worker. For example: some hotel rooms in rural Italy might not have Internet access.
- A best-effort view of the document is acceptable; the workers don't have to have "the" latest version of the document.
- The Specialists "roam" between multiple connection points as they go around their daily tasks.
- Some collaborators might prefer to "travel light" and only pack their tablet PCs instead of their normal desktop replacement notebooks.

•

	Connected Collaboration	Mobile Collaboration	Casually Connected Collaboration
Actor			
Type	Both	People	Both
Location	Dispersed	Co-located	Dispersed
Movement	Static	Roam	Roam
Community	Open public	Small Closed	Small Closed
Control	Full	Limited	None
Schedule	Scheduled	Scheduled	Unscheduled
Time Period	Known	Known	Unknown
Operation			
Type	Communication	Collaboration	Communication
	Collaboration		Collaboration
	Coordination		
Mode	Synchronous	Asynchronous	Asynchronous
	Asynchronous		
Reason	High importance	High importance	Low importance
Environment			
Technical Ability	Known	Partially known	Unknown
Social Ability	Known	Partially known	Unknown
Reason	Task related	Task related	Non task related
Level of Control	Managed	Partially managed	Unmanaged

Table 1 - Document authoring

Although document authoring shares properties with mobile collaboration, it is clear that document authoring can be classified as a Casually Connected Collaboration task.

2.2.6 Scenario: Scientific research project.

A non-profit scientific research organization is doing research on the behavioural trends of chimpanzees, and wants to improve the collaboration between scientists. Some of the scientists are situated in the remote areas of the African bush, while others are doing research in high-tech laboratories. The financial support organisation wants to be able to view the current state of the deliverables at any given time.

2.2.7 Collaboration classification

- Researchers might make use of mobile devices to capture and store data.
- Scientists will work together to compile reports.
- We can introduce a Data Mining server, which takes the results from the field researchers and identifies trends. These reports will then be viewed, updated, and validated by the field researchers.

- Different Media types will be shared. Field researchers might capture the chimps on video while lab researchers might draw graphs and write long detailed reports.

	Connected Collaboration	Mobile Collaboration	Casually Connected Collaboration
Actor			
Type	Both	People	Both
Location	Dispersed	Co-located	Dispersed
Movement	Static	Roam	Roam
Community	Open public	Small Closed	Small Closed
Control	Full	Limited	None
Schedule	Scheduled	Scheduled	Unscheduled
Time Period	Known	Known	Unknown
Operation			
Type	Communication	Collaboration	Communication
	Collaboration		Collaboration
	Coordination		
Mode	Synchronous	Asynchronous	Asynchronous
	Asynchronous		
Reason	High importance	High importance	Low importance
Environment			
Technical Ability	Known	Partially known	Unknown
Social Ability	Known	Partially known	Unknown
Reason	Task related	Task related	Non task related
Level of Control	Managed	Partially managed	Unmanaged

Table 2 - Scientific research project

The scientific research team might be best off using a Casually Connected Collaboration system.

2.3 Connected Collaboration

2.3.1 Introduction

The aim of a connected collaboration system is to provide actors reliable collaboration operations in a managed environment. Actors will typically have access to fast and reliable network connections and other fixed office resources. In the event of a technical failure, the actors will be able to resolve the failure themselves or have access to an admin that can resolve the failure. One can also assume the actors will be available during a scheduled/pre determined time period, usually office hours.

2.3.2 Actors

Actors in a connected collaboration system, typically, have the following properties:

- ***Type***: Both

  The actors can be people or machines.

- ***Location***: Dispersed

They can be geographically dispersed but are located at fixed locations.

- ***Movement***: Static

Actors are, predominantly, static. Actors might roam between known fixed locations, but they usually perform the same task at the same position, therefore, we can argue that relative to the task they perform, they are static.

- ***Community***: Open public

The actors can collaborate in a community of any size.

- ***Control***: Full

They have full control over the technical abilities of their environment. The actor has limited control over the connection between him and his collaborators.

- ***Schedule***: Scheduled

The actor, typically, performs actions in a scheduled/predetermined time-period.

- ***Time Period:*** Known

The actor has a limited, but known, time period available to perform an operation.

2.3.3 Operation

The operations that can be performed during connected collaboration have the following properties:

- ***Type***: Communication, collaboration, and coordination

All three types of operation; communication, collaboration, and coordination can be performed during connected collaboration.

- ***Mode***: Synchronous and Asynchronous

Both, synchronous and asynchronous, operations are feasible modes of operation in a connected collaboration environment.

- ***Reason***: High importance

The reasons for performing the collaboration function are usually of high importance, and a reliable solution is required after an operation has been performed.

2.3.4 Environment

A connected collaboration environment has the following properties:

- ***Technical ability***: Known

The availability and quality of the technical abilities provided by the environment are known and controlled.

- ***Social capabilities and restrictions***: Known

The social capabilities and restrictions of the environment are known.

- ***Reason***: Task related

The reason that the actor is at a given location, typically, means that he is there to perform the given task.

- ***Level of Control:*** Managed location

Actors reside at known, fixed locations; therefore, the environment is managed and controlled.

2.3.5 Scenario: Student administration

In a complex environment such as the University, multiple lecturers and tutors are responsible for giving marks and comments on a students' progress. Management of the University has received complaints, from various students, that the marking of assignments and projects takes too long. The Management wants to view the progress of marking, at any given time, to be able to identify the bottleneck in the marking process. Each marker is required to mark a small amount of students' work and fill these marks into a spreadsheet.

Student administration typically has the following properties:

- Because of the tedious nature of marking other people's work, the markers might decide to work in coffee shops. They will, therefore, only take their PDAs along to enter in the marks.
- Some markers – especially the tutors, do not have 24-hour Internet connection, and only go online once or twice a day.
- Management wants to be able to draw a mark sheet, at any given time, which will show the latest results of students for a given subject.

	Connected Collaboration	Mobile Collaboration	Casually Connected Collaboration
Actor			
Type	Both	People	Both
Location	Dispersed	Co-located	Dispersed
Movement	Static	Roam	Roam
Community	Small Closed	Small Closed	Small Closed
	Open public		
Control	Full	Limited	None
Schedule	Scheduled	Scheduled	Unscheduled
Time Period	Known	Known	Unknown
Operation			
Type	Communication	Collaboration	Communication
	Collaboration		Collaboration
	Coordination		
Mode	Synchronous	Asynchronous	Asynchronous
	Asynchronous		
Reason	High importance	High importance	Low importance
Environment			
Technical Ability	Known	Partially known	Unknown
Social Ability	Known	Partially known	Unknown
Reason	Task related	Task related	Non task related
Level of Control	Managed	Partially managed	Unmanaged

Table 1 - Student administration

Student administration can be classified as best suited by a Connected Collaboration system.

2.3.6 Scenario: Software development

A lot of software development companies want to improve their efficiency by working 24 hours a day. This can be achieved by employing programmers from all over the world different time zones. Multiple programmers will work on the same software component and will require the most up to date version of their component.

2.3.7 Collaboration classification

- Programmers can be distributed not only by space, but also by Time.
- It is critical that all programmers work on the same/latest piece of code available.
- Programming tasks must be coordinated sufficiently.

	Connected Collaboration	Mobile Collaboration	Casually Connected Collaboration
Actor			
Type	Both	People	Both
Location	Dispersed	Co-located	Dispersed
Movement	Static	Roam	Roam
Community	Open public	Small Closed	Small Closed
Control	Full	Limited	None
Schedule	Scheduled	Scheduled	Unscheduled
Time Period	Known	Known	Unknown
Operation			
Type	Communication	Collaboration	Communication
	Collaboration		Collaboration
	Coordination		
Mode	Synchronous	Asynchronous	Asynchronous
	Asynchronous		
Reason	High importance	High importance	Low importance
Environment			
Technical Ability	Known	Partially known	Unknown
Social Ability	Known	Partially known	Unknown
Reason	Task related	Task related	Non task related
Level of Control	Managed	Partially managed	Unmanaged

Table 4 - Software development

The task of software development is best suited by a Connected Collaboration system.

2.4 On-site/Mobile Collaboration

2.4.1 Introduction

The aim of on-site/mobile collaboration systems is to provide systems that will support collaboration operations, while the actors are on the site where the work needs to be performed. On-site collaboration systems, typically, make use of mobile devices such as PDAs, cell phones, and tabletPCs to give the actors access whilst at a specific geographic location.

2.4.2 Actors

The following properties holds true for actors in an on-site collaboration system:

- ***Type:*** People

The actors are typically people.

- ***Location:*** Co-located

The actors are typically co-located but at unknown/unmanaged locations. Co-located actors often require access to resources that are not at the same physical location.

- ***Movement:*** Roam

The actors roam between unmanaged locations.

- ***Community:*** Small closed

The actors collaborate in small, closed communities.

- ***Control:*** Limited

The actor has limited control over the connection between him and his collaborators. The amount of control available can improve dramatically with proper planning – the actors know where he will go and where he will collaborate.

- ***Schedule:*** Scheduled

The actor typically performs the action at a scheduled/predetermined time-period.

- ***Time Period:*** Known

The actor has a known, but limited, time period available to perform the action.

2.4.3 Operation

Operations that are performed in an on-site collaboration system have the following properties:

- ***Type:*** Collaboration

On-site collaboration systems tend to focus on the collaboration functions; due to the fact that collaborators are typically co-located, communication and coordination are usually verbal.

- ***Mode:*** Asynchronous

Mostly, asynchronous operations are performed during on-site collaboration.

- ***Reason:*** High importance

The reason for performing the operation is usually of high importance, thus a reliable solution is required after an operation has been executed.

2.4.4 Environment

The environment of an on-site collaboration system has the following properties:

- ***Technical ability:*** Partially known

The availability and quality of the technical abilities provided by the environment can partially be controlled.

- ***Social capabilities and restrictions:*** Partially known

The social capabilities and restrictions of the environment are partially known. The system has to deal with environmental challenges such as Micro-mobility.

- ***Reason:*** Task related

The reason that the actor is at a given location, typically means that he is there to perform the given task.

- ***Level of Control:*** Partially managed

Actors roam between known, but possibly unmanaged, locations. The environment can be managed to a certain degree through proper planning.

2.4.5 Scenario: Route management

A large Parcel Delivery Service company wants to be able to draw reports at any time of the day to show the progress of deliveries and pickups. These reports can be used by them to re-route some of the drivers. The company makes use of freelance drivers and is therefore not allowed to install tracking devices into the vehicles. When a driver arrives at a stop, he must capture the signature of the client on his PDA. The mobile device will then give him driving directions to his next stop.

- Drivers can only connect to the Internet via Wireless Internet, such as GPRS. They might roam into spots that do not have any connectivity options.
- Management requires a best-effort view on the progress of deliveries from all drivers.
- Drivers use Mobile Devices to capture and store their own progress.

	Connected Collaboration	Mobile Collaboration	Casually Connected Collaboration
Actor			
Type	Both	People	Both
Location	Dispersed	Co-located	Dispersed
Movement	Static	Roam	Roam
Community	Open public	Small Closed	Small Closed
Control	Full	Limited	None
Schedule	Scheduled	Scheduled	Unscheduled
Time Period	Known	Known	Unknown
Operation			
Type	Communication	Collaboration	Communication
	Collaboration		Collaboration
	Coordination		
Mode	Synchronous	Asynchronous	Asynchronous
	Asynchronous		
Reason	High importance	High importance	Low importance
Environment			
Technical Ability	Known	Partially known	Unknown
Social Ability	Known	Partially known	Unknown
Reason	Task related	Task related	Non task related
Level of Control	Managed	Partially managed	Unmanaged

Table 5 - Route management

Route management will be best suited by a Mobile Collaboration system.

2.5 Summary

Table 1, presents the summary of our model which can be used to distinguish between connected, mobile, and Casually Connected Collaboration.

	Connected Collaboration	Mobile Collaboration	Casually Connected Collaboration
Actor			
Type	Both	People	Both
Location	Dispersed	Co-located	Dispersed
Movement	Static	Roam	Roam
Community	Open public	Small Closed	Small Closed
Control	Full	Limited	None
Schedule	Scheduled	Scheduled	Unscheduled
Time Period	Known	Known	Unknown
Operation			
Type	Communication	Collaboration	Communication
	Collaboration		Collaboration
	Coordination		
Mode	Synchronous	Asynchronous	Asynchronous
	Asynchronous		
Reason	High importance	High importance	Low importance
Environment			
Technical Ability	Known	Partially known	Unknown
Social Ability	Known	Partially known	Unknown
Reason	Task related	Task related	Non task related
Level of Control	Managed	Partially managed	Unmanaged

Table 6 - Summary of types of collaboration

Furthermore, we have successfully grouped scenarios into a collaboration classification, and identified the technical concerns applicable to each scenario. These technical concerns can be summarized in the table below.

	Application Scenario	Technical concerns
Connected Collaboration	Software development. Student administration.	Reliability. Speed. Scalability.
Mobile Collaboration	Route management.	Security. Connectivity. Human computer interfacing.
Casually Connected Collaboration	Document authoring. Scientific research project.	Reduced replication. Ease of use. Best effort approach.

Table 7 - Technical restrictions of collaboration scenarios

Chapter 3. Nomad

3.1 Introduction

Nomad is a framework for a Distributed Resource Management system, with special emphasis placed on the accessibility of information stored on detachable devices, such as flash-disks, PDAs, Pocket PCs, laptops, and personal computers. Nomad is intended to address the emerging problem of information spread and neglect, where users have the same copy of the same work stored on various conventional devices, but no way to keep track of where the most up-to-date version resides. Nomad does not restrict itself to personal computing environments, but is also intended to be used amongst members in a project group. Unlike conventional centralized repository-based information sharing applications, Nomad is intended to be an integrated behind-the-scenes operator. Information gathering is only performed on request from a user (a pull-based-system), while information sharing on any of the interacting devices is always active; as long as that device remains on and connected to the network. We can argue that the main purpose of Nomad is to enable a team of geographically dispersed users to work on the same project; editing the same documents, pictures, code, and other artefacts whilst Nomad will automatically, and transparently, collect all the artefacts needed to compile the most up-to-date project or section thereof available. Centralized repository based systems require the repository or server to be reachable at all times. Nomad distinguishes itself by taking a best-effort approach to gathering information. Instead of relying on the availability of a single server, it hunts for all available nodes, and then gathers as much information from them as possible. In a typical environment, featuring mobile devices, many of those devices might be switched off, or disconnected. This cannot be helped, but Nomad can still report on which devices could, or could not be reached, and what the results are.

We define Nomad as a collaborative groupware that enables effortless Casually Connected Collaboration within a group of collaborators. Nomad focuses on artefact distribution and management, hiding the complexities of the nomadic environment from group members. What sets Nomad apart from most currently available artefact management groupware is the pull-based hunter-gatherer framework for artefact collection. Users of Nomad are considered to be casually connected collaborators. These are people who roam geographically, and typically have incomplete knowledge of their collaborators' connectivity status. Casually connected collaborators are dependent on team members but are not bound to them – best-effort results are acceptable in the event of a collaborator being off-line. These collaborators typically operate within a small closed community, working together towards a common goal by means of electronic communication.

A crucial element of Nomad is the hunter-gatherer framework; this component can be classified as collaborative middleware. Collaborative middleware is defined as the set of services that support the interchange of artefacts independent of artefact definition, network topology, or application. The primary tasks of the middleware are to provide artefact description, discovering, and distribution services to users. The Hunter-Gatherer Framework is designed to provide best-effort artefact collection to casually connected collaborators. It relies on a non-centralized, pull-based architecture for collecting and distributing artefacts.

3.2 Hunter and Gatherer

Hunter-Gatherer is a framework that supports pull-based artefact collection. This means that artefacts are collected from various sources only when they are requested/in need. A benefit of this approach is that members sharing these artefacts do not have to worry about getting their pieces to the person who requested the artefacts. The drawback behind this approach is that the framework can only deliver a best-effort result. Another important aspect of the Hunter-Gatherer Framework is that it is built on a non-centralized architecture. This implies that the system is not dependant on a single node, and can execute with only a single node connected. A node can be defined as a system with processing power that has been registered to be used in a Nomad project. The node can contain multiple storage resources (that it makes available at its own discretion) to form part of the distributed repository for the project. Each node also has a specified set of permissions to govern additions, updates, and removals of artefacts.

The framework can be separated into two main subsystems:

Hunter

Hunter is the distributed application that searches (hunts) for files on various registered machines also called Nodes. The application resides on each node in the network and then it hunts for the most up-to-date version of a project, or section thereof, available on all connected nodes. It accomplishes this by sending a shopping-list-file to all available nodes. When a node receives its shopping list it kicks of the gatherer process. When all the available nodes complete their gathering process, hunter receives all the artefacts, and compiles a most-up-to-date version of the project available at that moment.

Gatherer

Gatherer is an application that runs on a local node; it is typically invoked by a call from a remote hunter. Gatherer will, after receiving a shopping list from hunter, search all the storage resources the node has chosen to make available for the sharing. All files found are then gathered together into the correct structure, and sent to the hunter.

3.3 Requirements

The scenario of co-authoring a book or grant proposal, was the instigator of the Nomad project. Compiling a grant proposal is usually a joint effort between researchers of various geographically dispersed research institutions. These grant proposals are normally compiled in parallel, with normal daily tasks, and are therefore pressed for time. Researchers tend to do a lot of travelling, spending majority of time in hotels and airports with a questionable level of connectivity. All these factors make it extremely difficult to collaborate with team members, and more often than not, a lot of collaborative efforts are lost. Currently, e-mail is used as the primary means of collaborating in this environment, but this introduces the tedious and error-prone task to manually merging multiple documents. Another problem with e-mail is that mailboxes are usually cluttered with other not-as-important messages and attachments, and a collaborative document could easily be overlooked or lost.

Nomad was designed to assist these researchers by collecting and distributing artefacts. A researcher will continue to work on his/her pre-assigned task without worrying about distributing it to other members. When a team member requests a version of a document, Nomad will go and collect the latest version of the artefact it can find. Nomad will then compose an artefact and report on pieces that could not be collected, and collaborators that could not be found. Should a collaborator not be available, Nomad will attempt to collect the artefact as soon as it comes available. The purpose of Nomad in this scenario is to assist in the distributing and assembling of the artefact.

As one can see from these requirements, and taking previous Chapters into consideration, this is a typical Casually Connected Collaboration scenario; collaboration between people within a group where members might work at different times of the day and at different locations. Their work might even be spread over multiple devices with different capabilities and processing power. The work performed by a casually connected collaborator is usually not highly dependent – one member can continue with his work for an extended period of time without requiring any input from another collaborator.

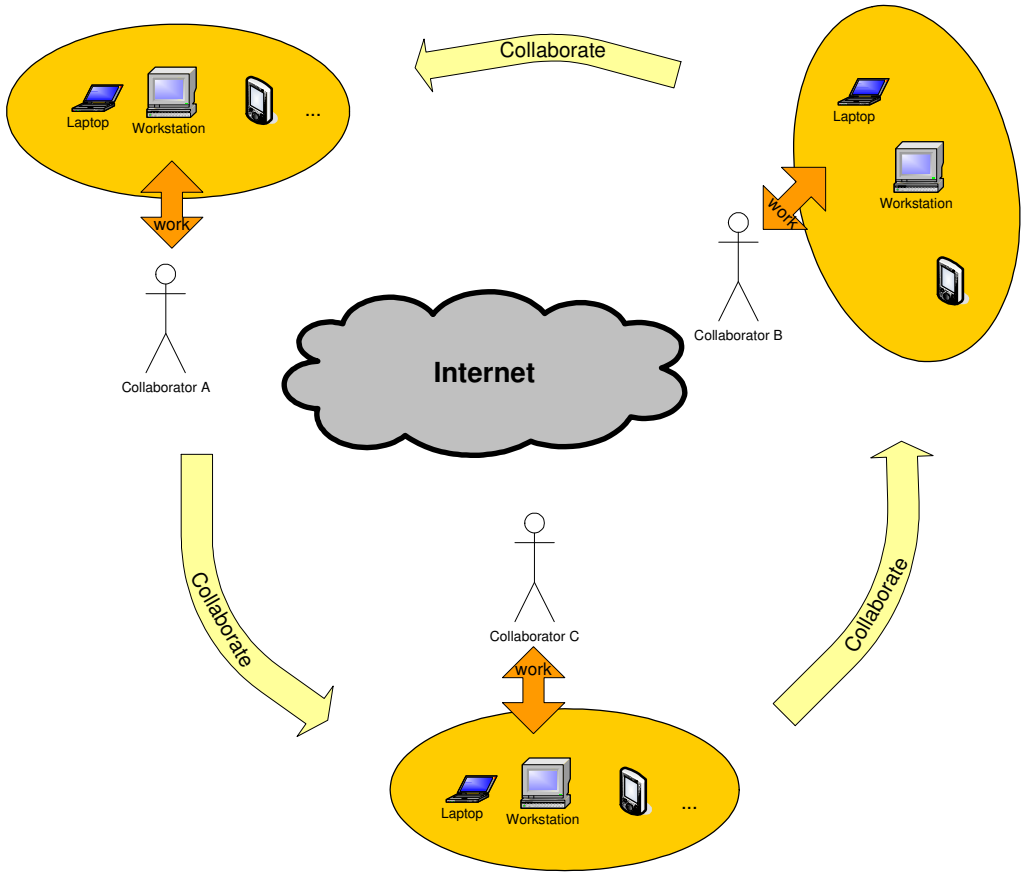

Figure 8 - A high level model of the collaboration environment

Figure 8 shows a high level model of the collaboration environment. The following aspects are worth mentioning when looking at the model:

(i) Collaboration activities occur between people

Collaborators do not know the internals of the people they collaborate with. Collaboration occurs between people. A user request an artefact from a person, this artefact can reside on any of the collaborators nodes – the physical location of the artefact is hidden from the requester. His only knowledge is from whom the artefact must be retrieved.

(ii) Work is independent of time or place

No correlation can be made between the time and place at which collaborators perform their work. Collaborators can be geographically separated and can work on different schedules; we can argue that work is independent of time or place.

(iii) Multiple devices might be used to finish a specific task

Modern information workers have, in general, a whole arsenal of technological devices at their disposal. Each device specializes in performing a specific task or portion of the work at hand. These devices range in technical ability and can be anything from a PDA, Cell phone, notebook, or PC.

(iv) Interconnectivity between team members is unreliable

Due to the fact that our collaborators are casually connected, no assumptions towards the technical ability of the connection between collaborators can be made. We can assume the worst, and argue that the connection between our collaborators (when available) is unreliable, at best.

(v) Workers want to follow their normal process of work

People are familiar with existing applications and ways of performing a specific task. They are reluctant to change the way in which they perform these tasks.

(vi) No centralized server

Due to the fact that actors, in a Casually Connected Collaboration system, do not have control over their location and its technical abilities – the use of centralized server architecture is not optimal. In a centralized server architecture users are required to have access to a central server. Casually connected users cannot guarantee that they will ever have access to a central server.

3.4 Functional Requirements

Figure 9 highlights the major functional requirements, required in any collaboration framework.

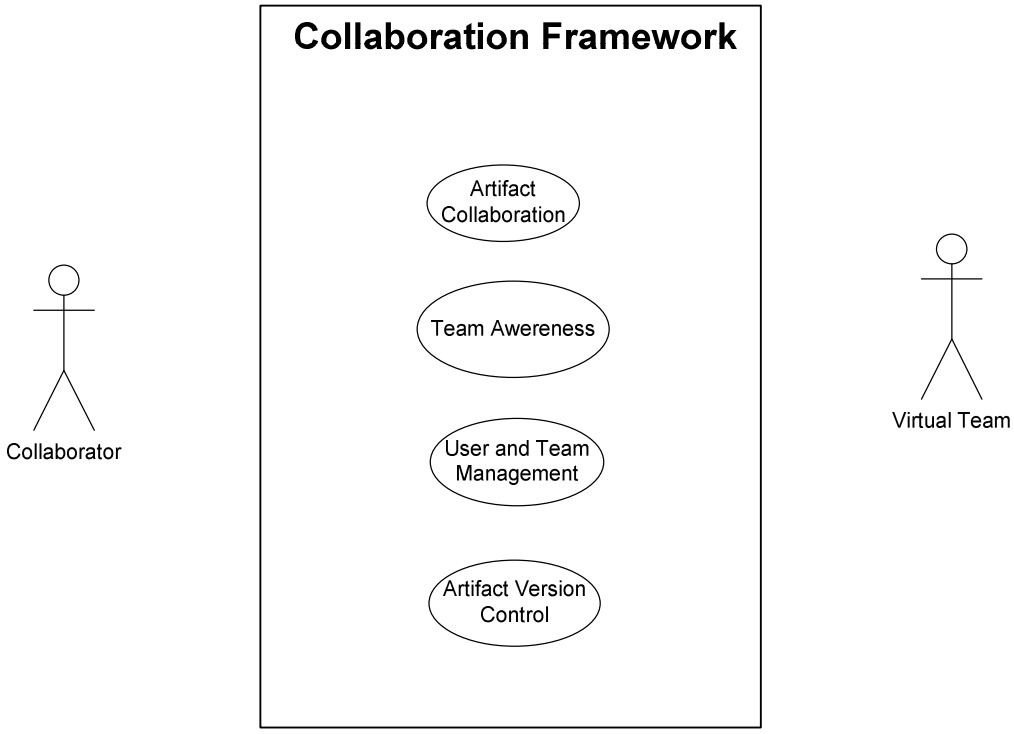

Figure 9 - The functional requirements of any collaboration system

3.4.1 Artefact collaboration

Artefact collaboration can be described as the task of distributing and sharing of an artefact between multiple collaborators.

This task can be broken into:

- Locating a collaborator;
- Request list of available artefacts;
- Request details of an artefact; and
- Pull a copy of an artefact.

3.4.2 Team awareness

Creating awareness amongst team members entails:

- Member status (Online/Offline);

- Task oriented notifications; and
- Informal communication/messaging.

3.4.3 User and team management

- Create/Edit/Delete users and its details; and
- Create/Edit/Delete teams and their details.

3.4.4 Artefact version control

- Add/Edit/Delete an Artefact;
- Keep track of available versions of an artefact; and
- Keep track of which collaborator has which version of an artefact.

3.5 Non-functional Requirements

Figure 10 highlights the major non-functional requirements, required in a Casually Connected Collaboration framework. Non-functional requirements are dictated by the environment of the system.

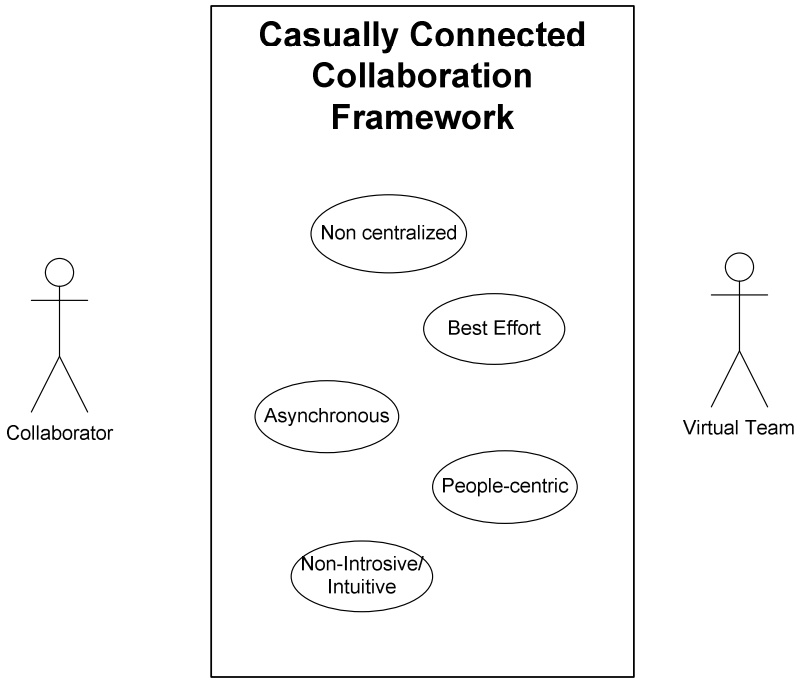

Figure 10 - The non-functional requirements for a Casually Connected Collaboration framework

3.5.1 Decentralized

- Servers can be introduced to improve reliability, but they should act as peers in the network.
- Artefacts are distributed amongst the users of the network.
- All collaboration, communication, and coordination tasks of the system should be independent of a centralized server.

3.5.2 Best-Effort approach

- Users must be able to get the best version currently available.
- A best-effort approach implies that the user can pull the latest version of all the artefacts he requested which are currently available on the network.
- If an artefact is not available, it will be pushed to the user as soon as it becomes available.

- Requests can be propagated through other peers until the artefact eventually reaches its target.

3.5.3 Asynchronous

- The framework should allow asynchronous communication.
- The framework should alert a user once a request has been completed.
- The framework should allow the user to cancel asynchronous requests that have expired.

3.5.4 People-centric

- The complexity of the network should be hidden from the users.
- Users should feel as if they are collaborating with other users and not with devices/drivers.

3.5.5 Non-intrusive/Intuitive to use

- The framework must enable users to work without having to change their normal way of working.
- The user must be able to continue with his work independent of his/her connectivity status.
- The framework must be able to give clear indication of the status of a user and the artefacts available on the network.

3.6 Protocol

The Nomad protocol is essentially a combination between information push-and-pull protocols also referred to as the *Hunter-Gatherer Framework*. Nomad does not perform automatic artefact synchronization and updates, these are only sent to users on request - they are "pulled" to the user. If a user is not currently online, the requesting user subscribes to a service that will "push" the information to him when it becomes available. Figure 11 highlights the process of artefact retrieval as described by the Nomad protocol.

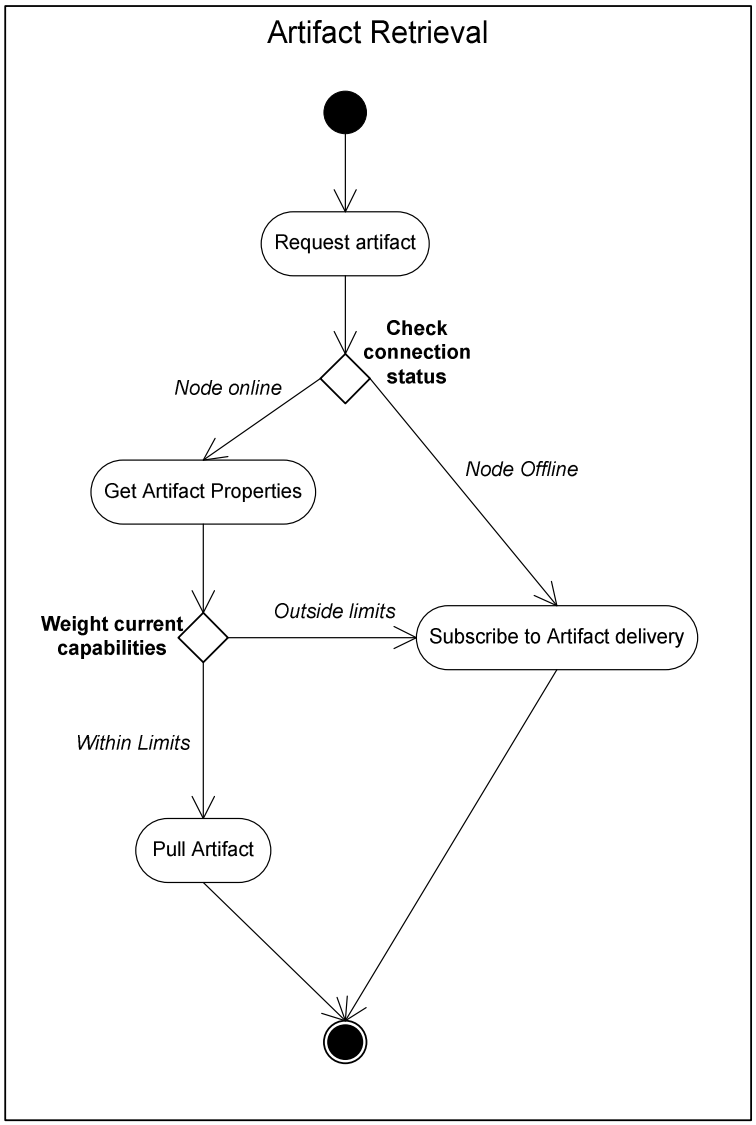

Figure 11 - The artefact retrieval process as defined by the Nomad Protocol

Request Artefact.

When a user initiates the *request artefact* action, Nomad will launch the hunter process to compile a list of all artefacts needed to build the requested, compound artefact. Hunter will compile this list using metadata that describes the dependencies between artefacts. This list will then form the shopping list passed onto all other nodes.

Check connection status.

Hunter will then try to contact all registered nodes. The first step in contacting the nodes is to check its connectivity status. If a node cannot be reached it is assumed to be offline and the hunter will

launch the *subscribe to artefact delivery* action, which will "push" the file to hunter when the node becomes available. If the node is online, then it will launch the *get artefact properties* action.

Get artefact Properties.

The *get artefact properties* action is used by hunter to determine whether it wants to download a specific artefact or not. The protocol does not limit or restrict the properties stored, and they will, typically, differ per implementation. These properties are stored on each node as Metadata; it describes the type of artefact, its size, and various other attributes.

Weigh current capabilities.

This step uses the *artefact properties* and *connection status* to determine whether it wants to download the artefact now, or wait until a better more reliable connection is available. If the current capabilities are outside of predefined threshold then hunter can decide to *subscribe artefact delivery* instead of resuming the download immediately. This step allows users to restrict downloading when they are on expensive high bandwidth connections such as 3G or HSDPA connections. It also gives users the flexibility to restrict downloads when bandwidth are too low or during peak, network time. The implementation of the weigh function is not defined by the protocol and will, typically, be different in each implementation of Nomad.

Subscribe to Artefact delivery.

When hunter decides to *subscribe to artefact delivery*, the artefact will be delivered to the user as soon as the requesting node comes online or as soon as the result of the weigh function falls within the desired threshold. In the scenario where the source node (the node we want to retrieve the artefact from) is offline or not available, the subscription will then be propagated to all other available nodes in the network. As soon as the source node comes online the subscription will be delivered to it, and it will initiate the gatherer process which will deliver the artefact to the requesting node. If the requesting node cannot be found, the artefact will be distributed to all available nodes in the network which will satisfy the weigh function. These surrogate nodes will then be used to propagate the artefact through to the requesting node. This enables us to deliver an artefact from one node to another, even if never online at the same time.

Pull Artefact.

The *pull artefact* is the action that physically downloads the artefact from the source node to the requesting node. An artefact can either be "pulled" directly from the source node, or – if the same artefact is available at multiple nodes then more advanced algorithms, such as the ones used by the Bittorrent P2P File-Sharing System (Pouwelse, et al, 2005) – it can be employed to "pull" the artefact from multiple nodes in the network to the destination or requesting node.

3.7 Comparing Nomad

To narrow down the number of systems/projects compared, we used the functional requirements as specified earlier in the chapter, and only included projects that met the following requirements:

- Systems that work within closed communities and small groups.
- Systems that focus on providing the three basic CSCW functionalities, namely: communication, collaboration, and coordination (Ellis, et al, 1991).
- All systems execute in a distributed environment using common network protocols such as TCP and HTTP, with the Internet as backbone.

3.7.1 Commercial

Novell Open Enterprise Server (Novell iFolder 3.6: Novell Open Enterprise Server, 2007)

The aim of Novell Open Enterprise Server is to provide an easy way to back up, synchronize, share, and access all your files from anywhere at anytime. Files are automatically backed up to your Novell Server, and changes are synchronized to all your workstations – this automatic propagation of changes enables multiple users to work on the same file simultaneously by ensuring that they always work on the latest version.

Microsoft Office Groove (Groove Home Page - Microsoft Office Online, 2007)

Microsoft Office Groove facilitates communication, collaboration, and coordination amongst members of a small group. Groove is aimed at providing a set of tools to group members who are regularly offline, and do not share network settings and capabilities. The tools provided by Groove include: calendars, discussion boards, file sharing, outliner, notepad, sketchpad, etc…

SubEthaEdit (SubEthaEdit, 2008)

SubEthaEdit is a powerful collaborative text editing tool that enables members of a group to simultaneously edit a text document. The collaboration is document-centric and any member of the group can edit the document at any time without blocking the access to the document for other users.

3.7.2 Academic

LaCOLLA (Marquès, et al, 2007)

LaCOLLA is a middleware that enables group members to perform collaboration functions by using only their own resources. This self-organization of the network removes the dependency from central regimes, and lets the group control the resources that are available.

CoCoDoc (ter Hofte and van der Lugt, 1997)

CoCoDoc provides compound document editing functionalities to facilitate members of a group to collaborate through editing the same document. The focus of CoCoDoc is to provide rich text-editor functionalities together with rich collaboration functions.

Basic Support for Cooperative Work (Appelt. 1999)

BSCW aims to support cooperation amongst team members through shared workspaces. The shared workspaces make use of existing Web technologies and are accessible from standard web browsers.

3.8 Functional Criteria

Groupware applications can be described by its functional criteria. The Functional Criteria specify the functionalities a user can expect of the system regardless of its environmental/ non-functional constraints.

3.8.1 Messaging

Messaging provide users the functionality to communicate via synchronous and asynchronous messages. Synchronous Messaging systems are Instant Messaging systems (IRC, MSM) while asynchronous are mostly E-mail–like applications.

3.8.2 Conferencing and Electronic Meeting Systems (EMS)

Conferencing and Electronic Meeting systems provide users with a shared communication channel/interface/workspace where they can work/talk/share simultaneously

3.8.3 Group Decision Support

Group Decision Support systems are portal-like applications that ensure all collaborating users have access to the same, accurate, and most up-to-date data on a given subject.

3.8.4 Document Management

Document Management systems provide features such as indexing, searching, and distributing documents to authorized users.

3.8.5 Document Collaboration

Document Collaboration systems include Document Management functionalities, and extend them by providing history, versioning, and change management.

3.8.6 Compound Document Management

Compound Document Management systems provide the functionality to see one document as a collection/combination of smaller documents; it allows users to view the single merged/compound version of the document

	Messaging	Conferencing & EMS	Group Decision Support	Document Management	Document Collaboration	Compound Document Management
Nomad				X	X	X
Novell Open Enterprise Server				X		
SubEthaEdit	X	X		X	X	
Microsoft Office Groove	X			X	X	
CoCoDoc				X	X	X
Basic Support for Collaborative Work (BSCW)	X	X		X	X	

Table 8 - Comparison of CSCW systems on functional criteria

3.9 Architectural Criteria

The Architectural Criteria of Groupware application defines where and how the collaboration is managed.

3.9.1 Central Architecture

Collaboration is managed at a central server. All data is exchanged via a central point of access. This architecture relates directly to client–server architecture.

3.9.2 Replicated Architectures

Collaboration is managed by all the peers in the network. Data and information are exchanged between peers, and all peers are equal in such a network. This architecture relates to pure Peer-to-Peer architectures.

3.9.3 Hybrid Architecture

The Hybrid Architecture can be seen as a Peer-To-Peer architecture where some nodes are more important than other nodes. Data and information are shared among the peers, but there exist a master peer(s) that can override information received from peers, or they can guide peers to other nodes. This architecture relates to Peer-to-Peer networks with Supernodes.

	Central	Replicated	Hybrid
Nomad			X
Novell Open Enterprise Server		X	
SubEthaEdit		X	
Microsoft Office Groove			X
CoCoDoc	X		
Basic Support for Collaborative Work (BSCW)	X		

Table 9 - Comparison of CSCW systems on architectural criteria

3.10 Focus Criteria

Focus area criteria defines the focal point of collaboration. This means that all collaborative tasks are centred on this particular area, and that no collaboration is possible without the focus criteria being present.

3.10.1 User-Centred

Collaborative tasks focus on the user - this implies that the user is the most important aspect in User-Centred Collaboration. User-centred groupware creates a communication channel between collaborating users; the groupware is not interested in what the users do with the channel.

3.10.2 Artefact-Centred

All collaborative tasks focus on the artefact. Artefact-Centred groupware provide methods to collaborate on a specific artefact. The groupware will, typically, store information with regards to the structure and history of the artefact. The communication channel between users is transparent.

3.10.3 Workspace-Centred

Workspace-Centred groupware can be seen as an extension to User-Centred groupware with the exception that a workspace can exist without users. The workspace can store state and in this way allows asynchronous User-Centred Collaboration. Collaborative users share the same workspace.

	User-Centred	Artefact-Centred	Workspace-Centred
Nomad		X	
Novell Open Enterprise Server		X	
SubEthaEdit	X		
Microsoft Office Groove		X	
CoCoDoc			X
Basic Support for Collaborative Work (BSCW)			X

Table 10 - Comparison of CSCW systems on focus criteria

3.11 Time Criteria

Defines the restrictions placed on the time of collaboration.

3.11.1 Synchronized

Collaboration must happen in a structured manner at the same time. Synchronized groupware will handle locking and collisions detection in real-time.

3.11.2 Unsynchronized

Collaboration can happen totally unsynchronized, without any interaction between the various collaborators. Groupware supports people working together; completely separate from each other. Collaboration only kicks in when requested from a user, otherwise all work performed does not affect other collaborating users.

3.11.3 Mixed (Synchronized & Unsynchronized)

Collaboration can be either synchronized or unsynchronized. The application supports both types of synchronization activities.

3.11.4 Serial

Serial Collaborations are unsynchronized with the exception that one user must perform a specific task before another user can continue with another task. E-mail is a classical example of Serial Collaboration.

	Synchronized	Mixed	Serial	Unsynchronized
Nomad				X
Novell Open Enterprise Server				X
SubEthaEdit	X			
Microsoft Office Groove				X
CoCoDoc	X			
Basic Support for Collaborative Work (BSCW)	X			

Table 11 - Comparison of CSCW systems on time criteria

3.12 User involvement Criteria

Defines the level of involvement required from the user to gain advantages provided by the groupware.

3.12.1 High

High level of user involvement means that the user is forced to work with a different interface that he is used to, in order to access the collaboration functionalities. This is typical to shared workspace environments.

3.12.2 Medium

Medium level of user involvement implies that users can work with their normal user interfaces, and only need to execute collaborative commands at any given time.

3.12.3 Low

Low level of user involvement means that the user is only involved in setting up the collaboration environment, and can then continue to work as if he is not collaborating. All collaboration functions are automated and transparent to the user.

	Low	Medium	High
Nomad	X		
Novell Open Enterprise Server	X		
SubEthaEdit			X
Microsoft Office Groove		X	
CoCoDoc	X		
Basic Support for Collaborative Work (BSCW)	X		

Table 12 - Comparison of CSCW systems on user involvement criteria

3.13 Final Comparison

This Section shows how Nomad matches up against the other projects.

	Function	Architecture	Focus	Time	User Involvement
Nomad	Compound document management, Document Collaboration, Document Management	Hybrid	Artefact	Unsynchronized	Low
Novell Open Enterprise Server	Document Management	Replicated	Artefact	Unsynchronized	Low
SubEthaEdit	Messaging, Conference and EMS, Document Management, Document Collaboration	Replicated	User	Synchronized	High
Microsoft Office Groove	Messaging, Document Management, Document Collaboration	Hybrid	Artefact	Unsynchronized	Medium
CoCoDoc	Compound document management, Document Collaboration, Document Management	Central	Workspace	Synchronized	High
Basic Support for Collaborative Work (BSCW)	Messaging, Conference and EMS, Document Management, Document Collaboration	Central	Workspace	Synchronized	High

Table 13 - Summary of comparison between academic and commercial CSCW systems

3.14 Conclusions

Even though existing systems may provide similar functionality to Nomad, on a non-functional level, none of them have sufficient support for collaboration in a casually connected environment. Both centralized and replicated architecture systems imply that the collaborator must have a managed environment; on the hand of the model devised in Chapter 2 we can argue that this is not suitable for Casually Connected Collaboration. Workspace-oriented systems also require users to alter the way in which they work, and this is also not desirable for a casually connected environment. Non-functional requirements are the main driver behind technological and architectural decisions, we can thus argue that there is a need for technological innovation to support Casually Connected Collaboration.

Chapter 4. Technology decisions

4.1 Introduction

In this Chapter we present the user with a more in-depth view on the implementation details around a system such as Nomad. We highlight the system architecture, network organization, and enabling technologies found in the Microsoft .NET Framework version 3.5 that we used during implementation.

4.2 Technology Environment

In previous Chapters we have already described the functional aspects of a casually connected environment. Although these have a significant impact on the technology decisions and system architecture, we also have to look at the technological environment in which the collaborators will

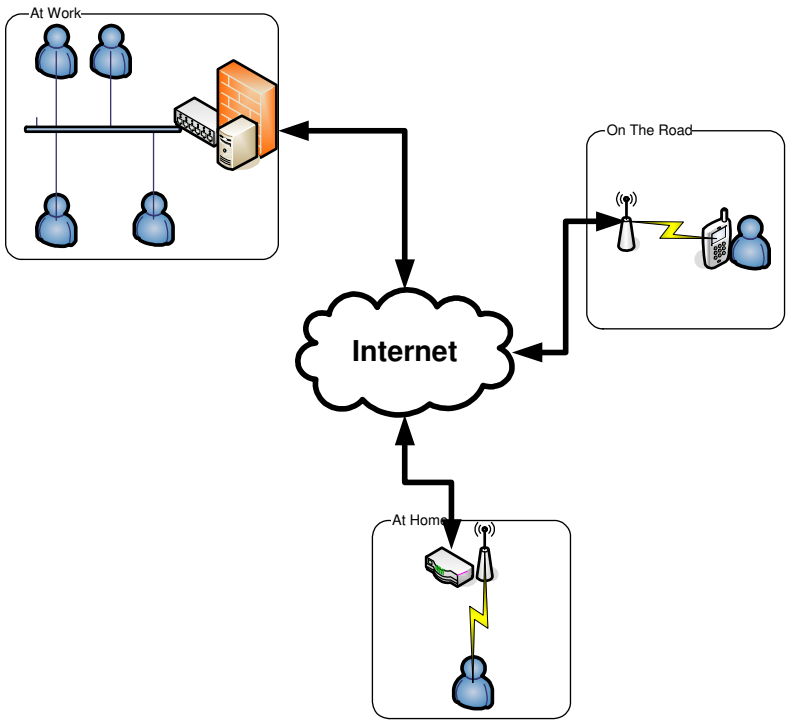

Figure 12 - The technology environment of casually connected collaborators

find themselves. The following technological devices have a significant impact on system design:

4.2.1 The Internet Address Space

The Internet is a worldwide, publicly accessible network of interconnected computer networks that transmit data using the standard Internet Protocol (IP). A core requirement of IP is the ability to uniquely identify or address each node in the network. In the classical view of the Internet, each node had a globally unique IP address which can be used directly by other nodes for purpose of communication. This classic view of the Internet Address Space has made way to the architecture of a global address realm and many private address realms interconnected by Network Address Translators (Ford, et al, 2005).

4.2.2 Network Address Translation

With the explosion of the number of Internet users, many concerns have been raised with regards to exhaustion of the IP address space, IPv4 only has 32bit addresses, and this was one of the motivation factors behind IPv6 which supports 128bit addresses. Another technology (supported in IPv4), Network Address Translation (NAT), became popular as a method of preserving the address space. The idea behind NAT is that all hosts on the Internet do not need globally unique addresses, some hosts can be assigned private/locally unique addresses (Wikipedia, 2007). These hosts are free to communicate with hosts within their private/local network context, but when they want to communicate with a host on the outside/global network a Network Address Translator (NAT router) needs to translate their private IP address into a globally unique address. This allows a NAT router to use a single Global IP address for many local IP addresses. As clearly stated by Peterson and Davie (2000), the biggest drawback of NAT is that it breaks one of the key assumptions of the IP service model – that every node has a globally unique address.

This is perhaps better explained with an example scenario:

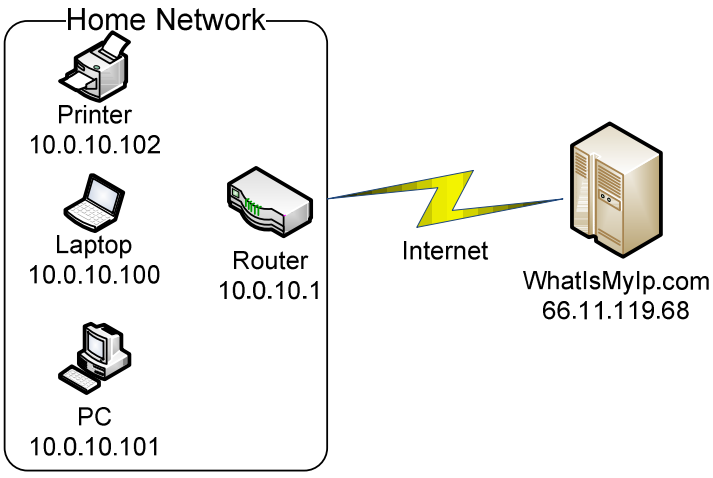

Figure 13 - An Example of a home-network connected to the Internet and the effect of NAT

When I am connected to the Internet from home, my PC gets an IP address of 10.0.10.100, this IP address enables me to see all network devices on my Local Area network, including other PCs, printers, and my ADSL Router. The router is responsible to route all traffic intended for the external network onto the Internet. When I query my IP address on the Internet through a site like WhatIsMyIP.com (2007), I get an IP address of 41.241.141.142. My router performs the Network Address Translation between me and WhatIsMyIP.com – all network packets they sent to 41.241.141.142 will be translated by my router as network packets for 10.0.10.100, and vice versa. When I access the Internet from my other PC with an internal IP address of 10.0.10.101, WhatIsMyIp.com will return the same IP address, this is because my NAT router only has one external IP to work with, and will forward the response to the correct internal IP address. In essence all PCs in my internal network are sharing a single external IP when connecting to the Internet.

4.2.3 Port Forwarding

As previously mentioned NAT shares a single IP address with multiple internal nodes. This greatly reduces the total number of IP addresses being used on the Internet, but it comes at a price. Nodes behind a NAT router cannot be accessed directly by external nodes. To overcome this problem a technique called Port Forwarding has been developed to allow incoming requests made on a specific port to be forwarded to an internal network node. This technique is commonly used in setting up Peer-to-Peer networks or exposing a Web server at home. A drawback of Port Forwarding is that it has to be setup on the NAT router. This approach is definitely workable in a connected/managed environment where the user has access and control over the resources available to connect him on the Internet. In Casually Connected environment Port Forwarding is not an option because it assumes you have access to the router and the knowledge to setup and configure Port Forwarding.

4.2.4 Firewalls

A Firewall is a network node that restricts access to other networked resources. Firewalls are common in most managed environments, and serve as the first line of defence against malicious attacks from other Internet nodes. Although firewall configuration can be a particularly complicated topic, the most basic of firewall settings tend to limit the open ports and allowed protocols being accessed by external network resources.

4.2.5 Consequences of environment

The effect of this environment on system design is as follows:

- Collaborators cannot be accessed directly through their IP addresses as seen by the outside world, because this IP address will be one that has been dynamically assigned by a DHCP or NAT server.

- The system must make use of standard ports and protocols, otherwise Firewalls will not allow access.

- Port Forwarding is not an option because of the Nomadic nature of the users we do not have control over the NAT routers.

4.3 Peer-to-peer networks

4.3.1 Definition

Peer-to-Peer networks can be defined as the technology that enables two or more network nodes to collaborate directly with each other without the necessity for central coordination (Schoder and Fischbach, 2003). This is in contrast with a client–server network topology where a large number of clients are being served from a small number of servers (Danzfuss, 2003).

Client–Server architecture is the classical network model, and has proved to be very successful in current implementations such as the WWW (Berners-Lee, et al, 1994). Some of the drawbacks introduced by Client/Server technology are highlighted by Singh (2001) as:

- Performance bottlenecks.
- The overall performance of a system is dependant on the load and performance off the server. Various techniques of Cluster computing around redundancy and replication can improve the scalability of Client/Server architecture, but these are often very expensive solutions.
- Single-point of failure.
- The system as a whole is dependant on the reliability and availability of the central server.
- Client control.
- In a classic Client/Server model, a client makes a request and receives a response from the server. The client decides when to make request and what type of request to make.
- Independent interaction.

- Clients interact with the server independent of other clients. This isolation that exists between clients is not ideal for a collaborative environment.

Peer-to-Peer technology moves away from this centralized approach towards distributed network control. Even though Peer-to-Peer networks are not a new idea, it is only recent file-sharing networks such as Napster (2007), Gnutella (2007), and KaZaA (2007) that brought P2P technology back into the spotlight. The modern day success of P2P networks is attributed to the low cost and high availability of computing power, and the increase in reliable network connectivity (Ripeanu, et al 2002). Some of the potential advantages that Peer-to-Peer brings to the table as presented by Danzfuss (2003) and Clark (2001) are:

- Dynamic operability.
- Applications using P2P technologies can operate even though peers join and leave the network frequently.
- Scalability.
- The performance of a P2P Application is not limited by the capacity of a single server. Responsibilities on the network can be shared between more peers as the network grows.
- Network value.
- Resources on the network are shared between peers. The more peers there are on the network, the more resources the network have on offer.
- Fault tolerance,
- Redundancy can be introduced by replicating resources amongst peers, this increases fault tolerance and availability, eliminating the single point of failure found in client/server architectures.
- Content based addressing.
- Resource addressing in Peer-to-peer models are typically independent of physical location and more geared towards content.

The important distinction to recognize between client/server and P2P models is the purpose or the functions of each node in the network. In a P2P model, nodes are considered equal in function (peers) whilst in a client/server model a node can either be a user (client) or a producer (server) node.

4.3.2 P2P Justification

As described by Oram (2001) the key driver behind peer-to-peer technology is to empower users to collaborate in the producing and consuming of information; he argues that the information is owned and managed by the nodes in the network, and not by a single centralized entity. Furthermore, collaboration in a casually connected environment builds on top of all the principals underlying P2P systems (Aberer and Hauswirth, 2002).

- Sharing of resources
- Decentralization
- Self-organization

We do however recognize that a lot of collaboration solutions available today make use of client/server technologies. Solutions such as Google Documents (Google) and Microsoft Live Mesh (Microsoft) aim to provide users '"access, anytime and anywhere"'.

Up until May 1999, almost all downloads from the Internet were done from centralized file servers with the FTP or HTTP protocol. Napster (2007), a Peer-to-peer file sharing client, changed the face of file sharing on the Internet forever. Peer-to-peer technologies such as Torrents are now the de-facto standard when it comes to downloading and sharing files over the web.

We argue that collaboration technologies on the Internet can follow the same evolutionary trend as file and content sharing over the Internet has done.

4.3.3 P2P Purity

Peer-to-Peer purity is the term used to describe the various degrees of centralization/distribution of services within a P2P network.

Hybrid peer-to-peer networks, such as Napster (2007), do rely on central servers to offer some of the required network services. Even though peers are served by a central server, they still collaborate and share on an equal level amongst themselves.

In pure peer-to-peer networks, Gnutella (2007) and Freenet (2007) as example, all network functions are equally distributed amongst the nodes; any arbitrary node in the network can be removed or replaced without the network suffering a loss of network service (Schollmeier, 2002).

Super-peer P2P networks such as KaZaA (2007) present a cross between pure and hybrid P2P networks. A super-peer network can be seen as a layered P2P network where all the nodes on a specific layer share responsibility for a function, and are considered as being equal peers. Yang and Garcia-Molina (2003) defines a super-peer as a node that acts as both a centralized server to a subset of clients, and as a peer amongst other super-nodes. Super-peer networks are considered extremely

effective, because they combine the efficiency found in Hybrid and Client/Server architectures with the autonomy, load balancing, and robustness of pure P2P architectures.

4.4 Network topology for a casually connected environment

From the discussions above it is clear that a peer-to-peer network topology is naturally more suitable for collaboration than a client/server model. Especially with qualities such as dynamic operability, content based addressing, and fault tolerance that P2P brings to the table which are highly sought after in a Casually Connected Collaboration framework as discussed in Chapter 3.

It is only when you start to analyse the restrictions imposed by NAT routers, firewalls, and other environmental constraints, as discussed in Chapter 7, that one can recognize the problems that might occur when implementing a P2P solution for casually connected environments.

After considering these issues we have decided to implement Nomad on a Super-peer P2P network topology. Our implementation will consist of 2 layers:

1. Relay and network layer.
   The purpose of the relay layer is to implement relaying techniques as discussed by Ford, et al (2005) to overcome the restrictions imposed by firewalls and NAT.

2. Collaboration layer.
   The collaboration layer will consist of the physical peers interacting and collaborating using the Nomad protocol as defined by Rama and Bishop (2006).

4.5 Application Architecture

Nomad adheres to a layered architectural approach. Each application layer is responsible for a specific set of tasks, and it communicates with the layer below it through well defined interfaces. Figure 14 shows the basic building blocks of Nomad.

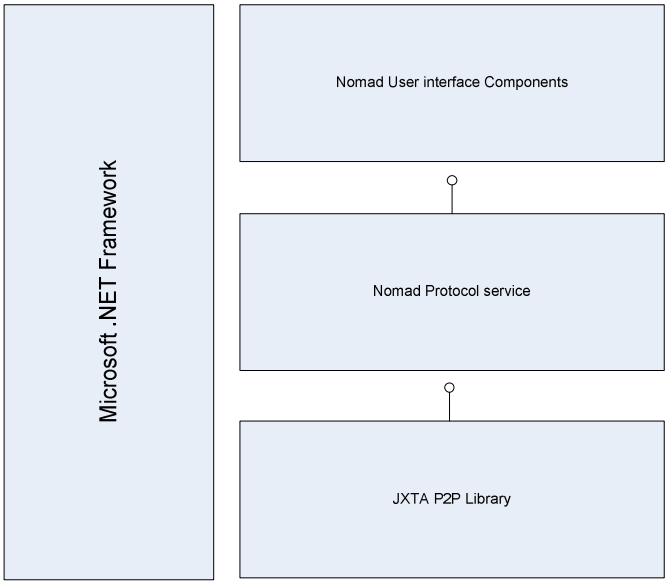

Figure 14 - The basic building blocks of Nomad

Furthermore, Nomad makes use of the Microsoft .NET framework on all levels of the implementation to provide it with a stable set of tools and libraries that significantly simplify the implementation.

4.5.1 Nomad User interface components

The Nomad user interface is built using Windows Forms technology in Microsoft .NET Framework. Windows Forms technology provides a stable platform for developing high-performance easy to use Windows applications.

The Nomad user interface was built around the Model View Controller design pattern. This design pattern can clearly be seen when looking at Figure 15. Figure 15 represents a subset of the User Interface component Classes - only classes necessary for project and artefact were included.

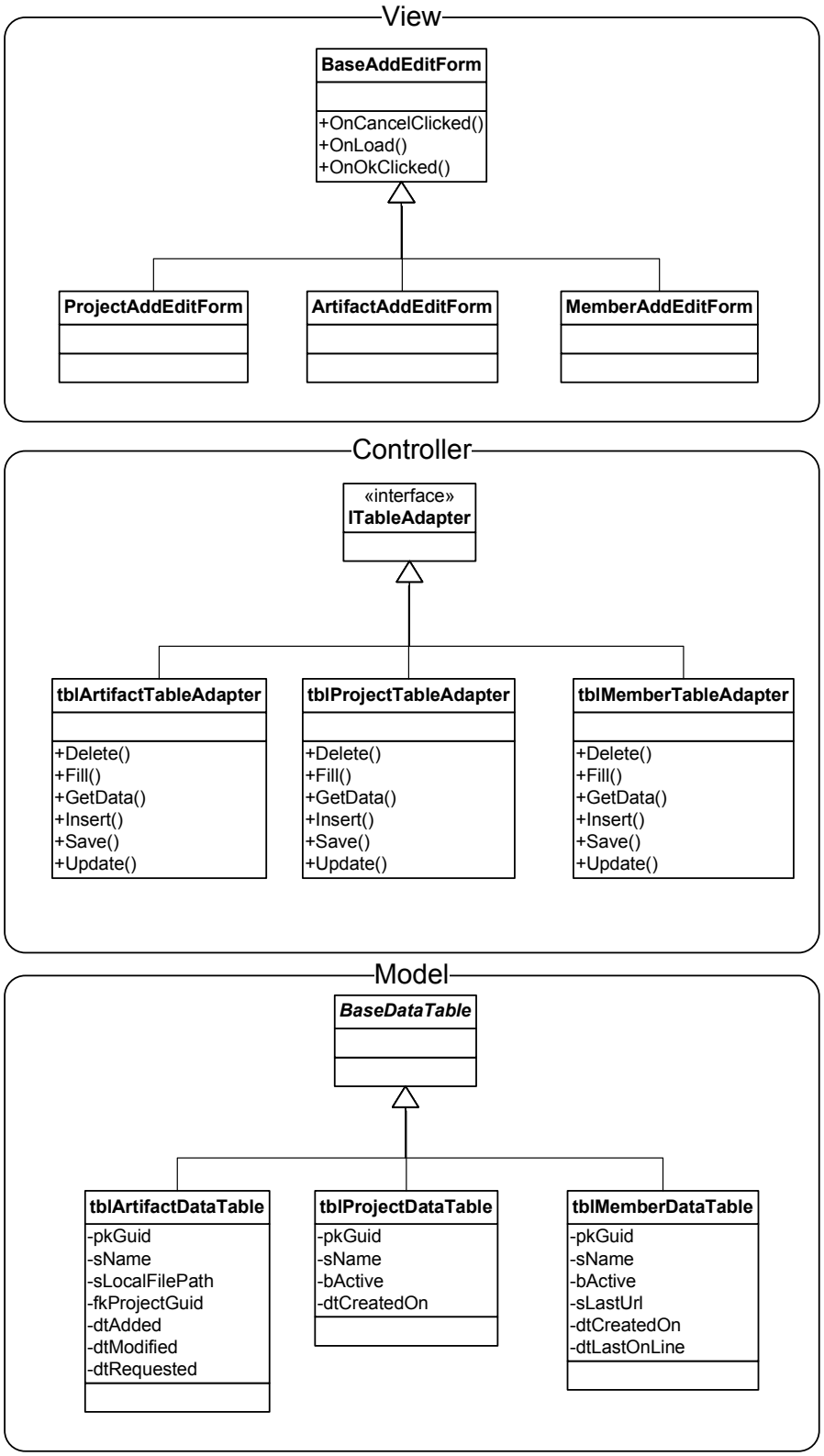

Figure 15 - Class diagrams for the Nomad user interface components

All the View components notify the Controllers of user interaction, Controller components are then responsible for performing a task using the underlying model, and then update the View components again.

The Nomad User Interface was designed with simplicity in mind. Figure 16 shows the basic screen layout of the Nomad application.

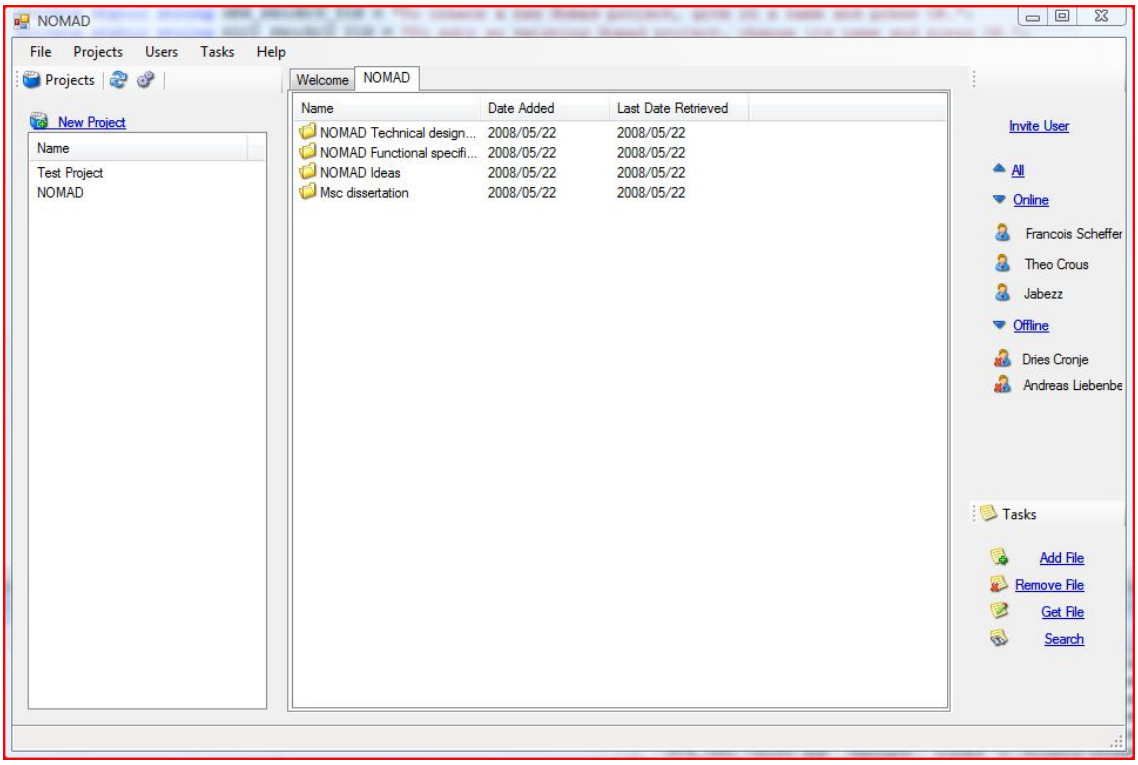

Figure 16 – The basic screen layout of Nomad

The left hand navigation pane allows you to manage your projects, while the left hand pane allows you to manage your collaborators and artefacts. The main workspace allows you to view and retrieve your artefacts per project. Your list of collaborators is filtered on the active project.

Adding new users or artefacts to the active project is very simple, as can be seen from Figure 17 and Figure 18.

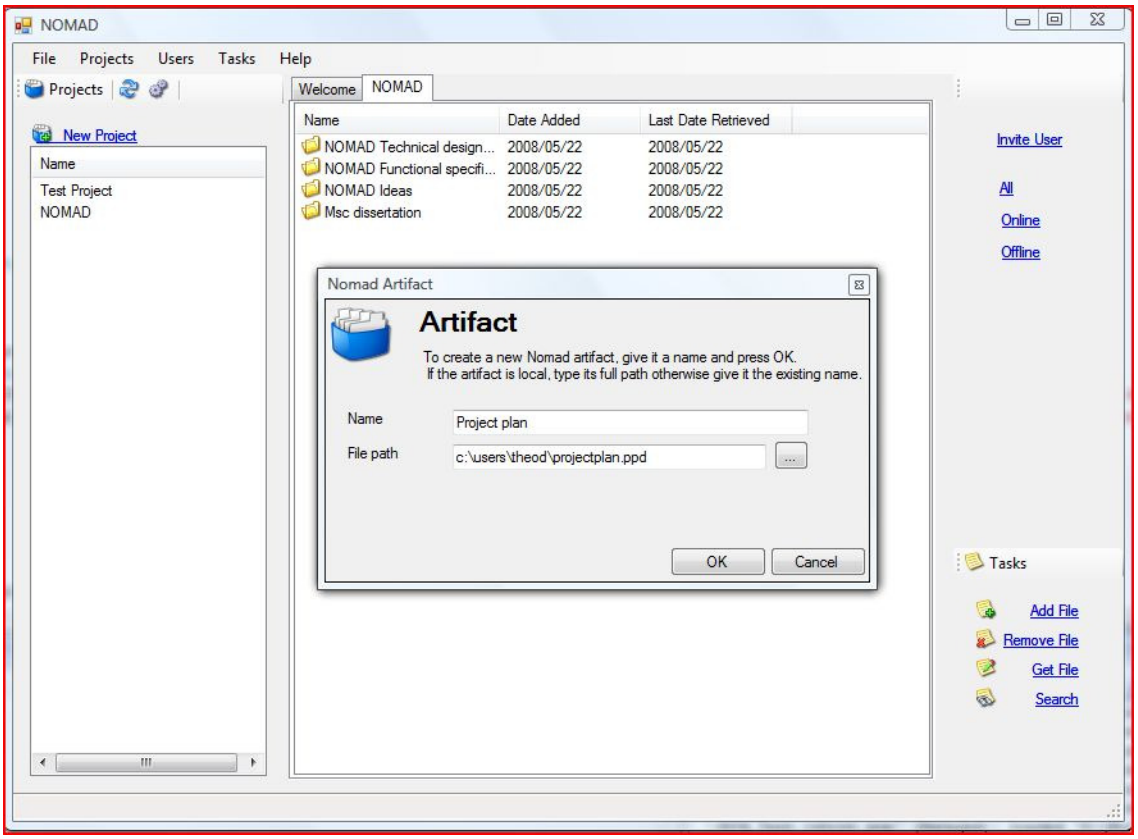

Figure 17 - Screenshot of registering an artefact in Nomad

When a new Artefact is created on your Nomad, you automatically become the owner of the artefact and you have to specify the current local file path of the artefact. This information will be propagated to all your collaborators, and they will have the ability to retrieve the artefact from you. Any file can be registered in your Nomad repository as long as the application has access to the file.

Figure 18 shows how a new user is registered in your Nomad repository. After a user is registered in the application, the user will be invited to join the Nomad community via e-mail. The e-mail will contain a link to the website where the latest version of Nomad can be downloaded, this link will be tailored for the specific user, and once he has downloaded and installed the application he will have all the information needed to collaborate on the project for which he was invited to join. Figure 19 shows the customized website where the user will be redirected to.

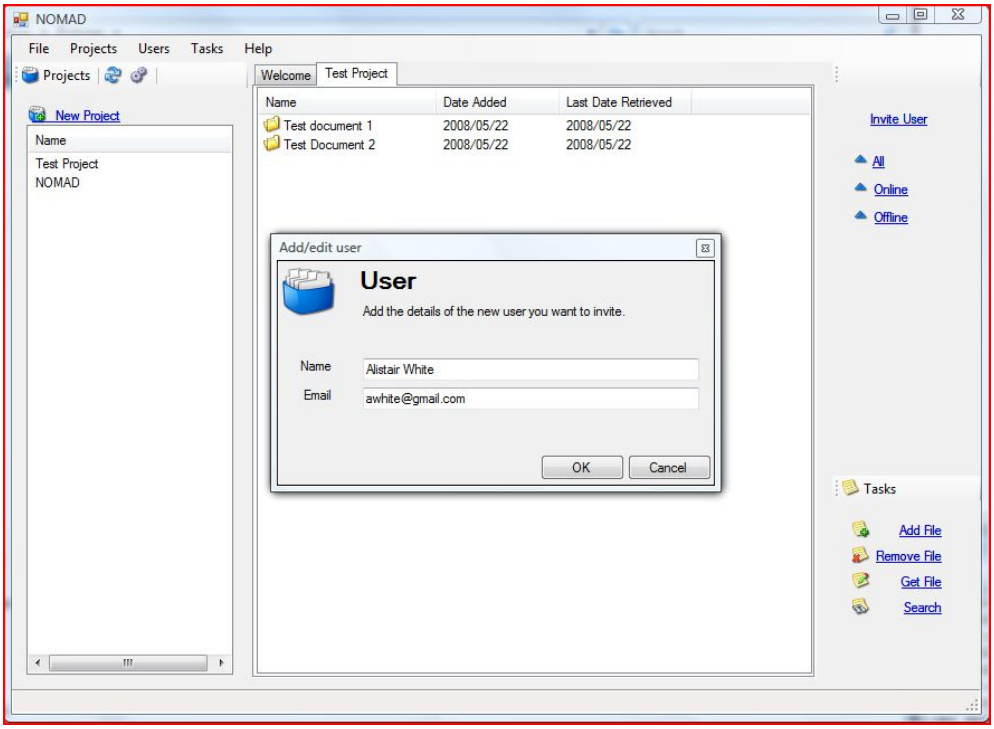

Figure 18 - Screenshot of registering a new user in Nomad

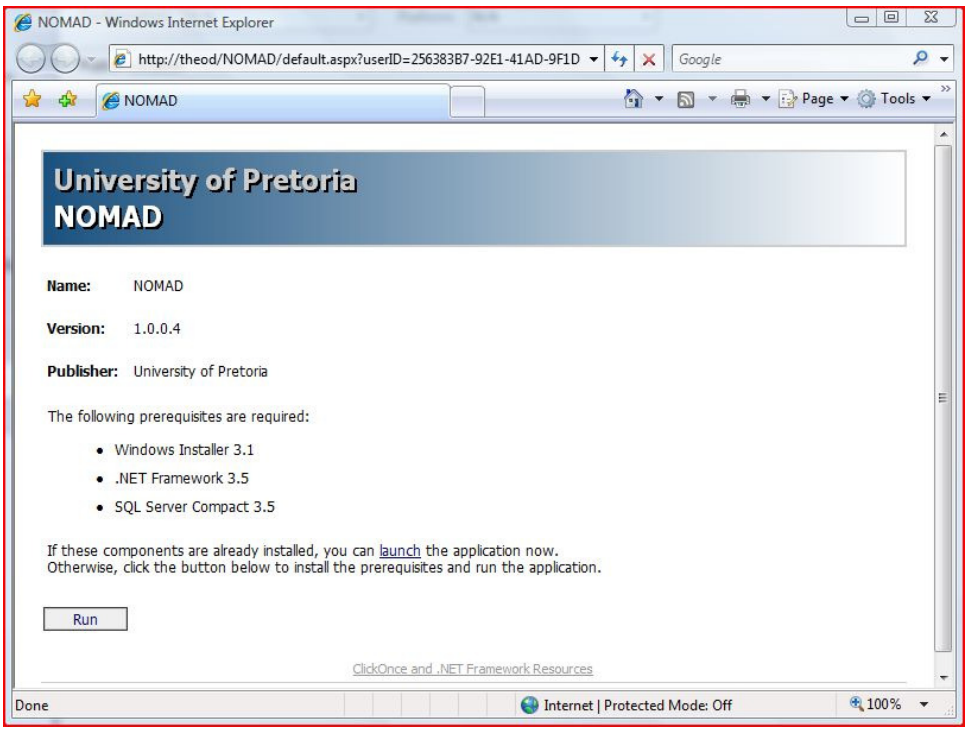

Figure 19 - Screenshot of the website where new users can download Nomad.

4.5.2 Nomad protocol service

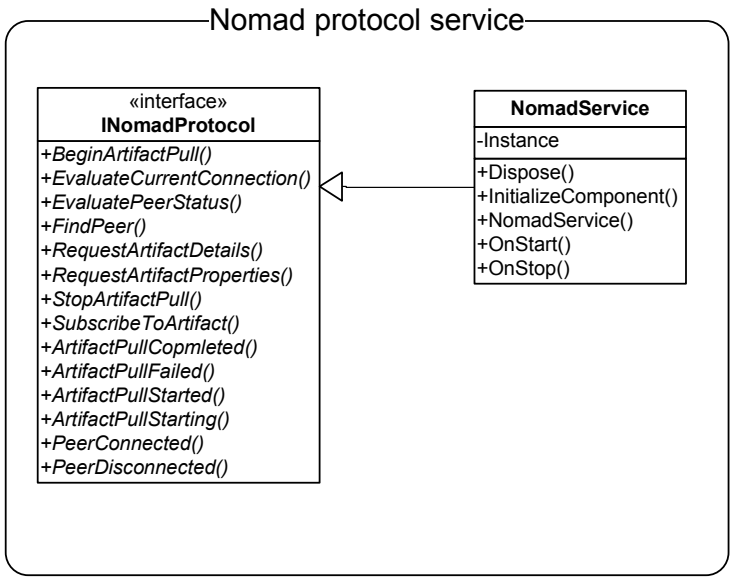

Figure 20 - Class diagram of the Nomad protocol service components

The Nomad protocol service consists of a Windows service that hosts the implementation of the INomadProtocol interface. The INomadProtocol interface defines the public methods that can be initiated from the user interface components. Figure 21 shows the sequence of events when a user requests a version of an artefact. The user interface only interacts with the INomadProtocol service, which hides the complexity of distribution from the user interface. The first step of the protocol is to load a description of the artefact from a local data store. This description stores details such as local file version, local file path, local file size, and a list of peers who own a version of the artefact. This Metadata of the artefact is synchronized in the background when a peer is connected to other peers, a peer only stores information on artefacts he owns, and a list of other peers who might own a copy of the artefact.

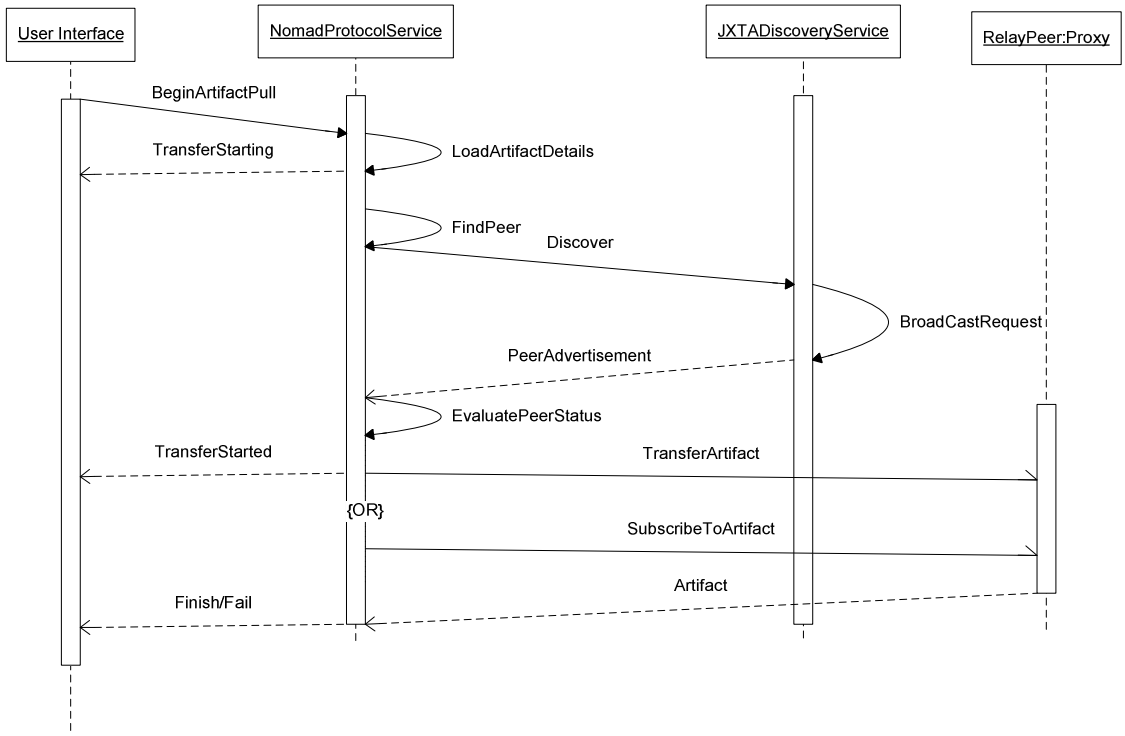

Figure 21 - The sequence of events initiated by an artefact request

The next step of the protocol is to find and evaluate connected peers. Peers are discovered through the JXTA Discovery service. The discovery service broadcasts the discovery request to all nodes through the underlying transport layer. The transport layer is responsible to traverse the virtual peer network to discover the requested peer. If the peer could not be found or the connection to the peer is too slow, its status would be set updated, and the EvaluatePeerStatus method of the Protocol service would fail. This failure will trigger the protocol service to register a subscription for the artefact. If the peer is found and a reasonably stable connection could be made, the protocol service will initiate an asynchronous transfer of the artefact. During artefact transfer the connection is monitored at frequent intervals; if the connection drops below the specified connection criteria, the transfer will be cancelled and a subscription to the artefacts will be registered by the protocol service. The attached code snippets show the implementation details of the Nomad protocol service – this highlights the sequence of events discussed and gives a brief discussion describing some of the implementation details.

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Runtime.Serialization.Formatters.Binary;
using JXTA;
using JXTA.Core.Implementation;
using JXTA.Core.API;
using JXTA.JXTADataSetTableAdapters;

namespace JXTA.Nomad
{
  public class NomadProtocol : INomadProtocol
  {
    DiscoveryService discoService;

    public event EventHandler ArtifactPullStarting;

    public event EventHandler ArtifactPullStarted;

    public event EventHandler ArtifactPullCompleted;

    public event EventHandler ArtifactPullFailed;

    public event EventHandler PeerConnected;

    public event EventHandler PeerDisconnected;

    …

  }
}
```

As can be seen by the class description of the Nomad protocol in the above code snippet, it makes extensive use of events. This approach allows any client application, such as a graphical user interface, to be notified when important events occur. As an example, client applications can use the *PeerConnected* event to toggle a flag or an icon on the user interface which will then show the user that one of his/her peers are online.

```
public void BeginArtifactPull(Guid artifactGuid)
{
      // Let the UI know Im starting the artifact Pull process
      OnArtifactPullStarting(EventArgs.Empty);

      // Read the artifact details from my local artifact   datastore
      JXTADataSet.tblArtifactDataTable tbl =
      tblArtifactTableAdapter.Instance.GetData(artifactGuid);

      // Loop through all the artifacts returned by the datastore
      foreach (JXTADataSet.tblArtifactRow artifactRow in tbl.Rows)
      {
        // Get nodes for this artifact, then check its connectivity status.
        JXTADataSet.relPeerArticatDataTable relatedPeers =
        relPeerArticatTableAdapter.Instance.GetData(artifactGuid);

        foreach (JXTADataSet.relPeerArticatRow peerRow in relatedPeers)
        {
          // Send request to the JXTA Discovery service to locate the peer
          ConnectionStatus peerStatus = FindPeer(peerRow.fkPerGuid);

          // If peer is online and his connection status is within criteria
          // Then we initiate an Artifact Pull
          if (peerStatus.Online && peerStatus.ConnectionValid)
          {

            AsyncArtifactPull(artifactGuid, peerRow.fkPerGuid);

            // Let the UI know that we've succesfully initiated a transfer
            OnArtifactPullStarted(EventArgs.Empty);
          }
          // Peer is offline or connection is outside performance criteria
          // Then we subscribe to the artifact delivery
          else
          {
            // Let the UI know that we are unable to pull the artifact.
            OnArtifactPullFailed(EventArgs.Empty);

            SubscribeToArtifact(artifactGuid, peerRow.fkPerGuid);
          }
      }
   }
}
```

The *BeginArtifcatPull* method forms the core of the Hunter component of Nomad. It starts off by raising an event to inform the user interface that it is initiating the *artefact pull* process. The shopping list of artefacts is compiled by querying the node's local artefact data store for all artefact detail. For each artefact in the shopping list, Hunter will now get all the nodes that have a version of this artefacts and then start to check the node's connectivity status. This is accomplished by sending a request to the JXTA Discovery service which will try to locate the peer. The JXTA Discovery service will also evaluate the connection, and return a connection criteria property which will indicate whether the connection is within the allowed criteria. If the peer node is online and within criteria then we will initiate an asynchronous *artefact pull,* and raise an event to inform client applications that we have

started an artefact transfer. When the peer could not be located, or if its connection status falls outside of the desired performance criteria, we will subscribe to *artefacts delivery* and inform client applications of this by raising an event.

```
private void AsyncArtifactPull(Guid artifactGuid, Guid peerGuid)
{
      SimplePeer peer = discoService.LoadPeer(peerGuid);

      // Hook in on download finished event
      peer.TransferFinished += new
      EventHandler<ValueEventArgs<JXTAObject>>(peer_TransferFinished);

      // Hook in on download interrupted event
      peer.TransferInterrupted += new
      EventHandler<ValueEventArgs<Guid>>(peer_TransferInterrupted);

      // Add the transfer worker to the thread pool.
      ThreadPool.QueueUserWorkItem(new WaitCallback(peer.TransferArtifact),
      peerGuid);
}
```

The *AsyncArtifactPull* method makes use of .NET Framework's ThreadPool to kick off an asynchronous artefact download. The physical implementation of downloading the artefact resides within the JXTA library. Extensive use of events enables the protocol to keep track of the status of the download.

```
void peer_TransferInterrupted(object sender, ValueEventArgs<Guid> e)
{
      // Let the UI know the peer has disconnected
      OnPeerDisconnected(EventArgs.Empty);

      // Let the UI know that we are unable to pull the artifact.
      OnArtifactPullFailed(EventArgs.Empty);

      // Subscribe to continue downloading
      SubscribeToArtifact(e.Value, e.Value);
}
```

Should the transfer of an artefact be interrupted before completion, Nomad will raise an event to inform the user interface that the transfer has failed and will automatically subscribe to *artefact delivery*.

```
void peer_TransferFinished(object sender, ValueEventArgs<JXTAObject> e)
{
      // Artifact was downloaded successfully, now save it to localpath
      JXTAObject artifact = (JXTAObject)e.Value;
      BinaryFormatter formatter = new BinaryFormatter();
      FileStream fs = new FileStream (artifact.LocalFilePath,
      FileMode.OpenOrCreate);

      formatter.Serialize(fs, artifact);

      // Let the UI know we are done transferring the artifact
      OnArtifactPullCompleted(EventArgs.Empty);
}
```

Once the transfer of artefact is complete, the artefact is serialized from memory to a file on the local node's storage device. The exact location of where the artefact is stored is specified within the artefact's local data store. The user interface is then notified that artefact transfer has completed successfully.

```
public void StopArtifactPull(Guid peerGuid)
{
      discoService.
        LoadPeer(peerGuid).
        StopTransfer();
}
```

The *StopArtifactPull* method enables the user to manually stop an artefact transfer. Nomad will not try to download this artefact again until the *BeginArtifactPull* method has been explicitly initiated for the artefact.

```
public void SubscribeToArtifact(Guid artifactId, Guid peerGuid)
{

      tblArtifactSubscribtionTableAdapter.Instance.
      AddSubscribtion(artifactId,
        peerGuid,
        DateTime.Now,
        DateTime.Now.AddDays(SETTINGS.SUBSCRIBTION_EXPIRY_DAYS));

      discoService.SubscribeRelayAtPeers(artifactId, peerGuid);
}
```

The *SubscribeToArtifact* method creates a subscription on all nodes in the peer-to-peer network to download the requested artefact from the requested peer. It uses the JXTA library to distribute the subscription to all peers in the network. To prevent subscriptions from just hanging around, we have introduced a day-limit setting which will mean that subscriptions will expire after a certain amount of days. This setting is a Network-wide setting that should be consistent amongst all peers.

```
public ConnectionStatus FindPeer(Guid peerGuid)
{
      // Send an Asynchronous request to the JXTA Discovery Service to
      locate the peer
      ConnectionStatus status = new ConnectionStatus();
      status.RequestStartTime = DateTime.Now;

      List<IAdvertisment> peerAdvertisements = discoService.Discover(
        DiscoveryQueryType.Peer,
        SETTINGS.TIMEOUT_THRESHOLD,
        SETTINGS.PEER_UID,
        peerGuid.ToString());

      foreach (IAdvertisment add in peerAdvertisements)
      {
        if (add.ID.Equals(peerGuid) &&
            (add.QueryType.Equals(DiscoveryQueryType.Peer)))
        {
          status.Online = true;
          status.RequestEndTime = DateTime.Now;
        }
      }
      status.ConnectionValid = EvaluatePeerStatus(peerGuid);
      return status;
}
```

The *FindPeer* method uses the JXTA library's discovery service to locate peers in the network. This is accomplished by creating an advertisement to which peers who meet the requirements will respond. If a peer does not meet the advertised requirements then it will not be allowed to respond, this enables us to filter out all peers that are either not online, or peers whose connection status does not meet the requested connection criteria.

```
public bool EvaluateCurrentConnection()
{
      return ((NetworkHelper.Instance.IsConnected())
        && (NetworkHelper.Instance.AvgResponseTime <=
      SETTINGS.RESPONSE_TIME));
}

public bool EvaluatePeerStatus(Guid peerId)
{
      return NetworkHelper.Instance.EvaluatePeerStatus(peerId);
}
```

Both the *EvaluateCurrentConnection* and *EvaluatePeerStatus* methods in the code snippet above can be used to check whether a peer is still online and whether its connection still meets the required connection criteria. The *NetworkHelper* class periodically sends a ping request to each node and then updates its *AvgResponseTime* property. This property can then be used to determine whether the connection to a specific peer is still reliable enough.

```
public ArtifactDetails RequestArtifactDetails(NomadArtifact artifact)
{
      return
      tblArtifactTableAdapter.Instance.LoadArtifactDetails(artifact.Id);
}

public ArtifactProperties RequestArtifactProperties(NomadArtifact artifact)
{
      return
      tblArtifactTableAdapter.Instance.LoadArtifactProperties(artifact.Id);
}
```

The *RequestArtifactDetails* and *RequestArtifactProperties* methods read artefact data from the local artefact data store. This is accomplished by making use of ADO.NET's *DataTableAdapter* classes.

```
public void OnArtifactPullStarting(EventArgs e)
{
      if (ArtifactPullStarting != null)
        ArtifactPullStarting(this, e);
}

public void OnArtifactPullStarted(EventArgs e)
{
      if (ArtifactPullStarted != null)
        ArtifactPullStarted(this, e);
}

public void OnArtifactPullFailed(EventArgs e)
{
      if (ArtifactPullFailed != null)
        ArtifactPullFailed(this, e);
}

public void OnArtifactPullCompleted(EventArgs e)
{
      if (ArtifactPullCompleted != null)
        ArtifactPullCompleted(this, e);
}

public void OnPeerDisconnected(EventArgs e)
{
      if (PeerDisconnected != null)
        PeerDisconnected(this, e);
}
```

The above methods ensure that all events raised in the protocol implementation are only raised if there is a client interested in the event. If no clients are hooked onto the event, then the event is just ignored.

4.5.3 JXTA P2P Library

Our Peer-to-peer network is based on a basic implementation of the JXTA Protocol. The JXTA Protocol defines the minimum required network semantic for peers to form and join a virtual network. It is language independent,and can be implemented on top of any transport protocol (TCP/IP, HTTP, Bluetooth, etc…)

JXTA is a set of open, generalized P2P protocols that allow any connected devices on the network (from cell phone to PDA, from PC to server) to communicate and collaborate as peers. The JXTA Protocol standardised the manner in which peers:

- Discover each other;
- Self organize into peer groups;
- Advertise and discover network services;
- Communicate with each other; and
- Monitor each other.

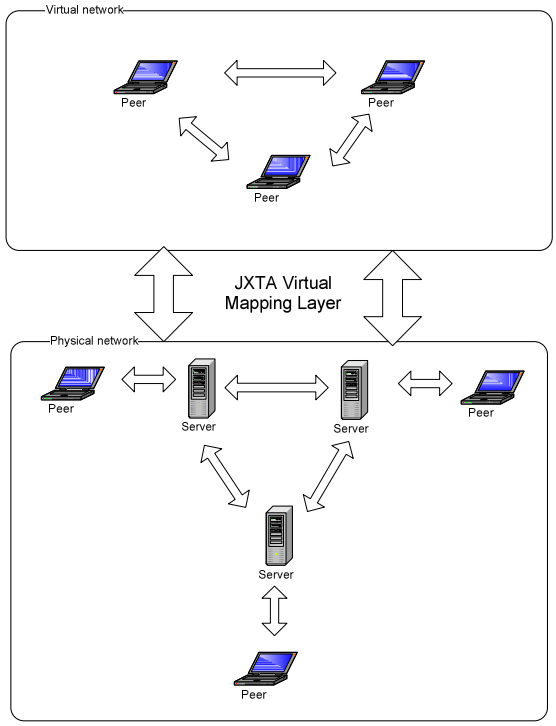

Figure 22 - Shows the how JXTA maps between the physical and the virtual network

JXTA forms a virtual network layer on top of a physical network, this hides the complexity and technical challenges a developer is faced with when developing P2P applications.

The six basic protocols defined by JXTA are:

Peer resolver protocol (PRP)

The PRP defines the mechanism to send a request and receive a response to one or more peers.

Endpoint routing protocol (ERP)

ERP defines the mechanism a peer can use to discover a route between itself and another peer.

Peer discovery protocol (PDP)

PDP defines the protocol a peer can use to advertise its resources and services to other peers. It also defines the method to discover resources and services available from other Peers.

Peer information protocol (PIP)

PIP enables peers to acquire status information on other peers. This information includes, but is not limited to: state, uptime, and traffic load.

Pipe Binding Protocol (PBP)

PBP defines the mechanism by which a peer can establish a virtual communication channel between one or more peers.

Rendezvous Protocol (RVP)

RVP is the mechanism used by peers to subscribe to a propagation service; this allows a peer to send a message to all peers listening to a rendezvous peer.

(i) Peers

A peer is any network device that implements the JXTA Protocols. A peer does not have to implement all six of the JXTA protocols, but it must implement at least the Peer resolver Protocol (PRP) and the Endpoint Routing protocol (ERP) to be addressable in the virtual network.

A peer must not be confused with a user, no explicit relationship exist between peers and users. The term “User” is an application level term whilst “Peer” is a network level term.

A peer must have a unique ID with the following characteristics:

- It must be persistent, and is not allowed to change.
- Location independent.
- Unambiguous. It must be a complete reference to a single resource.
- Unique.

- Canonical. References to the same resource must have the same ID for the resource.

Peers may use the JXTA Protocols to advertise their available resources and discover network resources available from other peers.

(ii) Peer Groups

A Peer group is a collection of peers with a common set of interest. Each group must also be uniquely identified by an ID. The JXTA Protocol recognizes three reasons for forming groups:

1. Creating a secure environment.
2. To provide a Scope within an environment.
3. Create a monitoring environment.

A Group can specify a set of services it provides to its members. JXTA specifies a list of core group services that groups need to implement, these services are discussed in more details under the (v) Services section.

(iii) Pipes

A Pipe defines a virtual connection to send and receive messages between two endpoints. JXTA recognizes that multiple Quality of Service pipe implementations exist, but only require an implementation to provide a Uni-directional asynchronous Pipe. Furthermore, the JXTA Protocol requires a pipe to be able to send a message to a single endpoint or to multiple endpoints.

(iv) Messages

A message forms the envelope for transporting information between peers. The content of a message can be of an arbitrary type, the message itself is an XML document with any number of named-subsections.

(v) Services

The JXTA Protocol recognises two types of network services: Peer services and Peer group services.

A Peer service is a service that is only provided by one peer, if the peer is unavailable then the service is not available to the network.

Peer Group Services, in contrast, are services provided by multiple peers. Peers can either collaborate to provide the service, or they can serve as a redundant peer for when other peers are not available.

JXTA specifies the following core Peer Group Services:

- Discovery Service.

- Membership Service.
- Access Service.
- Pipe Service.
- Resolver Service.
- Monitoring Service.

4.5.4 Microsoft .NET Framework

(i) The Common language runtime (CLR)

The .NET Common Language Runtime (CLR) is a virtual machine environment that provides runtime services and support for applications written on the .NET Framework. Source code that executes in the .NET CLR is referred to as managed code. During compile time, managed code is compiled into an intermediate language called the Microsoft Intermediate Language (MSIL). At runtime the Just-In-Time compiler (JIT compiler) of the Common Language Runtime converts the MSIL into native operating specific code. Introducing MSIL as a layer of abstraction between the written code and the executing code. .NET enables developers to write code in any language that can compile into MSIL. The list of supported languages is ever growing but amongst the favourites are:

C#, C++, J#, Visual Basic .NET, IronPython, P#(ProLOG), and even a version of COBOL.

Apart from the JIT compilation, the CLR provides other important runtime services that ease the process of software development, such as:

- Common type management
- Memory management
- Process and thread management
- Exception handling
- Object lifetime control
- Security

(ii) Windows Forms

The Windows Forms library provides developers with the building blocks to build highly responsive and interactive user interfaces for windows based applications. Even though replacement technologies such as XAML are already included in .NET 3.0 under the Windows Presentation Foundation (WPF), Windows Forms are still widely being used, mostly because of its ease of use. The Windows Forms library provides developers with direct access to the native windows user interface elements by wrapping the existing Windows API in managed code.

(iii) ADO.NET

ADO.NET is the .NET library that provides a common set of classes to access a collection of data sources and database management systems. Classes in the ADO.NET library can primarily be broken into two parts: Data providers and Data sets.

Data Providers

Data providers give developers CRUD access to a large variety of data sources. Each supported data source need to implement its own set of data provider classes. Included in the .NET Framework are data provider classes to access all the major Relational Database Management Systems (RDBMS) such as Microsoft SQL Server, Oracle. ADO.NET Data providers are implemented as partial classes, this enables developers to customize and extend on the features provided by the .NET platform. The attached code snippet shows how the Custom *GetData* and *Save* methods can be implemented without the need to override or inherit from the generated code.

```
public partial class tblProjectTableAdapter
    {
      private static tblProjectTableAdapter instance;
      public static tblProjectTableAdapter Instance
      {
        get
        {
          if (instance == null)
            instance = new tblProjectTableAdapter();

          return instance;
        }
      }
        public JXTADataSet.tblProjectDataTable GetData()
        {
            JXTADataSet.tblProjectDataTable tbl =
              new JXTADataSet.tblProjectDataTable();

            SqlCeDataAdapter adapter = new SqlCeDataAdapter(
              "SELECT * from tblProject where bActive=1",
              Connection);

            adapter.Fill(tbl);
            return tbl;
        }

        public bool Save(JXTADataSet.tblProjectRow dataRow)
        {
            if (dataRow.RowState ==
              System.Data.DataRowState.Added)
                return (Insert
                  (dataRow.pkGUID,
                  dataRow.sName,
                  dataRow.bActive,
                  dataRow.dtCreated) > 0);

            else if (dataRow.RowState ==
              System.Data.DataRowState.Modified)
                return (Update(dataRow) > 0);

            else if (dataRow.RowState ==
              System.Data.DataRowState.Deleted)
                return (Delete(
                  (System.Guid)dataRow["sName",
System.Data.DataRowVersion.Original])
                    > 0);

            else
                return false;
        }
    }
```

Data sets

Data sets provide a set of classes that represent a disconnected in-memory representation of relational data. Disconnected data sets are especially useful in situations where a stable long-running connection to the database is not feasible. There are two approaches for using datasets in .NET – generic or typed

datasets. The first code snippet shows an example of using typed datasets to assign default values to a new data row. The advantages of using typed data sets are that you get compile time support when accessing any of the table, column or relationship entities; this ensures type safety.

```
public JXTADataSet.tblArtifactRow NewtblArtifactRowWithDefaults()
    {
JXTADataSet.tblArtifactRow returnRow = NewtblArtifactRow();
returnRow.pkGuid = System.Guid.NewGuid();
returnRow.sName = string.Empty;
returnRow.sLocalFilePath = string.Empty;
returnRow.dtAdded = System.DateTime.Now;
returnRow.dtRequested = System.DateTime.Now;
returnRow.dtModified = System.DateTime.Now;
       return returnRow;
}
```

The same method as above can easily be transformed into generic datasets. Generic datasets are especially useful when writing a standard method that needs to perform the same function on many different tables. Even though using generic datasets are more error-prone, because of its lack of compile time support, it can be implemented to great effect when you need to perform a generic task.

```
public DataRow NewtblArtifactRowWithDefaults()
       {
         DataRow returnRow = new DataRow();
         returnRow["pkGuid"] = System.Guid.NewGuid();
         returnRow["sName"] = string.Empty;
         returnRow["sLocalFilePath"] = string.Empty;
         returnRow["dtAdded"] = System.DateTime.Now;
         returnRow["dtRequested"] = System.DateTime.Now;
         returnRow["dtModified"] = System.DateTime.Now;
         return returnRow;
       }
```

(iv) Language-Integrated Query References (LINQ)

Language-Integrated Query References, or better known as LINQ, present a common query language that developers can use to query and update various types of data sources. LINQ aims to eliminate the need for developers to learn a new query language for each different data source it wants to access (Microsoft Corporation, 2008). A typed dataset provides developers with compile time support when accessing an element of the dataset; LINQ takes this one step further by adding compile-time support to the query language. LINQ embeds a set of general purpose query operators that facilitates traversal, filter, and projection operations into any .NET based programming language. These standard query operators can be applied to any *IEnumerable<T>* based data source. This open query architecture is clearly demonstrated by the set of LINQ Providers implemented in the .NET Framework, namely: LINQ to SQL, LINQ to XML, LINQ to Objects, and LINQ to Datasets. Each one of these providers

interprets the common LINQ expression trees and performs data source specific execution logic. In order to demonstrate the ease of using LINQ let us examine the following code snippet.

```
    public bool HasAdminRole(string userName)
    {
      // Initialize the LINQ DataConttext
      NomadDataContext context =
        new NomadDataContext(SETTINGS.CONNECTIONSTRING);

      // Specify the query operations
      var users =
        from c in context.securityUsers
        where c.fk_securityUserRole > 1 && c.username.Equals(userName)
        select c;

      // Execute the query and return count
      return users.ToList<securityUser>().Count() > 0;
  }
```

This example makes use of the LINQ to SQL provider to select data from the underlying SQL server database for a given filter. The query is translated into T-SQL syntax by the .NET Framework, and the database specific implementation details are completely transparent to the developer.

(v) Windows Communication Foundation (WCF)

The Windows Communication Foundation (WCF) has been designed to simplify the development of distributed and connected applications. One of the main design requirements of WCF was to enable developers who are familiar with existing Microsoft enterprise technologies, such as: ASP.NET Web services, .NET Remoting, Message queuing, and COM+ technologies, to seamlessly migrate to the service oriented approach provided by WCF (Microsoft Corporation, 2007). This has been accomplished by breaking the complicated task of distributed application development into five steps: define the service contracts; implement the service contracts; configure the service; host the service; and finally build the clients. These steps form the basic programming lifecycle of any WCF application.

```
  [ServiceContract]
  public interface IJXTAService
  {
    [OperationContract]
    string Ping();

    [OperationContract]
    void InitiateArtifactTransfer(Guid artifactGuid, Guid peerGuid);

    [OperationContract]
    object StreamArtifactFromPeer(Guid artifactGuid, Guid peerGuid);

    [OperationContract]
    void CancelArtifactTransfer(Guid artifactGuid, Guid peerGuid);

  }
```

The code snippet above shows an example of a service contract definition. The service definition is a standard C# interface tagged with special WCF attributes that identifies the contract operations. Contract implementations only need to implement the interface; they are completely hidden from the fact that it will be exposed as a service. Furthermore, the .NET Framework enables developers to configure endpoint and service behaviour through configuration files and host it in Internet Information Services (IIS). After the service is configured and hosted, the last step to perform is the implementation of a WCF client which will call the service methods. The .NET Framework creates a client-side proxy which converts normal method calls into messages which are serialized and sent through to the remote service. Apart from creating the client-side proxy object and configuring the WCF framework, the WCF client can consume a remote service without any further knowledge of the underlying connections, message transfers, and serializations taking place.

Chapter 5. Conclusion and future work

5.1 Summary

The phrase "access, anytime and anywhere" has been used by both researchers and industry to describe various forms of mobile work and collaboration (Harper, et al, 2001). We argued that it is imperative to extrapolate the phrase and elaborate on the multiple facets it covers. Based on existing research in the CSCW arena (Grudin, 1994) we propose a model that can be used to distinguish between connected, disconnected and Casually Connected Collaboration scenarios. The aim of our conceptual model is to classify collaboration scenarios in terms of the actor, operation, and the environment. Chapter 2 concludes that a scenario can be classified as Casually Connected Collaboration when a small group of geographically dispersed people or machines, with little control over the technical and social capabilities of their environment, engage in collaboration activities. These actors perform their tasks in unscheduled time for an unknown amount of time, and the reason for them being out of office is usually not related to the task at hand. Chapter 3 uses the model to classify some known collaboration scenarios and finds that authoring a book or technical report is one of the best scenarios that fall under Casually Connected Collaboration whilst working in a software development team is least suited. We then present Nomad, a Peer-to-Peer Framework that will support collaboration between casually connected members. Chapter 4 elaborates on the challenges one encounters when implementing a true peer-to-peer solution on current Internet technologies. Firewalls, NAT, and other technologies used to overcome the lack of IPv4 address space forced us to build Nomad on a hybrid peer-to-peer architecture rather than a pure peer-to-peer architecture. The Microsoft .NET framework version 3.5 proved to be a good choice when it came to implementing Nomad, especially with the help of technologies such as ADO.NET, LINQ, and WCF.

5.2 Progress

During the initial phases of the research at the end of 2002, the functional and non-functional requirements for a system that would assist colleagues in writing reports and accessing documents anytime and anywhere where recognized. This triggered research into comparing these requirements against current available options such as CVS, SourceSafe, NFS, and other centralized collaboration products. It soon became apparent that even though these systems fulfil our functional requirements, they fall short when it comes to the non-functional requirements, not one of these projects sufficiently catered for unreliable connections and they all required the user to change his normal way of performing a task. We were convinced that we had a unique idea and research turned towards implementation details around a framework that will support both the functional and non-functional requirements of a Casually Connected Collaboration scenario. This framework was termed, Nomad.

The major output from this phase of the project was an Msc. produced by Jiten Rama that describes the Nomad protocol in detail. Even after the details of the protocol had been finalized, we realized that there were no clear guidelines to distinguish between the various types of connectivity, industry and researchers alike used the phrases ""access, anytime and anywhere"" with reference to different scenarios. We then focused on developing a model that can be used to classify collaboration scenarios into connected, mobile, and Casually Connected Collaboration. To validate this newly defined model we used it to classify the same projects researched at the beginning of the project, this classification is displayed in Chapter 2. On paper, our research was looking great, we were able to prove our theory, but lacked a physical implementation. It is during this implementation phase that we came across the many obstacles one has to overcome when developing a pure peer-to-peer system using current Internet technologies. The Nomad architecture was slightly modified towards a hybrid P2P system that would be feasible to implement using current available technologies.

5.3 Evaluation

This thesis presents a peer-to-peer framework as viable technology option for use in Casually Connected Collaboration scenarios. Implementation details are discussed on the hand of Nomad, an example implementation of the recommended framework. As discussed in the work done by Rama (2006); Nomad can be seen as an appropriate solution for collaboration between members in a small secure group. Performance concerns, especially during artefact transfer, have been raised during the simulation and evaluation of the Nomad protocol. These concerns have been addressed with the introduction of a connection evaluation function in the Nomad protocol, and through the use of advertisement-based node discovery in the JXTA library. The use of propagation nodes to transfer an artefact via its peers is seen as one of the major obstacles behind scaling Nomad to larger communities. Future research should focus on techniques to prevent propagation nodes from flooding the network with artefacts and consuming unnecessary bandwidth. Critics have suggested the use of mobile agents instead of a peer-to-peer approach. Due to the small size of Nomad communities and the simplicity of the current protocol, we have decided that the advantages that can possibly be gained by introducing Mobile agents are not worth the overhead and complexities involved. That being said, we do envisage that mobile agents can be useful in a Casually Connected Collaboration environment, this can be seen as a future research topic. Another suggestion from colleagues have been to simply replicate artefacts on multiple servers, even though we are fond of the simplicity of the idea, it goes against one of our core design criteria – avoid information spread and neglect. The framework aims to reduce the replication of artefacts on multiple servers and to ensure that collaborators have access to a single copy of the correct artefact.

5.4 Technology Trends

In hindsight, one would almost always implement a project differently when you are done. This is mostly due to the incredible pace at which computer science and software development technology evolves. In light of this, it is important to keep up with emerging trends and to evaluate how these can improve future designs and implementations of a Casually Connected Collaboration platform.

Microsoft recently announced the release of Microsoft Visual Studio 10 and the .NET Framework version 4.0. Cloud computing and especially support for the Windows Azure platform was the driving force behind most of the new features introduced in the framework. I will highlight some of the important new features, and how it can be used, in the design and implementation of a Casually Connected Collaboration framework.

5.4.1 .NET Framework 4.0 enhancements

One of the most exciting enhancements made to the upcoming version of .NET framework is the inclusion of the next version of Windows Communication Foundation (WCF 4.0). Microsoft's decision to include better support for Web 2.0 technologies moves us ever closer to creating the programmable web; this is highlighted by WCF 4.0's support for creating RESTful services. A RESTful service is defined as a service that conforms to the design constraints as defined by the Representational State Transfer (REST) architectural style (Fielding. 2000). The key design constraint that sets REST apart from other distributed architectural styles is its emphasis on a uniform interface between all components. The web as we know it today conforms to the REST architectural style, it was built completely on HTTP's uniform interface of which GET, PUT, DELETE, POST, HEAD, and OPTIONS are the most commonly used HTTP methods. Building RESTful services are in stark contrast to building Remote Procedure Call (RPC) based services, which have been the default way to build services up to now. When using RPC-based style a new interface has to be introduced with each new component you want to include in the system. Once the interface has been discovered by clients, they must then first familiarize themselves with the intricacies of the new interface. As the number of components and services grow, so does the number of interfaces and ultimately the complexity of the system as a whole. Building a Casually Connected Collaboration framework that is RESTful will have many advantages. These advantages include: simplicity, standardized clients, inherent security, and platform independence.

Another exciting enhancement made to the .NET Framework 4.0 is the inclusion of parallel extensions. These extensions comprise of the Task Parallel Library (TPL), PLINQ, and Coordination Data Structures. The TPL exposes parallel constructs like parallel *for* and *while* loops. Because of its inclusion into the Framework itself, these constructs can be called from any language supporting the .NET Framework. PLINQ is a concurrent query execution engine for LINQ: which allows you to execute LINQ queries in parallel. A framework that support Casually Connected Collaboration

requires very little user interaction; it is therefore feasible to write the framework in such a way that it runs in parallel while the user is working. Also, multiple nodes have to be contacted, queried, and processed – these operations can all be executed in parallel to improve performance. Making use of multi-core processors and the new enhancements to the .NET framework, one can significantly increase performance and user experience.

The .NET Framework version 4.0 will also include the Windows Azure SDK which provides developers the APIs needed to develop scalable cloud based services which can be deployed and managed on the Windows Azure platform. Windows Azure can be seen as an operating system for the Microsoft Cloud which provides developers on-demand computing power and storage space on Microsoft's managed data centres.

5.4.2 C# 4.0 Language enhancements

One of the most talked about enhancements made to the C# language is built in support for dynamic types. Dynamic type inference enables the C# language to perform type resolution during runtime as opposed to compile time, which is the case with static type inference. C# developers frequently have to integrate with systems, documents, and languages which are not type-safe such as the HTML DOM, XML documents without schemas, Ruby and Iron Python which poses problems for a language that only supports static types (Hejlsberg, 2008). To get past this shortcoming developers had to make use of the reflection libraries in .NET, which made code unnecessarily complicated, verbose, and difficult to read. With Microsoft's focus on cloud computing where a component might have to integrate with many other loosely typed services, they decided to introduce dynamic types which will improve consistency and readability when working with loosely typed objects. As a simple example the code snippet below shows how developers had to call a method using reflection; the next code snippet shows how dynamic types can simplify the same task.

```
public void ExecuteDynamicMethod()
{
      Execute("ExternalAssembly.dll",
            "External.TestClass",
            "RepeatHelloWorld",
            new Type[] { typeof(string), typeof(int) },
            new object[] { "Hello World", 10 });
}

public object Execute(
      string asmPath,
      string typeName,
      string methodName,
      Type[] parmTypes,
      object[] parmValues)
{
      Assembly asm = Assembly.LoadFile(asmPath);
      Type tp = asm.GetType(typeName);
      object instance = Activator.CreateInstance(tp);
      MethodInfo method = tp.GetMethod(methodName, parmTypes);
      return method.Invoke(instance, parmValues);
}
```

In the above code snippet the developer is invoking a method *RepeatHelloWorld* using Reflection. In order to combat the complexity a helper method named Execute is written to wrap the reflection and method invocation.

```
public void ExecuteDynamicMethod()
{
      Assembly asm = Assembly.LoadFile("ExternalAssembly.dll");
      dynamic instance =
      Activator.CreateInstance(asm.GetType("External.TestClass"));

      instance.RepeatHelloWorld("Hello World", 10);

}
```

In the above code snippet the developer is invoking a method *RepeatHelloWorld* using the Dynamic types provided by C# version 4.0.

The next version of C# will also provide support for optional and named parameters. Optional parameters allow you to omit a parameter during method invocation if a default value for the parameter has been defined. Named parameters will enable you to specify the value of a parameter by using its name, instead of relying only on its position in the method declaration. These two additions to C# will further simplify your code by reducing the need to create many overloaded methods for the same operation. Furthermore, it holds true to Microsoft's pledge of co-evolving the C# and Visual Basic languages as Visual Basic already have support for optional and named parameters.

C# version 4.0 will provide much better support for COM-interoperability. Apart from dynamic types and optional parameters, C# 4.0 will enable you to call COM methods without having to distribute the Primary Interop Assembly (PIA). This feature, more commonly referred to as no-PIA, will solve two problems commonly faced with COM-interoperability, namely: versioning issues and large executables because of the size of the PIA. In future the C# compiler will extract the small part of the PIA that you use in your application and include it directly into your assembly, this removes the need to deploy and load the whole PIA with your application.

All these new features in the C# language will enable developers to write code more concisely. It will also improve C#'s ability to integrate with external services and applications. This is particularly exciting for the development of a collaboration framework as it will be much easier to integrate with tools such as Microsoft Office, and scalability concerns can be addressed by developing the collaboration framework as a service on Windows Azure, Microsoft's cloud computing platform.

5.4.3 Visual Studio 2010 enhancements

Microsoft Visual Studio 2010 will support two key advances made in software development; developing cloud services and constructing applications that can take advantage of multi-core hardware. The Integrated Development Environment (IDE) will support the enhancements made in both the .NET Framework and the C# language by providing tools that makes it easier to write, analyze, debug, and manage code.

Noteworthy enhancements included in the Visual Studio 2010 IDE are:

- User interface tools for building and deploying cloud computing services on the Microsoft Windows Azure platform.
- Debug and code profiling tools for parallel development.
- Modelling tools to help developers understand existing code.

5.5 Future Work

The need for a framework that supports Casually Connected Collaboration has been recognized and a model has been provided to distinguish between casually connected and other collaboration scenarios. We also discussed the technology necessary to implement a framework that will support Casually Connected Collaboration and implemented Nomad, a proof of concept thereof. This highlighted some of the technical challenges one faces when implementing such a framework. The next logical step would be to compare the physical performance of a Nomad implementation, and measure whether it adheres to the functional and non-functional requirements specified early on in the project. It would be of special interest to compare a peer-to-peer approach with a classic centralized or client/server model, especially in terms of non-functional requirements and performance.

During the proof of concept implementation we started to recognize the shortcomings of current IPv4. Future research into the effect of IPv6 on peer-to-peer architectures might be of great value to the casually connected environment. Research into Mobile IPv6 (Perkins & Johnson, 1996) has already lead to some very interesting results and discussions. We also acknowledge the advantages that mobile agents can introduce in a Casually Connected Collaboration environment; a future project can focus on the introduction of agent based solutions.

In recent times collaboration has evolved to include both humans and machines, these trends are especially visible in systems such as Unmanned Aerial Vehicle (UAV). Interesting research might be conducted towards applying Casually Connected Collaboration approaches into a scenario where collaboration is conducted between both machines and humans. Even through the current technology, solutions presented provide a viable solution to the problems faced by casually connected collaborators. New technologies such as the Microsoft .NET Framework version 4.0 can significantly reduce implementation complexity and provide possible performance enhancements – future research into the application hereof should be considered.

References

Aberer, K., & Hauswirth, M. (2002). An Overview of Peer-to-Peer Information Systems. *Distributed Data & Structures 4, Records of the 4th International Meeting (WDAS 2002)* (pp. 171-188). Paris, France: Carleton Scientific.

Agre, P. E. (February 2003). P2P and the promise of Internet Equality. *Communications of the ACM* , 39-42.

Appelt, W. (1999). WWW Based Collaboration with the BSCW System. *Book Series Lecture Notes in Computer Science. SOFSEM'99: Theory and Practice of Informatics*, vol. 1725/1999 pp 762.Berlin /Heidelberg. Springer

Bardram J.E. (September 2005). Activity-based computing: support for mobility and collaboration in ubiquitous computing. *Journal Personal and Ubiquitous Computing*. Volume 9,Number 5. (pp 312-322)

Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H. F., & Secret, A. (1994). The world-wide web: Internet technology. *Communications of the ACM* , vol. 37 pp 76-82.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001, May). The Semantic Web. *Scientific American 284*, (pp. 34–43).

Billinghurst, M., Kato, H., (July 2002). Collaborative augmented reality. *Communications of the ACM* (pp 64-70)

Clark, D. (January 2001). Face-to-Face with Peer-to-Peer Networking. *IEEE Computer* , 18-21.

Danzfuss, T. L. (2003). *Resource Sharing in Distributed Peer-To-Peer Internet Applications.* Pretoria: University of Pretoria.

Dyson, E. (1990, November 30). *Release 1.0: Groupware preview: A new framework for Groupware.* Retrieved January 3, 2008, from Release 1.0: Groupware preview: A new framework for Groupware: http://downloads.oreilly.com/radar/r1/11-90.pdf

Ellis, C. A., Gibbs, S. J., & Rein, G. (1991). Groupware: some issues and experiences. *Communications of the ACM* , 39 - 58.

Emmerich, W. (2000) *Engineering Distributed Objects*, Wiley. (pp. 15)

Fielding, R, T. (2000) *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine (Chapter 5)

Fitzpatrick, G. (2003). *The Locales Framework: Understanding and Designing for Wicked Problems*. Springer (pp. 15-17)

Ford, B., Srisuresh, P., & Kegel, D. (2005). Peer-to-Peer Communication Across Network Address Translators. *Proceedings of the 2005 USENIX Technical Conference.* Anaheim, CA.

Freenet Website. (2007, December 20). Retrieved December 20, 2007, from Freenet Website: http://freenetproject.org/

Gnutella website. (2007, December 20). Retrieved December 20, 2007, from Gnutella website: http://www.gnutella.com/

Google. (n.d.). *Google Docs*. Retrieved July 20, 2008, from Google Docs: http://docs.google.com

Groove Home Page - Microsoft Office Online. (2007). Retrieved January 3, 2008, from Groove Home Page - Microsoft Office Online: http://office.microsoft.com/en-us/groove/FX100487641033.aspx

Grudin, J. (1994). Computer-Supported Cooperative Work: History and Focus. *IEEE Computer* , 19-26.

Harper, R. H. (2005). The Local and the Global: Paradoxes of the Mobile Age. *Communications for the 21 st Century* (pp. 31-38). Vienna: Passengen Verlag.

Harper, R., Brown, B., Sellen, A., O'hara, K., & Perry, M. (2001). Dealing with mobility: understanding access anytime, anywhere. *ACM Transactions on Computer-Human Interaction* , 323-347.

Hejlsberg, A. (2008) C# 4.0 - Questions and reasons behind the answers. MSDN Channel 9. http://channel9.msdn.com/posts/VisualStudio/C-40-Questions-and-reasons-behind-the-answers

Hua, C., Jianfei Q., Qingming H. (2006) A P2P Architecture for Supporting Group Communication in CSCW Systems. *Computer Supported Cooperative Work in Design, 2006. CSCWD '06. 10th International Conference on* (pp. 1-5) Nanjing.

ISO/IEC 42010:2007: Systems and software engineering -- Recommended practice for architectural description of software-intensive systems. (2007). Retrieved from ISO/IEC 42010:2007: Systems and software engineering -- Recommended practice for architectural description of software-intensive systems: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=45991

KaZaA website. (2007, December 20). Retrieved December 20, 2007, from KaZaA website: http://www.kazaa.com/

Kleinrock, L. (2000). Nomadic Computing and Smart spaces. *IEEE Internet Computing* , 52-53.

Lamming, M., Eldridge, M., Flynn, M., Jones, C., & Pendlebury, D. (2000). Satchel: Providing access to any document, any time, anywhere. *ACM transactions on Computer-Human Interaction* , 322-352.

Li, S. F. (May-June 2000). Integrating Synchronous and Asynchronous Collaboration with Virtual Network Computing. *IEEE Internet Computing* , 26-33.

Ma, J., Shizuka, M., Lee, J., & Huang, R. (2003). A P2P Groupware system with Decentralized topology for supporting Synchronous Collaborations. *Proceedings of the 2003 International conference on Cyberworlds CW03* (pp. 1-8). IEEE Computer Society.

Marquès, J. M., & Navarro, L. (2005). Autonomous and Self-sufficient Groups: Ad Hoc Collaborative Environments. *Groupware: Design, Implementation, and Use, 11th International Workshop* (pp. 57-72). Porto de Galinhas, Brazil: Springer-Verlag.

Marquès, J. M., Vilajosana, X., Daradoumis, T., & Navarro, L. (2007). LaCOLLA: Middleware for Self-Sufficient Online Collaboration. *IEEE INTERNET COMPUTING* , 56-64.

Mendoza-Chapa, S.,Romero-Salcedo, M.,Oktaba, H. (2000). Group awareness support in collaborative writing systems. *Groupware, 2000. CRIWG 2000. Proceedings. Sixth International Workshop on*. (pp. 112-118) Madeira, Portugal

Microsoft Corporation. (2007). *MSDN Library*. Retrieved August 2008, from What Is Windows Communication Foundation?: http://msdn.microsoft.com/en-us/library/ms731082.aspx

Microsoft Corporation. (2007). *MSDN Library*. Retrieved from WCF Client Overview: http://msdn.microsoft.com/en-us/library/ms735103.aspx

Microsoft Corporation. (2008). *MSDN Library*. Retrieved August 2008, from LINQ: .NET Language-Integrated Query: http://msdn.microsoft.com/en-us/library/bb308959.aspx

Microsoft. (n.d.). *Live Mesh*. Retrieved 04 23, 2008, from Live Mesh: http://www.mesh.com

Napster Website. (2007, December 20). Retrieved December 20, 2007, from Napster Website: http://www.napster.com/

Novell iFolder 3.6: Novell Open Enterprise Server. (2007). Retrieved January 03, 2008, from Novell iFolder 3.6: Novell Open Enterprise Server: http://www.novell.com/products/openenterpriseserver/ifolder.html

O'Hara, K., Harper, R., Unger, A., Wilkes, J., Sharpe, B., & Jansen, M. (2005). TxtBoard: from text-to-person to text-to-home. *CHI 2005* (pp. 1705-1708). Portland, Oregon, USA: ACM Press.

Oram, A. (2001). *Peer-to-Peer: Harnessing the Power of Disruptive Technologies.* Sebastopol, CA, USA : O'Reilly & Associates, Inc.

Osais, Y., Abdala, S., & Matrawy, A. (2006). A Multilayer Peer-to-Peer Framework for Distributed Synchronous Collaboration. *IEEE INTERNET COMPUTING* , 33-41.

Peterson, L. L., & Davie, B. S. (2000). *Computer Networks: A Systems Approach, Second Edition.* San Francisco: Morgan Kaufmann Publishers.

Pouwelse, J., Garbacki, P., Epema, D., Sips, H. (2005). The Bittorrent P2P File-Sharing System: Measurements and Analysis. *Lecture Notes in Computer Science Volume 3640/2005 Peer-to-Peer Systems IV* (pp. 205-216). Springer Berlin / Heidelberg

Rama, J., & Bishop, J. (2006). A survey and comparison of CSCW groupware applications. *Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries* (pp. 198 - 205). Somerset West, South Africa: South African Institute for Computer Scientists and Information Technologists.

Rama, J (2006). *The Design of a Protocol For Collaboration in a Distributed Repository - Nomad.* Pretoria: University of Pretoria.

Ripeanu, M. (2002). Peer-to-Peer Architecture Case Study: Gnutella Network. *Proceedings of the First International Conference on Peer-to-Peer Computing (P2P01).* IEEE.

Ripeanu, M., Iamnitchi, A., & Foster, I. T. (January/February 2002). Mapping the Gnutella Network. *IEEE Internet Computing* , 50-57.

Schoder, D., & Fischbach, K. (February 2003). Peer-to-Peer Prospects. *Communications of the ACM* , 27-29.

Schollmeier, R. (2002). A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications. *Proceedings of the First International Conference on Peer-to-Peer Computing (P2P01).* IEEE.

Scott, S.D., Grant, K.D., Mandryk, R.L., (2003) System guidelines for co-located, collaborative work on a tabletop display. *Proceedings of the eighth conference on European Conference on Computer Supported Cooperative Work.* (pp. 159-178) Helsinki, Finland

Shackel, B. (2000) People and Computers: Some Recent Highlights, *Applied Ergonomics*, edition 31. (pp. 595–608).

Singh, M. P. (FEBRUARY 2001). Peering at Peer-to-Peer Computing. *IEEE INTERNET COMPUTING* , 4-6.

SubEthaEdit. (2008). Retrieved January 3, 2008, from SubEthaEdit: http://www.codingmonkeys.de/subethaedit/

ter Hofte, G. H., & van der Lugt, H. J. (1997). CoCoDoc: a framework for collaborative compound document editing based on OpenDoc and CORBA. *Proceedings of the IFIP&IEEE International Conference on Distributed Platforms* (pp. 15 - 33). Toronto: Chapman & Hall, Ltd.

University, C. P.-E. (January-February 2000). CCF: A Framework for Collaborative Computing. *IEEE Internet Computing* , 16-24.

Wetteroth, D. (2001) *OSI Reference Model for Telecommunications*. McGraw-Hill Professional.

WhatIsMyIP.com. (2007, December 19). Retrieved December 19, 2007, from WhatIsMyIP.com: http://whatismyip.com/

Wikipedia. (2007, December 12). Retrieved December 19, 2007, from Network address translation: http://en.wikipedia.org/wiki/Network_address_translation

Yang, B., & Garcia-Molina, H. (2003). Designing a Super-Peer Network. *Proceedings of the 19th International Conference on Data Engineering (ICDE'03)* (pp. 49-60). IEEE.

York, J., Pendharkar, P. C. (May 2004) Human–computer interaction issues for mobile computing in a variable work context. *International Journal of Human-Computer Studies: HCI Issues in Mobile Computing*. Vol. 60 Issues 5-6, May 2004, pp 771-797.

Table of Figures