CHAPTER 5
SUMMARY AND FUTURE WORKS

From the description of this thesis work, from Chapter 3 and Chapter 4, it can be summarized as written in this chapter. It should be clearly enough how this thesis work contributes to the new highly tailorable and flexible groupware system.

5.1. Summary

The system is intended to provide synchronous communication strategy. This is translated into the use of available different tools and application at the same time for the users who joined the same session.

The design will use component based software. The communications of the groupware is basically communications between its components. The components will be separated based on its functions.

Coordination will be defined in some different way of work flow. We classify the applications based on the type of work (phase) that the applications can support, so that the user can use applications that support completion of their work type.

To provide a continuous coordination, a database to save the state of the work is undeniably needed. The database will be hold by the main server. But the state of running history will be stored in each replicated master.

In this system, authoring mechanism will be translated into the use of different colors for each user.

This system is aimed to give a highly distribution ability of the users. This groupware is designed to give its service to spatially distributed (worldwide) users. Hybrid architecture is shown to provide the best design tradeoffs. Thus in our system the distribution architecture mainly will implement hybrid

architecture. Only if all users join a session is located in the near area, the system implements centralized architecture.

During start up system will calculate cost of all variables concerning network latency between server to client as well as client to client intra and inter region, processing power of each client. This information added with the information of number of users and type of application used in a session, will determined type of distribution architecture of the program.

Because the system will have a possibility to implement distributed system strategy, in the case of hybrid/semi-replicated architecture, some basic problem of distributed system should be considered. Distributed system should provide some kinds of transparency; access, location, replication and fragmentation.

The system will serve a variable number of sessions. The users in a session may varies, according to the needs of the group own the session. The system serves a large scale of users worldwide. User should associate to at least a group. A group should have profile regarding its organization, expertise, and purpose of using the groupware. A maximum of users in a session should be different between applications. The system restricts the number of users in a session range between 2 and 8 people.

Everyone can use its services. Every user who wish to use the advantages of this groupware system, may register for a session, ask for the slot time, and he/she will be able to use all functionality the system has. There is no restriction from which country he/she comes, from which organization, or for what usage.

But the openness of a running session is determined by the session owner—who register the session to the system before the session starts.

This system will exploit the advantages of World Wide Web to support the asynchronous part of the system to handle management of users, groups and sessions.

The proposed approach is:

- This system will provide 2 alternative of architecture of its components, namely centralized and hybrid/semi replicated architecture.
- This distributed system will also allow partitioning of the network. Network partition in this case will be based on client physical locations.
- Late joiners are allowed by the system. There are 3 possibilities of adaptability. The adaptability is based on the network connection to each user and condition of users.
- There are 4 roles that are available in a session; moderator, member, floor handler, observer. Moderator will have double role at the same time, which is a role as moderator and another role from 3 roles available.
- At start up, moderator are given the right to choose between three types of floor control available from the system; Free Flow, Sequential Flow and Shared Floor Control Model.
- To support flexibility, in this system we will separate objects or components of the software based on its functionality into program, data, and user interface objects.
- The extensibility pattern of components is by using a Proxy. Client will always communicate to the object through the Proxy. The creation of object is passed from client command to a proxy and to a Creator to crate an Object.
- Concurrency control deals with the issues involved with allowing multiple people simultaneously access to shared entities. To prevent those problems, in this system we will use locking and updating mechanism. Locking will be implemented for the role Floor Holder. While updating mechanism is implemented to all users.
- In this system, to implement logging mechanism, there will be two dedicated component which functioning as logger; History Logger (HL), and Temporary Logger (TL). The histories of all committed events are stored in the HL. While the events before a user who has the lock decided to commit or abort are stored in TL.

- In this proposal, a logger is not only store the data, but also implements synchronization and updating protocol. Therefore, loggers will actively communicate with each other. This mechanism is also applied when generating components. Through a proxy, client command the proxy to generate objects. Local proxy will communicate to the other proxy with the same mechanism.
- Locking mechanism will be applied when a user changes his role from Member to Floor Holder. Locking mechanism begins by the time the floor hands to him. There will be no further problem of locking in sequential and free floor control model, since there is one floor holder at a time. But when shared floor control model is used, by the time user click an object, locking is applied. After he finished working on an object, he should choose either to commit or to abort. If for some period of time, he does not make any decision, the system will prompt him to make a decision.
- The development of this groupware is designed to be implemented by using JavaBeans which based on Java language. Classes, which are best written in modules, will be written in Java classes, not in the form of Beans. The communication of distributed components will be using the advantages of CORBA that is supported by Java.

5.2. Future Work

This thesis work describes only the design part of a synchronous groupware. Therefore, implementation of the design through coding and sufficient testing should be performed. In this design, there are still no security feature, QoS and fault tolerance. Therefore, further design concerning those features should be performed.

In this design, we did not consider how such complicated creation of objects, updating and logging mechanism and many others, will it be smoothly work on a congested internet. Therefore such implementation and testing in a real worldwide network should be performed.

CSCW groupware developers always work individually. Applications are usually developed from the scratch, while many ready, well knowingly by the users could not be implemented in the newly developed groupware. In the future work, it should be investigated how to use a ready made application, for example open source application or freeware, inside this groupware.

In the software design, we did not concern about versioning. In mobile application era like now, a light version of software is needed since everyone move so fast. High end handheld devices, like PDA should be considered as a good market, and in vise versa the groupware application will be a value added service to the telecommunication vendor/operator.