Construction Phase Project Status Assessment

18 September 2017

1 Executive Summary

Confide Instant Messaging App has achieved the majority of requirements identified in the Project Plan. The Project Plan was submitted to the project sponsor as part of the final Life Cycle Architecture Milestone (LCAM) submission on 14 July 2017.

1.1 Project Plan Achievement

The functional requirements are summarised as follows:

1. Create an account on the service (username/password)
2. Single user (one-to-one) chats
3. Group chats (set up group chat, participate in group chats)
4. Contact management (invite/accept/reject/delete) for single and group chat.

1.1.1 Functional Requirements

Functional requirements (1) to (4) have all been achieved. App users are able to create their own accounts (1), initiate contact with other users and accept or reject requests from other users. Users are also able to create group chats and invite users to join these. Alternatively, it is possible to accept an invitation to join someone else's group chat.

Non-functional requirements included:

1. Availability on the Windows desktop platform
2. Open source chat client and chat server
3. High availability for 24/7 chat with "real-time" user experience
4. User workflow (provide an intuitive, quick way to communicate using the app)
5. Secure chat

An open source Java server and client chat library was chosen to meet requirements (1) and (2). These products use XMPP, which has become the de-facto messaging standard. Our chat server has demonstrated stability and speed over the entire course of Confide development.

Team Orange is confident that it will be able to scale up to higher volumes of use under real world conditions. Therefore requirement (3) has also been satisfied.

Requirement (4) will be tested during the User Acceptance Testing (UAT) phase. Significant improvements in presence and new message alerts over the final construction iterations have made the app more intuitive to use.

1.1.2 Non-functional Requirements

Non-functional requirement (5) was nominated as a stretch target in the use case revision document. We did not achieve "Message Confidentiality" which was related to a low priority use case (UC0006 Secure Chat) and a low priority performance requirement (2.2 message confidentiality), and was only in scope if we were to go for our stretch target. Without completing this use case, we were still able to reach our standard target. Due to a lack of time, we decided that

our efforts would be better spent improving the application and adding other features, instead of attempting “Message Confidentiality”.

1.2 Alpha Testing

All planned functionality has been achieved, and tested thoroughly with our alpha UAT scripts. There were no major defects to be addressed. The software is now ready for UAT/Beta Testing.

1.3 User Manual

A User Manual has been prepared to assist with the testing, and we believe that it is sufficient to allow any user to easily achieve any task within our application, therefore sufficient to support beta stage UAT.

1.4 Beta Testing

A set of beta testing forms were prepared in spreadsheet format. In order to expedite the testing, which will occur in three geographic regions, an online form has been prepared using Google Forms. It is anticipated that this will streamline the process of interpreting the test results. The beta tests follow a sequential order. The later tests rely on results achieved in the earlier ones. The tests are:

1. Account Logon/Logoff
2. Add a Contact and chat with them
3. Receive an invite to chat /chat management
4. Contact Management
5. Create a Chatroom

These tests should exercise the main functionality that users will want and need in their daily interaction with the app.

2 Detailed iteration analysis

This is more of a detailed project management report. What I want you to do here is to go through each iteration, identify the primary goals of that iteration and report on how you went in that iteration in regard to achieving that iterations goals. In particular, I want you to identify any issues that arose in that iteration, and report on what you did in response to that issue. This includes issues that arose and were resolved quickly.

2.1 Construction Iteration 1 (C1)

2.1.1 Goals

C1 kicked off the second part of the project (309) with a large list of features planned to be implemented. These were:

1. Team meeting planning
2. LCAM Resubmission
3. Implement Group Chat in BabblerBase section
4. Implement Group Chat in core app (MVC)
5. Test Group Chat functionality

2.1.2 Work Completed

Shaun:

Worked on the Multi-User Chat GUI. Also performed research on the JavaFX, lambdas, and the MVC pattern. Functionality of Multi-User Chat messaging screen was postponed due to lack of time. Some functionality within the Multi-User Chat management screen was postponed due to a lack of time.

Tim:

Worked on the Multi-User Chat backend and XMPP layer. Functionality of becoming a Multi-User Chat Member was left, as it was troublesome to implement, and being an occupant seemed to be sufficient for our needs. Functionality of destroying a Multi-User Chat Room, inviting contacts to Multi-User Chats, and responding to Multi-User Chat Invites; was postponed for now due to a lack of time.

Murray:

On leave and unable to work on this iteration.

2.1.3 Issues

Time restraints implementing all of multi user chat functionality. The work load to implement all features was underestimated. The team decided to break components up over the coming construction interactions.

2.1.4 Summary

The team made significant headway towards implementing the group chat functionality. However the implementation included quite a bit of complexity including the "XMPP way" of doing group chat. Murray was on pre-arranged leave for the entire iteration, and the LCAM resubmission was due. These factors meant that some of the group chat functionality needed to be postponed until the following iteration.

2.2 Construction Iteration 2 (C2)

2.2.1 Goals

1. Implement and test group chat functionality
2. Implement contact management (delete, mute, block, presence)
3. Add ability to logout
4. Improvements to the GUI, including some incomplete Multi-User chat features from C1.

2.2.2 Work Completed

Shaun:

Multi User Chat feature fully integrated and functional test performed. Low priority feature Accept/decline invitation deferred due to time restraints.

Tim:

Implemented backend and XMPP functionality of blocking, deleting, and presence. Implemented logout functionality. Attempted to implement functionality of joining a Multi-User Chat as a member, however there were some issues and it was not working; postponed until a later iteration. Implemented and improved a bunch of random features.

Murray:

Created presence UI object, however ran into issues connecting it to the given user's actual presence. Discovered more about the architecture, and how to go about implementing this feature after discussions with Tim and Shaun. He was not able to complete this. At this stage of the project it was not a good use of resources to divert Tim and/or Shaun to assist Murray with coding. The team decided Murray should focus on testing and documentation instead.

2.2.3 Issues

- Unable to join Multi-User Chats as a member, the functionality was postponed until a later iteration as it was not necessary at this time

- Unable to connect the UI presence object to the given user's actual presence, the functionality was postponed.

2.2.4 Summary

The team continued to make good progress towards implementing the main functional requirements. Some issues were encountered with multi-user chat membership and presence, these were not completed this iteration as planned.

2.3 Construction Iteration 3 (C3)

2.3.1 Goals

The main focus of this iteration was refinement of usability. We wanted the user to be able to move easily between the various functionality provided by the app with minimal mental overhead. In particular, it should be clear to the user what state of the chat process they are in at any given time.

2.3.2 Work Completed

Shaun:

Large efforts were made to accommodate the message notification feature. Fundamental changes had to made in the way the contact screen view was updated and created to accommodate the notifications components. Due to time constraints, the following features were not implemented: displaying a user's presence, and displaying the name of the user currently logged in.

Tim:

Implemented a stronger connection and message reliability. Resolved issue of Multi-User Chats no longer being created as persistent, users had to have entered the room before configuring it to be persistent. Tim resolved chat room discussion history retrieval issue, had to set up listeners before logging in, and request discussion history when logging in.

Murray:

Focussed on planning and beginning documentation for the IOCM submission. He had some issues with his build environment. He also reviewed critical core use cases.

2.3.3 Issues

- Multi-User Chats no longer being persistent (resolved)
- Chat room discussion history was no longer being retrieved (resolved)
- Murray had an issue with his build environment (resolved)

2.3.4 Summary

The application was becoming more polished and feature complete. Some issues were encountered during the iteration but these were all resolved.

2.4 Construction Iteration 4 (C4)

2.4.1 Goals

Focussed on implementing:

- User interface for some backend features that already exist.
- Multi-User Chat bookmarks and invitations.

2.4.2 Work Completed

Shaun:

Implemented UI features for: block contact, delete contact, presence, displaying contacts in Group Chats. Major time spent restructuring app to comply with MVC and load pattern. Resolved a bug where notifications were incorrectly set upon login, this was resolved during the restructure. Resolved a bug where contacts would be doubled upon logging in then out then back in, fix not needed because Tim implemented a reset function.

Tim:

Created and mostly completed a user manual. Implemented Multi-User Chat bookmarks (persistence) and invitations. Resolved the bug where Multi-User Chats were not loading new messages, the issue was that a lambda was not being set when the Multi-User Chats were being created from a bookmark. Resolved a bug where new users going online was not reflected in their presence, there was actually no issue, the issue was that we were using two old users that were created in an old version of our program, and were not properly subscribed to each other.

General improvements: Back button added to chat sessions, in a chat session your partner's presence is displayed, navigation buttons reflect which screen you are currently on.

Murray:

Created drafts for the Project Status Assessment. Worked on alpha and beta stage UAT tests. Reduced complexity of test model. Reviewed Tim's User Manual.

2.4.3 Issues

- A major restructure was necessary to comply with MVC and load pattern, and to allow further features to be added on.
- Bug fixes as follows:
    - notifications were incorrectly set upon login (resolved by Shaun)
    - contacts were doubled upon logging in (resolved by Shaun and Tim)
    - Multi-User Chats were not loading new messages (resolved by Tim)
    - new users going online was not reflected in their presence (resolved by Tim)

2.4.4 Summary

This iteration continued refinements and improvements to the applications operation. Several bugs were identified and resolved during this iteration.

2.5 Construction Iteration 5 (C5)

2.5.1 Goals

Final missing functionality was fully implemented:

- contact delete and block functionality.
- Multi-User Chat invite and delete functionality.

Additionally, finalising alpha testing to ensure that the application had no known bugs and was ready for beta testing.

2.5.2 Work Completed

Shaun:

- Finished UI delete block features
- Slight modification of invite MUC workflow
- Review of ICOM documents

Tim:

- Finished contact delete and block functionality
- Finished off MUC delete and invite functionality
- Completed user manual
- Provided input for Alpha Testing
- Worked on status assessment

Murray:

- Conducted and documented alpha tests
- Worked on status assessment
- Provided review and editing of user manual
- Finalised beta testing documentation and test model documentation

3 Concerns and Deviations:

3.1 Technology Stack/Design Decisions

During the project, we faced the usual challenges of understanding the problem domain, in particular learning a framework which abstracts away the complexities of the underlying layers. The XMPP protocol imposes certain ways of doing things. These are sometimes open to different interpretations. In addition, none of the team had previous experience with the server technology chosen. We could observe some side effects and possible issues, but at times it was difficult to determine whether the server or our code was at fault.

At the outset, the team decided on a Model View Controller Architecture. This was enhanced by Java 8's lambda feature. In hindsight, this was a good move, however it did impose a fairly significant learning curve in the first part of the project, for those unfamiliar with using it.

3.2 Rational Unified Process

Our team was able to utilise the RUP to a significant extent. As with any project methodology, it is difficult to estimate the time to complete tasks with certainty. Due to the geographic separation of the team and the need to complete other (unrelated) projects, it was sometimes difficult to get good coordination of effort, in a way that would be more easily achieved in a co-located environment. However, the communication tools chosen by the team worked well.

In terms of design, it would be fair to say the design “evolved”. GUI design and usability are a very large component of a chat app, as far as the user is concerned they want the interface to get out of the way. The user should be able to “just communicate” with minimal mental overhead in actually operating the software.

Our team was probably a bit ambitious in its early expectations of what could be achieved, and this had to be reined in. Fortunately, we were able to define which functionality was “stretch” targets. This avoided the disappointment (and potential penalty) of not delivering promised features.

In terms of a “classic” RUP project, the distinction between the construction and testing phase was blurred. This proved to be beneficial in our case, as already stated, GUI and user experience forms a large part of the viability of the application. Alpha testing became formalising that what we had previously tested still worked, plus a few minor bug fixes.

3.3 Avoidable issues

3.3.1 Skills imbalance

There was a difference in coding proficiency in the team. Tim and Shaun were quite proficient in Java, whereas Murray was not as experienced. Murray tried to compensate by completing more of the administration tasks, but it meant that he slipped further from the possibility of making meaningful and timely contributions to the code. His absence at the beginning of semester 2 of the project made the knowledge gap even wider.

To avoid this, we should have made sure that all team members were maintaining a working understanding of the code base.

3.3.2 Technology problems

Murray had an issue with his build environment. The root cause of this issue is unknown. This issue may not have been avoidable; however, it could have been resolved very quickly if there was greater team communication, and we all put our efforts into resolving it.

3.3.3 Code Style/Discipline

A major restructure was necessary to comply with MVC and load pattern, and to allow further features to be added on. The root cause of this issue was allowing ourselves to deviate from the agreed upon patterns. This is a classic problem in development, i.e. “just get it done” can sometimes lead to technical debt and code which is hard to debug and modify.

This issue may have been avoided if there was greater team communication, then we could have discussed the issue and found a way to complete the functionality without deviating from our patterns.