# Elaboration Phase Status Assessment

## 1. Assessment against Objectives of the Elaboration Phase

### 1.1 Did we complete the expected outcomes for LCAM?

The aim of the project is to build a tool for resource management for an organisation﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿ this is embodied in the completed Critical Core Risky Difficult (CCRD) Demonstrator.

We understand the main CCRD use case to be Allocate Resource.

This is shown in the completed Full Use case Description Allocate Resource

We understand the main functions of this use case are:

Enable quick locating of skilled resources which are available for a project

Ensure that resources do not get double booked

Ensure resources only get booked if their allocation is suitable.

We have a good idea of how we are going to achieve our aims. We are going to use a web application which stores data in a database, using various persistence frameworks. This is shown in the completed Architecture notebook

We have a good understanding of the project specific risks facing our project and how we are going to deal with them. The risks are:

Appropriately securing information

Sponsor maintains support

The team’s skillset is adequate enough

Scope creep / Under scoped

Our evolving understanding of risks is shown in the ongoing risk list and discussed further below in Section 4.

We have a good understanding of how we are going to check that our application delivers the intended functionality and system properties. Our key areas of concern and the test strategies we will use to address these concerns are as follows:

Functional testing

Unit testing

Integration Testing

User acceptance testing

This is shown in the completed Master Test Plan

We have a good understanding of the dependencies and likely completion times for different parts of the project. Target completion dates for key aspects of the project are as follows:

*Architecture:*

***13/05/2019***Implement 1st & 2nd highest priority to support CCRD use case

*Functionality:*

***24/6/2019***Top 2 CCRD use cases to be implemented

*Testing:*

*Iterative testing will be performed as new functionality is added. Once stable a UAT of that functionality will be performed.*

*Documentation:*

***26/08/2019***Complete user documentation - how to use the system for each user, admin documentation that explains how to configure and setup system.

This is shown in the Project Plan.

### 1.3 Skills required

Our project requires skills using the following key tools and technologies:

*MySQL*

*Java*

*Maven*

*Git*

*Tomcat*

*SpringBoot*

*JPA*

*Hibernate*

*JavaScript*

*HTML*

*CSS*

We have demonstrated that we have the skills to use these technologies through the implementation of the CCRD Use Case.

## 2. Deliverables

### *2.1 Project vision*

*This document contains only a rough outline of the intended application; there were no issues in producing the document*

### *2.2 Requirement model*

*Fleshing out further requirements for this component was performed throughout the Elaboration phase. We will be implementing a lot of these during construction phase. Outlining the CCRD use case was a high priority and we managed to achieve it.*

### *2.3 Proposed architecture*

*This document consists of how we will combine all required components together: No issue.*

### *2.4 Risk List*

*The risk list is a document that can never be completed until the project is finished, as new risks and issues can arise at any time. This document is well written and contains key components which the team must look out for. There were no issues in producing this document*

### *2.5 Master test plan*

*No issues*

### *2.6 Initial project plan*

*We had issues where our plan was not properly outlining the work that needed to be performed we have fixed these issues and now the plan outlines the work required more thoroughly.*

### *2.7 CCRD Implementation*

*We had some issues implementing the persistence layer with JPA and Hibernate as they are quite strict. They do not permit having objects in the memory cache not match the persistence layer after an update. This means all relationships in terms of one-to-one, one-to-many and many-to-many and their directionality of uni or bi-direction* ***must*** *be correctly defined in each of the models. The code must reflect the directionality by cascading changes to the children of a parent and the parent of a child if an update is made to one. This ensures the state is consistent at the memory cache and persistence layer levels, which is great in that is supports our requirements for reliable and bug free code, but that level of thoroughness is also a lot more difficult and time consuming to implement. We spent an inordinate amount of time just tweaking the relationships to make sure everything cascaded as it should without throwing an error. This was especially an issue for the CCRD as it involved a three way many-to-many relationship, Many projects which require many skills and allocate many resources that are proficient with many skills.*

*Unit and Integration testing also had many growing pains, due to the SpringBoot architecture chosen. Spring is noteworthy in that it uses automated annotation based dependency injection to build the applications ‘context’. This means at any level the programmer can’t just instantiate a new object that uses Spring’s @Autowired annotation, that has to be done by Spring for it to be aware of that object in the application context. This took some time to adjust to, especially for unit tests as typically in unit tests one would mock all objects in the context except for the Class under test. That being said, once we had worked that out, it was a great time saver setting up and tearing down the testing environment before and after each test, ensuring there was no contamination between tests.*

## 3. General Issues

### 3.1 *Communication within the team*

*The team had problems during the elaboration phase where we did not communicate when tasks were completed or when tasks were available for the team to work on. This lead to at least one wasted iteration.*

*This issue has now been resolved and the team is communicating more frequently.*

## 4. Risks

### *4.1 Poor use of version control*

*This risk can cause project delays if the team does not appropriately use version control which will cause the project to fail.*

*The mitigation strategy is to use a common version control system with available help online.*

*This risk is resolved.*

### *4.2 Team has inadequate skill set*

*This risk can cause the project delays in the form of team members only learning requisite technologies when they need them which can make developing the application take longer than required.*

*The mitigation strategy for this is to use skills taught during the CSU Computer science bachelors and to occasionally have skill quizzes to ensure the teams skills are up to date.*

*This risk is still open but being managed*

### *4.3 Team members leave the project*

*Team members leave the project, due to the small project team some use cases will have to be removed.*

*The mitigation strategy is to ensure that team members are all well communicated to and for the team to be open with each other to manage expectations.*

*This risk is still open and being managed through ensuring the team airs any issues when possible.*

### *4.4 Inadequate security*

*If the program is not secure enough to store data then the users will not use it.*

*Will ensure the security of the data is appropriate to the requirements.*

*This risk has been mitigated through HTTPS and implementing a login page*

### *4.5 Incapable of providing needed functionality*

*The risk of not meeting the client’s needs can lead to refusal of product purchase thus a waste of time and money on part of the developer’s alongside a potential loss of reputation.*

*The mitigation strategy is to establish the project requirements and selecting technologies that can deliver the services required by the client.*

*This risk is being mitigated by iteratively performing UAT with the client.*

### *4.7 Team unable to identify or procure technologies*

*If team members are unable to determine the technologies that should be used, the project will cease to proceed past the planning phase.*

*The mitigation strategy is to investigate early into the technologies to be used.*

*This risk is resolved.*

### *4.8 Team continues to communicate poorly.*

*During the Elaboration phase the team had communication issues which are now resolved.*

*If the team continues this trend the project will likely fail.*

*This risk is being managed by ensuring an iteration manager is responsible to ensure delivery of all artefacts.*

## 5 Summary – Overall Project Progress

*Over all our progress against the Elaboration phase is complete.*

*the team needs to ensure that we better focus our attention to the required skill sets specifically:*

*Hibernate: JPA persistence.*

*Java Servlet pages.*

*The team has had some issues with communication, Attention should be focussed on ensuring communication is maintained into the Construction Phase*

*While the team did some upskilling early on a lot of the development of the current CCRD use case was performed by a key team member, the rest of the team needs to ensure they upskill over the break to ensure that they can equally contribute to development.*