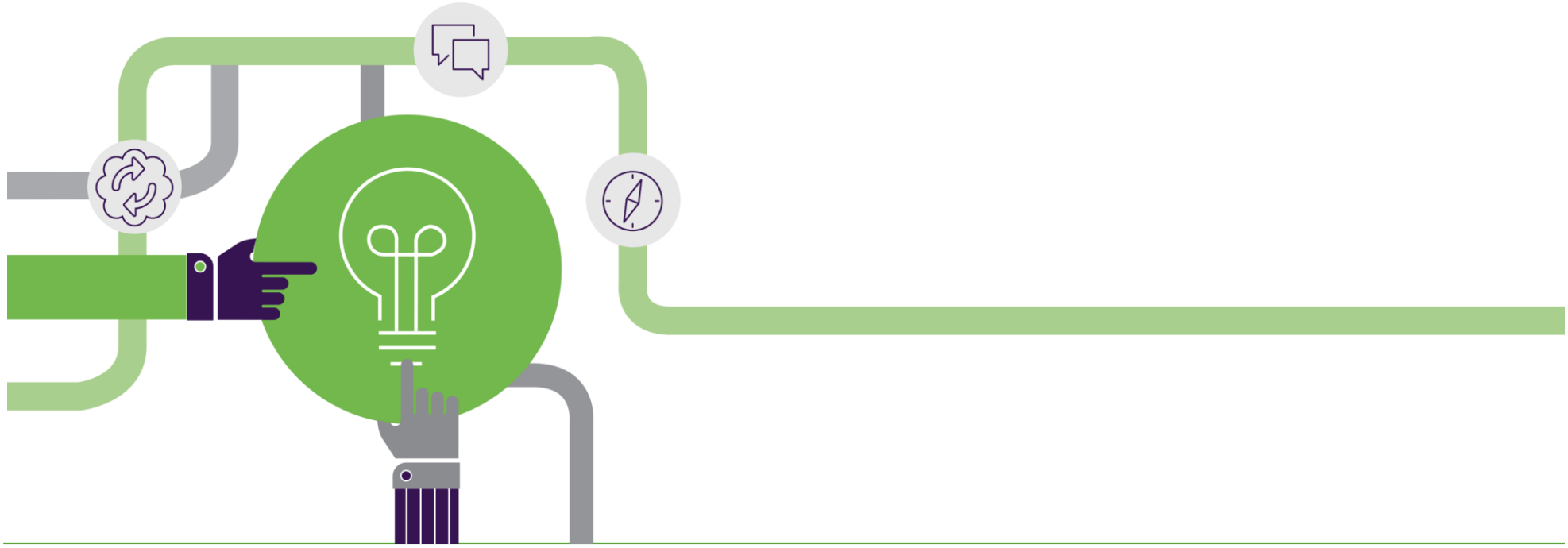
Bluefruit®

Client: Take Me There Lifts
Document no: TMT-001-D01-A
Author: Byran Wills-Heath

Inception

Software Feasibility Study

Up and Down Lift – Revision A



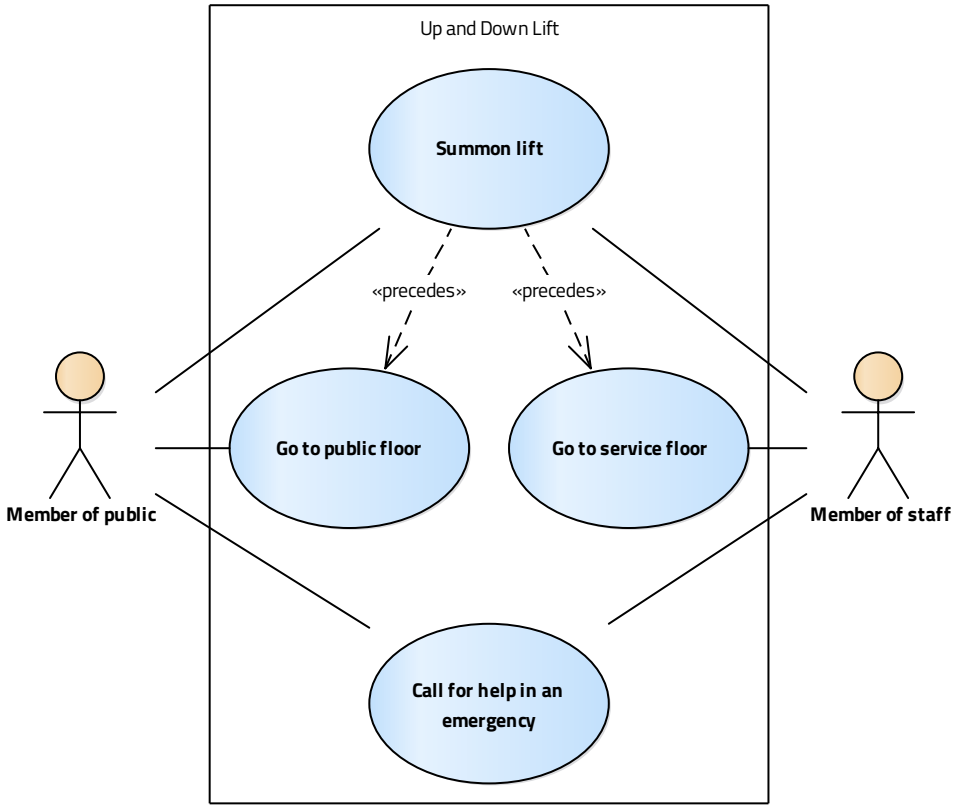
Embedded Software Specialists

Model

Overview

The main purpose of the system is to transport members of the public between different floors of a building. The system will allow members of staff access to designated service floors while blocking member of the public from accessing these floors.

Initial System Use Cases



Other

The software must be compliant with BS EN 81-41:2010 “Safety rules for the construction and installation of lifts. Special lifts for the transport of persons and goods. Vertical lifting platforms intended for use by persons with impaired mobility”.

Implementation

Bluefruit will develop initial hardware prototypes to facilitate initial development using a 3rd party hardware developer.

Test

Proposal

The proposed testing consists of four levels:

- **Unit testing** – this the testing of the software components. It is delivered using a method called Test-Driven Development (TDD), which is an approach which involves writing and validating tests before functionality is written. These unit tests are automated.
- **Executable specifications** – this is the testing of the business logic of the software. It is delivered using a method call Behavioural-Driven Development (BDD). This involves agreeing acceptance criteria at the start of a sprint in a plain English format, and then the team automate these tests. In addition to verifying specifications have been fulfilled, these tests also act as regression tests.
- **Manual testing** – this testing is performed by the Software Testers, it includes confirming the software conforms to the above acceptance criteria, when run on the target hardware. It also includes break testing.
- **Acceptance testing** – this is the final testing performed by Take Me There Lifts to confirm that the software performs as desired. This testing can act as a final safety net to catch defects, but more importantly its focus is to ensure the software is aligned with the business case and branding of the product.

Costs and Benefits

Each of the four levels of testing has its own costs associated costs and benefits described below:

Unit Testing (TDD)

Unit testing is entirely delivered by the Software Developers. Research has shown that TDD can take 15%-35% longer than writing code without TDD. For this cost, it delivers the following benefits:

- **Better tested code** – defect density is typically reduced by 40-90%
- **More habitable code** – TDD enables developers to continually optimise the design. Habitability of code is critical where maintainability and scalability are considered important.

- **Guaranteed up-to-date documentation** – the unit tests perform a key role of documenting the code. This documentation is guaranteed to be up-to-date (otherwise the tests fail). The existing Take Me There Lifts code is 54% comments, TDD offers a much more valuable and efficient alternative to this effort.
- **Improved architecture** – to write unit tests the code must be well-architected. This is important for the future maintenance and scaling of the code base.
- **Team ownership** – unit tests enable developers (other than the original developer) to modify the code with confidence. This is important where you don't want to be tied to a single developer either for contingency or scaling reasons.
- **Regression testing** – automated tests help ensure old functionality is not broken when new features are added.

Executable Specifications (BDD)

The main impact of the BDD process is the Three Amigos meeting at the start of each sprint and involves the whole team and a Product Owner (from Take Me There Lifts) for approximately 2 hours. This time is arguably needed in any case, as there is always a need for some investment of time in the acceptance criteria for features.

The benefits of this approach include:

- **Clarity** – experience has shown that bringing all documents and people into a single point to verbally confirm a common understanding using rules and examples is the most effective approach to eliminating ambiguities.
- **Reduced waste** – by removing these ambiguities and "closing the loop" on a regular basis, this maximises developer effort by avoiding "rabbit holes" and deviations from the optimal development path.
- **Traceability** – the executable specifications robustly bridge the gap between requirements documents and the delivered software in an extremely traceable way. This approach provides an easy way to track requirements coverage and also perform impact analysis on change requests.
- **Regression testing** – as with TDD, automated tests help ensure old functionality is not broken as the project progresses.

Manual Testing

Manual testing is performed by the Software Testers. We allocate one Software Tester to every four Software Developers, so nominally a team of two Developers will get half of one Testers time. Tester time is not charged in addition to the Developers' daily rate (see above) and is part of our investment in Quality.

The benefits of manual testing occurring at Bluefruit are:

- **Early feedback** – our Testers perform software testing throughout the sprint. This gives the Developers early feedback on issues, which reduces the effort required to address issues.
- **Break testing** – our Testers are all trainee developers, so they have a deep understanding of how software development works. This gives them strong insights into how to break software. The perceived reliability of software by end-users is strongly related to how well "rainy days" have been covered.
- **Strategic testing** – our Testers are focussed on how to test your project. For example, from the very first design meeting, they will be asking "how will this feature be tested". Experience has shown having dedicated members of the team focused on this aspect dramatically improves software quality.

Acceptance testing

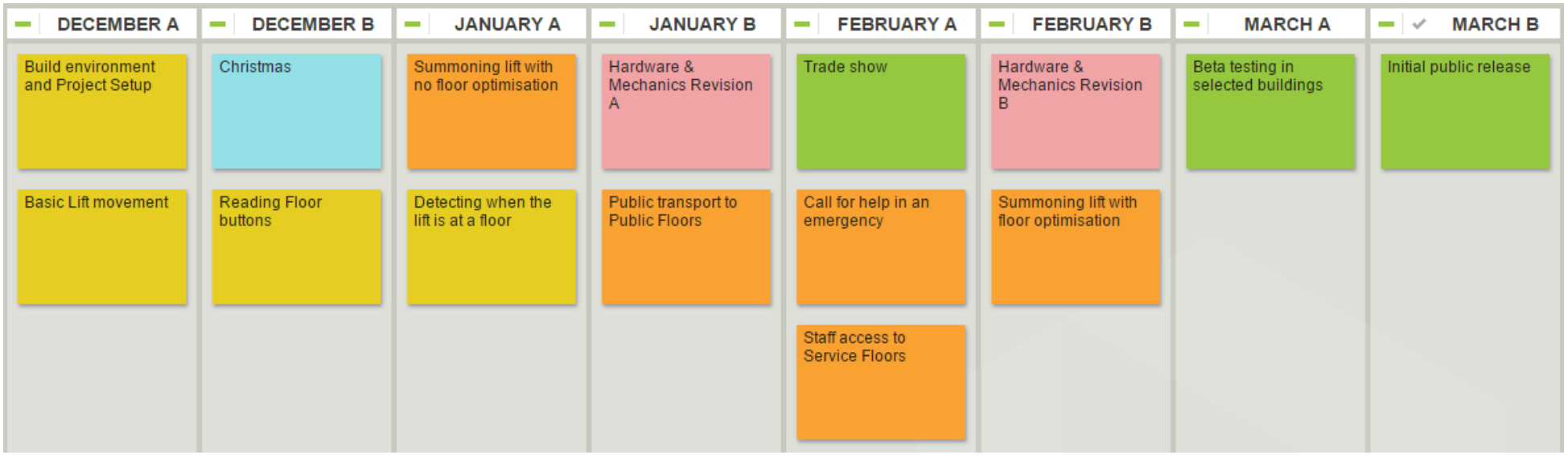
We would not suggest that any of the above testing is a substitute for client acceptance testing on the delivered software, although we usually see that the value from this effort is increased as our client's time is more focussed on ensuring the software is fit-for-purpose (i.e. well aligned to the business case and branding of the product) and less time spent bug-tracking.

Deployment

Timescales

- Prototype Version – 5th February
- Beta Version – 3rd March
- Release Version – 27th March

Roadmap



Key:

Events	Goals	Features	Hardware/Mechanics	Technical Tasks

Process

Our aim is to produce the highest quality software for you, to facilitate this our process is tailored towards assisting in producing high quality software.

Iterative and Incremental

We propose that the software is developed in an Iterative and Incremental manner. Short iterations allow you to provide us with regular feedback. This makes sure what is being developed is what you want and allows you to change the requirements as you see your product evolve over time. Incremental development allows the development of the key sections or high risk areas of the software first providing a basic but functional version of the product at an early stage and polishing the features in later iterations.

Iteration process

To achieve regular communication and feedback between our team and you we recommend working in two week iterations. This allows us enough time to develop software that adds meaningful value to your product while minimising the time it takes for us to receive feedback from you about how the product is evolving.

To facilitate this, we propose the following happens during an iteration, although you are only required to attend some of the meetings, the more you participate in, the smoother the project will run.

Three Amigos meeting (required attendance)

This occurs on (or before) the first day of the iteration, it's purpose is to discuss the work that is going to happen during the iteration and for the participants to agree the completion criteria for the work. It's called a Three Amigos meeting because it requires at least one representative from the customer, development team and test team to be present.

Planning (optional attendance)

This occurs as soon as practically possible after the Three Amigos meeting and involves the development and test team. Its purpose is to decide how the work that was discussed in the Three Amigos will be broken down. Although you are welcome to participate in this meeting, most of our customer choose not to as it's about how the software will be implemented instead of what the software will do. This tends to be a very technical meeting. (Do we mention pig and chicken roles?)

Stand ups (recommended attendance)

Every morning we meet to discuss how the iteration is progressing and what everyone will be working on during the day. We recommend you have a representative at this meeting but you do not need to physically attend, most of our customer use a form of teleconferencing (usually Skype) to join in.

Demonstration and Iteration review (required attendance)

At the end of every iteration a working increment of the software will be released. We will demonstrate the features that have been implemented to you. Although you may have had a representative at the Stand ups who knows how the development is progressing, this allows a wider group of people from your organisation to see progress without needing to participate on a daily basis.

After the demonstration a summary of the work that is targeted for the next iteration is planned.

Retrospective (recommended attendance)

Every iteration ends with a Retrospective; this allows the team to discuss and plan improvements that they will enact in the next iteration.

Configuration Management

All source code and resources will be stored in a Git repository hosted on GitHub.

Access to the GitHub repository can be provided to Take Me There Lifts if required.

Documentation will be stored on Bluefruit Software's Wiki or in the Git repository.

Source code can also be provided in a different format on request.

It has been suggested that:

- LeanKit will be used for Project Scheduling.

Source Code Management

To track and keep safe the source code safe we use a Source Code Management system called Git. Git is an open source SCM system originally developed for the Linux Kernel which has gained wide spread usage within the software development community.

We propose to use GitHub to store the main copy of you source code. GitHub is an online git repository service providing a globally accessible secure location to store and access to your source code. All of our customers GitHub repositories are private meaning that they are only accessible to people who we grant access.

Project Management

Take Me There Lifts Contacts

- Project Management/Technical – **Chris Roberts**
- Commercial – **Nicholas Smith**

Bluefruit Contacts

- Project Management/Technical – **Matthew Verran** / Byran Wills-Heath
- Commercial – **Paul Massey** / Steve Forth / Byran Wills-Heath

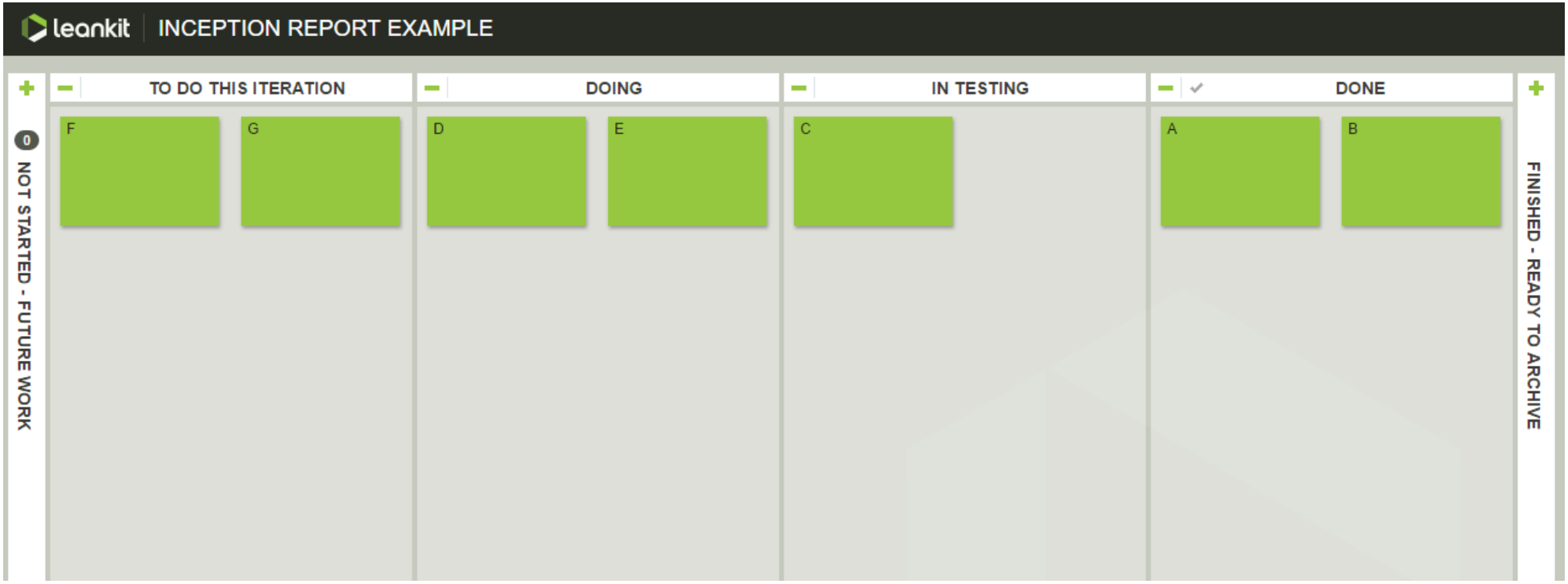
Budget

The budget has been allocated for this project.

Progress Tracking

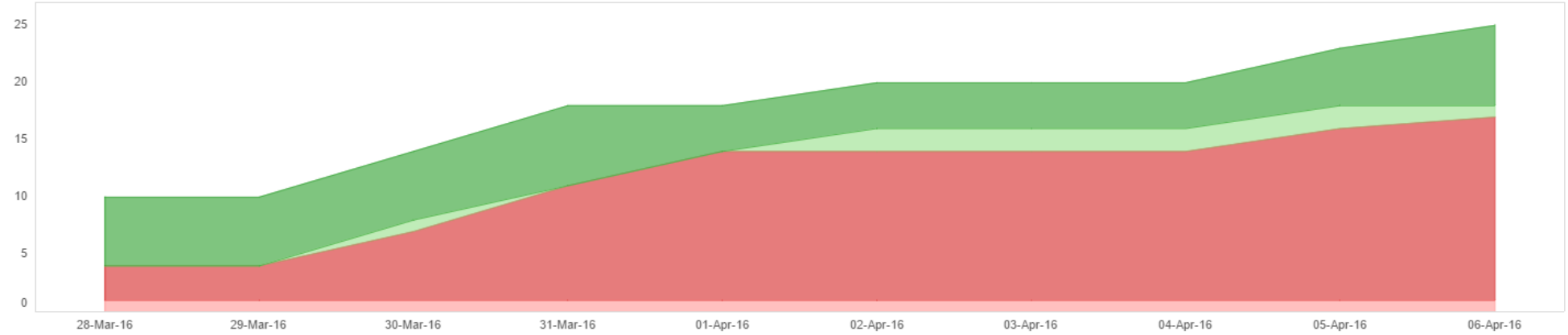
Kanban Boards - LeanKit

We propose to use LeanKit to track the progress of the project. LeanKit is an on-line Kanban board that allows virtual cards representing items of work to be tracked in a form that is accessible to the entire team.



LeanKit is used to track all work that the team performs; as well as the features they are developing, it tracks writing of any documentation that is requires, issues found in the current release, etc…

Within LeanKit there are many different ways you can view the progress of the project, we recommend using a Cumulative Flow graph to see the progress of your project.



Bluefruit day to day tracking

As well as using LeanKit we track various other metrics and progress indicators about your project. These are displayed on Information Radiators that located at various locations around our offices and allow any member of the team to see at a glance the current progress and status of your project.

Team

We propose to provide you with a team consisting of a Team lead, Team deputy, 2 Developers and 1 Tester.

Team lead

The Team lead will oversee all of the aspects of your project and will be your primary point of contact. The team lead will arrange with you, times for the various meetings that will occur throughout the Iteration and will be available to discuss with you any aspects of you project.

Team deputy

The Team deputy provides cover for the Team lead in case they are ill or on holiday and will be your second point of contact.

Developers

The developers write the code, unit and integration tests for your software.

Testers

The Testers are responsible for ensuring that the software is system tested. They will along with you and a representative of the development team develop a test suite for your software that is used as a completion measure for an Iteration.

The testers maintain a regression suite of tests which they will run at least once every three Iterations to ensure that changes introduced in recent Iterations has not broken features from previous Iterations.

Cover (i.e. for sick)

We propose to provide you with 4 covered developers (1 x Team lead, 1 x Team deputy and 2 x Developers). This means that there will on average be 4 developers working on your project for the entire Iteration. In reality this may mean there are 5 developers working on your project for the first half of the Iteration and only 3 during the second half of the Iteration.

Barriers/Risks

Risk	**Likelihood (1-5)**	**Impact (1-5)**
Late delivery of hardware & mechanics revision A	3	1
Late delivery of hardware & mechanics revision B	2	5

Environment

The system will be developer using GCC ARM Embedded compiler version 5.0.

Access to online systems

The following people from Take Me There Lifts will be given access to Bluefruits online systems

- LeanKit – Chris Roberts
- GitHub – Chris Roberts

Core quality

Metrics

We track several industry standard code metrics to monitor the quality of the code we are producing for you. These metrics are used as early indicators of problems in the code and act as an early warning of impending problems.

We measure several metrics, the two most important to us are Cyclomatic Complexity and Depth. Both measure how complex the code is getting but in different ways. The higher the complexity the harder the code is to test, for this reason we keep our codes Cyclomatic Complexity and Depth low.

Build Servers

We propose to setup a build server at Bluefruit for you project to build all the releases you receive. Build servers provide a 'clean' environment (compared to developer PCs) from which your releases are created. The advantage of a clean environment is that the software installed on the system is known, so there are no hidden software dependences. In contrast using developer PCs to build software can become problematic because one developer may build the software and it works correctly but when another developer builds the software it doesn't functionally correctly because it was built using a different version of a certain tool or it will not build because of a missing tool.

As well as proving a clean build environment we also use our build servers to automatically run all the tests we write so software cannot be released without all the tests passing. This removes the opportunity for human error where a developer ships a release forgetting to run a certain set of tests.

As a quality check our build servers also calculate key industry standard code metrics and fail the build if the code does reach an acceptable level. This provides an additional automated quality check on the code that cannot be forgotten about.

Code Reviews

We propose to use peer code review to check the code quality of the source code entering your code base. Code reviews occur before a developer merges their changes into the main line version of the code base. This provides a safety net to check that potentially incorrect or substandard code does not enter the main code base.

Training

We want all of our staff to be up to date with the latest development practices and technologies. We believe that this gives the projects we work on the best possible chance on succeeding by producing high quality software using the latest techniques. To accomplish this, we use a mixture of in-house training and external training.

We run weekly training sessions on topics that are either useful to the current work being undertaken or on new techniques that we could potentially use. Once a month we have external trainers visit who assist the teams with their development practices assisting them on how they can improve what they are doing. We allow all of our staff to attend conferences so they stay up to date with the latest advances in development practices and technology.

For subjects that require more intense training we regularly bring in external trainers to run multi day courses.