Loughborough University

Atos IT Challenge

Project Initiation Document

Team Hydra:
Oliver Woodings
Jay Vagharia
Simon Kerr

October/November 2014

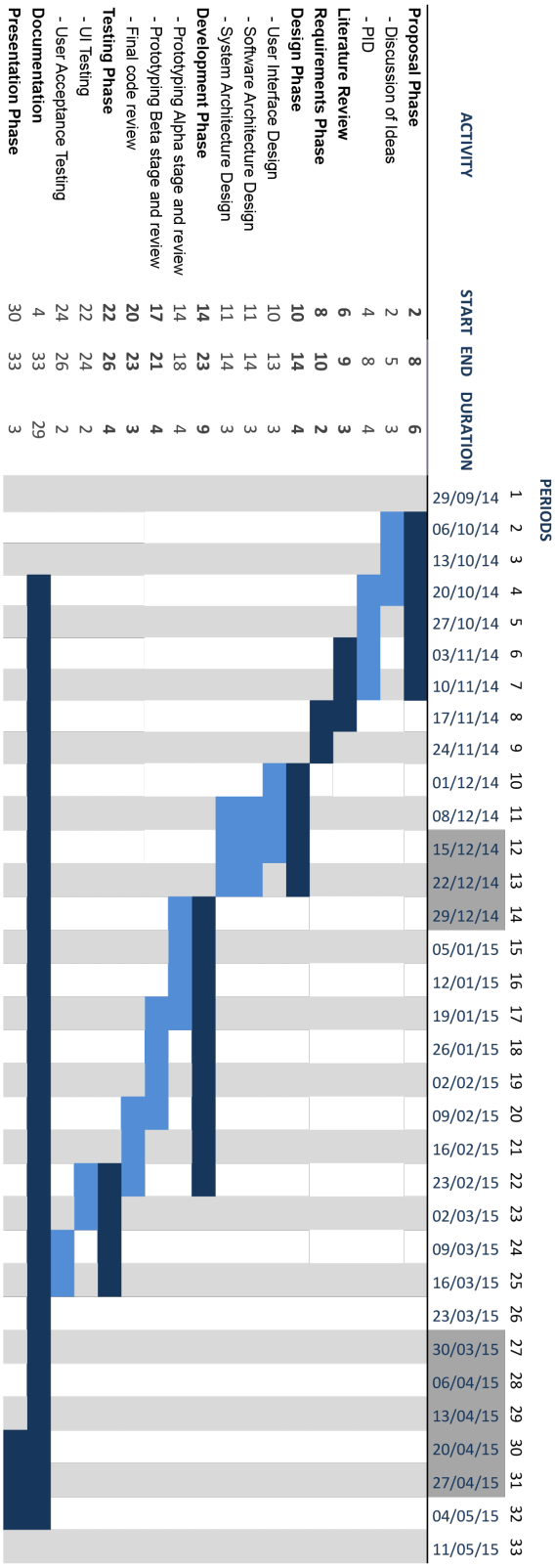
1 Introduction

The definition of *Connected Living* differs from person to person, but Atos believe that it is homes, workspaces and cities all seamlessly connected through smart devices providing connectivity anytime, anywhere. Consumers are always wanting to feel more connected to their workspace, their homes and their cities. With this in mind, we have decided to take the approach of connecting people with music.

Imagine being able to control and connect with the music in your surroundings, anywhere. Imagine being able to walk in to a coffee shop, library, bar and restaurant and suggesting your favourite songs that then play through the public audio media. Choona is an app focused around the 'Connected Music'. Our research has shown there is no music sharing concept that allows people to collaboratively listen, interact and suggest music to people in your surrounding area, home or business through an intelligent, cloud-driven playlist system.

Choona is a public music player that lets you have a say in what you listen to in public. It allows you to suggest songs you wish to be played at your location either right away or at a specified time. It provides you with the option to like/dislike songs suggested by others. Furthermore, you can connect through your mobile device allowing you to listen to the music privately (via your headphones).

2 Project Plan



3 Project Management

4 Ideas

Choona will be an app that can connect people to music in their location. Music is something that can be listened to at home, in the office or in public. We have created an app that allows you to do so.

- **Home** - In the home, music can be played throughout the home using one medium, but different playlists could be set up for each room, so different family members can listen to the music they want.

- **Office** - In the office, employees can stipulate what music they wish to listen to. This can be via the main system where the music is just something heard in the background. There is the option of using a mobile device to play the music privately where the user can plug in their earphones, listen to the music and zone-out the office background.

- **Surrounding area** - When we speak of your surrounding area, we are talking about restaurants, bars, coffee shops retailers and more. In this type of scenario, users can queue songs they want to hear. They can also interact by liking/unliking a song choice. For businesses, this could be a way to enhance a customers visit and entice them to come back again. The potential for this is huge because as well as homes, surrounding area and the office, it could be rolled out in hotels, library's and potentially public transport creating a large user base.

The app could allow the user to suggest songs they wish to be played either right away or at a specified time. The thought of having a song played at a particular time could almost be used an alarm clock for e.g. user could wish to play a song at 5pm which would subconsciously indicate the end of their working day or lunchbreak etc.

The system infrastructure is going to be built upon four main layers, as illustrated in Figure 1:

- **Source** - When we listen to music, we need to take it from a source. There are two different types of sources we need to consider for this project. First of all there are the virtual sources. Virtual sources are your software streaming packages such as 'Spotify', 'Youtube', 'Google Play Music' etc. At the moment, we will not consider iTunes as a virtual source because iTunes music needs to be purchased before streaming whereas with sources such as Spotify, a subscription to the service only needs to be purchased. Secondly, there are the physical sources. These include iPods, MP3 players and other hardware devices where the songs are stored as data files.

  We want to make sure that the app is not just configured for one music source, but a host of sources leading to a larger audience and a greater variety of music. As all these sources are different, we shall need to have adaptors in place where the data

is converted into the same format so it can be understood by our service application making it available to many sources and hardwares.

- **Service Layer** - We shall have a service layer that serves our app, provides us with value-added service such as song queueing and gives us access to our data sources. It will consist of a database that in essence will contain all the data we need to access the different music sources, communication between the API's.

- **Access APIs** - We will need to have certain API's in place that accomplish specific tasks. We will need an API that is responsible for streaming the right music from it's source, an API that takes care of the control aspect of the project e.g. which songs will play next, denoting the available songs etc. A management API will generally manage the whole system in terms of access rights etc.

- **Client Layer** - Finally we have the client layer which consists of the app and the hardware that outputs the music. The UI (User Interface) design must be 100% accurate if we are to gain a huge user base. The app makes it easy for a user to carry out a task and that it is intuitive in the sense that it's not hard to learn. The design needs to be consistent and minimilistic and finally bug free. When it comes to the output (i.e. speakers), adaptors may need to be in place so that the app can communicate with it. However, in the case of WiFi speakers, we should be able to stream the music wirelessly to the speakers.

There is potential to eventually integrate NFC within this app. Something that could be considered would be placing NFC tags around a shop, tables in a coffee shop etc. the playlist for that location would autmatically pop up in the app. This makes finding the playlist much simplier for the user.

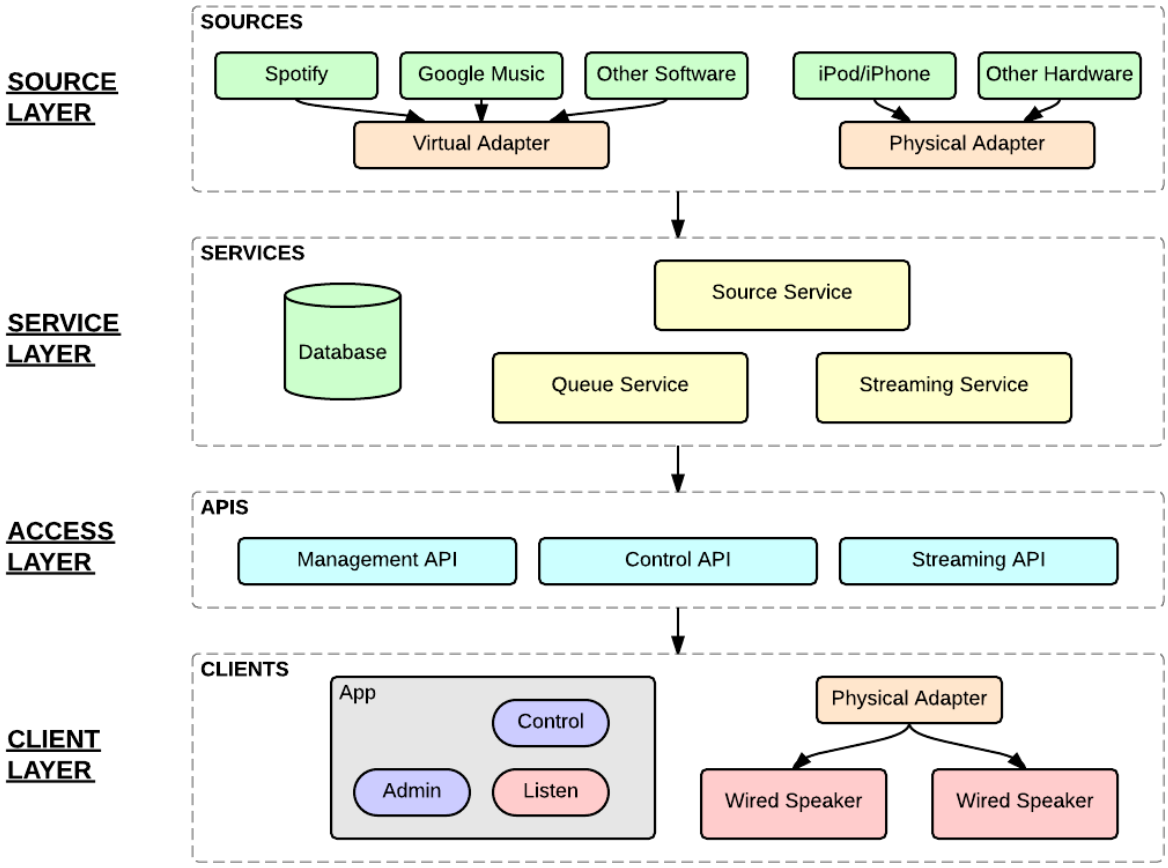


Figure 1: Concept system architecture

5 Literature Review Outline

Below, we have identified literature that will be looked at and discussed in order to feed the requirements stage with valuable information.

- **The Internet of Things** - The Internet of Things (IoT) is something that is spoken more frequently. It is the connection of identifiable devices within an existing internet infrastructure. It allows for data transfer across a network without human-to-human interaction. A detailed look into this will determine how we connect people with music in a very efficient and sleek manner.

- **Competitor Analysis** - We have taken a look at other apps and services on the market and feel that nothing matches our idea. The closest service available is from **Sonos**. Sonos is a smart system of speakers and audio components that unite your digital music collection in one app and can then be controlled from any device. The rise

of digital music has allowed for us to bring our music wherever we go, through media such as iPods, MP3 players etc. However, there hasnt been the same advancements in terms of systems that don't move around. 'Wireless' is the word that comes to mind when thinking of a solution. It is now possible to stream audio to a wireless device (speaker) and without compromising on the sound quality. Sonos provides the user with the ability to play music from a device wirelessly anywhere in the **home**. However, there are two issues with this; it is only available for the **home** and you need their **hardware** which is not cheap. The cheapest speaker you can purchase is £169. If you are a music-orientated person, you may wish to purchase their high-end hardware which can cost anything up to £1200. Unfortunately, Sonos does not support other wireless speakers. An adaptor can be purchased to allow these to be connected to their system, the *Connect* device, but that costs £279.

Pure have also moved into this market where they provide wireless speakers and hardware for *wireless music* in the **home**. They too allow you to purchase hardware to link your current speakers with their system and at £69, it is much cheaper than Sonos but is still not cheap. It allows you to wirelessly play your music from any music app or streaming service you want. **Bose** also provide a very similar service to Pure, but the hardware costs are more expensive.

From this, we can see that there are no services available for the wireless sharing of music outside the home. Our app would unite music into your whole day routine, whether it be at the office, coffee shop, restaurant as well as the home. We also want to make sure that no expensive costs are applied. Competitors can appear at anytime during the development of a project, so it is important that we keep looking for emerging customers and that we can identfy how our product is unique to theirs.

- **Music Sources** - In today's maket, there are many music sources available to an individual. We have virtual music from services such as 'Spotify', 'Google Music' and 'iTunes' as well as physical music on 'iPods', 'MP3 players' and other hardware devices.

  Spotify is a music streaming service that offers access to a library of over 20 million music tracks with over 40 million active users. It is available across 58 markets including the UK, USA, France, Germany, Hong Kong and Argentina. It is available on iOS, Andriod, Windows as well as PC and Mac. One chain that is affiliated with Spotfiy is Costa. Costa have their own playlist that people can access frm their device. **Google Music** is another streaming service and offers the same service as Spotify. Again, they have a large library of songs (around 18 million) and the service is available in over 57 countries on all android devices as well as web browsers. **iTunes** A year ago, Apple's iTunes accounted for 75% of the digital music market and with a huge 575 million active users. Although this may have decreased slightly in the last year, that is still a huge user base. As well as general users, the likes of Starbucks are affiliated with iTunes and use this service to hand-pick and play music throughout their stores. The idea of allowing customers to put forward their music preference may be of interest to a chain like Starbucks amongst others.

The above figures suggest that the music streaming industry is vast and that music is a part of many people's lives. Coffee shops have integrated these sources and music into their environment, but without the customer interaction. Choona would provide this interaction. Over the course of this project, we shall identify more sources because having more sources creates a larger user base as well as a better music library. We shall look at how the different sources work to try and make sure we have adaptors in place that can cover the wide variety of sources available.

- **Legal Issues** - Music playing for customers or staff through media such as radio, MP3, TV etc. is considered a *public performance*. The *Copyright, Designs and Patents Act 1988* means that an agreement is needed from the copyright owner before the material can be played in public. A music license (PPL) will grant this agreement. In most cases, a license is required but there are a few instances when one is not required. One example of this is where PRS (PRS for Music represents the rights of over 100,000 artists in the UK, licensing organisations to play, perform or make available copyright music on behalf of our artists and overseas societies, distributing the royalties to them fairly and efficiently) artists have waived their rights. Another example is a hotel, guest house or B&B that has fewer than 25 rooms with no areas open to non-residents. Any business such as a coffee shop, bar or gym that plays recorded music in public will legally require a PPL. The likelihood of our service being used in places that don't have a PPL and require one is small. Most coffee shops, restaurants, gyms etc. will already have the license in place. It will be work places deciding to implement our service that will have to go about retrieving a PPL. As part of this section, we shall look at the process of obtaining a PPL, the costs involved and potential constraints.

- **NFC** - Near-Field Communication (NFC) is a form of short range wireless communication (4cm or less to launch a connection) allowing for radio communication and the passing of data packets between two devices, or smart tags that work with NFC. NFC is an advancement to RFID systems because it allows for two-way communication. This two way communication can then be used for authentication purposes or two-way communication. At the moment, not all devices support NFC and it is suggested around 20% of phones worldwide will have NFC capability by the end of 2014. This figure is set to increase dramatically over the next five years meaning more devices will have the capability and more users will be familiar with the technology. This widespread reach of NFC phones could mean one day that NFC tags become as commonplace as bar codes, so it makes sense to make use of this technology.

  Using NFC tags with a mobile device, a user could access the playlist at their current location without the need to search for it. The idea is very durable as NFC tags are small and cheap enough to integrate anywhere. They do not need a power source but instead draw power from the device that reads them. We shall look in more detail at how we can install these into facilities, the upgrading process and determine whether they can be managed internally by the business themselves or will outside assistance be required.

6 Software Development Approach

Software Development Approaches have existed since the 1970s. They range from rigid, long-term methods such as *Waterfall*. through to experimental and risky approaches like *Extreme Programming*. Factors such as team size, business goals and clients all contribute to choosing the correct approach.

To identify the correct software development process to use we must first decide on some attributes of the project of which we can compare across various approaches. The following have been derived from the project ideas and plan:

1. **Team Size** - Most of the development throughout this project will be done by one person. This removes the need for large-scale approaches such as *Waterfall* and *Spiral* and instead opens up the possibility of a faster, looser process such as *Agile*.

2. **Client Interaction** - If the project idea is accepted we will be allowed to work very closely with a representive from Atos. This means our development approach needs to incorporate this throughout the entire lifecycle, making methodologies such as *Waterfall* less relevant since they do not encourage client interaction during the development stage.

3. **Timeline** - The entire development and testing cycle for the project is only a few months, so whatever approach is chosen needs to be dynamic and allow for quick iterations without the need to go through the entire process again. This requirement makes approaches such as *Incremental Development* inappropriate because they add in repeated requirements analysis and architecture design steps that may not be necessary.

4. **Deliverable** - The client is expecting a working prototype to be delivered at the end of the project. The technology must be built to a satisfactory standard, but does not need to be production-ready nor be immaculately written. This allows for the testing, integration and implementation steps in the software development approach to be removed as a priority and opens up the possibility of using less formal approaches such as *Prototyping* and *Extreme Programming*.

These requirements narrow the scope down to three relevant approaches:

- **Agile** - Agile development promotes principles such as continous improvement, early delivery, adaptive planning and evolutionary development. It breaks tasks down into small increments, with the goal of each increment being a new release. There is a big focus on quick communication and easy feedback, normally achieved by daily standup meetings and regular planning sessions. Whilst the communication principles would be useful for this project, the approach also tends to have a relience on multiple teams tackling the project which is not relevant.

- **Prototyping** - Software prototyping is perfect for projects where a production-ready deliverable is not required. Each iteration of the project has no requirement to fully

work or meet the users' requirements. Instead, the software goes through an iterative modification process until user acceptance is met or the prototype is discarded. The downfall of prototyping is that it is a very relaxed approach. Client interaction is suggested, not enforced, and often the developers may lose sight of the completed project by getting too involved in a limited prototype which ultimately is only a partial representation of the desired product.

- **Rapid Application Development (RAD)** - RAD is used to describe alternatives to the *Waterfall* approach that focus on iterative development rather than formal, structured planning cycles. The most common definition of RAD, created by James Martin, involves an initial requirements planning stage followed by an interative construction and user feedback stage.

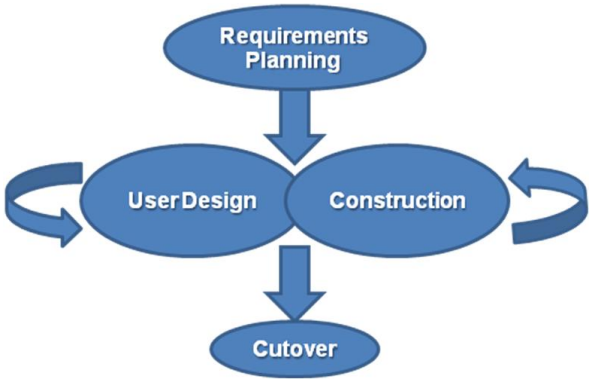


Figure 2: Phases in the James Martin approach to RAD

This fits in nicely with the four project attributes we defined above:

1. **Team Size** - RAD imposes no requirements on number, size or iteractions of teams.
2. **Client Interaction** - The UserDesign stage requires development to go hand in hand with client feedback and testing.
3. **Timeline** - There is no restriction on the length of the UserDesign/Construction process in RAD, allowing for the development cycle to last as long as possible before the deadline is reached without having to estimate iterations during early planning stages.
4. **Deliverable** - There are no formal requirements for repetitive testing and quality control. This can be implemented independently from the software development approach, allowing for testing to be suited to the individual project.

The analysis above clearly identifies *Rapid Application Development* as the most appropriate software development approach for this project.

7 Technical Considerations

The technical considerations for this project can be divided into three sections: mobile application, server-side systems and hardware.

7.1 Mobile Application

One of the basic top-level requirements for this project is that the end product is centered around a mobile application. Therefore we have to ensure that all technology used is compatible with as many mobile devices as possible. If the app starts to have a dependency on niche features, such as NFC and 4G, there is a risk of excluding large quantities of users.

- **Platform** - There are currently three major mobile platforms that support apps: Android, iOS and Windows Phone. Others such as Blackberry and Firefox OS either do not have a thriving application community or have a very small userbase. This project should ideally support all three of these major platforms, however each one requires applications to be written in different languages. There are two options here which will need to be considered:
  - *Native Applications* - A native application is one that is written to work directly with the device's operating system. For example on Android this would mean that the application is written in the programming language Java and that it directly uses the device's native libraries. The advantage this approach is mainly performance, but there are also certain features that are only available for native applications. The disadvantage is that the application has to be written separately for each platform you want to support.
  - *Framework Applications* - To solve the difficulties of supporting multiple platforms, frameworks such as Apache Cordova and Adobe Phonegap have been created. These allow developers to write their applications once and have it automatically work across all major platforms. The downside here is that the application normally exists inside a ẃrapperẃhich can have a negative impact on performance.
- **Technology** - Features such as NFC are starting to become prevalent in many high-end smartphones, however they have not penetrated the lower-end of the market. The project needs to make sure that proper analysis is done on the prevalance of these features if they are to be used.

7.2 Server-Side Systems

Many ideas and concepts in this project revolve around connected data and technology. There are likely to be many different services and systems required in order to complete the project. Designing and architecturing will be an important part of ensuring the project is a success.

- **Architecture** - There are many different styles of software architecture. For this project we will be using the *Microservices Architecture*. This is centered around separating out the application into individual services, each one with its own unique responsibility. This fits in nicely with the chosen Software Development Approach - *Rapid Application Development* - since it allows for entire services to be completed in a single development cycle.

  Advantages

  - Easier debugging
  - Better fault isolation
  - Independent service development and deployment
  - Removes commitment to a specific technolog stack
  - Avoids monolithic applications and systems

  Disadvantages

  - Requires inter-service communication
  - Testing can be more complicated
  - Multi-service deployment can be hard to orchestrate
  - Multiple services means more memory usage

- **Infrastructure** A solid infrastructure is essential in order to adequately support the numerous software services and systems required for the project. There are several options available when it comes to selecting infrastructure components (servers, database management systems etc):

  - **Cloud Computing** - Cloud-based systems such as *Amazon Web Services* (AWS) and *IBM SoftLayer* allow systems engineers to easily configure and deploy scalable services across the globe. They often support many different types of services (application servers, databases, data processors, queues etc) and can be configured quickly and easily through user interfaces, rather than having to manually set up each service. The downside to cloud-based infrastructure is usually the cost and also the added abstraction; the systems engineer is no longer in direct control of each moving part which can sometimes make fault finding a more laborious process.
  - **Dedicated Servers** - Dedicated servers are the more traditional approach to supporting software systems. They give the systems engineer absolute control over the entire infrastructure, however this goes hand-in-hand with the additional responsibility involved in configuring everything by hand. Another concern is redudancy. In a cloud-based system it is very easy to spin up copies of a service in the event of failture, however with a dedicated system you normally need to have servers on standby all the time. Compared to cloud services, dedicated servers can often work out cheaper.

7.3 Hardware

There is the potential for some hardware to be developed and made available to enhance the mobile application. For prototyping it is likely that pre-existing devices such as the Raspberry Pi will be used. Since the main goal of the project revolves around the app, it is unlikely that much time or development will be put into the hardware itself and focus will instead be given to the software that it runs.

8 Risk Assessment