**Project Proposal**

**Title:**

**Topological Navigation Editor**

**(ToNE)**

By

Laxwell Nyama

MComp Computer Science

University of Lincoln

2015/2016

Supervised by

Dr Marc Hanheide

**Table of Contents**

**Page**

1. Abstract
2. Introduction
   1. Overview
   2. Relevance
   3. Aim
   4. Objectives
   5. Literature review
3. Background Research
   1. Overview
   2. Topological Navigation
   3. Simulation Agents
   4. Graphical User Interface
4. Project Plan
   1. Overview
   2. Gantt Chart
   3. Pert diagram
5. Methodology
   1. Overview
   2. XP Programming
6. Scope and Constraints
   1. Overview
   2. Scope
   3. Constraints
   4. Risks
   5. Risk Matrix
7. Resources
   1. Overview
   2. Development Platform
   3. Hardware requirements
   4. Software requirements
8. Conclusion
9. References
10. **Abstract**

This interactive design project will seek to develop an innovative graphical user interface to be used in robotics for topological navigation editing. In this proposal, is the intended approach to designing a topological map navigation editor interface (GUI), which is robust, functional, usable and accessible, to be used by people with no technical knowledge or background, to control mobile robots. “Graphical user interfaces divide the standard user actions and working paradigms into various categories which are presented graphically in a way that reflect their functionality.” (Ellis Horowitz and Zafar Singhera, 1993). These features make graphical user interfaces more easy to use.

1. **Introduction**
   1. **Overview**

“Mobile robotics and robot navigation is a growing area of scientific research. Without navigation the creation of self-propelled, household machines, guard robots, or planet surveyors is beyond imagination”. (Richárd Szabó 2004). An easy and effective way of controlling such robots will vastly increase their applicability.

In recent years we have seen exponential growth in mobile robotics and robot use, however, the control and use of these robots requires a high level of technical knowledge. The way these robots are currently controlled is too complex and technical for an average person, hence the need to design and implement a graphical user interface that is simple, effective and easy to use for people with less technical ability.

It is however, also important that robot controlling softwares be written in several high level computer languages like C/C++ or Java, Python for the simple reason of maintenance and compatibility. Above all, usability is fundamental and draws from the field of Human Computer Interaction. In this project, Python is used for writing the program as it one of the requirements for this project due to the benefits it offers.

**2.2 Relevance**

This project is part of STRANDS project; STRANDS aims to enable a robot to achieve robust and intelligent behaviour in human environments through exploitation of long-term experience in security and care scenarios. This project aim to develop a real world topological navigation system editor (GUI) by improving on the existing techniques to advance the robot navigation capabilities, usability and user interfaces. The development of robust and effective navigation editor that can be used by people with no technical background will play a great part in the deployment of these robots. Autonomous Mobile Robotics is a growing scientific field and it is one of the modules I will be undertaking.

**2.3 Aims and objectives**

**Aim**: The aim of the project is to develop a Topological navigation editor user interface for robot systems that can be used by people with no technical background.

**2.4 Objectives**

* Establish requirements
* To identify, investigate and evaluate current systems (Topological navigation editors)
* To identify, investigate and evaluate tools and techniques
* Create and evaluate a prototype
* To design and develop the topological navigation editor user interface
* Test, analyse and evaluate the topological navigation editor
* Demonstrate the effectiveness of the artefact
* Artefact to be delivered on time

The navigation editor design is to be simple for ease of control of the robots by people of all walks of like with little to no technical ability. The editor will include features like drag and drop, the user will be able to drag and drop a node, edge and also edit these through their properties tab. However, while the graphical interface is aimed at user with no technical knowledge, expert users will also benefit.

**Establish requirements**

According to (Preece et al, 2011) “Whatever the initial situation and whatever the aim of the project, the users’ needs, requirements, aspirations, and expectations have to be discussed, refined, clarified and probably rescoped”. This will be the first objective to be met so I can establish what it is that is required. With requirements gathering I aim to understand as much as possible about the users, their activities and the context of that activity, so the system under development can support them in achieving their goals and it also produces a set of stable requirements that form a sound basis to start design.

My requirements gathering process comprises of sub activities, which will inform and refine one another iteratively through out the product life cycle, this is due to the fact that requirements evolve and develop as the stakeholders interact with the design through prototypes and see what is possible and how certain facilities can help them.

For the project both “functional requirements: which say what the system has to do and non-functional requirements: which says what constraints there are on the system and its development will be gathered. Requirements gathering techniques like interviews, focus groups and direct observation on users will be conducted to collect enough information.The project requires the creation of tools that will simplify the map creation/edition process making the system more user friendly. This includes simplifying the use of tools already available for topological map creation, direct map edition from text files and also improving and completing the tools for editing previously created maps using Rviz.

**To identify, investigate and evaluate current systems (Topological navigation editors)**

This first milestone of this objective is aimed at identifying any existing Topological navigation systems. The investigation of pre-existing systems, not only does it help in discovering the industry standard techniques, hardware and software requirements but will also aid in the design of the artefact. The background research and literature review will shed some light on what is needed to be done and in some measure how to do it effectively and efficiently.

The identification, investigation and evaluation of current systems will be conducted through different medium such as the internet, academic literature, IT journal, books, publications and newspapers for current news and interacting with real life robots. Case studies from past and present projects relating to the project will also be explored and critically analysed to aid in the design and implementation of the navigation editor “for example, Green *et al* (2009) describes how they adapted a large open source piece of software, called Bugzilla, to design and develop a ‘Problem Reporting, Analysis and Corrective Action System’ for NASA’s next generation vehicle to replace the shuttle.” (Preece et al, 2011).

Evaluation and critical analysis of current systems will give insight in some of the difficulties others have encountered in the past and how best to improve on their experiences and systems.

**To identify, investigate and evaluate tools and techniques**

The aim of this objective is to research into the tools and techniques to employ, this objective is fundamental as it facilitate on learning on how to use the tools and employ techniques that will be utilised to develop the topological navigation editor graphical user interface (GUI). Here, investigating ROS and STRANDS robot systems, which are some of the tools to be used to achieve the goal of this project.

“Since the advent of graphical user interfaces in the 1980s many tools have been developed to support creative thinking, design sketching, simulation, brainstorming, mind-mapping, most aimed at visual interfaces.” (Julie and Andrew, 2003) and the challenge is to choose the best tool for the job. However this project has specific tools to be used as part of project requirements MongoDb and PyMongo just to mention a few are also other tools to conduct some research on as they are integral to the project development.

The tools that will be used range from low-level i.e. requiring a lot of programming to high levels tools. The lowest level being the *graphical libraries* that provide hardware independence, *User interface toolkits* to structure an interface as a tree of interactive objects and widgets, whereas user interface builders provide an interactive application to create and edit those widget trees. *Application frameworks* build on toolkits and UI builders to facilitate creation of typical functions such as cut/copy/ paste, undo and help.” (Julie and Andrew, 2003), which are part of functionalities needed in the editor. This objective will help understand the structure of the tools and how they can be used effectively for the project.

**Conceptual design/ Create and evaluate a prototype**

After conducting all the necessary research and requirements gathering as stipulated by the first three objectives, here the design is conceptualised by a Low-Fidelity prototype. “A Low-fidelity prototype also known as Lo-fi is one that does not look very much like the final product, it uses materials that are very different from the intended final version, such as paper and cardboard rather than electronic screens and metal.” (Preece *et al*, 2011).

Two lo-fi prototyping techniques that are storyboarding and paper prototyping will be employed. The former will aid with showing how a user might progress through a task using the product under development while the latter will facilitate to show the design interface.

The reasons lo-fi prototype is used is for the benefits it offers such as being cheap and quick to produce. Lo-fi prototypes are also simple and quick to modify, hence they support the exploration of alternative designs and ideas. This is particularly important during the conceptual design of the navigation editor user interface.

Paper prototyping involves use of paper, index cards, cardboard etc. to build up a paper representation of the system. One example of this was the “cardboard computer” (Ehn and Kyng, 1991). Due to the technical and costly nature of the hardware to be used this type of prototyping would the most appropriate to use because the process will be iterated a few times to come up with a final design solution that meet user requirements.

**To develop and implement the topological navigation editor user interface**

A user-centred approach will be applied to design the interface. “In 1985, Gould and Lewis (1985) laid down three principles they believed would lead to a ‘useful and easy to use system’:

1. Early focus on users and tasks.
2. Empirical measurement
3. Iterative design” (Preece *et al*, 2011).

These three principles are now accepted as the basis for a user-centred approach e.g. see (Mao *et al,* 2005).

Since the aim of this project is to develop a topological navigation editor graphical user interface for robot systems that can be used by people with no technical background, in this objective both types of design will be employed. “Broadly speaking there are two types of designs: conceptual and physical. The former is concerned with developing a conceptual model that captures what the product will do and how it will behave, while the latter is concerned with details of the design such as screen and menu structures, icons and graphics.” (Preece et al, 2011).

The development process will itself be iterative, that means when problems are found they are fixed and more testing carried out and so on. A user study and testing will be conducted each time new features or components are added to make sure that the user is involved and their feedback incorporated. A conceptual design of any changes to be made will be presented to the user before the physical and actual implementation. User centred design advocates that the focus be on the user, empirical measurements and iterative design. This design approach compliments the Extreme Programming methodology that will be used in the project. This agile methodology also emphasises on gathering feedback from the user during implementation and getting the user involved through out the project life cycle.

**Test, analyse and evaluate the topological navigation editor interface**

“Nowadays users expect much more than just a usable system, they also look for a pleasing and engaging experience. This means that it is important to carry out an evaluation.” (Preece et al, 2011)

Testing is fundamental and it is an obligation to test the artefact to ascertain functionality and make sure it is fit for the purpose and also its usability (Usability testing). Once the navigation editor is completed, a functionality test and a usability test with a focus group of users will be conducted. An application with high functionality is not enough if its user interface makes it difficult for most of the users to be able to make use of its functionalities. “Computer systems with poor user interfaces can have a financial cost.” (Stone D *et al*, 2005).

The testing of the navigation editor interface will be conducted using the following criteria:

* Minimal requirements
* Interface usability
* Performance and effectiveness

There are various methodologies of testing graphical user interfaces. Graphical user interface (GUI) testing is inherently more difficult than traditional, command line interface testing. The nature of GUI applications, their asynchronous mode of operation, nontraditional input/output and hierarchical structure for user interaction make their testing significantly difficult. Each object on the application has to be tested.

The evaluation process will be guided by the DECIDE framework. “The DECIDE framework provides a check list to help plan evaluation studies and to remind you about the issues you need to think about. It has the following six terms:

* **D**etermine the goals
* **E**xplore the questions
* **C**hoose the evaluation method
* **I**dentify the practical issues
* **D**ecide how to deal with ethical issues
* **E**valuate, analyze, interpret and present the data.” (Preece *et al*, 2011).

This framework also allows for iterative movement back and fourth the list after taking a first pass through each of them. This flexibility ties in with the development methodology.

**Demonstrate the usability and effectiveness of the artefact**

After accomplishing all the objectives above, the artefact will be ready for final demonstration to the client. The demonstration will involve deploying the application on a simulation. This is to show the stakeholder or the user how the artefact functions, give them a walk on the application so may they may have some idea on how they will use it and to also make sure that the client is satisfied with the product. It is also an effective way to address the prospect's specific product-related concerns. The ability to see, feel and sometimes smell a product is generally more appealing to prospects than simply listening to a pitch.

Demonstration also helps combat product-related concerns presented by the user. Users often come with preconceived notions or misinformation based on things they have heard in the marketplace. If a user questions key claims or doubts the product and its performance capabilities, a demo can ease those concerns. The key is to view the users’ concerns as a request for more information and not a condemnation of the solution.

**Artefact to be delivered on time**

There are going to be some difficulties and constraints along the way, as is the case with most software projects, however it is important to deliver the artefact in the time stipulated, this can be achieved through effective planning, time management and adhering to the plan and time schedules. The use of different tools like the Gantt chart for the project will also aid in delivering the artefact.

Delivering software is hard, and delivering quality software in time is even harder.

**2.5 Literature review**

Overview

In this section presented is the literature review undertaken in order to understand the problem domain and allow me to design the system to meet the user requirements and critically evaluate already available materials.

The point of literature review as identified by Borg and Gall (1989, cited by Saunders et al. 1997: 39) is among other things:

* To refine research questions and objectives
* To highlight research possibilities that have either been explicitly identified by other authors or have possibly been overlooked in the past.
* To avoid repeating the works of others
* To identify search methods and strategies

This is achieved with references to past and current literature in the field of robotics, user interfaces and Human computer interaction.

Navigation for autonomous mobile robots is a new and growing field of interest with a lot of research still underway. Having said that there is quite a fair amount of materials that gives a foothold in this field of study. The editor to be designed is to enable the edition of topological maps via an interface. Due to the capabilities provided by topological maps, P. Ranganathan *et al*, (2002) argues that topological navigation paradigm offers a lot of advantages:

* The node information may be meet points, land- marks detected by a vision system or any other distinguishing features of the environment that can be reliably recognized by a robot;
* The path between two nodes does not have to be traced exactly; it is sufficient if the robot can traverse a general path (not exactly defined) between two nodes;
* Small storage capacities are required, since only information about the nodes are stored;
* There is no need to maintain a global coordinate frame, so this method is suitable for exploring large-scale environments;
* Path planning from a topological map can be very fast and without complex computations.

However, two important properties have to be guaranteed: first, the nodes have to be detected and identified with certainty and accuracy, and secondly, the navigation operations must lead the robot from one node to another. Most scholars propose the use of both paradigms in conjunction for best results.

1. **Background research**

**3.1 Overview**

“A topological map represents the robot environment by a graph. Paths are defined as sets of two distinct points, or “places” which must be detected and recognized by the robot using sensors data.” (P. Ranganathan *et al*, 2002). These points provide the nodes of the map. Only little relevant information about the places is required to locate and identify them. The edges between two nodes correspond to navigation operations such as climb ramp. These navigation operations take the robot from one node to another.

**3.2 Autonomous mobile robots and navigation**

Autonomous mobile robots must be able to acquire and maintain models of their environments to efficiently carry out complex missions in indoor and outdoor environments. The research on mobile robot navigation has produced two major paradigms for mapping indoor environments: grid-based and topological.

“Grid based approaches, such as those proposed by Moravec/Elfe (Moravec 1988) and many others, represent environments by evenly-spaced grids. Each grid cell may, for example, indicate the presences of an obstacle in the corresponding region of environment. Topological approaches, such as those described in (Engelson & McDermott 1992; Kortenkamp & Weymouth 1994; Kuipers & Byun 1990; Matarić 1994; Pierce & Kuipers 1994), represents robot environments by graphs.” (Sebastian and Arno, 1996). Nodes in such graphs correspond to distinct situations, places, or landmarks (such as doorways). They are connected by edges if there exists a direct path between them.

According to Sebastian and Arno 1996, both approaches to robot mapping exhibit orthogonal strengths and weaknesses. While grid-based methods produce accurate metric maps, their complexity often prohibits efficient planning and problem solving in large-scale indoor environments. Approaches in the *metric paradigm* generate fine-grained, metric descriptions of a robot’s environment (Moravec 1988; Lu & Milios 1997). The metric framework is the most common for humans and considers a two-dimensional space in which it places the objects. The objects are placed with precise coordinates. This representation is very useful, but is sensitive to noise and it is difficult to calculate the distances precisely.

Topological maps, on the other hand, can be used much more efficiently, yet accurate and consistent topological maps are often difficult to learn and maintain in large-scale environments, particularly if momentary sensor data is highly ambiguous. Topological maps are usually more compact, since their resolution is determined by the complexity of the environment. Consequently, they permit fast planning, facilitate interfacing to symbolic planners and problem-solvers, and provide more natural interfaces for human instructions. This project will deal with the topological based navigation paradigm.

**3.3 Simulation agents**

The use of simulation is well presented in Explorations in evolutionary robotics (Cliff et al., 1993; Bongard, 2013), but it is a tool that must be used with caution; the benefits in terms of speed and cost have to be balanced against issues arising from the validity of the simulation - the so called ‘reality gap’ (Jakobi et al., 1995). I will be using a simulation in a project so a bit of research in this field will help with overall undertaking of the project.

* 1. **Graphical User Interface Development (IDE)**

“A GUI is a computer program that enables a person to communicate with a computer through the use of symbols, visual metaphors, and pointing devices.” (Steven Levy, Jr. 2015). These include graphical elements such as windows, icons and buttons.

There is a need to be well versed with the development environment that will be used for the project; however researching on other development environments is valuable. Xerox developed the first commercially available GUI, called “PARC,” which was used by the Xerox 8010 Information System, released in 1981.

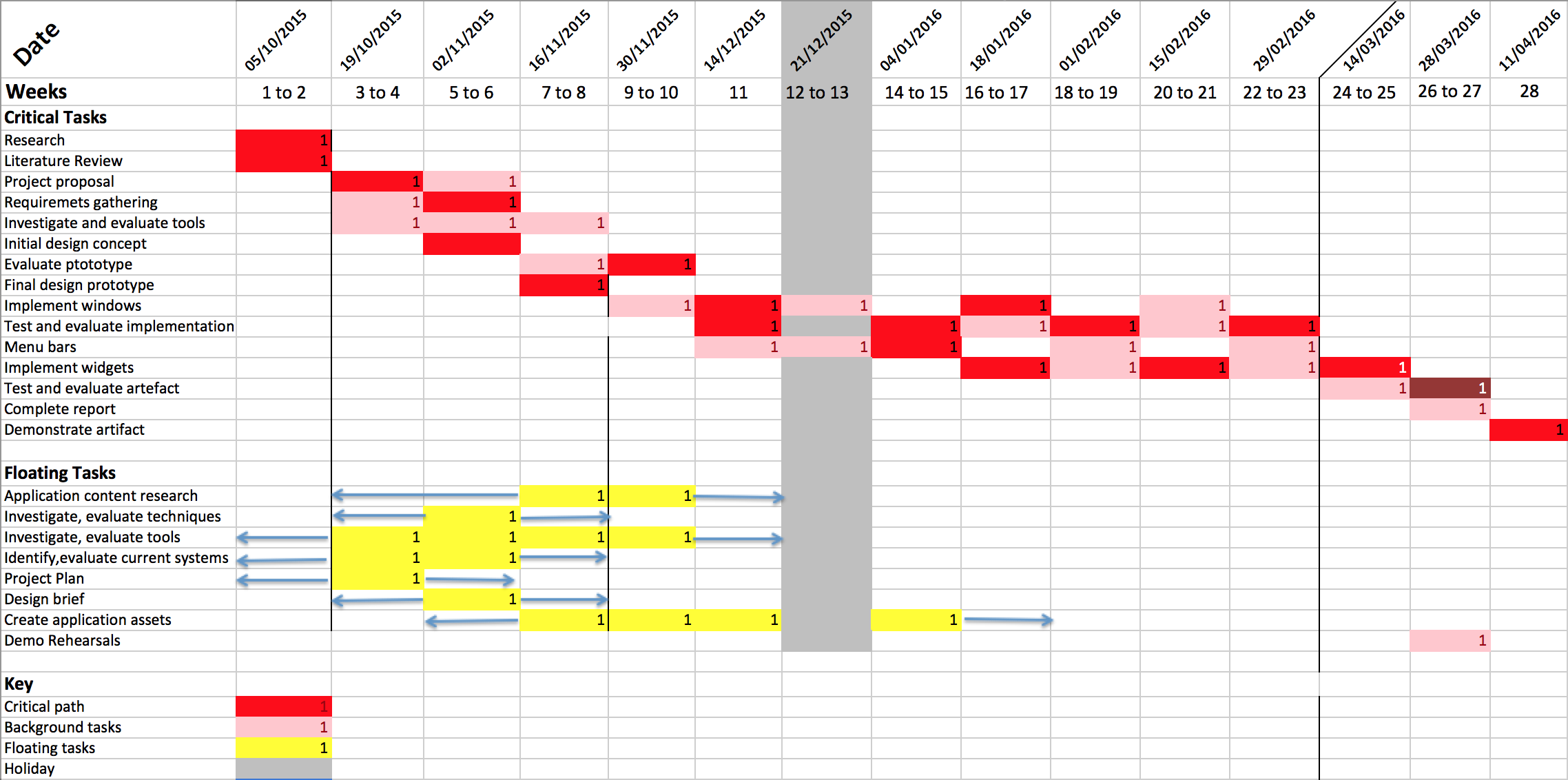
There are many tools now available on the market used to develop the GUI. PyQt4 is the tool to be used on the project. It optimized to work well with the Python programming language to be used.

A robot, just like a computer system needs a user interface to communicate with users. Graphical user interfaces have become very popular due to their ease of use. WIMP represents the most common graphical elements: window, icon, menu, and pointer. A window is an area on the screen that displays information. The contents of the window are displayed independently from the rest of the screen. “Designing a good user interface is critical to the success of a system. A good user interface encourages an easy and natural interaction between a user and a system. Ideally, a user can forget that she is using a computer and get on with what she wants to do.” (Paul Zandbergen, 2003). This is particularly important for this project as it aims to develop a user interface for users with no technical expertise.

1. **Project plan**
   1. **Overview**

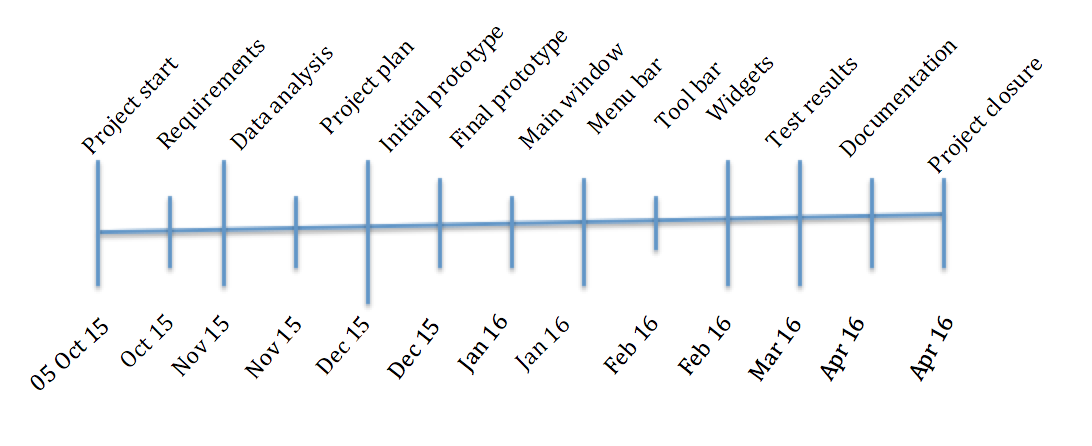
In this section of the proposal I will present some of the tools I have commissioned to help me in organising, time management and ensuring that the project stays on track to facilitate on time delivery. I have employed the Gantt chart for the project plan and a Pert diagram for the flow of activities.

* 1. **Gant chart**



In the Gantt chart above the critical path is in red showing the scheduling of main activities and their precedence, background task are shown in pink and finally the floating tasks show in yellow and holiday break greyed out. The chart is scheduled in

**Milestone Timeline**



* 1. **Pert diagram**

The pert diagram below further shows stages where the iterations will happen be it on the conceptual design, actual development or the testing and evaluation. These iterations are shown in loop like format.

Research

Gather user requirements

Create Project Plan

User study

Conceptual Design/ Lo-FI Prototyping

Evaluate prototype

Develop widgets

Develop GUI

Implement functions

Testing and evaluating artefact

Report and Demonstration

1. **Methodology**
   1. **Overview**
   2. **Extreme Programming**

An agile software development methodology known as Extreme Programming (XP) will be used for developing the interface. Extreme Programming (XP) is a software engineering methodology, the most prominent of several agile software development methodologies. Like other agile methodologies, Extreme Programming differs from traditional methodologies primarily in placing a higher value on adaptability than on predictability. “XP also focuses on excellent application of programming techniques, clear communication, feedback, courage and respect, which allows us to accomplish things we previously could not even imagine”(Kent Beck and Cynthia Andres, 2005). These are the reasons for choosing to use this methodology.

This is type of agile software development involves developing and delivering functionality in fragments or in incremental stages. The XP methodology allows for iterations and changes are encouraged and are seen as natural and desirable happenings that occur during the course of the projects development cycle. This approach is beneficial and appropriate for this particular project where the exact requirements and potential difficulties are not always foreseeable due to the experimental and dynamic nature of the project. The methodology involves splitting up the different aspects of the projects functionality steps giving the developer a greater degree of control over the projects development process.

“XP builds on best practises such as unit testing, pair programming, and refactoring” (Erich Gamma, 1999). However, since this is an individual project, pair programming will not be practised, as this requires two people to undertake. “Extreme programming uses an object oriented approach as its preferred development paradigm and encompasses a set of rules and practices that occur within the context of four framework activities: planning, design, coding and testing.” (Roger S. Pressman, 2010).

The language to be used is Python, which complements this methodology and the advantages it offers. Python is a general-purpose, dynamic, object-oriented programming language. The design purpose of the Python language emphasizes programmer productivity and code readability.

1. **Scope and constraints**
   1. **Overview**

This section present some of the risks involved with projects and their impact, however the list is not exhaustive. Risks are conditions and situations that may affect the progress of the project.

* 1. **Scope**

The scope of this project will be limited due to the nature of the project and a number of constraints some of which are discussed below. A narrower scope means shorter processes and fewer resources required, however the scope will determine the process, but it also must fit project goals.

* 1. **Constraints**
* Ethical issues – this may arise if the user participating in the test decides to quit before testing is completed. Participants are allowed to exercise their right to withdraw from the test at any point in time as they wish, this will however have huge implications on the project completion and delivery.
* Expertise – This project is a real challenge to me as I am entering into uncharted territory. My knowledge and expertise is virtually none existent. This will be the biggest constraint in this project which might have impact on project delivery as I will consumed a lot of time getting to know the tools techniques.
* Time – The fact that I have to undertake a lot of research and learning on how to use the tools in a short space of time will impact on the project schedule.
  1. **Risks**

These come in different shape and forms; here I will highlight two major risks categories that may impact my project:

*“Project risks* – threaten the project plan, if a risk manifest it is likely that the project schedule will slip. Project risks identify schedule, personnel, resources, stakeholder and requirements problems.

*Technical risks* – threaten the quality and timeliness of the software to be produced, if they manifest, implementation may become difficult. Technical risks identify potential design, implementation, interface, verification and maintenance problems.” (Roger S. Pressman, 2010).

* 1. **Risk Matrix**

The table below is a risk matrix, showing the probability of a risk manifesting and the impact it will have on the whole project.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risks | Probability | Impact | Score | Mitigation |
| Failure to gather requirements | 0.1 | 5 | 0.5 | The probability of this occurring is virtually none existent. However in the case that it occurs this has a crippling impact on the project as a whole. To avoid this meetings with the user should be arrange and attended. |
| Failure to understand the project requirements | 0.4 | 4 | 1.6 | Due to the scale and dynamic nature of the project, there is a chance that requirements may not be understood leading to the development of a product that does not meet requirements. Frequent user feedback is needed to avoid misunderstanding. |
| Objectives not carried out properly | 0.3 | 3 | 0.9 | Failure to carry out objectives properly could be disastrous for the project. Milestones are put in place to make sure every goal is met and on time. |
| Milestones not being met on time | 0.3 | 3 | 0.9 | Failure to meet each milestone is critical to the project. This can cause a host of problems including project not being completed in time. |
| Software issues | 0.5 | 4 | 2 | Issues with software have a marginal impact and the probability of it occurring is quite high. However if it happens, there are lot of machines in the computer labs that can be used. |
| Hardware issues | 0.4 | 5 | 2 | Issues with hardware have a marginal impact and the probability of it occurring is quite high. However if it happens, there are lot of machines in the computer labs that can be used. |
| Time constraints | 0.7 | 4 | 2.1 | Time management if fundamental to avoid delivering the project late. The scale of the project is large and might take a lot of time than planned. Meeting all objectives and sticking to the plan this may be avoided. |
| Lack of resources | 0.1 | 2 | 0.2 | This risk is very limited due to the fact that most of the resources used are available as open sources. |
| Personal issues | 0.3 | 5 | 1.5 | The risk of unforeseen events in our day-to-day lives for example getting ill, this will effectively hamper the project. In case any of personal isssues arise that affect the project, the supervisor will need to be notified |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** |  |  |  |  |
| **Probability** | **0 - 1** |  | **Impact** | **1 - 5** |
| Almost certain | 0.9 - 1 |  | Catastrophic | 5 |
| Likely | 0.7 – 0.8 |  | Critical | 4 |
| Possible | 0.5 – 0.6 |  | Marginal | 3 |
| Unlikely | 0.3 – 0.4 |  | Negligible | 2 |
| Rare | 0 – 0.2 |  | Insignificant | 1 |

1. **Resources**
   1. **Development platform**

The user interface will be implemented using the Ubuntu operating system, which runs on the Linux Kernel (Ubuntu Linux 14.04LTS 64bit (trusty)). The reason being the STRANDS project is already underway and using that platform, so for compatibility purposes the artefact has to run the same system.

* 1. **Hardware requirements**
* Computer running on Linux kernel – Linux is, in simplest terms, an operating system. It is the software on a computer that enables applications and the computer operator to access the devices on the computer to perform desired functions. This is a open source software that is stable and has a large support base. It is arguably the most dominant operating system in the market. Hence the reason why it is a requirement for the project, it also ensures compatibility with most hardware components.
* 40GB hard drive memory - A hard disk drive is a device used to permanently store and also retrieve information. This will be used for installing necessary software on the computer to run the project and store files and information concerning the project.
* 1GB working memory (RAM) - Random access memory is used as the working memory of a computer system. It stores input data, intermediate results, programs, and other information temporarily. It can be read and/or written. It is usually volatile, that is all data will be lost when the power is turned off. The size of RAM is the optimum for running all the process fast enabling the progress of the project.
* Mobile robot that provides odomerty data and is equipped with a horizontally mounted, fixed, laser range finder (to use slam\_gmapping).
  1. **Software requirements**
* Ubuntu14.04.3 LTS (trusty) - is a Debian-based Linux operating system and distribution, with Unity as its default desktop environment for personal computers including smartphones in later versions. Ubuntu also runs network servers. It was decided that every fourth release, issued on a two-year basis, would receive long-term support (LTS). LTS releases are typically used for large-scale deployments. Due to the scale of the project it is the most appropriate OS to use. By using this version of the software ensuring compatibility and advantages it offers, it is part of meeting the user requirements.
* ROS Indigo – Robot Operating System (ROS) is a collection of software framework for robot software development. ROS is in fact a meta-operating system, something between an operating system and middleware. The navigation editor being developed is to be used on a STRANDS project robot called LINDA. This project is using this operating systems for its robots.
* STRANDS software – Strands is a project aimed at enabling a robot to achieve robust and intelligent behaviour in human environments through adaptation to, and the exploitation of, long-term experience. Most of the STRANDS software is based on ROS and made available free on GitHub. Most of the software being used for this project is open source. This is a big advantage as it reduces the cost and also has large support communities. However this is part of a STRANDS project and by using these tools, the project is meeting some of the user requirements.
* Python - he design purpose of the Python language emphasizes programmer productivity and code readability. Guido van Rossum initially developed Python. It was first released in 1991. Programming languages such as ABC, Haskell, Java, Lisp, Icon, and Perl inspired Python. Python is a high-level, general-purpose, multiplatform, interpreted language. It is a minimalistic language. Python is maintained by a large group of volunteers worldwide and is open source software. Its simplicity, general-purpose and multiplatform features makes the appropriate language to use. It also has stable base as a open source software and supports several programing styles.
* MongoDb - is an open source, document-oriented database designed with both scalability and developer agility in mind. Instead of storing your data in tables and rows as you would with a relational database, in MongoDB you store JSON-like documents with dynamic schemas. MongoDB was founded in 2007 by the people behind DoubleClick, ShopWiki and Gilt Groupe. At DoubleClick, the site could never be down and there were daily challenges with processing, storing, and scaling data, forcing them to write their own software to handle specific problems. MongoDB and the MongoDB drivers are open source software. As stated in sections above, the scale dynamic nature of this project requires a large space and dynamic system for storing information about the project, map produce by the robot, and a whole host of different data sets.

MongoDB allows deployment of projects as big as anyone can possibly dream. From startups to enterprises, for the modern and the mission-critical, MongoDB is the database for giant ideas. With is the reason for using MongoDB and it is also part of the project requirements and by using MongoDB, not only is the project meeting its requirements but also provides high performance, high availability, and easy scalability.

* PyMongo - is a Python distribution containing tools for working with MongoDB, and is the recommended way to work with MongoDB from Python. It is a requirement to use MongoDB and recommended to use Python for the development of the navigation editor, this make PyMongo a necessity to deploy an effective system for the project. However it is a requirement that we use PyMongo to interface between the programming language to be used and the database.
* PyQt4 - PyQt4 is a set of Python bindings for v4 and v5 of the Qt application framework. Wikipedia defines PyQt as a Python binding of the cross-platform GUI toolkit Qt. It is one of Python's options for GUI programming. PyQt is free software.

.

1. **Conclusion**

“User interface(UI) software is often large, complex, and difficult to implement, debug, and modify. As interfaces become easier to use, they become harder to create.” (Myers 1994). This project will be a challenge. There has been some robot interfaces developed before, however It is clear that the research on UI software tools has had enormous impact on the process of software development. Now, UI design is poised for a radical change, primarily brought on by the rise of the World Wide Web, ubiquitous computing, recognition-based user inter- faces, handheld devices, wireless communication, internet of things and other technologies.

**References**

Jenny Preece, Yvonne Rogers, Helen Sharp (2011) *Interaction Design beyond human-computer interaction 3rd Edition*. Wiley.

Julie A. Jacko, Andrew Sears (2003) *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications.* Lawrence Erlbaum Associates. London.

Roger S. Pressman, (2010) *Software Engineering: A Practitioner’s Approach Seventh Edition.* McGraw-Hill.

D, Stone., C. Jarret., M. Woodroffe., S. Minocha (2005). *User Interface* *and Evaluation*. Morgan Kaufmann Publishers.

Cliff, D., Husbands, P. and Harvey, I. (1993) ‘*Explorations in evolutionary robotics’, Adaptive Behavior* 2(1), 73–110.

Ehn, P. and Kyng, M (1991) *Cardboard Computers: Mocking it up or hands-on the future*. New jersey.

Kent Beck with Cynthia Andres (2005) *Extreme Programming Explained, Embrace Change.* Pearson Education, Inc.

Steven Levy, Jr. (2015) Graphical user interface.

<http://www.britannica.com/technology/graphical-user-interface> [Access on 01 Nov 2015]

P. Ranganathan , J.B. Hayet b, M. Devy, S. Hutchinson, F. Lerasle (2002)

*.Robotics and Autonomous Systems. Topological navigation and qualitative localization for indoor environment using multi-sensory perception.* Elsevier Science B.V. USA

<http://www-cvr.ai.uiuc.edu/~seth/ResPages/pdfs/RanEtAl02.pdf> [Accessed 01 November 2015]

Ellis Horowitz and Zafar Singhera, “*Graphical User Interface Testing*”, Proceedings of the Eleventh Annual Pacific Northwest Software Quality Conference, October, 1993.

<http://www.cs.usc.edu/assets/004/83254.pdf> [Accessed on 31 Oct. 15]

Elizabeth Parlemo

[www.livesciences.com/51834-exoskeleton-helps-clumpsy-robots.html](http://www.livesciences.com/51834-exoskeleton-helps-clumpsy-robots.html) [accessed on 08 Oct 2015]

Richárd Szabó Department of General Computer Science Eötvös Loránd University, Budapest, Hungary. Topological navigation of simulated robots using occupancy grid.

<http://www.jataka.hu/rics/ars2004/ars2004.pdf> [Accessed 15 Oct 2015]

<https://docs.mongodb.org/getting-started/python/>

[Accessed 15 Oct 2015]

Sebastian Thrun and Arno Bücken, 1996 Integrating Grid-Based and Topological maps for Mobile Robot Navigation. Portland. Oregon.

<https://www.ri.cmu.edu/pub_files/pub1/thrun_sebatian_1996_8/thrun_sebastian_1996_8.pdf>

F. de la Paz Lopez and J. R. Alvarez-Sanchez Dept. of Artificial Intelligence – UNED (Spain)

<http://link.springer.com/chapter/10.1007%2F3-540-45723-2_55#page-1>

https://en.wikipedia.org/wiki/PyQt