Summary

Outlined within this report is the inspiration, analysis and eventual implementation of the design of a Graphical User Interface (GUI) to support scission point identification. Following good practice of delivering IT projects, cycles of design, implementation, testing, and evaluation were delivered incrementally and prototyping approach was employed to evolve from a low fidelity prototype to final product. Subsequently in each cycle of iteration evolutionary prototype is implemented, tested and scientifically evaluated in order to determine its eligibility to be delivered as the final product of the project.

Acknowledgments

I would like to thank my supervisor, David Duke, for his invaluable advice on both theoretical and technical issues throughout the course of the last few months, as well as for his useful feedback on the mid-project report I presented to him. I would also like to extend my gratitude to my assessor Olaf Beyersdorff, who provided valuable insight during the progress meeting and feedback on the mid-project report.

Contents

Chapter 1

Introduction

1.1 Overview

The scission of a nucleus into two fragments is at present the least understood part of the fission process, though the most important for the formation of the observables (Ivanyuk, 2013) in field of Quantum Physics and researches in Nuclear Energy because of major role it plays in fundamentals of Nuclear Fission either theoretically or in practice. Reliable predictions of fission product yields are also important for the technical application of nuclear reactions for the energy production and transmutation of nuclear waste in reactors and for developing advanced new generation nuclear reactors.

Even today, the most widely used theories of fission rely on the macroscopic–microscopic approach to nuclear structure and semi-classical dynamics based on the Langevin equations(Schunck, 2013) In the recent years, the rapid development of computers power has given scientists the opportunity to successfully implement the full microscopic theory of fission. Recently physical-mathematical approaches like The Skyrme nuclear energy density functional theory (DFT)(Schunck, 2013) or formulations like the Strutinsky's optimal shape based on the changes in level of energy at session point (Ivanyuk, 2013) are used for understanding the fission process and calculating scission point.

In fact getting benefit from significant raise in computer power in last decade was limited to developing computational implementation of mathematical approaches used manually for calculations of fission process or it has focused on techniques for representing individual properties of the phenomena (David, 2012) where applying other computational approaches like use of Information Visualisation techniques for simulating multiple properties of the process has not been widely used. Applying mentioned techniques would aid scientists to draw a visual analysis of the process, gaining more insight to its components.

It is argued that getting benefit from applying well-known Information Visualization techniques would be a useful approach which aids scientists in visual exploration of information gathered from real data helping them gaining insight to less known aspect of this physical phenomenon although it seems this area needs to be brought to more attention of computer visualization experts. The latest issue is subject of a research done by a group of researchers including David Duke, Hamish Carr

(David, 2012) from University of Leeds. Their approach in applying Topological Analysis Technique in particular Joint Contour Net (JCN) to real data for reducing data fidelity results producing both 2D and 3D visualization of the Scission Point process where both two sorts of visualization are represented in slides generated from running substantial amount of codes written in two different programming languages.

This project aims to develop a Graphical User Interface for representing visualization output of codes written by the research group. The GUI should be developed taking into the account special user requirements, HCI principles and usability goals defined at the design stage of project helping scientists getting benefit from the visualization output without concerning about mechanics of underlying code. Also it is designed to provide functionalities aiding scientists to make visual sense of input data for more investigation and deriving scientific conclusions.

1.2 Aim

The aim of this project is to design and develop a GUI for representing the visualization slides generated by the underlying code to scientists. The GUI should provide functionality of accessing to different levels of details from abstract level to low level data aiding scientists to review represented slides from different points of process, finding points of interest, flagging them out and adding notes to them for further exploration. Also it should give opportunity of visual comparison between outputs of different experiment runs aiding scientists deriving scientific conclusions from data explored; also it should include functionality of representing the visual output efficiently in case of probable extension in quantity of given point steps within the experiment. The GUI should be extensible to the point it interacts with underlying code for generating visualization slide in case the slide is not already generated for a specific point in the experiment.

1.3 Objectives

The objectives of project are:

- To develop a well-designed GUI representing visualization slides aiding scientists gaining more insight to Session Point by exploring information from the real experiments data.
- To evaluate the GUI design and functionalities based on usability goals and HCI principles.

1.4 Minimum Requirements

- To design and develop a GUI for representing already generated visualization slides based on established requirements, capable of providing the set of functionalities necessary for efficient interaction between user and the interface including:
    1. Providing functionality of zooming in and out to different levels of details in represented output of underlying code.
    2. Functionality of flagging out points of interest, adding text data to them for further investigation.
    3. Functionality of providing parallel visual comparison between outputs of different experiment runs.
- Design and develop the GUI in an extensible manner so the interface functionalities could be extended by adding new modules to the source code.

1.5 Deliverables

Project would deliver a functional GUI software package accompanying with relevant documentation on how to use the GUI.

1.6 Possible extensions

Possible extensions to project could include:

- Interface interacting with underlying code to run the code, generating new slide in case the slid is not generated for a given point of data.
- Implementing capability of adding other sort of data properties for example audio to the points of interest.
- Implementing Capability of search for particular pattern in given data set using search filters.

1.7 Schedule and Project Management

The first scheduling work was done in this project's first week. The project was broken down to sensible milestones and for each millstone subtasks were defined taking into account project deadlines. By doing this, the work was done safely in the knowledge that if the deadlines were kept to

that point, there would be enough time to complete all the milestones of the project. This helped to prevent the enormity of the task from becoming too overwhelming.

Below is set out the initial breakdown of project milestones, the list is laid out by week number of the project, there being 6 weeks in total.

Stage 1. Week 1:

This week is dedicated to initial investigation and background research for achieving first milestone which includes both progressing on write up and submitting Mid Project Report. Other sub tasks for first week milestone are defined as:

- To investigate previous FYPs to figure out any project completed in this field and gain more insight to general approach of completing the project, also devise the general break down of the report and confirm it in discussion with my supervisor.
- To decide on the most appropriate approach for managing project in discussion with my supervisor, accordingly devise project management plan for the project defining milestones and subtasks, calculating the time resource needed for each task and reasonably distributing time between tasks.
- To complete background research, collecting resources on Time Series Data, their characteristics and well known approach in field of Information Visualization for exploring facts from Time Dependant Data Series.
- To complete background research, collecting resources on Nuclear Fission process in particular Scission point both from physical and computational point of view.
- To discuss the scope of problem with my supervisor to have brief and clear understanding of scope of the problem and appropriate approach for tackling the problem producing the solution to problem.
- To discuss possible evaluation techniques with my supervisor to take the most appropriate approach in order to be able to overcome possible problems and barriers and propose the adequate solution.
- Completing writing of first two chapters of the project report, as well as preparing the mid project report to submit using gathered information.

Stage 2. Week 2 to 4:

These three weeks are dedicated to design the GUI based on established requirements from users and produce a low fidelity prototype using Story Boarding approach and receive user feedback then start coding in Python. By the end of this stage of project a high fidelity prototype should be produced. Following Agile method for creating the prototype, also taking into the account incremental delivery of the project, this prototype would be improved in two separate iterations to be delivered as the final product of the project. Subtasks for this milestone are defined as below:

- To complete gathering requirement from users and start designing GUI followed by producing a low fidelity prototype using Story Boarding approach, presenting the prototype to users for receiving feedback.
- To complete investigation on choosing the most appropriate GUI Toolkit Package for the project from number of available solutions in discussion with my supervisor.
- To complete research on new technologies required to be adapted for this project both on GUI Toolkit and new Python libraries to obtain knowledge how the underlying code should be structured.
- To gain confidence on adapting new technologies by complete practicing tasks on how to use them for the purpose of project.
- To start coding in Python aiming producing a high fidelity data based on the received user feedback and the choice made on the GUI Toolkit Package.
- To complete writing up Chapter3 and Chapter 4 of the report by the end of this stage of project.
- To produce a high fidelity prototype as the main deliverable for the next stage of project.
- To test the high fidelity prototype for rectifying possible technical issues including debugging.
- To evaluate the prototype based on the chosen approach in last stage of the project. Completing this task successfully would be vital for next stage of project.

Stage 3. Week 5:

This week of project would be spent on completing first iteration for improving high fidelity prototype delivered from last stage. Both two iterations would include main steps in designing

interface from HCI and usability point of view including Design, Implementation, Testing, and evaluation. By evaluating the interface at the end of this stage we expect improvement in user experience. Subtasks of this stage are defined as below:

- To make reasonable decisions on improving the design of GUI based on the evaluation result from last stage.
- To implement improvements by manipulating existing code or adding new modules.
- To test the prototype for rectifying technical issue in particular debugging.
- To evaluate the prototype on the chosen approach in first stage of the project.
- To complete writing up of Chapter 5 of report based on the information gathered from incremental experience.

Stage 4. Week 6:

This week is dedicated to complete the final iteration of improvement and delivering the final project. At this stage by evaluating the GUI we expect the evolutionary prototype to be distinguished "Good enough" and we can deliver it as GUI software package for the project. Subtasks for this stage are defined as below:

- To make reasonable decisions on improving the design of GUI based on the evaluation result from last stage.
- To implement improvements by manipulating existing code or adding new modules.
- To test the prototype for rectifying technical issue in particular debugging.
- To evaluate the prototype on the chosen approach in first stage of the project to conclude the product is good enough to go live.
- To complete writing up of Chapter 6 of report based on the information gathered from incremental experience.
- To review the report for making improvements and proof reading it.
- Submitting the Final Project Report by 31st of May 2013.

1.8 Relevance to Degree Programme

This project is relevant to several modules studied during my IT BSc course. Principles of designing user interface discussed in details in Usability Design module (COMP3442) in third year of my study.

Also the use of well-known visualization techniques for visualizing different data sets is taught in Information Visualization module (COMP3736) in the third year. Programming Computer module (COMP2245) taught in second year provides me by deep knowledge of programming in Python which is assumed as a vital skill for this project. By Nature this project could be assumed as a proper instance of IT projects, the set of skills I had developed by completing Software Project Management module(COMP1945) during first year of my study is helping me choosing the appropriate methodology for efficient management of this project.

Chapter 2

Background research

2.1. Context

"A time series is a set of observations measured sequentially through time. These measurements may be made by continuously through time or be taken at discrete set of time points. By convention, these two types of series are called continuous and discrete time series, respectively, even though the measured variable may be discrete or continuous in either case"(Janacek, 1993).

Because of consequences of variables' state transforming from one stage to another by time in for casting, gathering and analysing time series is widely used in business and economic forecasting, econometric and finance, earth science, physics etc. The analysis of time series data is one of the most common problems in any data domain, and the most common techniques of visualizing time series data (sequence charts, point charts, bar charts, line graphs, and circle graphs) have existed for hundreds of years(Maciejewski, 2011).

The aim of time series visualization is to put into context relationship of past, present and future(Maciejewski, 2011). In order to understand the current state of phenomena we need to take into account the fact of time therefore it is always helpful to try understanding the historical concepts of phenomena. Applying visualization techniques to time series data would provide the opportunity of exploring information in regards to analysing how different processes continue or evolve through time, the fact which has considerable importance in science.

How we go about analysing and visualising data on one or more time series? The special feature of time series data is that successive observations are usually not independent(Janacek, 1993) so analysis and visualisation should take into account the order in which the data is collected. A well designed visualization can aid in answering the following questions for unknown dataset(MacEachren, 1995):

- Does a data element exist at a specific time? (Existence of a data element)
- When does a data element exist on time? Is there any cyclic behaviour? (Temporal location)
- How long is the time span from beginning to end of the data element? (Temporal interval)
- How often does a data element occur? (Temporal texture)

- How fast is a data element changing or how much difference is there from data element to data element over time? (Rate of change)
- In what order do data elements appear? (Sequence)
- Do data elements exist together? (Synchronization)

The visual exploration of time-dependent data requires a special treatment of the factor time. On the other side, a number of special-purpose visualization techniques for time-dependent data have been developed lately. Each approach has specific characteristics and advantages. One has to decide, which technique to apply for a given problem (Schumann, 2003). For finding the appropriate approach we need to consider set of attributes including static vs. dynamic representation, data vs. event visualization and conventional vs. multivariate display (Schumann, 2003). Since there is more than one technique available for each characteristic, further decisions have to be made based on the characteristics of the time-dependence.

2.2 Visualizing Time Series Data

Data Visualization has been used for exploring time series data for a long time therefore variety of visualization techniques has been adapted based on the different characteristics of data set. Based on the time dependence of the visual representations we can distinguish two cases for time series visualization (Schumann, 2003):

1. The visual representation is not time-dependent: The visual representation does not automatically change over time. Modifications result from user interaction only. In this case, we speak about static representations.
2. The visual representation is time-dependent: The visual representation changes dynamically over time and is a function of time. We call this form dynamic representation.

The decision of which kind of representation should be used has to be made based on the nature of task as both two forms have their benefits. In case of static representation a number of well-known techniques have been developed through years (Schumann, 2003), however most of them are limited to representation of a single variable over limited number of time steps which are not matter of our interest as scission point does not fall in this category.

Dynamic representation makes direct use of time to map some data. Dynamic representation techniques are able to demonstrate the general temporal behaviour, the dynamic of data and processes

very well. Let the data elements d_i be given over a time period from *tstart* to *tend*. The most natural approach is to map the temporal aspects of the data directly onto the time control of a dynamic representation, where the data elements' representation changes over time (e.g. size, shape, colour, texture, transformation)(Schumann, 2003).

In case of continuous linear time it is a natural approach to represent the time *t* found in the data directly and simply to generate appropriate representations for the data values *d* depending on the time *t*. In fact, we have to sample *f(t)* continuously regarding time to get data elements d_i to be visualized. This leads to a computer animation, where each element d_i corresponds to a frame *i* in the animation(Schumann, 2003).

For interval data, the depiction of smooth movements for changes is not of interest. Nevertheless, time can be mapped to the temporal aspect of the presentation. In this case, this leads to a presentation similar to a slide show, where data elements d_i are presented sequentially over time(Schumann, 2003). The difference with respect to animation is that in this case no attempt is done to achieve the perception of a continuous movement(Schumann, 2003).For discrete time steps we can assume the underlying time model is continues which lead us to take the same approach and visualize data in slide show.

All the approaches discussed above are based on the assumption the underlying time model is linear discrete or continuous. As a specific instance we can point to event-based data, where we assume that an event represents an occurrence in time signalling a change in data. A general aspect of event-based data is that time between following events is usually not constant(Schumann, 2003).

In case of real time monitoring of event-based data, the applicability of information visualization general approaches is proven e.g. by Matković (Matcović, 2002) who applied successfully level-of-detail and focus and context techniques for real-time monitoring of processes(Schumann, 2003).

Events are matter of special interest as they represent point of time that changes are going to happen in data and the underlying model. Dynamic visualization is a valuable tool to detect these events and the corresponding changes in time-series data(Schumann, 2003). Currently techniques for detecting features in spatial data sets are applied to detect event in given data set. Post (Post F.H. and Doleisch, 2002) apply this idea to the analysis of flow field data. He interprets the as the counterpart of spatial features in the evolution of features. Based on this assumption they distinguished between the following types of events on spatial features (Post F.H. and Doleisch, 2002):

- Birth or death,

- Entry or exit,
- Split or merge, and
- Topology change

Visualization can surely support the exploration of time-dependent data. Further investigation seems however to be necessary to couple the presented visualization methods with those depicting also the spatial context, since many time-dependent data sets exhibit spatial dependencies as well(Schumann, 2003). Moreover, in the field of event-based visualizations scientific work just has started, and further concepts must be developed.

2.3 Scission Point: A Specific Instance of Time Series Data

In last few sections the importance of accurate calculation of scission point was discussed also an overview of time series data structure was provided aiming to familiarise reader with approaches taken for visualising time dependant data series. In next sections of this report we will focus on problem we consider to solve, developing interface for supporting scientists identifying the scission point. For this purpose we define the scission point from physics perspective and we discuss why we can assume this definition has the characteristics of time series data model and how the assumption we made will help us to solve the problem efficiently.

"In the course of a fission process, we may single out a particular time from which the nuclear density in the neck region between the nascent fragments may be considered as vanishing. In an independent-particle approximation or for a correlated system considering the canonical basis states , after such a time we would ideally expect that the wave function of any nucleon spreads over only one fragment. Therefore, after this event of separation, a given nucleon should then belong to only one fragment and the dynamics of nucleons in one of the fragments F1 become independent of their nuclear interactions with nucleons in the other fragment F2. The dynamics of nucleons in F1 would be therefore governed only by the internal interactions and the external Coulomb potential created by F2 and decreasing with time. This particular point in the fission process is called the scission point"(Bonneau et al., 2007).

The process of nuclear fission or fusion and position of Scission Point in the process from Physics point of view is not matter of interest in this project however familiarity with the nature of process and understanding which variables play the crucial role in the process over time helps us to gain better insight to the data set provided for the project. By investigating the nature of event and provided data set we distinguish that the factor of time is not one of the variables in the data set. The data set is an

example of multivariate data which includes real data gathered from experiments done in different approaches where for each specific approach several runs are completed based on eight different values of quantum variables of nuclei so result of each single run we have a dataset of the quantum variable representing energy level for the given point. However physically the fission process and Scission Point happens continuously over time and time plays a crucial role in changes in different variable values in particular nuclei energy level, therefore based on the discussion about the Time Series Data Visualization definitions and techniques, we can assume that data gathered of changes in different quantum variables of nuclei results a dynamic multivariate time dependant data set which represents Scission Point as an event-based example of dynamic time dependant data series.

Exploring information from the dataset adapting well-known approaches in visualizing dynamic time series data sets is aiding scientists in understanding a fundamental physical phenomenon and giving them new insight into the detection of nuclear scission (David, 2012). However nature of event results a complex dataset including variation of different variables taken in numerous time steps which leads to getting benefit from new approaches of reducing data fidelity for simplifying locating the crucial scission point. Also complexity of dataset would impose limitations on volume of information and level of details GUI represents to user, changing the way user interacts with GUI and it will have impacts on what kind of functionality and to which level GUI provides to user.

2.4 The Problem

The aim of this chapter is to explain hybrid nature of problem which means finding solution for this problem is dependent on understanding principles and taking advantage of codes written for problem solved by David's team(David, 2012). Therefore at the beginning of this chapter abstract view of David's problem(David, 2012) and its solution are reviewed, followed by defining my problem and possible methodologies could be taken for solving it.

2.4.1 Visualizing Nuclear Scission through a Multifield Extension of Topological Analysis(David, 2012)

In the article published in IEE journal(David, 2012) it is briefly explained how JCN approach is applied to time dependant data series of nuclear scission for visualising nuclear scission process supporting scientists to identify scission point. This section will draw an abstract view of the problem they considered and the applied approach for solving it.

Early models of fission were based on an empirical liquid drop picture of the nucleus (David, 2012), in these models the initial assumption is nucleus is a drop, by stretching this drop it breaks in two, that's how fissions occurs also the moment of breaking down is Scission Point (Bohr and Wheeler, 1939). Modern approaches are try to derive the scientific conclusion based on studying nucleon nucleon interaction, explaining fission process (David, 2012). The major theoretical approach in the recent case is Density Functional theory(DFT) (Bender et al., 2003).

DFT central assumption is that the complex interactions of protons and neutrons within the nucleus can be hierarchized (David, 2012).In First approximation , everything happens if all nucleus were moving independently from another in some average quantum potential, the nuclear mean field(David, 2012) (David, 2012). Spontaneous symmetry breaking is invoked to deform the mean field introducing first class of correlations(David, 2012).

Because fission involves 'stretching' the nucleus, the DFT treatment of the problem begins with identifying relevant deformation degrees of freedom Q of the mean field(David, 2012). A realistic description of fission involves at least N >= 4 degrees of freedom such as elongation, triaxiality, mass asymmetry, degree of necking, etc.(David, 2012).The list if degrees of freedom defines what is called the " collective space" (David, 2012).

Following Density Functional Theory we can assume different degrees of freedom as variables *(Q0, Q1, Q2 ...)* which can have different possible values. We can use 2D to nD space for representing them; in that case the intersection of values of degrees of freedom would represent the energy level of nucleus at intersection point.

Density Functional Theory explains how nucleus energy level varies through the fission process especially at Scission Point. When one nucleus breaks in two nuclei massive amount of energy is released, this assumption is basis of nuclear reactors functionality for producing nuclear energy. Therefore the energy level in the state before Scission Point drops down significantly to the state after Scission Point. Conclusion is derived that the radical change in energy level is representing the Scission Point so investigating changes in nucleus energy level plays a crucial role in calculation of Scission Point.

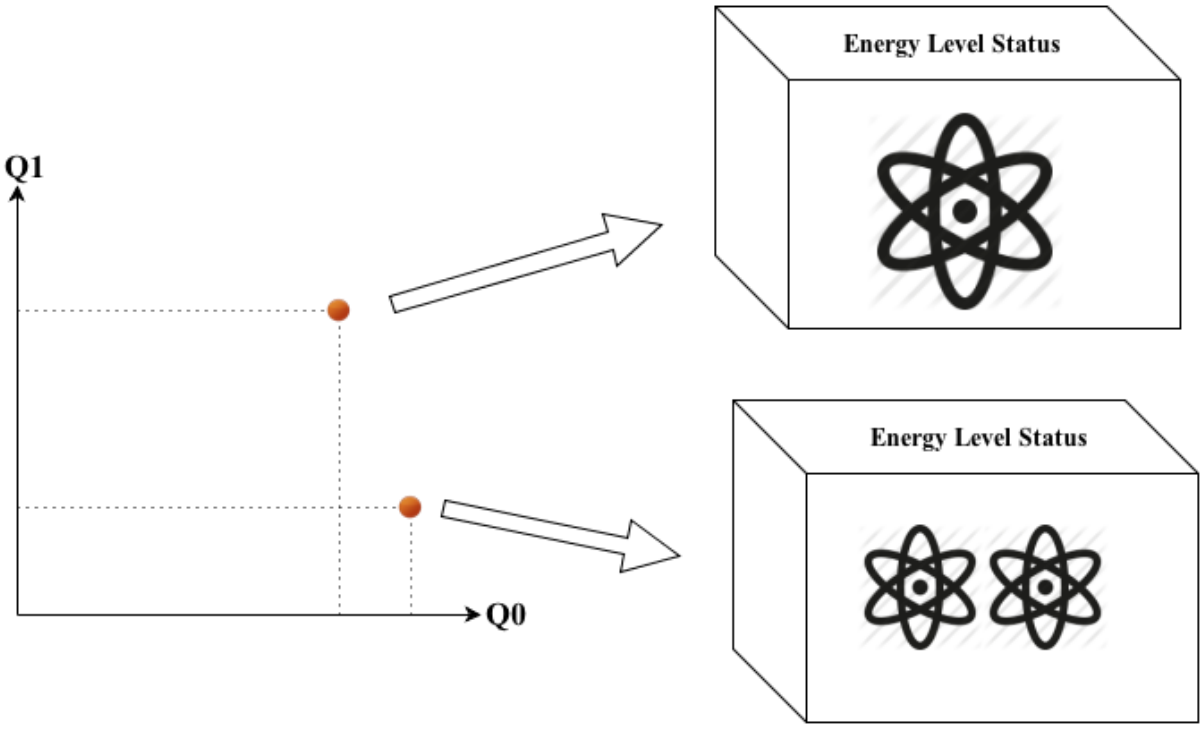

Figure 1. Different values of quantum variables of nuclei represent different energy level status.

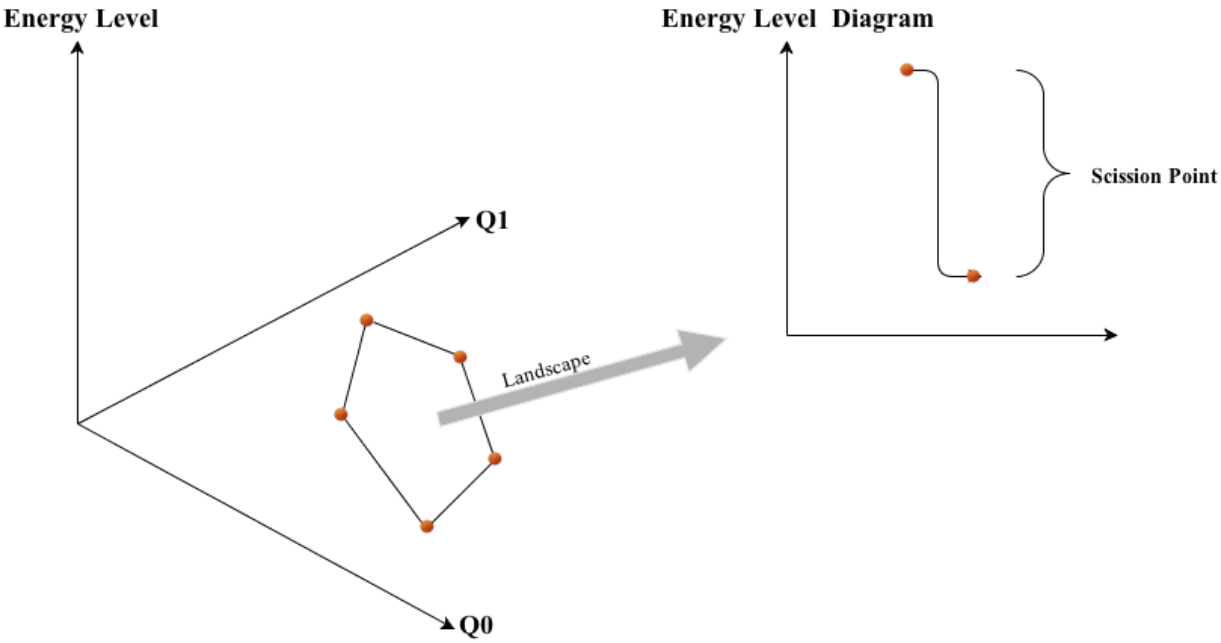

Figure 2. Different values of Q shape a landscape representing different levels of energy at given points, significant drop in the energy level points out Scission Point.

For investigating variations in nucleus energy level we can draw an nD space for showing points which represent intersection of values of Q (1, n) and the energy level at that state. We can calculate different points based on various values of Q (1, n) in the nD space, by connecting calculated points we would derive a landscape representing rise and falls in energy level through the points. Based on

the discussion above by investigating variation in energy level in the landscape, we can assume that a significant fall in energy level in the landscape is representing the Scission point.

However high fidelity of data and complexity of the landscape makes working out using this approach a complicated task, therefore we need to take reasonable approaches for reducing data fidelity simplifying the task. Since we have to detect events between objects encoded in a multi field composed of multiple scalar fields this problem is well suited to topological analysis.

2.4.1.1 Scalar Topological Analysis

Topological techniques for scientific data analysis have become increasingly popular and it has been applied to the analysis and visualization of scientific data sets (Pascucci et al., 2010). In this context two complementary approaches have been developed: contour based analysis and gradient analysis (David, 2012). Contour based analysis detects objects and their relationships also this approach is computationally cheaper and simpler than gradient based analysis therefore the contour based analysis is adapted for this task (David, 2012).

2.4.1.2 Joint Contour Nets (David, 2012)

Joint Contour Nets (JCN)s (Duke, 2011), (Duke, 2012.) approximate the Reeb space for sampled multivariate data. As with scalar topology, the first step is to reduce the data to a graph - in this case, based on quantization of the input data (David, 2012). Based on recent work on the relationship between histograms, sampling, isosurfaces and quantization(B. Duffy, 2012) , the Joint Contour Net is constructed in three stages. In the first, each cell of the mesh is explicitly subdivided along contours (isosurfaces) of the individual variables to decompose the domain into slabs of constant (quantized) values in each cell. In the second, the adjacency graph of the slabs is extracted: i.e. the graph representation of isovalued regions. Finally, adjacent slabs (in neighbouring cells) with identical isovalues are collapsed to compute the Joint Contour Net (Duke, 2012.). This representation then encapsulates the topological structure of the multi field at the chosen level of quantization (David, 2012). Moreover, varying the coarseness of quantization can be used to simplify the Joint Contour Net on demand, and the contour tree turns out to be a special case (David, 2012).

Applying JCN approach to the dataset results simplifying data, increasing data fidelity also generating contour tress for different values of Q1, Q2 etc. Result is represented in slides as one of the outputs of the underlying code. The other put of the code is generating 3D visualization slides taking advantage of visualization pipeline implemented in VTK version 5.8.

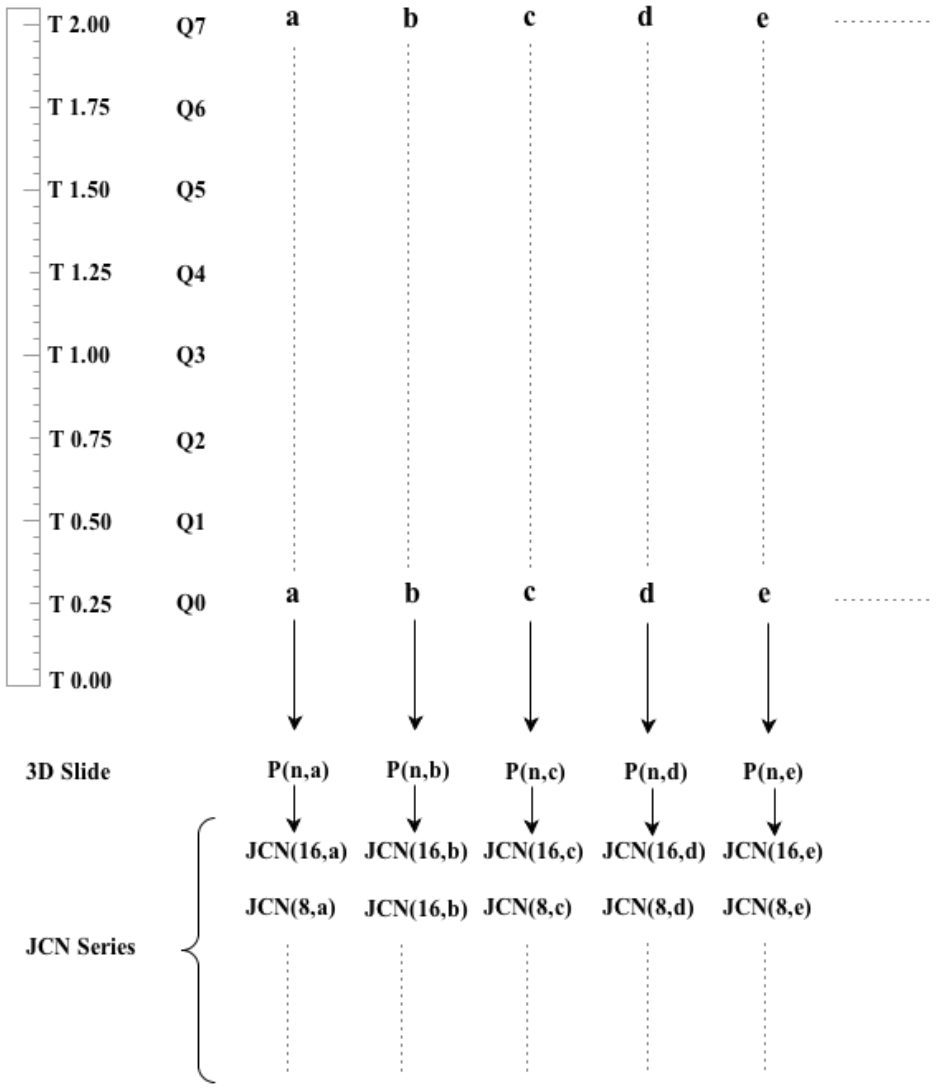

Figure 3. For any given point in data set result of variation in T values, a 3D slide and a series of JCNs are generated.

Figure 2 is representing process abstract; It is assumed for each given quantum variable Q (Q0, Q1 …) we have a dataset (a, b,c,…). Values of mentioned data set relies of value of T which is another quantum variable and can take 9 different values varying from 0.00 to 2.00. Therefore we can assume by applying JCN method to given Q dataset we can generate the JCN for different values of given dataset. Therefore the dataset includes dynamic multivariate data from occurred experiments based on different known approaches, when by taking each specific approach there could be various runs based on different values of T and result of each single run is a series of JCNs generated.

As seen in Figure 3 flow of tasks from raw data to generating JCNs is hierarchal therefore this hierarchy could be reused for organizing different levels of information GUI represents to the users and drawing boundaries for how much details would be represented from cognitive point of view.

Also this hierarchy could be useful for organizing files and folders on disk space, files and folders which are assumed as raw material for the GUI code.

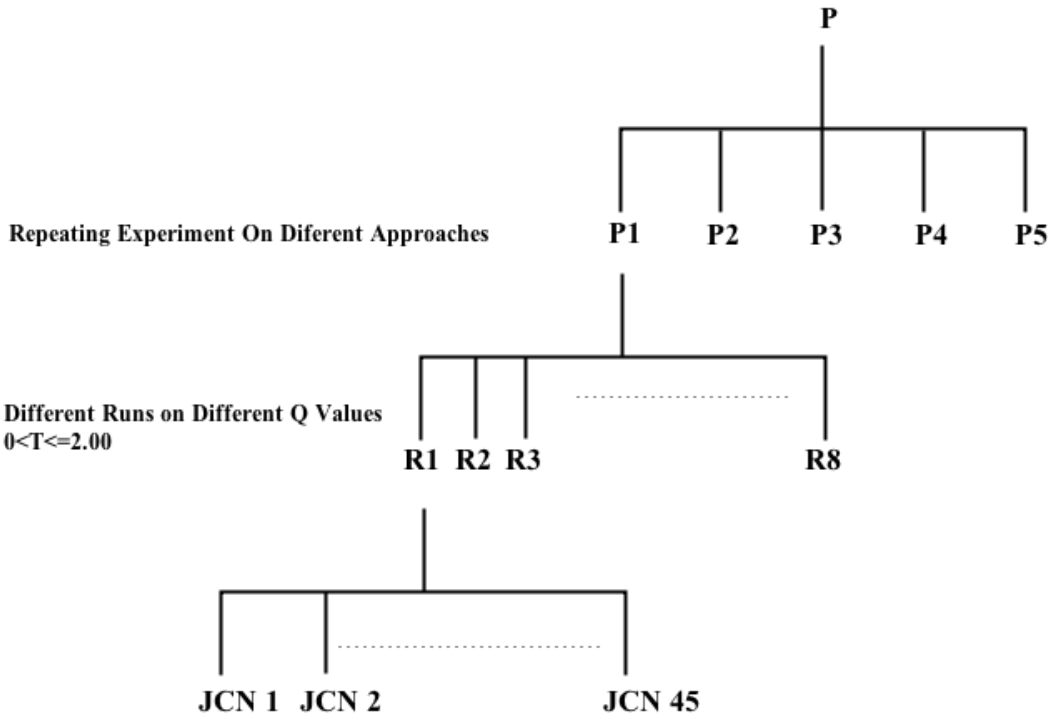

Figure 3. Abstract view of hierarchical structure and different levels of data in visualization

Figure 3 representing the hierarchy of JCN files on the storage space; at top of the hierarchy we have different experiments which are fundamentally different. At next level we are storing results of running same experiment P by taking different approaches, this will result P1, P2, etc. But these experiments are dependent on T values therefore at next level we have R1, R2, etc. which are result of experiments P1, P2, etc. done on different values of T. Finally for each of R1 to R8 we have a datasets of JCNs.

2.4.2 A GUI to Support Scission-Point Identification

The fundamentals of the precedent problem and different outputs of code written for solving the problem were discussed in last chapters. It is made clear that result of running the code is a series of JCN slides which are scientist's point of interest. Examining JCNs will provide scientists by opportunity of identifying interesting facts about scission process including scission point. Therefore

the current problem is designing, implementing and evaluating a GUI supporting scientists with their investigation on JCN slides.

The GUI should provide capability of investigating slides; this capability is gained by implementing zooming, inverting, mirroring facilities. Design process needs to take into account basic GUI functionalities like opening and saving files, designing menu bars and tool bars as well as HCI issues including cognitive load and cost of action for users.

Also it is necessary to highlight the point that specific type of users of this project has significant consequences on how the interface is designed and evaluated. In fact users of interface are scientists, this means their interactions with computer systems differ; consequently they differ in how they cognitively interact with the interface and how they use the interface on day to day basis for completing their tasks. These issues impact the use of real estate and cognitive load represented to the users.

The other issue is how different levels of complex information are represented efficiently by the interface. In last section it was discussed that the output slides are generated based on different values of other variables, also we mentioned the hierarchy which will useful for categorising slides on storage device. The same rational could be applied to different levels of complexity represented to user by the interface.

GUI should be designed and implemented addressing all mentioned issues, providing capabilities require by scientists supporting them with their tasks, also it should be evaluated based on the user type and their specific requirements.

2.5 Related Work

Designing and developing user interface for handling different sort of visualisation and representing the output to use has become popular due to increase in using visualisation for exploring information. There are number of commercial and open source visualisation toolkits developed by companies or academic organizations for purpose of data exploration. This chapter will review couple of them, trying to trade-off between their advantages and disadvantages, addressing issues with my project whenever it is appropriate.

2.5.1 Piccolo

Piccolo is an open source toolkit developed by The Human-Computer Interaction lab (HCIL) of University of Maryland. It is developed on 3 different platforms, all versions have the same core functionalities and few differences which make them preferable to be used on platform like mobiles etc., three versions are Piccolo. Java, Piccolo.NET, and PocketPiccolo.NET; they are developed with Java and C# aiming better speed and memory performance. It is declared that this toolkit is designed for 2D graphic visualization and has capability of high quality rendering and multiple views, zooming and panning.

Image 1 represents Piccolo.Java toolkit interface(HCIL, 2002) which is composite widget consists of multiple box widgets laid out from left to right which provides different functionalities to user. Using theses functionalities is not straight forward therefore a comprehensive documentation is implemented to help user with using the interface.

Simplicity and safety which is gained from providing user by useful feedback of interactions are keys in this interface, in other hand interface impose huge cognitive load to the user which results confusion especially for who uses the interface infrequently.

The other issue is colour hue used for presenting some parts of the interface. Inefficient use of colour hue is making problem for colour deficient audience, refraining the toolkit from making a good user experience for them. Also ambiguity in labelling widgets on the screen is adding more complexity to interface diminishing the simple design of it.

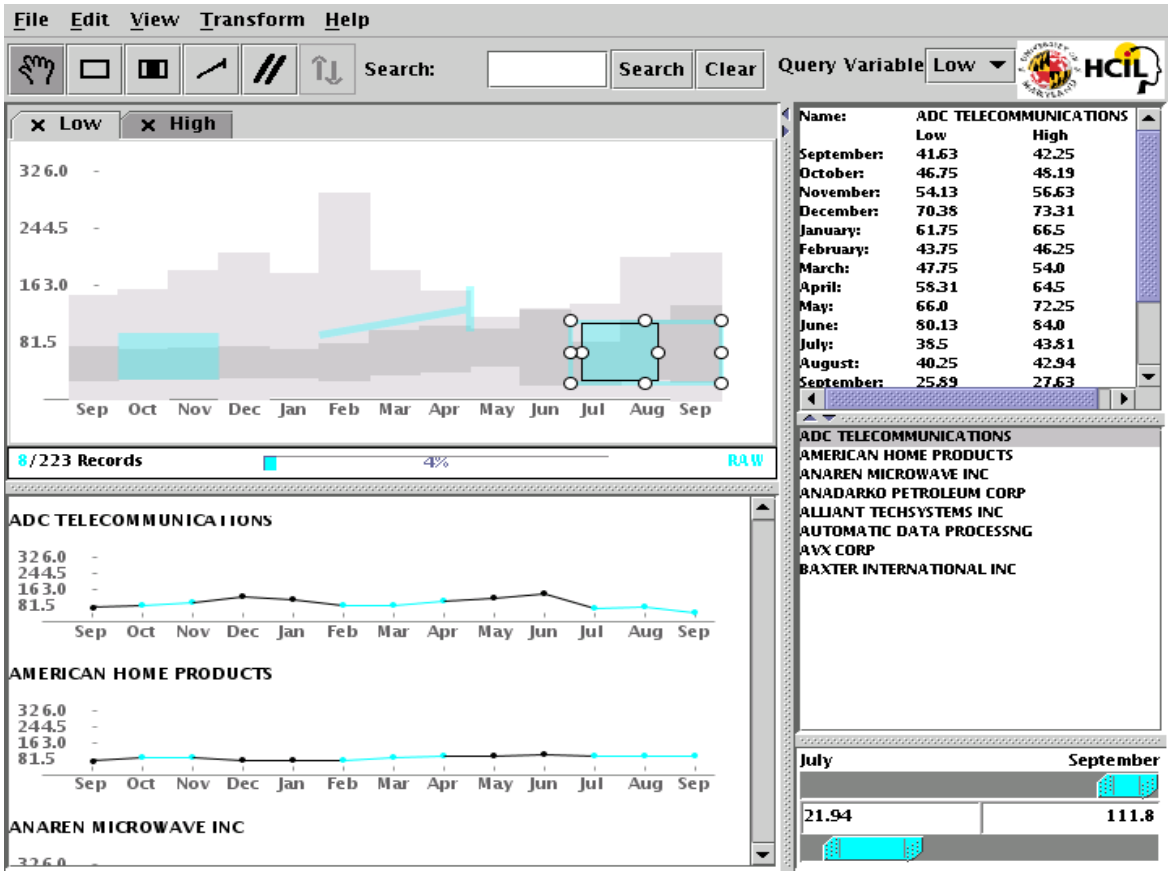

Image 1. Piccolo.Java toolkit user interface (HCIL, 2002).

On HCIL web site is mentioned most of development team time is spent on optimising the performance of the underlying code to increase efficiency in speed and memory or adding more built in functionalities to the interface. Therefore it could be concluded paying less attention to design principles and HCI goals is making the interface inefficient although the toolkit is highly utilized.

The interface has good utility and is it equipped by useful functionalities which help user completing their task although the interface is not easy to learn and it is not memorable which means user should put unreasonable effort and more time to learn and remember how to use the interface.

Image2. Showing Piccolo.Net interface (HCIL, 2002), in comparison with Image 1. Piccolo.Java toolkit user interface (HCIL, 2002) is even more complex. This user interface does not support users in supporting their task effectively as complexity of the interface and ambiguous widget labelling is taking user time to figure out how the interface carry out a particular task. Also like its precedent version it is not easy to learn how the interface works without referring to interface comprehensive help document and in case of learning, the probability of user remembering all the steps for delivering a simple task is not high.

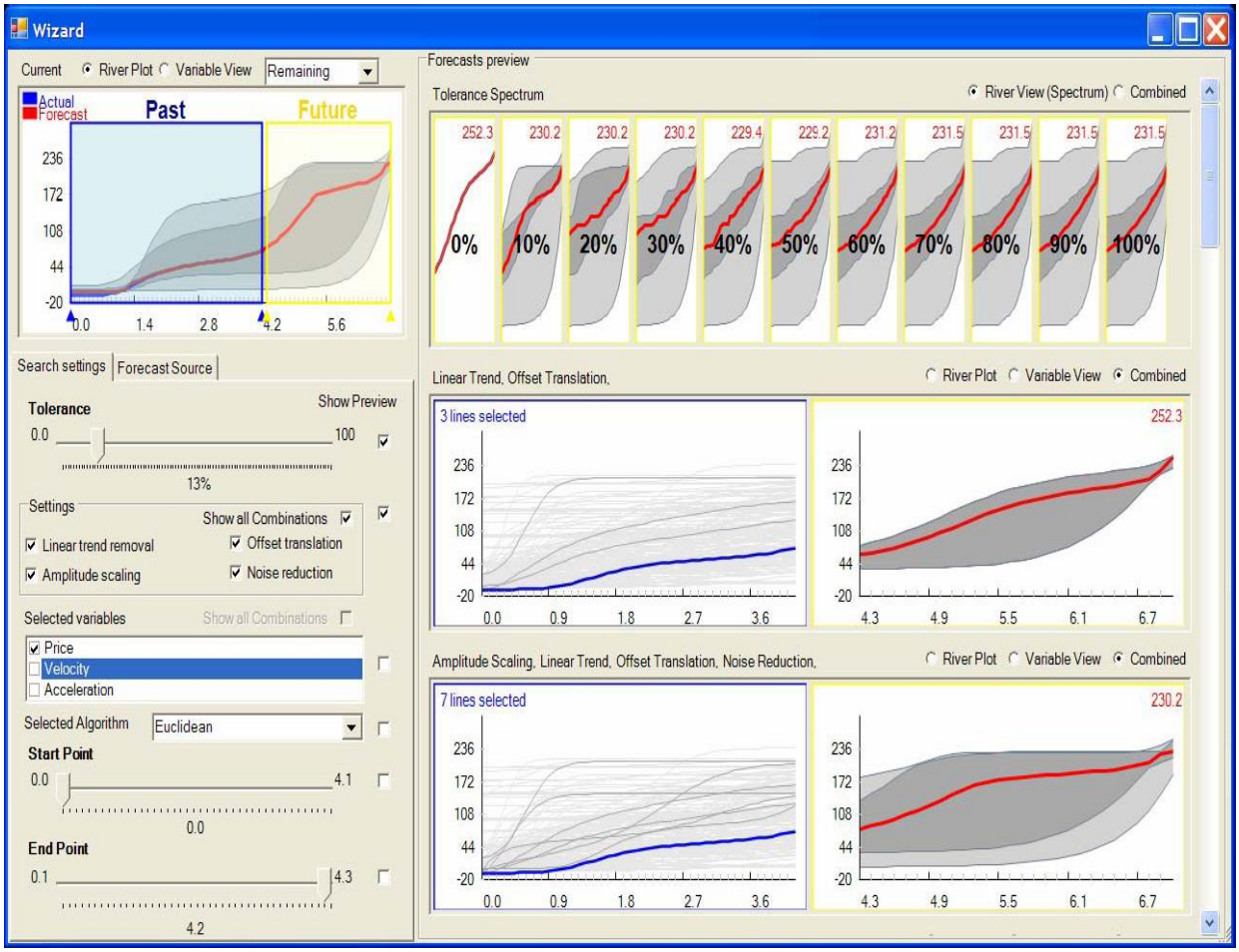

Image 2. Piccolo.Net toolkit user interface (HCIL, 2002)

However the interface is safe to use which means by providing appropriate feedback it refrains users from delivering dangerous tasks which can be assumed as the positive point of the interface although like its precedent interface design is suffering from complexity which results imposing high cognitive load to the user. All mentioned characteristics are leading user to unpleasant and frustrating user experience.

On HCIL web site is mentioned most of development team time is spent on optimising the performance of the underlying code to increase efficiency in speed and memory or adding more built in functionalities to the interface. Therefore it could be concluded paying less attention to design principles and HCI goals is making the interface inefficient although the toolkit is highly utilized.

In next section we will review another open source toolkit from design and HCI point of view to gain more practical knowledge of already designed interfaces, taking advantage of the knowledge in current project.

2.5.2 InfoVis

The InfoVis Toolkit is a Graphics Toolkit written in Java to ease the development of Information Visualization applications and component(Fekete, 2005). On InfoVis toolkit web site is declared that

It has unified data structure and by using homogeneous columns instead of compound types improves dramatically the memory required to store large tables, trees or graphs, and generally the time to manage them. Also it is mentioned that the InfoVis Toolkit can use accelerated graphics provided by Agile2D, an implementations of Java2D based on the OpenGL API for hardware accelerated graphics. On machine with hardware acceleration, some visualizations redisplay 100 times faster than with the standard Java2D implementation. the InfoVis Toolkit is meant to incorporate new information visualization techniques and is distributed with the full sources and with a very liberal licence. It could be a base for student projects, reseach projects or commercial products.

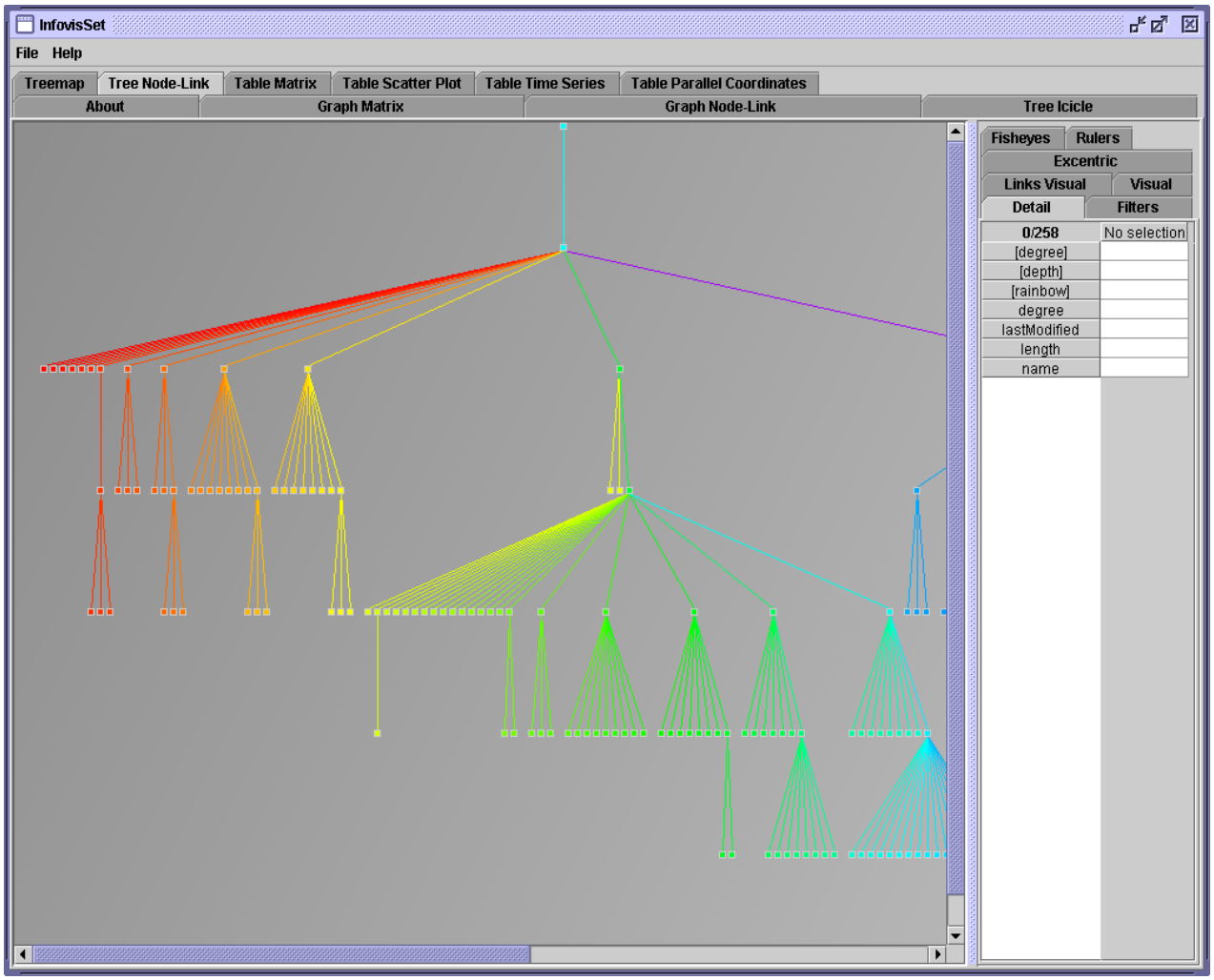

Image 3. InfoVis toolkit user interface(Fekete, 2005)

Image 3 representing InfoVis user interface(Fekete, 2005), it is divided to three main components which two from them are composite tabbed widgets and the other one is the main are for representing the presentation. Using tab widget is helping with reducing cost of action in other hand by relying the design just on the tabs instead of using toolbars etc. interface is becoming complex. Despite of the simple 3 component main frame the complex tabbed design is leading to frustration and confusion. Also like Piccolo toolkit inefficient use of colour hue which refrains colour deficient audience from interacting with interface remains an issue.

By taking to account usability goals it can claimed that the interface is not supporting the users in carrying their tasks efficiently without referring to the user manual although clear and sensible widget labels and feedback from the interface is useful to the user. The same as Piccolo project the emphasis on this project is on utilising interface with more functionalities rather than focusing more on the user experience and HCI side of the design which is adding more complexity to the interface affecting learnability and memorability of the interface. It is not easy to learn how the interface works as well as remembering how to use when users learn it.

2.6 Software Development Methodology

Selecting the appropriate approach for managing projects is the most important stage of the software project management which plays a vital role in either success or failure of the project (Bob Hughes, 2006). In order to select the appropriate approach for managing the project taking into the account specific project characteristics and risks is a necessity. In case of making decision of developing software in house, analysing project characteristics, risks and adapted technologies would light up the way to select the appropriate approach.

By reviewing the aim and objectives of the project it is concluded that we are dealing with a process oriented project (Bob Hughes, 2006) supposed to deliver a specific tool to be used by a small group of scientists. Also programming for this project would be done in Python based on justification that the substantial amount of underlying code is written mostly in Python therefore considering extensibility of the project, developing project in Python is the most preferred and sensible choice. In other hand by analysing resources we can distinguish high level uncertainty associated with product and resources of the project.

In fact this project would deliver the first generation of the GUI from this nature. Although designing and developing interfaces for time series visualization became popular, task specific characteristics, usage and users of this project make it unique.

Final product of the project is an UI, the fact which standalone emphasizes the iterative nature of the task for delivering the project, by taking into the account project characteristics and constraints getting benefit from agile methodology would be helpful for dealing with project high level risks, fast delivery and incremental nature of task.

It is necessary to make the point clear that specific characteristics of well-known agile approaches like Crystal Technologies, XP, Scrum etc.(Kenneth E. Kendall, 2005) which are defined to use collaborative approaches amongst a team of developers and adapting specific technologies for task delivery are not applicable to this project, however the principles and values of agile modelling would be applicable to be used for defining the general life cycle approach of the project.

Using agile approach brings fact of communication to the project. Systems projects require constant updating and technical design is especially prone to such errors. Add to this tight project deadlines, completing the first project of its nature and unavailability of the resources project has the potential for some serious communication problems which can delay the project. Agile methodology relies heavily on communication therefore problems are fixed rapidly, wholes are closed and weak thinking is quickly strengthen through to interaction with users (Kenneth E. Kendall, 2005).A second value the agile approach is that of simplicity. While working on the project from this nature, probability of becoming overwhelmed with complexity of task is high. Simplicity of agile approach will help the smooth run of the project. Feedback is the third value that is important when taking the agile approach. Concrete user feedback which is wrapped up with concept of time is useful to occur within certain time to help for making adjustments to make the progress flowing fast and smooth.

User interface design is an iterative process where users interact with designers and interface prototypes to decide on features organisation and look and feel of the system user interface(Sommerville, 2007). More commonly especially where iterative development is used, the user interface design proceeds incrementally as the software is developed. In both cases we should have developed and, ideally, tested some prototypes(Sommerville, 2007). Therefore developing an evolutionary prototype (Bob Hughes, 2006) to improve user involvement and clarification of partially known requirements is a major part of project. Prototyping would start by using “story boarding” approach to produce a low fidelity prototype and after getting feedback from users it will be implemented in Python throughout the project life cycle. This evolutionary prototype would be improved in two cycles of iteration to be qualified and delivered as final product of the project.

As we discussed nature of UI design lead to incremental delivery of the product (Sommerville, 2007). Taking advantage of incremental delivery approach we can improve later increments of prototyping by feedback from early stages which sound reasonable in absence of any similar system. Also incremental delivery would provide the possibility of changing the requirements because of the

shorter time span in design and delivery of the product. However the “requirement drift”(Kenneth E. Kendall, 2005) could be a possible disadvantage of taking this approach which would be mitigated by considering specific user requirements and evaluating prototype at final stage of each increment against the requirements. Also incremental delivery would give benefit to user by involving them in design and development process at early stages with a conventional approach. The other advantage for both users and developer would be facts that at the end of each increment some useful components are delivered to the users and breaking down the project to smaller sub-projects makes managing the project and controlling the risks easier.

Taking agile methodology and delivering an evolutionary prototype in three separate increments provide both developer and users by opportunity of better interaction and efficient use of available resources also it helps with mitigating the risks associate with project or in case of incidents occurring dealing with them effectively.

2.7 GUI Development Methodology

Wide variety of UI methodologies has been developed in recent years; each one is focusing on different aspects of user interactions and application design requirements for providing methods of design for diversity(Savidis and Stephanidis, 2004). Although taking each of appropriate approaches would be addressing issues in relation to managing the UI design and development process, new engineering methodologies and development tools emerge in cases where existing methods and instruments are not sufficient to effectively address the development challenges for new categories of interfaces. Therefore, there is a need for a systematic process in which alternative design decisions for different design parameters may be supported which lead us to hybrid design methodologies (Savidis and Stephanidis, 2004). Despite of reasonable differences in methodology details of current UI methodologies, core task flow from requirement gathering to design, implement, test and evaluation remains the same as the fundamentals of UI design.

The aim of this chapter is to justify the choice of most appropriate UI methodology for my project so I will review a wide spectrum of key existing design methods, their focus ranging from the low-level aspects of interaction to higher-level design artifacts, while addressing their appropriateness in accommodating specific requirements of my project and finally I will derive the conclusion of a well suited methodology for my specific task.

2.7.1 Task-oriented design methods

Methods such as hierarchical task decomposition (Johnson, 1988)and task-action(Payne, 1984), are perhaps the most representative examples of the task-based design methods. Those techniques do not encompass the capability to differentiate and represent design alternatives for the same task, mapping to varying design problem parameters. Instead, the design outcome reflects only a specific instance of the design parameters, and conveys only a single task-based structure. Consequently, following the design formalism of task-based methods, the differentiation at various levels of the task model would need to be represented via distinct alternative task-hierarchies, i.e. alternative designs. Each of those distinct design instances would need to be potentially produced for each different snapshot of values of the design parameters, leading to a large number of alternative designs(Savidis and Stephanidis, 2004).

2.7.2 Interaction-object oriented design methods

Such techniques are based on the asynchronous nature of user actions, providing models for representing concurrent user-activities, mainly through operators for expressing ordering relationships and progress control(Savidis and Stephanidis, 2004). They usually rely upon the theory of reactive systems, while the most representative examples in this category are CSP(Hoare, 1978) and UAN(Hartson, 1990). These models suffer from two main problems: (a) they are mostly suited for representing temporal action oriented dialogue properties, rather than constructional (i.e. they cannot represent structural interface relationships) and (b) they are quite low-level, by being very close to directly computable models, such as an implementation language, rather than to a design language, rendering them more appropriate for small design projects(Savidis and Stephanidis, 2004).

2.7.3 Visual/graphical design methods

Such methods are a type of straightforward physical design, in the sense that concrete visual interface snapshots are produced, via an iterative sketching and refinement interface prototyping process. The design process is conducted by managing design guidelines, usability criteria, artistic design, and task-oriented knowledge, towards more effective visual communication. In the context of this paper, such methods are considered to realize an artifact generation process, which can be employed at various stages of the interface design process, whenever the design needs to be instantiated into specific physical (e.g. visual) forms(Savidis and Stephanidis, 2004).

2.7.4 Scenario-based design methods

Such methods capture key usage-scenarios, annotated with design documentation which can be directly communicated to various levels of the design process, for further reviewing and enhancements. Such design documentation constitutes valuable reference material for early prototyping and engineering(Savidis and Stephanidis, 2004). Examples of use of scenario-based design are reported in (Royer, 1995). The employment of scenarios is mainly a complementary design technique, and has not been applied on its own to: (a) manage the conduct of the overall design process and (b) to facilitate the construction of the finalized design space as an appropriate organization of documented dialogue patterns(Savidis and Stephanidis, 2004).

2.7.5 Design rationale methods

Design rationale methods support argumentation about design alternatives and record the various design suggestions as well as their associated assessments. They are employed for the generation of design spaces, which capture and integrate design information from multiple sources, such as design discussions and diverse kinds of theoretical analyses(Savidis and Stephanidis, 2004). Questions, options and criteria (QOC) (McLean, 1995)design rationale is the most common method for constructing design spaces, capturing argumentation about multiple possible solutions to design problems.

2.7.6 Hybrid design methods

Hybrid design methods borrow elements mainly from task-based-, scenario-based-, and object-based-design techniques. Usually, one of these basic techniques becomes the centrepiece of the design process, while additional information can be extracted by performing parallel design steps through the deployment of any other method(Savidis and Stephanidis, 2004). For example, TADEUS (Stary, 1996) builds upon the task-model, providing also organizational and work-flow information. Hybrid models, until now, have emphasized the quick transition to an implementation phase, and have been applied to design projects in which work-flow information is a necessary ingredient (i.e. business process engineering/re-engineering)(Savidis and Stephanidis, 2004).

Based on the categories discussed above and following software project management methodology we are adapting visual/graphical design method. Getting advantage from prototyping by completing 3 iterations of development the final product would be delivered incrementally. Each increment is divided to five core stages of UI design from usability and HCI point of view, including requirement gathering, design, develop, test and evaluation(Yvonne Rogers, 2011). At first increment which is the largest in terms of time resource for developing the high fidelity prototype, requirement gathering is done by interviewing users then prototyping using story boarding approach for reflecting the user requirements gathered from interview. Sketching the paper prototype would help developer with receiving feedback from users and improve the user experience (Yvonne Rogers, 2011) by starting to develop an evolutionary high fidelity prototype in Python. By the end of each increment evaluation task would be completed which gives the developer opportunity of getting user feedback and comparing system functionally and user feedback against initial requirements; we can assume feedback from each increment as the requirement for the next increment. Improving the high fidelity prototype by getting benefit from user involvement in two iterations the prototype would evolve to the final product of the project.

In the requirement gathering stage usability goals (Yvonne Rogers, 2011) would be defined and design would be based on us HCI principles and usability goals taking to the account project characteristics affecting user interaction like type of users etc. First increment is aiming to make the user experience by achieving usability goals defined in requirements earlier. Two iterations would be focusing on improving user experience to the level well-designed GUI goes live.

2.8 Evaluation Techniques

Evaluation is integral to design process. Evaluation focuses both on the usability of the system and on the user experience when interacting with the system. There are many different evaluation methods. Which to use depends on goals of evaluation(Yvonne Rogers, 2011).

Conducting evaluation involves understanding not only why evaluation is important but also what aspects to evaluate, where evaluation should take place, and when to evaluate.

Controlled settings involving users approach enable evaluators to control what users do, when do it and for how long(Yvonne Rogers, 2011). This approach has been extensively and successfully used to evaluate software applications(Yvonne Rogers, 2011). Usability testing falls in this category. The primary goal is to determine whether an interface is usable by intended user population to carry out the tasks for which it was designed. A number of participants in one group or more would brought to the lab to carry out the given task. This approach is most efficient when interface is sketched for fair

population of users from diverse backgrounds, so data collected from experiment would be useful for improving the design. In my project the interface is designed for benefitting one user who is a scientist in field of computer science, therefore taking into account user population and type of user employing controlled usability testing seems inappropriate and other evaluation methods should be hired for evaluating this project.

Evaluations that take place without involving users are conducted in settings where evaluator has to imagine or model how an interface is likely to be used(Yvonne Rogers, 2011). Cognitive walk through falls in this category which involves simulating a user's problem solving process at each step in human computer dialog, and checking to see how users progress from step to step in these interactions(Nielsen, 1994). This approach is also known as an expert evaluation approach where usability specialists evaluate a product employing mentioned approach.

Based on the information provided of user population and taking into consideration that user for this project is an expert in usability evaluation, employing cognitive walk through for evaluating the project is reasonable and appropriate.

Evaluation would happen at two different occasions; first after developing the low fidelity prototype. For this purpose story boarding would be adapted for walking through the steps user take for completing the task and prototype would be evaluated. Based on the feedback from evaluation high fidelity prototype would be sketched. Second occasion is at the end of each cycle of developing high fidelity prototype when cognitive walk through would be employed for evaluating the prototype. Data collected from evaluation of each cycle of implementation would be used for improvement in next cycle.

Chapter 3

Development Overview

3.1 Architecture

This chapter explains the architecture of the interface in general in addition of its specific implementation in QT. Stepwise flow of GUI architecture starts with designing the interface either by hand coding or a visual tool. In this step all widgets are designed and then they are laid on the

interface, output of this step is a non-functional high fidelity prototype of the interface. In next step by making use of built in libraries defined actions are added to the interface which makes it functional.

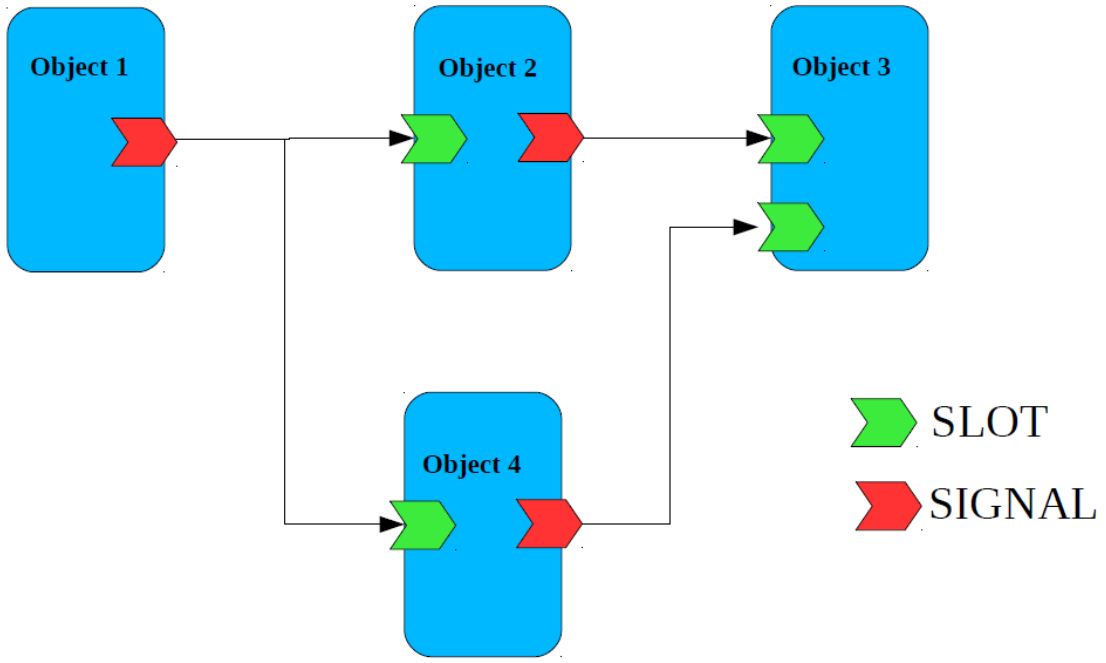

Diagram 1. QT Signal and Slot Flow

Every GUI library provides the details of events that take place, such as mouse clicks and key presses. For example, if we have a button with the text Click Me, and the user clicks it, all kinds of information becomes available. The GUI library can tell us the coordinates of the mouse click relative of the button, relative to the button's parent widget, and relative to the screen; and the precise time of the click and of the release; and so on. Similar information can be provided if the user "clicked" the button without using the mouse. The user may have pressed the Tab key enough times to move the focus to the button and then pressed Spacebar, or maybe they pressed Alt+C. Although the outcome is the same in all these cases, each different means of clicking the button produces different events and different information. The Qt library was the first to recognize that in almost every case, programmers don't need or even want all the low-level details: They don't care how the button was pressed; they just want to know that it was pressed so that they can respond appropriately. For this reason Qt, and therefore PyQt, provides two communication mechanisms: a low-level event-handling mechanism which is similar to those provided by all the other GUI libraries and a high-level mechanism which is called "signals and slots"(Summerfield, 2007).

Every PyQt's widgets support the signals and slots mechanism. In particular, they are capable of announcing state changes, such as when a checkbox becomes checked or unchecked, and other important occurrences, for example when a button is clicked. All of PyQt's widgets have a set of predefined signals. Whenever a signal is emitted, by default PyQt simply throws it away. To take notice of a signal we must connect it to a slot. In PyQt, slots can be any callable we like any function or method, and no special syntax is required when defining them(Rempt, 2008).

Most widgets also have predefined slots, so in some cases we can connect a predefined signal to a predefined slot and not have to do anything else to get the behaviour we want. PyQt is more versatile than C++/Qt in this regard, because we can connect not just to slots, but also to any callable, and from PyQt 4.2, it is possible to dynamically add "predefined" signals and slots to QObjects(Summerfield, 2007).

3.2 User Types and Establishing Requirements

The objective of completing IT project is satisfying user needs, therefore considering type of users and understanding their special requirements play a vital role in success or failure of the project.

The aim of requirement gathering process is 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 achieving their goals. Building on it the second aim is to produce a set of stable requirements to start designing. Also this process supposed to guarantee effective communication between developer and clients bringing benefits of user involvement to project.

The overall purpose of data gathering in the requirements activity is to collect sufficient relevant and appropriate data so that asset of requirements can be produced. For this purpose interview method was employed. Interview happened at David's office, I took semi structured interview approach for requirement gathering. Interview session took about 30 minutes in format of informal chat. By ending the session I could establish list of requirements as below:

Functional requirements: they are mentioned as minimum requirements in chapter 1.6.

Data requirements: data for the project would be provided to me by David. They are some series of slides in JPG format.

Environmental requirement: The interface is meant to be used in academic environment so it should have the same capabilities in different platforms including Linux, Mac and windows.

I case of usability goals the main goal is designing and implementing an efficient interface. At this stage considering the potential user types who are scientist familiar with computer systems and confident in their own field, it is concluded that user they do not need special training for using the interface and they can take more cognitive load than other type of users.

3.3 GUI Toolkit Choices

In last chapter we discussed the requirements establishment process, at this stage the most appropriate toolkit choice has to be chosen in accordance to the established requirements. The toolkit should be capable of helping developer to save time and energy and finally meet special characteristics this project has like time constraint etc. For choosing the appropriate toolkit some choices were reviewed and discussed with my supervisor and other lecturers to justify my choice.

3.3.1 GTK

GTK+, or the GIMP Toolkit, is a multi-platform toolkit for creating graphical user interfaces. Offering a complete set of widgets, GTK+ is suitable for projects ranging from small one-off tools to complete application suites. GTK+ is cross-platform and boasts an easy to use API, speeding up your development time.GTK+ was initially developed for and used by the GIMP, the GNU Image(GTK+Team, 2007-2012) Manipulation Program. It is called the "The GIMP ToolKit" so that the origins of the project are remembered. Today it is more commonly known as GTK+ for short and is used by a large number of applications including the GNU project's GNOME desktop(GTK+Team, 2007-2012).

GTK+ has been designed from the ground up to support a range of languages, not only C/C++. Using GTK+ from languages such as Perl and Python (especially in combination with the Glade GUI builder) provides an effective method of rapid application development(GTK+Team, 2007-2012).

Over time GTK+ has been built up to be based on four libraries, also developed by the GTK+ team:

- GLib, a low-level core library that forms the basis of GTK+. It provides data structure handling for C, portability wrappers and interfaces for such run-time functionality as an event loop, threads, dynamic loading and an object system(GTK+Team, 2007-2012).
- Pango, a library for layout and rendering of text with an emphasis on internationalization. It forms the core of text and font handling for GTK+(GTK+Team, 2007-2012).

- Cairo, a library for 2D graphics with support for multiple output devices (including the X Window System, Win32) while producing a consistent output on all media while taking advantage of display hardware acceleration when available(GTK+Team, 2007-2012).
- ATK, a library for a set of interfaces providing accessibility. By supporting the ATK interfaces, an application or toolkit can be used with tools such as screen readers, magnifiers, and alternative input devices(GTK+Team, 2007-2012).

GTK has been used on several of projects and can be used for this project as well; toolkit chosen for the project will be justified after reviewing other choice.

3.3.2 QT

Qt is a full development framework with tools designed to streamline the creation of stunning applications and amazing user interfaces for desktop, embedded and mobile platforms. Qt's cross-platform full framework and tools enables developers to reach various desktop, embedded and mobile operating systems and numerous devices with one code base. Qt brings freedom to the developer saving development time, adding efficiency and ultimately shortening time to project.

- Qt framework - intuitive APIs for C++ and CSS/JavaScript-like programming with Qt Quick for rapid UI creation(Oyj, 2013).

- Qt Creator IDE - powerful cross-platform integrated development environment, including UI designer tools and on-device debugging(Oyj, 2013).

With Qt, code can be reused efficiently to target multiple platforms with one code base. The modular C++ class library and developer tools easily enable developers to create applications for one platform and easily build and run to deploy on another platform. Also it is compatible with multiple platforms including Windows, Mac, Linux, Android, iOS(Oyj, 2013). Qt’s powerful full-framework capabilities allows for the creation of highly-performing native applications as well as for hybrid development where the developer can choose which tools provide the best user experience.

Qt Designer is a powerful cross-platform GUI layout and forms builder. It allows you to rapidly design and build widgets and dialogs using on-screen forms using the same widgets that will be used in your application. Forms created with Qt Designer are fully-functional, and they can be previewed so that you can ensure that they will look and feel exactly as you intended.

Features & Benefits

- Design user interfaces quickly with drag and drop functionality
- Customize widgets or choose from library of standard widgets
- Instantly preview forms in native look and feel
- Generate C++, Java or Python code from interface prototypes
- Use Qt Designer with Visual Studio IDE
- Build fully-functional user interfaces with Qt's signals and slots

Following project requirements toolkit should have flexibility in design and implementation also help the developer to save time and effort as time constraint is one of the major risks to the project.

QT introduces QT Designer which saves time by providing a visual toolkit for design called QT Designer which also helps with some bits of coding as it is capable to generate python code of sketched prototype. The other important issue is that QT package is already installed on all machines on school of computing clusters which refrains the developer from spending time on installing the package and deal with all issues could arrive during installation.

Based on all of the advantages mentioned QT was chosen as the toolkit for designing and developing the interface.

Chapter 4

First Cycle of Implementation

4.1 Design

Getting benefit from user feedback of evaluating low fidelity prototype, initial design was sketched by QT Designer Toolkit. The result was a non-functional high fidelity prototype in .UI for mat which is the built in format for out puts of QT Designer Toolkit. At this stage all the widgets were created and their properties were set up, also all relevant signals were made. Image 4 represents QT Designer environment and sketched design.

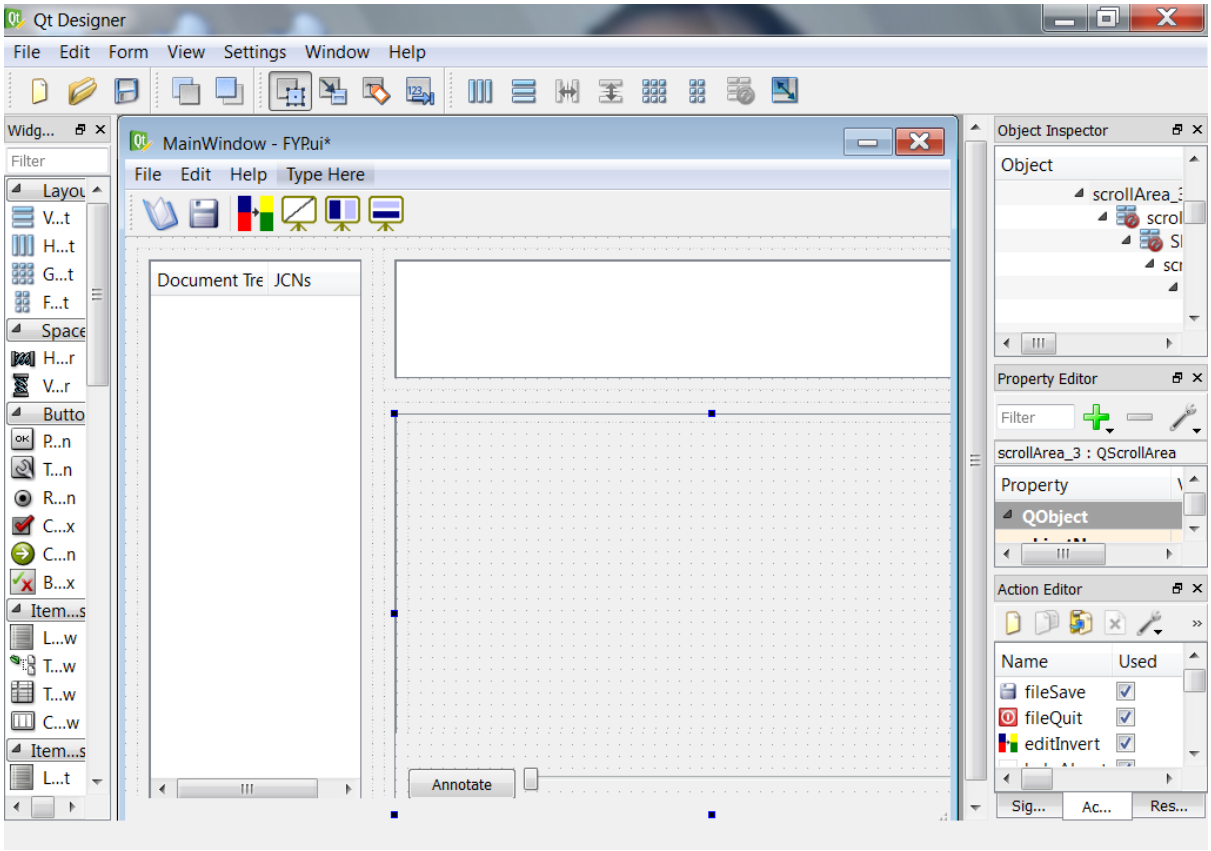

Image 4. Qt Designer Environment and Sketched Prototype

By competition of sketching the interface was exported to a .UI file. This files containing the user interface and can be used in different applications but it does not run as stand-alone file, so for getting benefit of it needs to be converted a Python file capable of running stand-alone from command line or

IDLE.

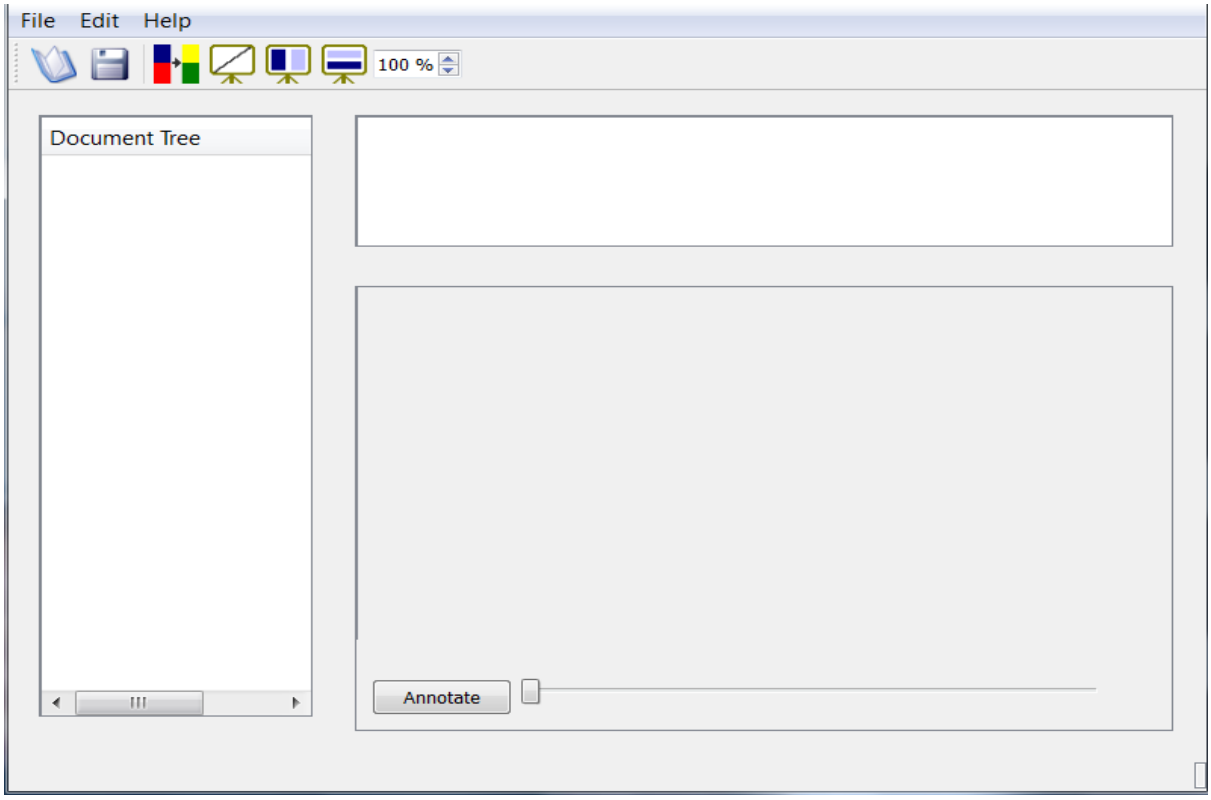

Image 5. The Output File of QT Designer

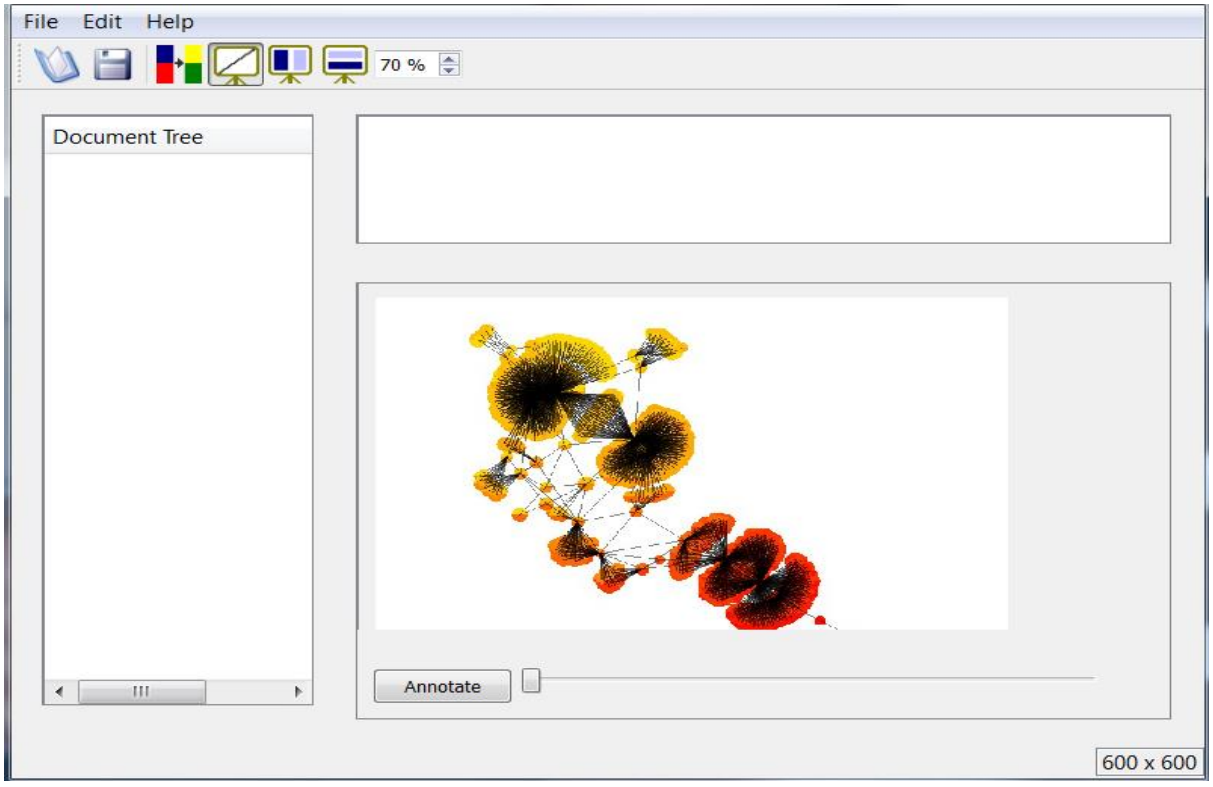

Image 6. Actual Interface Loading a JCN Slide

The sketched design includes the basic interface menu bar and relevant tool bars and it consists of three main box widgets, one in left hand side for showing the hierarchy discussed in chapter 2.4.1 and two widget on the right hand side one composite widget for showing for visual representation of the JCNs and underneath of it a widget for showing the actual slide.

We are now done by first step of the architecture, in next chapter we will discuss how the signal slot connection is implemented as well as any special feature in code for the interface.

4.2 Implementation

Implementation phase will be discussed in three steps including implementation of menu bar/ tool bar/ slide widget, tree widget and custom widget for visual representation of JCN Series.

4.2.1 Menu Bar, Tool Bar, Slide Widget

Qt Designer can be used to create user interfaces for dialogs, custom widgets, and main windows. The user interfaces are stored in .ui files, and include details of a form's widgets and layouts. In addition, Qt Designer can be used to associate labels with Buddies their "buddies", and to set the tab-order, that is, the order in which widgets get the keyboard focus when the user presses the Tab key.

This can also be done in code with QWidget.setTabOrder(), but it is rarely necessary for hand-coded forms, since the default is the order of widget creation, which is usually what we want(Summerfield, 2007). Qt Designer can also be used to make signal–slot connections, but only between built-in signals and slots.
Once a user interface has been designed and saved in a .ui file, it must be converted into code before it can be used. This is done using the pyuic4 command line program. For example:

```
C:\pyqt>pyuic4 -o FYP.py FYP.ui
```

One great benefit of using Qt Designer, in addition to the convenience of designing dialogs visually, is that if we change the design, we only have to regenerate the user interface module(using pyuic4 directly, or via mkpyqt.py or MakePyQt), and we do not need to change our code.

The only time that we must change our code is if we add, delete, or rename widgets that we refer to in our code. It means that using Qt Designer is much quicker and easier for experimenting with designs than editing hand-coded layouts, and helps maintain a separation between the visual design created using Qt Designer, and the behaviour implemented in code(Rempt, 2008).

When we create a user interface with Qt Designer, we create a subclass using multiple inheritances in which we put the code we need to give the user interface the behaviour we need. The first class we inherit is QDialog. If we were using the "Widget" template our first inherited class would be QWidget, and if we were using the "MainWindow" template our first inherited class would be QMainWindow. The second class we inherit is the class that represents the user interface we designed using Qt Designer.

It is common for applications to have a version string, and conventional to call it __version__; we will use it in the application's about box.

The initializer begins conventionally with the super() call. Next, we create a null QImage that we will use to hold the image the user loads or creates. A QImage is not a QObject subclass, so it does not need a parent; instead, we can leave its deletion to Python's normal garbage collection when the application terminates. We also create some instance variables. We use dirty as a Boolean flag to indicate whether the image has unsaved changes.

The filename is initially set to None, which we use to signify that either there is no image, or there is a newly created image that has never been saved.

For the application's status bar, we want the usual message area on the left, and a status indicator showing the width and height of the current image. We do this by creating a QLabel widget and adding it to the status bar. We also switch off the status bar's size grip since that seems inappropriate when we have an indicator label that shows the image's dimensions. The status bar itself is created for us the first time we call the QMainWindow's statusBar() method. If we call the status bar's showMessage() method with a string, the string will be displayed in the status bar, and will remain on display until either another showMessage()call supplants it or until clearMessage() is called. We have used the two-argument form, where the second argument is the number of milliseconds (5 000, i.e., 5 seconds), that the message should be shown for; after this time the status bar will clear itself.

```
self.sizeLabel = QLabel()
self.sizeLabel.setFrameStyle(QFrame.StyledPanel|QFrame.Sunken)
status = self.statusBar()
status.setSizeGripEnabled(False)
status.addPermanentWidget(self.sizeLabel)
status.showMessage("Ready", 5000)
```

Key Sequences

Many key sequences are standardized, some even across different windowing systems. For example, Windows, KDE, and GNOME all use Ctrl+N for “new” and Ctrl+S for “save”. Mac OS X is similar, with Command+N and Command+S for these actions. The QKeySequence class in PyQt 4.2 provides constants for the standardized key sequences, such as QKeySequence.New. This is especially useful when the standardized key sequences differ across windowing systems, or where more than one key sequence is associated with an action. For example, if we set a shortcut to QKeySequence.Paste, PyQt will trigger a “paste” action in response to Ctrl+V or Shift+Ins on Windows; Ctrl+V, Shift+Ins, or F18 on KDE and GNOME; and Command+V on Mac OS X.
For key sequences that are not standardized, we can provide the shortcut as a string; for example, setShortcut("Ctrl+Q").

Resource Files

The program was coded initially assumes that the application’s working directory is the directory where it is located. This is the normal case under but if the program is executed from the command line from a different directory none of the icons will appear. This is because we gave the icon’s path as images, that is, a path relative to the application’s working directory, so when invoked from elsewhere, the icons were looked for in the ./images directory which might not even exist.

Solution is to put all our icons (and help files, and any other small resources) into a single .py module and access them all from there. This not only solves the path problem (because Python knows how to look for a module to be imported), but also means that instead of having dozens of icons, help files, and similar, some of which could easily become lost, we have a single module containing them all. This resource could be implemented directly in QT Designer and exported with .UI file.

The resource module was generated by pyrcc4 using the following command line from .ui file:

C:\pyqt>pyrcc4 -o qrc_FYP.py FYP.qrc

We must run this command whenever we change the resources.qrc file.

Application Termination

The QKeySequence class does not have a standardized shortcut for application termination, so we have chosen one ourselves and specified it as a string. We could have just as easily used a different shortcut for example, Alt+X or Alt+F4.
The close() slot is inherited from QMainWindow. If the main window is closed by invoking the "file quit" action (which we have just connected to the close() slot), for example, by clicking File-Quit or by pressing Ctrl+Q, the base class's close() method will be called. But if the user clicks the application's close button, X, the close() method is not called.

The only way we can be sure we are intercepting attempts to close the window is to reimplement the close event handler. Whether the application is closed by the close() method or via the close button, the close event handler is always called. So, by reimplementing this event handler we can give the user the opportunity to save any unsaved changes, and we can save the application's settings.

In general, we can implement an application's behaviour purely through the high-level signals and slots mechanism, but in this one important case we must use the lower-level event-handling mechanism.

Restoring and Saving the Slide Window's State

Before we complete the initializer method we will restore the application's settings from the previous run (or use default settings if this is the very first time the application has been run).
Before we can look at application settings, we look at the creation of the application object and how the main window itself is created. The very last executable statement in the FYP. pyw file is function call main().Here is its code:

```
def main():
    app = QApplication(sys.argv)
    app.setOrganizationName("University of Leeds")
    app.setOrganizationDomain("leeds.ac.uk")
    app.setApplicationName("FYProject")
    app.setWindowIcon(QIcon(":/icon.png"))
    form = MainWindow()
    form.show()
    app.exec_()
```

Our primary use of them is for loading and saving application settings. If we create a QSettings object without passing any arguments, it will use the organization name or domain, and the application name that we have set here. So, by setting these once on the application object, we don't have to remember to pass them whenever we need a QSettings instance. These details are used by PyQt to save the application's settings in the most appropriate place for example, in the Windows registry, or in a directory under $HOME/.config on Linux, or in $HOME/Library/ Preferences on Mac OS X. The registry keys or file and directory names are derived from the names we give to the application object.

```
settings = QSettings()
self.recentFiles = settings.value("RecentFiles").toStringList()
self.restoreGeometry(
settings.value("MainWindow/Geometry").toByteArray())
self.restoreState(settings.value("MainWindow/State").toByteArray())
self.setWindowTitle("FYProject")
QTimer.singleShot(0, self.loadInitialFile)
```

This assumes that the geometry was saved when the applicationwas close-Event() terminated, as we will see when we look at the closeEvent().

The QMainWindow class provides a restoreState() method and a saveState() method; these methods restore from and save to a QByteArray. The data they save and restore are the dock window sizes and positions, and the toolbar positions.

In the last line of code we can see a single-shot timer is implemented. We want the main window to appear as quickly as possible so that the user knows that the launch was successful, and so that they can see any long-running processes, like loading large files, through the main window's user interface.

This works because a single-shot timer with a timeout of zero does not execute the slot it is given immediately. Instead, it puts the slot to be called in the event queue and then simply returns. At this point, the end of the main window's initializer is reached and the initialization is complete. The very next statement (in main()) is a show() call on the main window, and this does nothing except add a show event to the event queue. So, now the event queue has a timer event and a show event. A timer event with a timeout of zero is taken to mean "do this when the event queue has nothing else to do",

so when the next statement, exec_(), is reached and starts off the event loop, it always chooses to handle the show event first, so the form appears, and then, with no other events left, the single-shot timer's event is processed.

Handling File Actions

The Filemenu is probably the most widely implemented menu in interface applications, and it offers file handling facilities.

```
def FileOpen(self):
    if not self.okToContinue():
        return
    dir = (os.path.dirname(self.filename)
           if self.filename is not None else ".")
    formats = (["*.0}".format(unicode(format).lower())
            for format in QImageReader.supportedImageFormats()])
    fname = unicode(QFileDialog.getOpenFileName(self,
            "FYProject - Choose Image", dir,
            "Image files ({0})".format(" ".join(formats))))
    if fname:
        self.loadFile(fname)
```

If the user asks to open an existing slide, we first make sure that they have had the chance to save or discard any unsaved changes, or to cancel the action entirely.
If the user has decided to continue we want to pop up a file open dialog set to a sensible directory. If we already have an image filename, we use its path; otherwise, we use ".", the current directory.
In the case of the interface, we use the list of image type extensions for the image types that can be read by the version of PyQt that the application is using. At the very least, this is likely to include .bmp, .jpg (and List .jpeg, the same as .jpg), and .png.
If the user opens a file, the loadFile() method is called to actually perform the loading. We will look at this method in two parts.

```
def loadFile(self, fname=None):
if fname is None:
        action = self.sender()
```

```
if isinstance(action, QAction):
        fname = unicode(action.data().toString())
if not self.okToContinue():
        return
else:
        return
```

If the method is called from the fileOpen() method or from the loadInitial- File() method, it is passed the filename to open. But if it is called from a recently used file action, no filename is passed. We can use this difference to distinguish the two cases. If a recently used file action was invoked, we retrieve the sending object. This should be a QAction, but we check to be safe, and then extract the action's user data, in which we stored the recently used file's full name including its path. User data is held as a QVariant, so we must convert it to a suitable type. At this point, we check to see whether it is okay to continue.

We do not have to make this test in the "file open" case, because there, the check is made before the user is even asked for the name of a file to open. So now, if the method has not returned, we know that we have a filename in fname that we must try to load.

```
if fname:
         self.filename = None
         image = QImage(fname)
if image.isNull():
        message = "Failed to read %s" % fname
else:
        self.addRecentFile(fname)
        self.image = QImage()
        for action, check in self.resetableActions:
                action.setChecked(check)
        self.image = image
        self.filename = fname
        self.showImage()
        self.dirty = False
        self.sizeLabel.setText("%d x %d" % (
        image.width(), image.height()))
        message = "Loaded %s" % os.path.basename(fname)
        self.updateStatus(message)
```

We begin by making the current filename None and then we attempt to read the image into a local variable. PyQt does not use exception handling, so errors must always be discovered indirectly. In this case, a null image means that for add- some reason we failed to load the image. If File () the load was successful we add the new filename to the recently used files list, where it will appear only if another file is subsequently opened, or if this one is saved under another name. Next, we set the instance image variable to be a null image: This means that we are free to reset the checkable actions to our preferred defaults without any side effects. This works because when the checkable actions are changed, although the relevant methods will be called due to the signal–slot connections, the methods do nothing if the image is null.

After the preliminaries, we assign the local image to the image instance variable and the local filename to the filename instance variable. Next, we call showImage() to show the image at the current zoom factor, clear the dirty flag, and update the size label. Finally, we call updateStatus() to show the message in the status bar, and to update the log widget.

```
def fileSave(self):
if self.image.isNull():
        return
if self.filename is None:
        self.fileSaveAs()
else:
        if self.image.save(self.filename, None):
                self.updateStatus("Saved as %s" % self.filename)
        self.dirty = False
else:
        self.updateStatus("Failed to save %s" % self.filename)
```

The fileSave() method,and many others, act on the application’s data (a QImage instance), but make no sense if there is no image data. For this reason, many of the methods do nothing and return immediately if there is no image data for them to work on.

If there is image data, and the filename is None, the user must have invoked the “file new” action, and is now saving their image for the first time. For this case, we pass on the work to the fileSaveAs() method.

If we have a filename, we attempt to save the image using QImage.save(). This method returns a Boolean success/failure flag, in response to which we update the status accordingly.
When the “file save as” action is triggered we begin by retrieving the current filename. If the filename is None, we set it to be “.”, the current directory. We then use the QFileDialog.getSaveFileName() dialog to prompt the user to give us a filename to save under. If the current filename is not None, we use that as the default name—the file save dialog takes care of giving a warning yes/no dialog if the user chooses the name of a file that already exists. We use the same technique for setting the file filters string as we used for the “file open” action, but this time using the list of image formats that this version of PyQt can write.

If the user entered a filename that does not include a dot, that is, it has no extension; we set the extension to be .png. Next, we add the filename to the recently used files list (so that it will appear if a different file is subsequently opened, or if this one is saved under a new name), set the filename instance variable to the name, and pass the work of saving to the fileSave() method that we have just reviewed.

Handling Help Actions

When we created the main window’s actions, we provided each with help text, and set it as their status text and as their tooltip text. This means that when the user navigates the application’s menu system, the status text of the currently highlighted menu option will automatically appear in the status bar.

Similarly, if the user hovers the mouse over a toolbar button, the corresponding tooltip text will be displayed in a tooltip.

Whether or not we provide online help, it is always a good idea to provide an “about” box. This should at least show the application’s version and copyright notice, as Figure 6.10 illustrates.

```
def helpAbout(self):
QMessageBox.about(self, "About FYproject",
"""<b>FYProject</b> v %s
<p>Copyright &copy; 2013 Mohamamd Tari.
All rights reserved.
<p>This application can be used to perform
simple slide manipulations.
```

<p>Python %s - Qt %s - PyQt %s on %s""" % (
__version__, platform.python_version(),
QT_VERSION_STR, PYQT_VERSION_STR, platform.system()))

The QMessageBox.about() static convenience method pops up a modal OK-style message box with the given caption and text. The text can be HTML, as it is here. The message box will use the application's window icon if there is one.

We display the application's version, and version information about the Python, Qt, and PyQt libraries, as well as the platform the application is running on. The library version information is probably of no direct use to the user, but it may be very helpful to support staff who are being asked for help by the user.

Slide Widget

The Architecture of the slide widget is pretty simple; it is a QLabel which can accept most file types including images. The Show () method is assigning the loaded image file to QLabel widget, by doing this the image will be painted on the slide box widget.

Annotation Button

addText(self) method is called when the user clicks the Annotate button. It pops up a smart add/edit item dialog. If the user clicks OK, a new text file is added to the same directory that the slide is stored inheriting its name from name of the slide so if slide file is 1.jpg then the text file 1.txt will be added to the same folder. In case of user decides to edit the text the loader will search for the existing text, in case of success the text file is loaded and new text will be added to it, and in case of failure loader loads a blank text file for user.

```
def addText(self):
    dialog = TextItemDlg(position=self.position(),
                         scene=self.scene, parent=self)
    dialog.exec_()
```

4.2.2 Tree Widget

The left widget in main window is defined as a tree widget using QT Designer. At start-up of the interface defined method would walk through the files hierarchy discussed in chapter 2.1 using Python built in Function os.path.join() to load file details. In next step these file details will be added to the tree widget by iterating over a loop. The widget is designed on a live form so elements of the tree are sensitive to mouse click whenever user clicks on one of the elements consequently system loads related JCN series to appropriate widget on the screen.

```
QList<QTreeWidgetItem *> items;
 for name in (os.path.join(os.path.dirname())
     items.append(new QTreeWidgetItem((QTreeWidget*)0,
QStringList(QString("item: %1").arg(i))));
```

4.2.3 Custom widget for visual representation of JCN Series

One of PyQt's greatest and longest-standing strengths is the ease with which it is possible to create custom widgets. The custom widgets we create with PyQt are made the same way as the standard built-in widgets, so they integrate seamlessly and have no arbitrary restrictions on their appearance or behaviour.

Creating custom widgets in PyQt is not a matter of "one size fits all". Rather, we can choose from a number of approaches that give us increasing levels of control over our widgets' behaviour and appearance.

In design stage this widget is defined as a QtList widget which is capable of accepting several list items. For developing the actual custom widget for saving time the code was adapted from an already developed widget which had main functionalities we need for our custom widget.

The original widget had to show show a 3 × 3 grid, with each square either blank (showing just the background color) or with a red or yellow ellipse. The state of any grid square should change from blank to red to yellow and back to blank in an endless cycle.

Changes of state had to occur when the user clicks a square or presses the spacebar on a square. The keyboard focus was shown by drawing the square with a thick blue pen instead of the normal thin black pen used for the other squares. The user had to be able to change the focused square by clicking a square or by using the up, down, left, and right arrow keys to move the focus. The paint event is quite short, but slightly subtle; code was able to save and restore the painter's state, using QPainter.save() and QPainter.restore(), sothat pen and brush colours intended for one square don't propagate to others.

```
BLANK, BLUE, YELLOW = range(3)
class CountersWidget(QWidget):
    def __init__(self, parent=None):
        super(CountersWidget, self).__init__(parent)
        self.setSizePolicy(QSizePolicy(QSizePolicy.Expanding,
                                       QSizePolicy.Expanding))
        self.grid = [[BLANK] * 3 for i in range(3)]
        self.selected = [0, 0]
        self.setMinimumSize(self.minimumSizeHint())
    def sizeHint(self):
        return QSize(200, 200)
    def minimumSizeHint(self):
        return QSize(100, 100)
    def mousePressEvent(self, event):
        xOffset = self.width() / 3
        yOffset = self.height() / 3
        if event.x() < xOffset:
            x = 0
        elif event.x() < 2 * xOffset:
            x = 1
        else:
            x = 2
        if event.y() < yOffset:
            y = 0
        elif event.y() < 2 * yOffset:
            y = 1
        else:
            y = 2
        cell = self.grid[x][y]
        if cell == BLANK:
            cell = RED
```

```
        elif cell == RED:
            cell = YELLOW
        else:
            cell = BLANK
        self.grid[x][y] = cell
        self.selected = [x, y]
        self.update()
    def keyPressEvent(self, event):
        if event.key() == Qt.Key_Left:
            self.selected[0] = (2 if self.selected[0] == 0
                                else self.selected[0] - 1)
        elif event.key() == Qt.Key_Right:
            self.selected[0] = (0 if self.selected[0] == 2
                                else self.selected[0] + 1)
        elif event.key() == Qt.Key_Up:
            self.selected[1] = (2 if self.selected[1] == 0
                                else self.selected[1] - 1)
        elif event.key() == Qt.Key_Down:
            self.selected[1] = (0 if self.selected[1] == 2
                                else self.selected[1] + 1)
        elif event.key() == Qt.Key_Space:
            x, y = self.selected
            cell = self.grid[x][y]
            if cell == BLANK:
                cell = RED
            elif cell == RED:
                cell = YELLOW
            else:
                cell = BLANK
            self.grid[x][y] = cell
        self.update()
```

```
def paintEvent(self, event=None):
    painter = QPainter(self)
    painter.setRenderHint(QPainter.Antialiasing, True)
    xOffset = self.width() / 3
    yOffset = self.height() / 3
    for x in range(3):
        for y in range(3):
            cell = self.grid[x][y]
            rect = (QRectF(x * xOffset, y * yOffset,
                    xOffset, yOffset).adjusted(0.5, 0.5, -0.5, -0.5))
            color = None
            if cell == RED:
                color = Qt.red
            elif cell == YELLOW:
                color = Qt.yellow
            if color is not None:
                painter.save()
                painter.setPen(Qt.black)
                painter.setBrush(color)
                painter.drawEllipse(rect.adjusted(2, 2, -2, -2))
                painter.restore()
            if [x, y] == self.selected:
                painter.setPen(QPen(Qt.blue, 3))
            else:
                painter.setPen(Qt.black)
            painter.drawRect(rect)
```

The code was adapted to receive the number of circles which are number of JCN from the tree widget after user click on one branch of the tree and paint the same series of circles with ability to change their colours result of user clicks. So the new widget is representing number of JCNs in a series of connected circles, by clicking on each circle the relevant JCN slide would be shown on the slide

widget. Also in case user find interesting point about the slide, say finding the scission point, user is able to change the circle colour representing user findings.

4.3 Evaluation

It was discussed before that evaluation happens at the end of each cycle of iteration and user feedback would be used for improving the prototype through next iteration of development. First evaluation happened after story boarding when the low fidelity prototype was evaluated and user feedback was provided for sketching the high fidelity prototype in coding.

At this stage the high fidelity prototype should be evaluated and tested by user so the feedback would be used for next iteration of improvement. It was discussed that the special type of user for this project make usability testing an inappropriate approach for testing the user interface, so walk through approach was employed for all three phase of evaluation happens in the project life cycle.

For purpose of evaluation, a meeting was arranged with the user at his office. User walked through the interface testing the functionalities it is advertised to have. During the session developer was observing user also making note of communication in between for documentation.

By completing the evaluation user feedback was recorded and documented. User pointed out to two main areas that needs to be improved: utilising a zoom facility that can take integer numbers and zoom in/out the slide accordingly apart from zoom facility provided as slider at the bottom of image and utilising a colour bar for the custom widget which represents the colours custom widget can take and their meanings.

User feedback from this session is used for second iteration of development as the model of improvement. It is assumed after next cycle the interface is meeting the minimum requirements of the project and can be delivered as the final product of the project.

Chapter 5

Second cycle of implementation

5.1 Design

Getting benefit from QT Designer tool which makes separation between the design and implementation, the interface was redesigned efficiently in QT Designer, following user feedback a Combo Box was added to the toolbar which can take integer numbers as zooming percentage and has the capability of adjusting the loaded slide accordingly. Also the properties were added to the box using QT Designer facilities. After updating the .ui file it was converted to python code, so the signal slot can be added to the new widget of interface.

5.2 Implementation

It is convenient for users to be able to zoom in and out to see an image in more or less detail. We have provided a spinbox in the toolbar to allow mouse users to change the zoom factor (and which we will come to shortly), but we must also support keyboard users, so for them we create an “edit zoom” action which will be added to the Edit menu. When triggered, the method connected to this action will pop up a dialog box where the user can enter a zoom percentage.

There are standardized key sequences for zoom in and for zoom out, but there is not one for zooming generally, so we have chosen to use Alt+Z in this case. (We did not use Ctrl+Z, since that is the standardized key sequence for undo on most platforms.)

We want to provide the user with a quick means of changing the zoom factor, so we provide a spinbox in the edit toolbar to make this possible. Earlier, we put a separate “edit zoom” action in the Edit menu, to cater to keyboard users.

```
self.zoomSpinBox = QSpinBox()
self.zoomSpinBox.setRange(1, 400)
self.zoomSpinBox.setSuffix(" %")
self.zoomSpinBox.setValue(100)
self.zoomSpinBox.setToolTip("Zoom the image")
self.zoomSpinBox.setStatusTip(self.zoomSpinBox.toolTip())
self.zoomSpinBox.setFocusPolicy(Qt.NoFocus)
self.connect(self.zoomSpinBox,
SIGNAL("valueChanged(int)"), self.showImage)
editToolbar.addWidget(self.zoomSpinBox)
```

5.3 Evaluation

The same evaluation approach was employed for evaluating the second cycle of development. Meeting was arranged with user and cognitive walk through evaluation was completed with user and user feedback was recorded and documented. As I was expected the evolutionary prototype is good enough to be introduced as the final product of the project.

Chapter 6

Conclusion

6.1 Meeting Minimum Requirements

A list of the minimum requirements for this project and how they have been accomplished is laid out below:

- To design and develop a GUI for representing already generated visualization slides based on established requirements, capable of providing the set of functionalities necessary for efficient interaction between user and the interface including:

    1. Providing functionality of zooming in and out to different levels of details in represented output of underlying code.

    This requirement is achieved by utilising two different zoom mechanism(zoom slider and zoom box) which aid user of zooming either by the slider or choosing an integer as zoom base figure.

    2. Functionality of flagging out points of interest, adding text data to them for further investigation.

This requirement has achieved by implementing custom widget which has capability of changing its colour by user click, so each colour means differently and one of the colours, in this case blue is used for flagging out the points of interest.

3. Functionality of providing parallel visual comparison between outputs of different experiment runs.

Both two widgets that provide this facility which are custom widget and the slider are design as QtList widgets, which means numbers of items (JCN series and image slider) could be added to these two widget separately. In achieving this requirement we need to consider the number of items is restricted by monitor dimensions.

• Design and develop the GUI in an extensible manner so the interface functionalities could be extended by adding new modules to the source code.

This requirement is already achieved as the new widgets could be added to the interface using QT Design or hand coding the widget. Also the modular signal slot architecture of the code enables any further enhancement in future. By manipulating slots current functionalities would be enhanced or new functionalities would be added.

6.2 Future Enhancements

Many possible enhancements are laid out in section 1.6. Below is presented some of the most beneficial enhancements to the system:

- The tree widget could be enhanced in the way that instead of clicking on separate JCN sets for loading them to custom widget by clicking on a category of JCNs all of them would be added to the system.
- Interface interacting with underlying code to run the code, generating new slide in case the slid is not generated for a given point of data.
- Implementing capability of adding other sort of data properties for example audio to the points of interest.
- Implementing Capability of search for particular pattern in given data set using search filters.

Bibliography

B. DUFFY, H. C., AND T. M¨OLLER. 2012. Integrating Histograms and Isosurface Statistics. ***IEEE Transactions on Visualization and Computer Graphics*****.**

BENDER, M., HEENEN, P.-H. & REINHARD, P.-G. 2003. Self-consistent mean-field models for nuclear structure. ***Reviews of Modern Physics,*** **75, 121-180.**

BOB HUGHES, M. C. 2006. ***Software Project Management,*** **McGraw-Hill Education.**

BOHR, N. & WHEELER, J. A. 1939. The Mechanism of Nuclear Fission. ***Physical Review,*** **56, 426-450.**

BONNEAU, L., QUENTIN, P. & MIKHAILOV, I., N. 2007. A definition of scission points and consequences on some fission distributions. 343-346.

DAVID, D. 2012. Visualizing Nuclear Scission through a Multifield Extension of Topological Analysis. ***IEEE Transactions on Visualization and Computer Graphics,*** **18, 2033-2040.**

DUKE, H. C. A. D. 2011. Joint contour nets: Topological analysis of mutlivariate data. ***VisWeek Poster Compendium*****.**

DUKE, H. C. A. D. 2012. Joint contour nets: Topological analysis of mutlivariate data. ***IEEE Transactions on Visualization and Computer Graphics*****.**

FEKETE, J.-D. 2005. ***The InfoVis Toolkit*** **[Online]. Available: http://ivtk.sourceforge.net//#intro [Accessed 31/05/ 2013].**

GTK+TEAM, T. 2007-2012. ***The GTK + Project*** **[Online]. Available: http://www.gtk.org/.**

HARTSON, H. R., SIOCHI, A.C., HIX, D. 1990. The UAN: a user-oriented representation for direct manipulation interface design. ***ACM Transactions on Information Systems,*** **8, 181–203.**

HCIL. 2002. ***Visual Exploration of Time-Series Data*** **[Online]. Available: http://www.cs.umd.edu/hcil/timesearcher/ [Accessed 30/05/ 2013].**

HOARE, C. A. R. 1978. Communicating sequential processes. ***Communication of the ACM,*** **21, 666–677.**

IVANYUK, F. 2013. On the Scission Point Configuration of Fisioning Nuclei.

JANACEK, G. J. G. J. 1993. ***Time series : forecasting, simulation, applications***

JOHNSON, P., JOHNSON, H., WADDINGTON, P., SHOULS, A. 1988. Task-related knowledge structures: analysis, modeling, and applications. ***Cambridge University Press*****, pp. 35–62.**

KENNETH E. KENDALL, J. E. K. 2005. ***System Analysis and Design*****, Natalie E. Anderson.**

MACEACHREN, A. M. 1995. ***How maps work : representation, visualization, and design*** **New York ; London, Guilford Press.**

MACIEJEWSKI, R. 2011. Data Representations, Transformations, and Statistics for Visual Reasoning. ***Synthesis Lectures on Visualization,*** **2, 1-85.**

MATCOVIĆ, K., H. HAUSER, R. SAINITZER, AND M.E. GRÖLLER. Process Visualization with Level of Detail. IEEE Symposium on Information Visualization(InfoVis '02), 2002. 67-70.

MCLEAN, A., MCKERLIE, D. 1995. Design Space Analysis and Use-Representations. ***Technical Report EPC,*** **102.**

NIELSEN, J. A. M., R.L. 1994. ***Usability Inspection Methods,*** **New York, John Wiley & Sons inc.**

OYJ, D. 2013 ***QT*** **[Online]. [Accessed 30/05/ 2013].**

PASCUCCI, V., WEBER, G. H., BREMER, P. T., DAY, M. & BELL, J. 2010. Analyzing and Tracking Burning Structures in Lean Premixed Hydrogen Flames. ***Visualization and Computer Graphics, IEEE Transactions on,*** **16, 248-260.**

PAYNE, S. Task-action grammars. Proceedings of IFIP Conference on Human–Computer Interaction: INTERACT'84, 1984 North-Holland/Elsevier Science, London/Amsterdam. pp. 139–144.

POST F.H., H. H., F.H., B. VROLIJK, R. LARAMEE, AND H. & DOLEISCH 2002. Feature Extraction and Visualization of Flow Fields. ***Eurographics 2002 State of the Art Reports,*** **69-100.**

REMPT, B. 2008. Gui Programming With Python: Using the Qt Toolkit.

ROYER, T. Using scenario-based designs to review user interface changes and enhancements. Proceedings of the ACM DIS'95 Symposium on Designing Interactive Systems,, 1995 MI, USA. pp. 277–246.

SAVIDIS, A. & STEPHANIDIS, C. 2004. Unified user interface design: Designing universally accessible interactions. ***Interacting with Computers,*** **16, 243-270.**

SCHUMANN, W. M. A. H. VISUALIZATION METHODS FOR TIME-DEPENDENT DATA - - AN OVERVIEW. ***In:*** **S. CHICK, P. J. S., D. FERRIN, AND D. J. MORRICE, ed. 2003 Winter Simulation Conference, 2003.**

SCHUNCK, N. 2013. DENSITY FUNCTIONAL THEORY APPROACH TO NUCLEAR FISSION. ***ACTA PHYSICA POLONICA B,*** **Vol. 44 263-270.**

SOMMERVILLE, I. 2007. *Software Engineering,* England, Pearson Education Limited.

STARY, C. 1996. Integrating workflow representations into User Interface design representations, Software Concepts and Tools. vol. 17.

SUMMERFIELD, M. 2007. *Rapid GUI Programming with Python and Qt,* Michigan., Edwards Brothers.

YVONNE ROGERS, J. P., HELEN SHARP 2011. *Interaction design : beyond human-computer interaction* Hoboken, N.J, Wiley

Appendix A

Personal Reflection

The final year project is the most important piece of work that a student will undertake during their degree course, in terms of opportunity for personal development as well as in terms of scope. The process of choosing and extensively researching a subject area, establishing a problem and set of project goals and developing a solution to the problem is one which requires the application of several skills which are valuable for somebody wishing to work in either the academic or commercial world. The process of critically evaluating ones work and the published work of other researchers is one with which a student may not have had a great deal of experience, but this critical mindset is extremely important to scientific endeavour. Aside from the skills which are developed, a student undertaking a project will gain a considerable amount of knowledge relating to their chosen specialist subject area. The project may even inform a student's career choices as they discover a field or specialism which interests them (or indeed discover that they do not enjoy working in a particular area).

The particular value of this project to me was in developing a useful piece of code which represents a full IT project life cycle. Also my understanding of how the project management methods which were discussed as part of my course could be applied to non-trivial modelling tasks

The first challenge which faces the student is making a choice of project. Ultimately, the choice of project is a personal one and is driven by either an intellectual interest in the type of work the project involves or a desire to gain experience in a subject area they are interested in applying themselves to professionally. A good choice of project can help considerably in improving the experience of the student and ultimately the quality of the report:

If a project involving the production of a piece of code is chosen, there are usually innumerable areas of application for any particular computational technique

There are several challenges faced by a final year student during their project. Perhaps the most pressing of these challenges is the need to be disciplined and organise time effectively. The student must also be able to adapt to any unforeseen events which impact on the project's schedule or shifts in focus which occur as a result of the continuous research and investigation they undertake. There are some pieces of advice which I believe can be helpful in managing time effectively and maintaining flexibility:

• Begin gathering background research material as early as possible, being sure to make a note of any interesting pieces of work which are referenced frequently in the literature and obtain a copy of them sooner, rather than later, this allows the collection of background reading material to be built up and worked through at the student's own pace, avoiding the problem of trying to obtain material during the later stages of the project which must be ordered-in by the library service.

• If the project involves the production of software system it can be extremely challenging to estimate the amount of time its design and implementation will take therefore, it can be valuable to establish a outline of the design's broad structure as soon as enough background reading has been undertaken to make a reasonably informed decision.

• Do not wait until implementation and evaluation are complete before beginning the report. It is necessary to produce a background reading chapter for the midproject report, but aside from this I only made brief notes during the design and implementation phases. A better way to work might be to create draft versions of the various parts of the report as the relevant work is being undertaken.

Appendix B

External contributions

In the implementation phase of the interface, a piece of code was adapted from an online resource for saving time. The full piece of code is provided below:

```
from __future__ import division
from __future__ import print_function
from __future__ import unicode_literals
from future_builtins import *

from PyQt4.QtCore import (QRectF, QSize, Qt, SIGNAL)
from PyQt4.QtGui import (QApplication, QPainter, QPen, QSizePolicy,
        QWidget)

BLANK, RED, YELLOW = range(3)
```

```
class CountersWidget(QWidget):

    def __init__(self, parent=None):
        super(CountersWidget, self).__init__(parent)
        self.setSizePolicy(QSizePolicy(QSizePolicy.Expanding,
                                       QSizePolicy.Expanding))
        self.grid = [[BLANK] * 3 for i in range(3)]
        self.selected = [0, 0]
        self.setMinimumSize(self.minimumSizeHint())


    def sizeHint(self):
        return QSize(200, 200)


    def minimumSizeHint(self):
        return QSize(100, 100)


    def mousePressEvent(self, event):
        xOffset = self.width() / 3
        yOffset = self.height() / 3
        if event.x() < xOffset:
            x = 0
        elif event.x() < 2 * xOffset:
            x = 1
        else:
            x = 2
```

```
    if event.y() < yOffset:
        y = 0
    elif event.y() < 2 * yOffset:
        y = 1
    else:
        y = 2
    cell = self.grid[x][y]
    if cell == BLANK:
        cell = RED
    elif cell == RED:
        cell = YELLOW
    else:
        cell = BLANK
    self.grid[x][y] = cell
    self.selected = [x, y]
    self.update()


def keyPressEvent(self, event):
    if event.key() == Qt.Key_Left:
        self.selected[0] = (2 if self.selected[0] == 0
                            else self.selected[0] - 1)
    elif event.key() == Qt.Key_Right:
        self.selected[0] = (0 if self.selected[0] == 2
                            else self.selected[0] + 1)
    elif event.key() == Qt.Key_Up:
        self.selected[1] = (2 if self.selected[1] == 0
                            else self.selected[1] - 1)
    elif event.key() == Qt.Key_Down:
```

```
            self.selected[1] = (0 if self.selected[1] == 2
                                else self.selected[1] + 1)
        elif event.key() == Qt.Key_Space:
            x, y = self.selected
            cell = self.grid[x][y]
            if cell == BLANK:
                cell = RED
            elif cell == RED:
                cell = YELLOW
            else:
                cell = BLANK
            self.grid[x][y] = cell
        self.update()


    def paintEvent(self, event=None):
        painter = QPainter(self)
        painter.setRenderHint(QPainter.Antialiasing, True)
        xOffset = self.width() / 3
        yOffset = self.height() / 3
        for x in range(3):
            for y in range(3):
                cell = self.grid[x][y]
                rect = (QRectF(x * xOffset, y * yOffset,
                               xOffset, yOffset).adjusted(0.5, 0.5, -0.5, -0.5))
                color = None
                if cell == RED:
                    color = Qt.red
                elif cell == YELLOW:
```

```
                color = Qt.yellow
            if color is not None:
                painter.save()
                painter.setPen(Qt.black)
                painter.setBrush(color)
                painter.drawEllipse(rect.adjusted(2, 2, -2, -2))
                painter.restore()
            if [x, y] == self.selected:
                painter.setPen(QPen(Qt.blue, 3))
            else:
                painter.setPen(Qt.black)
            painter.drawRect(rect)


if __name__ == "__main__":
    import sys

    app = QApplication(sys.argv)
    form = CountersWidget()
    form.setWindowTitle("Counters")
    form.show()
    app.exec_()
```