EE4RPP: Project Proposal

The purpose of the project is to make an intuitive yet powerful bioinformatics search engine which provides online access to a large dataset of protein isoelectric points which has been compiled by Aston University researchers and students over the course of several years.

Background

Bioinformatics is a multidisciplinary field which uses computational methods to aid in biological research by creating systems for storing, organising and analysing complex biological data. Within this field there are many online databases categorising biological information at the molecular level, and one such purpose of these is for storing the functional and physical properties of proteins. Currently, no such database exists for one of the most widely-used, important, and useful properties of proteins: the isoelectric point (pI). An isoelectric point is the acidity (pH) at which a molecule carries no net charge; below the isoelectric point, proteins have a net positive charge, above it a net negative charge. Additionally, proteins are at their lowest solubility at their isoelectric point, and this makes the isoelectric point a vitally important property when both characterising and purifying proteins.

The dataset which has been compiled is a collection of entries stored as a non-relational table, and for each entry it records the name of the protein, its identity, origin, experimental conditions, its isoelectric point, and other pertinent data. There are also links to a heterogeneous collection of databases containing associated data, such as amino acid sequence, function, etc. A web-accessible database that warehouses this data and offers a robust and adaptable GUI for searching, viewing and downloading results would greatly increase the accessibility of the dataset.

Objectives

1. To build a free (as in freedom) web application for viewing protein isoelectric points.
2. To produce a bioinformatics tool with real world value for future research.
3. The application should provide intuitive but powerful searching facilities.
4. The application should provide a convenient means for a certified user to edit and upload additional data.
5. The application should present information in a usable and efficient form.
6. Users should be allowed to download generated results for offline use.
7. Adequate security precautions should be taken to minimise the risk of data being sabotaged or stolen.
8. The implementation should use a clean model view controller architecture.
9. Comprehensive test coverage of the API and common use cases should be automated.
10. The application should be fully scalable for much larger datasets.

Deliverables

Two primary deliverables can be derived from the initial project requirements:

1. An updatable relational database warehousing the provided dataset.
2. A web-accessible GUI with searching and downloading functionality.

Additionally, two further deliverables can be derived from the same requirements document as secondary features:

3. A web-accessible GUI to support dataset editing and uploading features.
4. Support for NCBI BLAST protein sequence matching (1).

Risk Assessment

Table 1 lists some of the potential project risks that were identified during the initial inception phase which could influence the success of the project and its ability to meet the objectives and deliverables.

Risk	**Category**	**Probability**	**Impact**
Design is not intuitive	Design	2	3
Project involves use of new technical skills	Development	5	5
High Level of technical complexity	Development	5	3
Project milestones not clearly defined	Planning	1	1
System requirements not adequately identified	Requirements	2	5
Change in project requirements during development	Requirements	1	5
Changes in dataset format during development	Resources	2	5
Unable to obtain required resources	Resources	1	1
Users not committed to the project	Users	2	4
Lack of cooperation from users	Users	1	4
Users with negative attitudes toward the project	Users	1	2

Table 1 A list of potential project risks and their severity

Mitigation Strategies

For each of the risks discovered in the assessment, mitigation strategies have been defined which provide techniques to avoid or minimise the threat of each risk.

Design is not intuitive – the key to mitigation of this risk is in frequent and effective user testing and an understanding of typical and common use-cases for the product.

Project involves use of new technical skills – in order to prevent this risk from having a serious impact on the project, it will be necessary to begin studying and reading about the technologies that will be used at a very early stage in the project, long before the start of the implementation.

High Level of technical complexity – avoiding this risk will involve ensuring that the scope of the project remains technically feasible, and that the software architecture is abstracted into small enough units that it is easier to focus on each one separately, as well as keeping small iterative development cycles and adequate test coverage to prevent regressions when implementing new functionality.

Project milestones not clearly defined – a thoroughly described and well thought out project plan will help to prevent scheduling issues and delays in development that would arise from this risk.

System requirements not adequately identified – a comprehensive specification of the finished product before implementation begins will help to mitigate this risk.

Change in project requirements during development – an agile approach towards accommodating for changes in the requirements should be used so as to keep the time between user feedback sessions and input from stakeholders low.

Changes in dataset format during development – it is not possible to entirely avoid this risk due its nature and the dependence on third parties, but steps can be taken to prevent any delays that this would cause, chiefly, a well abstracted data parsing component which can be switched and modified if necessary to accommodate for a new dataset format.

Unable to obtain required resources – since the project does not require many resources, it is important to acquire these as early on in the development process as possible, and alternative resources should be planned for, such as local test servers.

Users not committed to the project, lack of cooperation from users, and users with negative attitudes toward the project – the useful of the finished project will depend largely on ensuring that the needs of the users are considered the primary goals of the design. Violating this principle may cause disillusionment from the people who are volunteering their time to assist in the project.

Required Resources

The deliverable product for this project is a web service that will be publicly available. To this end, there are three required resources:

1. A server which can be host the web-service and process and respond to requests from clients.
2. A public IP address which this server can be assigned to for external access.
3. A domain name which will resolve to this IP address.

Additional stipulations for the requirements are that the server should use a GNU/Linux operating system so as to support most common webserver stacks, and that it should be a dedicated physical machine with root access so as to allow for more involved configuration and testing. Assuming that the University can supply the webserver and access to one of its IP addresses, the budget for the project need only cover the cost of domain registration, which is very cheap (less than £20 per year).

Development Process

The software development process used for this project is based the Open Unified Process (OpenUP), a part of the Eclipse Process Framework (2). The reasoning behind this choice is that, as Rational Unified Process derivative, OpenUP offers an open source process framework which is targeted at agile development in small teams and provides a number of development phases and activities which can be used when designing the project plan (Figure 1).

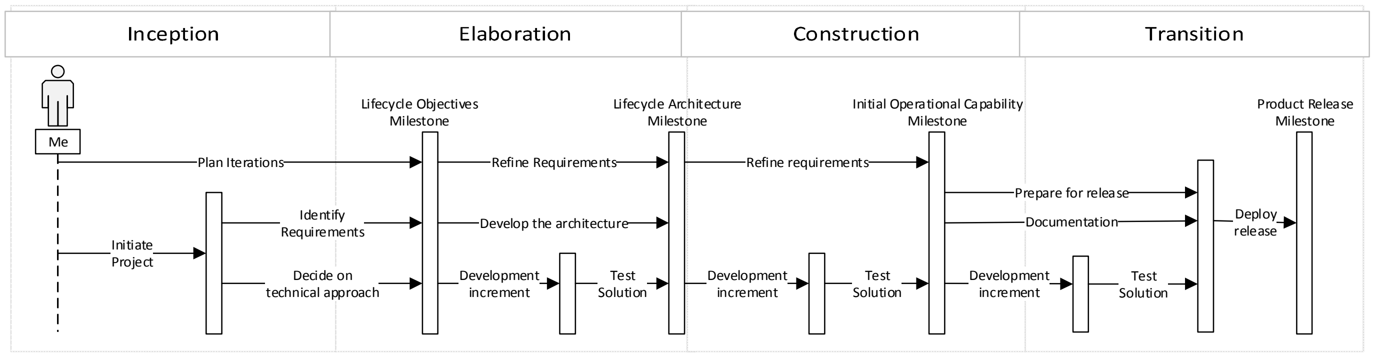


Figure 1 A sequence diagram showing a single full iteration of the OpenUP process

Work Breakdown Structure

The crux of OpenUP is in breaking down a large development project into four key phases to iterate on: the Inception Phase, Elaboration Phase, Construction Phase and Transition Phase (3).

Inception Phase

The inception phase represents the initial work which defines the scope and objectives of the project. Key tasks include generating a list of the core project's requirements, key features and main constraints, and developing an understanding of the general project use-cases and business case. The work in this phase culminates with a stakeholder concurrence on the project scope, cost and schedule, and a deep requirements understanding which covers the depth and breadth of the technical work to be undertaken. For this project, the inception phase should include meeting all project stakeholders and research into existing protein databases, their use-cases, and a deeper understanding of the scientific value of the dataset.

Elaboration Phase

The elaboration phase builds upon the work done in the construction phase by requiring deeper technical research into required technologies, and initial prototyping of early ideas. By the end of the elaboration phase, the product vision should be agreed upon and stable, and a full plan of the technical architecture should have been reached. Further iterations of the elaboration phase may be used after construction has begun to refine the architecture plan, or as a response to a change in the technologies used. The purpose of the phase is to turn the initial product vision into a realisable goal with quantifiable and achievable goals and objectives. For this project, the elaboration phase will involve investigation into some of the available technologies (PHP, MySQL, Node.js, MongoDB, etc.), and technical prototypes of the database backend.

Construction Phase

The construction phase covers the development of the main software architecture and associated documentation, and should result in "Initial Operational Capability" (3). Success criteria for this development phase includes whether the product is mature enough to be deployed to users, and so for this project will require meeting with stakeholders to ensure that the implementation of the plan is acceptable.

Transition Phase

The transition phase includes beta testing of the new system against user expectations, and includes a review of the completed product against the requirements and objectives established in the initial project plan to measure success. The phase culminates in a product roll-out and the associated distribution, marketing and training of users that is required. For this project, it will involve deploying the finished project to a public server and conducting extensive user testing.

Version Control

A revision control and source code management (SCM) system will be used during all development to keep an auditable and transparent log of progress, and Git will be used for this. There are numerous advantages that Git has over other SCMs, chiefly that it is entirely open source and GPL licensed (4), it has a very lightweight branching model and good support for rebasing and merging, and there are numerous sources which offer free hosting of open source licensed projects that are tracked by Git. A public repository of the source code and all relevant documentation for this project is available on GitHub (5).

Issue tracker

One of the additional benefits of the GitHub online repository hosting service is that it supplies a number of useful tools, namely an issue tracker and milestones list. This allows issues to be created online and categorised appropriately (e.g. bugs, tasks, regressions, documentation, etc.), and then referenced from the repository commits. Milestones can be created and individual issues assigned to them, allowing for quick and visible progress checking of development towards a specific goal.

Test driven development

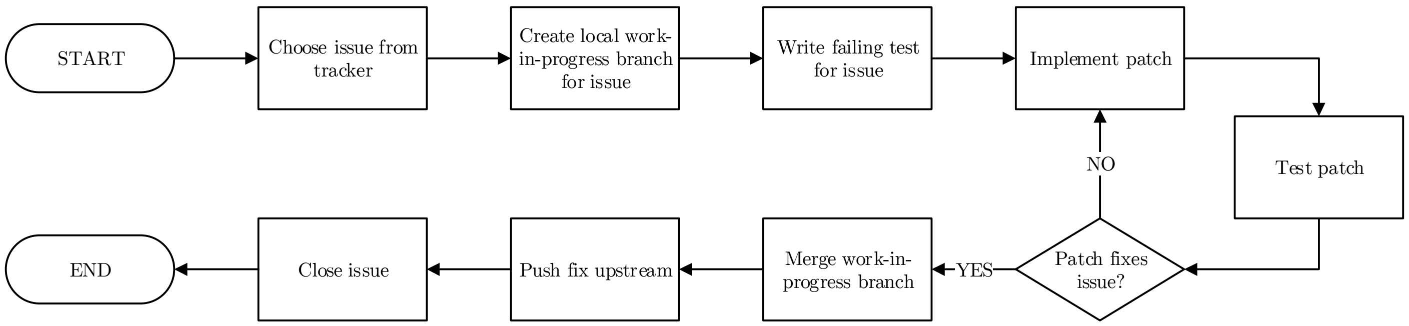

START
Choose issue from tracker
Create local work-in-progress branch for issue
Write failing test for issue
Implement patch
Test patch
NO
Patch fixes issue?
YES
Merge work-in-progress branch
Push fix upstream
Close issue
END

Figure 2 A single iteration of the project's test driven development workflow

By combining the available issue tracker with good version control practises, it is possible to implement a simple and functional test driven approach to development (Figure 2). This breaks down the development process into single-issue chunks, with each iteration beginning with creating a local development branch for an issue and then writing failing test cases which can then be patched. Using this model of development ensures that all work undertaken is relevant to the project and directly affects progress, minimising the amount of time wastage and increasing the stability of the codebase by ensuring adequate test coverage (6).

Project Schedule

The project development is spread over a 26 week period, with 11 weeks in the first teaching period and the remaining 15 in the second. In order to maximise the effectiveness of this time, a list of tasks for each of the four OpenUP development phases was constructed, and a time allowance associated with each. The final project plan consists of 8 phases: the inception phase and transition phases, and four iterations of elaboration and construction. The smaller elaboration and construction cycles were used so as to maximise the allowance for changes in the project specification caused by user feedback and review without causing delays in the development. This is to minimise the impact of the "Change in project requirements during development" risk. Once the list of tasks was assembled, a Gantt chart (page **Error! Bookmark not defined.**) was constructed which ordered each of these tasks and distributed them across the timespan. Careful ordering of the tasks ensured that there is the least chance for blocking between activities, where one task runs over the specified time allowance and causes later tasks to be postponed until it's finished. The final project plan allows for the maximum amount of parallel activities and development by ensuring that there are adequate gaps between activities that depend on each other.

Milestones

In order to provide a running measure of success for the project, a set of milestones were defined which track the development process from inception through to transition and provides completion deadlines for a set of activities. Two types of milestones are used: design and implementation. Design milestones cover the design of the user interface, such as the "look and feel" of the project, the interaction mechanisms, and other design implications. Each design milestone is preceded by a round of user testing, in which feedback and opinions can be gathered by the project stakeholders in order to influence the next design milestone. The implementation milestones cover the technical development, with each milestone marking a set improvement in the implementation of the backend, frontend, and controller, from the initial prototyping phase to the "feature complete" endpoint. Unlike the design milestones, the implementation milestones do not rely so heavily on input from third parties and so are more a personal measure of my own development.

Design Milestones

D1 First iteration design (week 3) – at this early stage of development, the design should consist of a set of non-interactive "paper prototypes" or static renders of the application interface, which can be used as a rough guide for beginning to prototype the interaction design.

D2 Second iteration design (week 13) – the user interaction design should be the primary focus of this second iteration, with many of the common tasks (searching for a result, looking up a record, etc.) being more tightly defined.

D3 Third iteration design (week 18) – by the third iteration, the interaction design should be complete, allowing the focus of development to be placed on polishing the look and feel of the application and establishing a common aesthetic style.
D4 Finalised design (week 24) – this last design milestone marks the endpoint of all design changes, and can be used to review the quality and effectiveness of the fully evolved product.

Implementation Milestones

M1 Initial prototype (week 11) – by the end of week 11, breath-first and depth-first prototypes of the system which some of the more common user tasks should have been implemented, although the underlying software architecture and technologies are free to change for the production system.

M2 Working system (week 18) – by week 18 the software architecture and choice of technologies should have been fully realised, and the functional backend of the majority of use-cases should have been implemented, along with good test coverage of each.

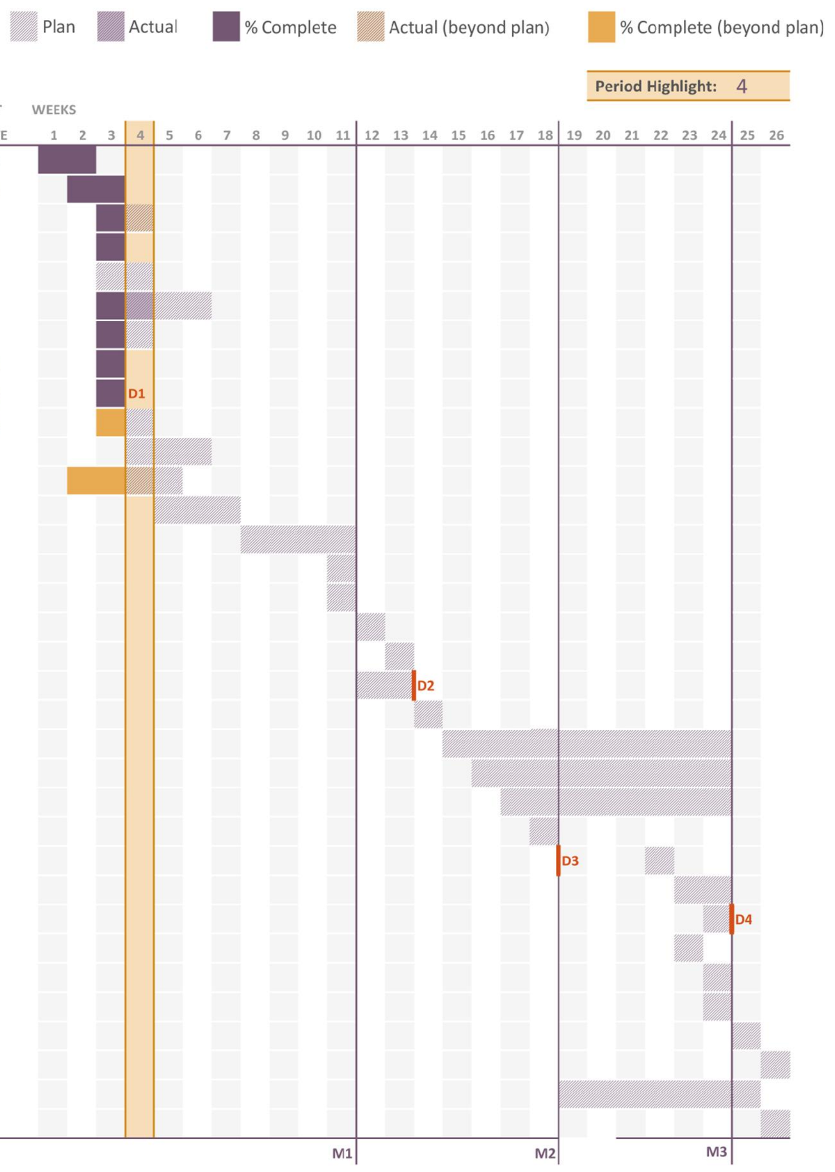
M3 Feature complete (week 24) – the feature complete milestone marks the end of the development of new features. By this point, the system should be fully functional and optimised, allowing for final stress and load testing to take place, and for the system to be deployed.

References

1. **National Center for Biotechnology Information.** About NCBI. *NCBI.* [Online] ND. [Cited: 31st October 2013.] http://blast.ncbi.nlm.nih.gov/Blast.cgi.
2. **The Eclipse Foundation.** OpenUP. *Introduction to OpenUP.* [Online] ND. [Cited: 31st October 2013.] http://epf.eclipse.org/wikis/openup/.
3. **Rational.** Rational Unified Process - Best Practices for Software Development Teams. *IBM.* [Online] November 2011. [Cited: 21st October 2013.] http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf.
4. **Free Software Foundation, Inc.** GNU General Public License, version 2. *GNU Operating System.* [Online] June 1991. [Cited: 31st October 2013.] http://www.gnu.org/licenses/gpl-2.0.html.
5. **Cummins, C.** Online profile. *GitHub.* [Online] 12th April 2012. [Cited: 31st October 2013.] https://github.com/ChrisCummins.
6. **Martin, R. C.** *Agile Software Development, Principles, Patterns, and Practices.* s.l. : Pearson, 2011.
7. **Maier, D.** *The Theory of Relational Databases.* s.l. : Computer Science press, 1983.

Final Year Project

		PLAN		ACTUAL		PERCENT
ITERATION	ACTIVITY	START	DURATION	START	DURATION	COMPLETE
1	Inception	Meeting stakeholders	1	2	1	2	100%
1	Inception	Iteration planning	2	2	2	2	100%
1	Inception	Previous FYP investigation	3	1	3	2	25%
1	Inception	Dataset investigation	3	1	3	1	75%
1	Inception	Resource acquisition	3	2			
1	Elaboration	Use cases and investigation	3	4	3	2	25%
1	Elaboration	Risk assessment	3	2	3	1	50%
1	Elaboration	Create fake datasets	3	1	3	1	100%
1	Elaboration	User testing (1)	3	1	3	1	100%
1	Elaboration	Prototyping (1) Static	4	1	3	1	100%
1	Elaboration	Software architecture planning	4	3			
1	Construction	Build system	5	1	2	3	95%
1	Construction	Prototyping (2) breadth-first	5	3			
1	Construction	Prototyping (3) depth-first	8	4			
2	Elaboration	User testing (2)	11	1			
2	Elaboration	Meeting stakeholders	11	1			
2	Elaboration	First iteration review	12	1			
2	Elaboration	Second iteration plan	13	1			
2	Construction	Artwork & assets	12	2			
2	Construction	Templating engine	14	1			
2	Construction	Back-end development	15	10			
2	Construction	Front-end development	16	9			
2	Construction	Presentation development	17	8			
3	Elaboration	User testing (3)	18	1			
3	Elaboration	Benchmarking (2)	22	1			
3	Construction	Optimsation and Security	23	2			
3	Construction	User testing (4)	24	1			
3	Construction	Benchmarking (2)	23	1			
3	Construction	Load testing	24	1			
3	Construction	Penetration testing	24	1			
3	Transition	Source code documentation	25	1			
3	Transition	Source code licensing	26	1			
3	Transition	Report Writing	19	7			
3	Transition	Report Proofing	26	1			



Design milestones

D1	Week 3	First iteration design ✓
D2	Week 13	Second iteration design
D3	Week 18	Third iteration design
D4	Week 24	Finalised design

Implementation milestones

M1	Week 11	Initial prototype
M2	Week 18	Working system
M3	Week 24	Feature complete