Gathering Requirements for an Integrative Biology Project

The project

The Integrative Biology project is a second generation e-Science project, funded by EPSRC, which will build on the output of first round projects and integrate these and other new developments into a customised Grid framework for:

- running large scale, whole organ HPC simulations,
- managing a growing database of simulation results and
- supporting collaborative analysis and visualisation.

The long-term goal driving the project is development of an underpinning theory of biology and biological function capable of sufficiently accurate computational analysis that it can produce clinically useful results.

The Requirements Process

Gathering requirements can be approached in many ways, from videoing current work-practises, to interviewing potential users, to rapid prototyping. The approach taken depends very much on the type of users who expect to use the system being developed and on what is to be delivered by the project. As a second-generation project, an integration project and a production system development project, Integrative Biology raises a number of interesting requirements gathering and representation issues. This paper aims to identify the variances in gathering requirements and outlines the approach that the Integrative Biology project has taken to address the needs of the user community for this project.

In the commercial field, the process of gathering requirements is driven by several key factors:

- The client budget
- The timescales for delivering a solution
- Pre selected technology

These factors dictate the most cost effective process for a company to determine what is to be built, or solution requirements. Where the client has a budget or time constraint, the process of gathering requirements is typically carried out in phases:

1. The exploratory stage

This process determines the opportunity for the vendor. Does the client have a large budget to spend? Do they have existing technology that should form part of the 'solution'? Do they have any preconceived idea of what the system should look like or do? Who are the users and how computer literate are they? Who will be the key stakeholders that determine what is to be developed? This process underpins the requirements exercise. If this process is not carried out, the vendor is likely to suffer conflicts later in the project with poorly met expectations.

2. The Scoping exercise

At this point in a project, the client is often paying for the delivery of a scoping document. This document contains details of what will be delivered (what is in scope) and what is out of scope. This involves interviewing project stakeholders and key users. The client then signs off this document when they are confident that it reflects their expectations. Only when this is signed off does the vendor proceed to capture the detailed user requirements. Frequently, this phase is separately funded from the main developments, and this document then forms the basis for the development contract.

3. The Detailed User Requirements

This process is carried out by more technical consultants and is charged to the client. This process is iterative and involves working directly with the users to document what they expect the system to do and how they want the system to look. Companies use a variety of methods to capture this information, from textual representations to case modelling. Again the user community signs off this document of captured requirements before moving onto technical design. The document then goes on to form the basis of the acceptance tests that the developed system has to meet. A key point to make here is that before the project commences, a resource is identified who takes the role of 'super-user' and who will act as an arbitrator for the user group.

4. The Technical Design

This process defines and documents technical architecture from the agreed requirements and is charged to the client. The primary objective of this phase is to produce a specification that the development team can build. The key issue is to ensure that the system built will actually satisfy the requirements. Ideally, two related documents are produced, an architecture document that is oriented towards the end users, so that they can fully understand the system being developed, and a high level system design document oriented towards the IT developers,

that provides the necessary decisions and structure needed for the implementation. Unfortunately, in most cases, only one document is produced to carry out these two functions. One key point is that it should be possible to trace each requirement to the part of the system that will satisfy it. Sometimes this is done formally; often it relies on the abilities of the users to understand the technical design documents.
By following the above process, the vendor is more likely to be successful in managing the expectations of the client.

Looking at research projects, there is often no clear user, and no clear deliverable. The key to these projects is innovation. Often though, there is an element of development in these projects. Determining what to build requires a process very similar to that used in the development of products in industry:

1. Identify Market or Research Opportunity

This process identifies the fundamentals of a research project in academia and identifies a new product direction for a commercial company. This exercise looks at competitors, looks at current trends and looks at innovative solutions. The output of this exercise may be a feasibility report and return on investment analysis. In the research world, it is more likely that this information is part of the research proposal and the acceptance of the proposal is taken as implicit approval of the “requirements” stated.

2. The Detailed User Requirements

If a feasibility study identifies a market opportunity or a research project, this process requires the resources to work with a cross section of users to determine the wish list for a potential product. There is often no user representative who acts as arbitrator, and indeed often no group of users with the motivation to contribute the significant effort needed to produce fully comprehensive requirements. This results in the requirements gatherer having to sift through many requirements, often conflicting, to determine a baseline system from which to work. The definition for the product or project deliverables therefore has to be taken by the vendor or project team in the face of incomplete requirements information. The only saving grace is that the vendor or project principals are frequently knowledgeable about the domain.

3. The Technical Design

This process defines a technical architecture to support the product definition.
Many of the projects undertaken in large multi-disciplinary research projects follow variants of the latter process. The deliverables are often prototypes or demonstration capability.

The eDiaMoND project followed a similar process and identified the challenges in working with a disparate user community with no standard ways of working. In this project, extensive effort was put into understanding the domain through observing clinical staff in situ and observing their work practices. The complications then stem from having to decipher this knowledge and determining baseline system requirements to enable an initial prototype to be developed.

Communicating with Users

The language used when communicating with users is crucial to ensure common understanding between users and project teams. Users who are technology aware or have existing and often deficient systems are usually able to articulate what they want from a system in a language understood by those developing systems. Users who have no existing systems on which to base their needs for improvements or are not as aware of what technology can do for them require assistance with this process, as they are often unable to articulate their needs.

The language used and the ways in which user needs are described and documented in terms of system functional or non-functional requirements varies, although Use Case Modelling has been used widely in industry for several years. This language used may be textual, diagrammatic or through the use of rapid prototypes to ‘mock-up’ systems to enable true feedback through experiential exercises.

Another problem with the use of UML in this particular project is that many of the requirements are defined at higher levels of abstraction as the interaction with the computer and the software system is an aid to expert decision making through data exploration. What is required is to develop a modular problem-solving environment that is flexible and dynamic to support data exploration and steering by a group of experts. As a consequence, we are likely to use UML based language and methodology more a tool for reasoning about the architecture and the modules amongst the architect and the software developers rather than as a tool to gather user requirements.

The Integrative Biology Challenges

The Integrative Biology Project will be developing a simulation framework, which will offer users the potential for more collaborative working, access to HPC resources and a more interactive way of working on these complex problems. The potential users of the system work in very different ways to each other. They typically are able to use mathematical modelling tools or develop their own software packages to enhance the science. Their working environment is usually their laptop or desktop machine and they work within these confines by adapting the way they work to fit these constraints. Their understanding of technology varies from able to develop parallel codes, which will run on HPC machines to obtain the best performance for their modelling codes, to little or no understanding of how collaborative systems could enhance the science. Also, each group of scientists have home grown software that encapsulates their knowledge of their field, their research focus and the familiarity they have of the desktop tools. The requirements for performance are different. A generic architecture needs to build in different views

Some of the scientists on the project have worked collaboratively for many years but that process has involved joint publications and the sharing of developed models, code and data informally.

For example, the team in Auckland have worked extensively with Denis Noble's group in the UK and with other groups in the US and have incorporated their models into the CMISS repository [1]. This code is provided to the research community as open source executables. These collaborators have not, to date, been able to work collaboratively on a daily basis and in real time so the vision we are portraying for the project are far from how these scientists work today.

Our Approach to the Initial Requirements for Integrative Biology

The Integrative Biology Project is seeking to introduce several changes in the way its end users work, providing access to more powerful resources, encouraging the use (and reuse) of available tools and supporting a much more collaborative working environment. It has therefore adopted an iterative approach to requirements capture through the development of an initial prototype, from which the project will iterate through reviews and further requirements definition. It is envisaged that the prototype will act as a tool for working with the scientists to determine the true requirements for a modelling 'collaboratory' for bio-simulations. This initial phase therefore required a different approach from the later requirements exercises, as the starting point was almost a clean slate, with little to show the users other than descriptions and demonstrations of what other first generation e-science projects have produced, and which may form a basis for improving the tools available to them.

The definition of the initial prototype required the project team to determine the essential features of a potential simulation framework to ensure that the development work started from a strong base and ensured the buy-in from the user community by developing a prototype that excited them.

With this in mind, the project underwent an initial requirements capture exercise with heart, cancer and molecular modellers. One of the advantages of the project is that it incorporates within itself a good cross-section of end users, so is not reliant on busy and somewhat unmotivated external users. The team explored the different methods for extracting these requirements and decided to utilise scenarios as the language for communication.

The various aspects of the system were split into clearly distinct areas:

- Visualisation and Computational Steering
- Data Management
- Simulations and Modelling
- Security

These areas were assigned leaders for the requirements process. These leaders were able to work with the users to articulate the possible benefits of potential tools. The individual groups devised questionnaires for the users, to ensure that key aspects of requirements in each of the above areas were considered. These questionnaires were sent to users to consider prior to a detailed briefing on the process so that the users understood why their contribution was important.

These scenarios were described as:

- Something users want to be able to do, described in terms of the steps required to achieve it

- Having no more than 5-10 steps per scenario
- Able to inter-relate to other scenarios
- Can be described by the potential users
- Have a distinct ID so traceable

The aim was that users articulated what they want to be able to do in terms of scenarios without having to consider the system requirements. Example scenarios were provided for reference from the Principal Investigator to both encourage the users to contribute and also provide them with samples of what the team require. The user community was briefed through presentations and access grid calls on what the requirements team required from them.

One of the sample scenarios was:

Simple: A collaboration between a team of mathematicians and experimentalists has resulted in the development of a simulation code that is sufficiently CPU intensive to require HPC.

The group take their code, write a (simple) wrapper, based on a template provided, that will interface their code's I/O to that of the HPC host, and rebuild with the supplied HPC versions of their libraries. The compiler will "optimise" for the HPC. Via a portal, they upload their input data and executable onto the HPC system and submit the job. Through the portal they can see their job's status, and when it completes they download the result datasets ready for visualization using their existing visualization tools.

The users were shown how these scenarios would be used to determine system requirements:

A Web based application portal needs to:

- *display a list of HPC resources and current load on these.*
- *allow the user to select the HPC on which the simulation needs to run.*
- *show the wrapper template in an editable text box (wrapper editor).*
- *show the appropriate libraries for the chosen HPC.*
- *ask the user for file names to store resultant data.*
- *submit the job for optimisation and execution as a proxy for the user.*
- *display the execution state via the portal interface.*
- *inform the user (possibly via email if the simulation takes long time) when the job is complete with enough information about where the result is stored, file formats and file sizes.*

A 'twiki' repository was set up for all staff to access, with a section created for the input of 'User Scenarios'.

The users were then asked to input their main scenarios, which described key aspects of how they work, into the project repository. These scenarios described their day-to-day work, whether it is running new or existing mathematical models on existing technology, or developing new tools to enhance the science.

The initial plan had been to take these scenarios and get the requirements team leaders to develop the system requirements from these scenarios. The scenarios, however, did not provide the level of details required to perform this task fully but they did provide sufficient information to identify the key elements of an initial prototype.
The plan had been to develop a list of specific requirements that would have the following properties:

- Each requirement will incorporate only one idea
- Each requirement will be clear and quantifiable and verifiable
- Each requirement will be categorized and prioritised
- Each requirement will be traceable, back to a scenario and forward to the system design

Reviews with users would then have qualify where possible that the requirements have been captured correctly and these requirements would then form the basis, together with an existing technology review, of the initial prototype for the project.

The results of the initial requirements exercise for Integrative Biology
The requirements process lasted for approximately four weeks and required extensive effort from the requirements team to cajole the users into documenting their scenarios. Where the barrier to providing this information was the use of the repository, the project team captured these scenarios from the

users and documented these for the users to sign off on. Many of the project staff had never used a project repository or an intranet before and were even resistant to signing up to this service despite the benefits articulated to them of a central repository for project knowledge. This resistance may have been an indication of the low levels of active collaboration between the users.

A key finding during this process was that users were used to working with what they had been given or provided with. They were not used to being asked for their opinions on what a system could do for them, let alone asked what their ideal way of working would be. The users therefore needed much handholding to give them the confidence to express their ideas.

Whilst it is considered poor practise to build systems which the developers think are suitable for users without determining the needs, we have experienced a sense of either fear or 'I don't want to be the first' in providing information.
One user stated,
'I have never been asked what I wanted before – we have always had to make do with the tools available'.
Another stated,
'I am only a lowly modeller and do not feel comfortable stating requirements for a prestigious community of experienced scientists'.

The requirements team had to stress the importance of the user's knowledge and the importance of steering the development team towards a truly useful prototype. One explanation for this may be that the scientists have immediate problems to solve, which they perform using existing, possibly primitive tools. Their methods and extent of collaboration are limited and true collaborative working will be a social as well as a technical hurdle. This tactical focus on removing immediate barriers within their existing environment rather than strategically deciding how they would ideally like to work was probably the most common problem the requirements team met.

One of the scenarios extracted by the authors of this paper was that of an integrated problem-solving environment for cancer modellers. The user requirement was gathered by one of the authors by sitting beside one of the modellers, just observing them carry through a routine modelling, simulation and testing activity the results of which had just been published by her. The user agreed to be interrupted with questions by the author. The user had asked the author to familiarise with the topic at the level of an intelligent layperson by reading through four research papers in the topic. The author also undertook background Web search to familiarise herself with the terminology and the barest basics, so that the above observation action could be undertaken more effectively. The user and the author used this session to discuss the activity that the user is undertaking and to discuss how the scenario could be used, by the author, to determine the requirements for a more integrated environment.

The user, as well as the cancer modelling community, uses Matlab^{TM} as their desktop modelling tool. Researchers are mathematical modellers and hence find the built-in capabilities of this generic problem-solving environment for symbolic computation, handling of modest sized matrices, algebraic manipulation and simulation testing and traditional visualisation of modest amount of data very useful. It provides a high-level scripting interface to test ideas and rapid prototype algorithms. The users want to retain the familiar high level scripting interface and the history mechanism and similar facilities that Matlab supports. That sets the client interface requirement.

While discussing how the software was used and what the user was trying to do (as opposed to what the interactions were or what the code/application was), it emerged that the modeller was trying to define the governing equations to model cancer cell growth *in silico* which will then be solved numerically and the predictive power of the model compared against experimental data. Components/modules of the governing equations and/or its numerical solutions and input into it are borrowed from literature some times. So the modeller needed access to such modules and codes and the associated literature, which describe the underlying assumptions behind those so that appropriate modules may be adopted into current model. This suggests the development of a server-side database and associated user interface tools that will allow collaborating researchers:

- to submit their models, with annotations on the underlying assumptions
- to submit a copy of the published work based on that model

- to submit the numerical simulation/code fragments of the model along with compilation instructions.

This will need to be accompanied by the development of client-side modules that can query, access and display the contents of the database, as well as fetch, compile and execute the codes as part of a workflow system that has modules of the users alongside those from the database. A Web Services and XML based methodology that will allow transparent and standard communication protocol between the two systems.

The third requirement gleaned from the observation and discussion was that an associate of each modeller carries out an equivalent in vitro experiment, and the data from that is used to validate the mathematical model. The in vitro experiment takes roughly about 30 days to complete and data is delivered after the completion of an experiment. This means that the modeller is designing and testing her model and equations many, many times without being guided by the data that can guide this prototyping, when such data is available but not accessible easily as the practice stands now. The author and user then discussed that a productive problem-solving environment should be developed

- to upload experimental data as it is being collected, build data query and access to that experimental data as it is being gathered (cell growth data is collected every other day by the experimental scientist)

and

- build a secure Web Services based access to the query and access the data service from the desktop environment.

Improving the performance of the solvers and numerical schemes will allow the user to increase the complexity of the model, including more realistic governing parameters. The Grid and Web services based application solving can then be built around with the simulation executing on a remote Grid compute layer.

Security needs were handled separately. The security questionnaire was circulated to all users to attempt to capture identifiable assets and the users view on what should be kept confidential, what should be kept integral and the availability needs of these assets. It was interesting to note that only the project manager returned this questionnaire initially and that users did not consider their software, hardware or data as assets! An in-depth interview with a key user by the security team concluded that valuable resources like key software packages were not adequately secured or backed-up (much to the user's horror) thus introducing key risk elements to the existing science and highlighting key deliverables in terms of policies and security infrastructure for the prototype. The process of performing this activity for all project members will be lengthy.

An alternative approach has been taken in defining a legal questionnaire for all project collaborators to complete. This questionnaire is a request for detailed information regarding software and its use and terms, data and its use and terms and the services to be provided and the conditions under which they will be offered. It is similar to the activity undertaken by legal departments in defining materials transfer agreements, but for software, hardware and services. The objectives here are to underpin the collaboration agreement and ensure that there are common understandings relating to the ownership of intellectual property. Any infrastructure developed must be able to support the range of collaborative scenarios envisaged by the users. This questionnaire also provides valuable information for defining the security policies for the prototype.

For this initial stage of requirements, we have limited our consideration of the user interface to the knowledge of the existing tools that are used and understand that detailed user requirements would need to be carried out in subsequent phases. The utilisation of developed prototypes will enable this process to be carried out more effectively.

Conclusions from the initial stages of requirements gathering

The team have had to encourage the users into feeling that all of their views will be taken into consideration. It was also evident that you need 'brave' users to set an example and to make others feel comfortable with imparting information. The use of focused questionnaires has certainly ensured that the right questions are asked and experience has shown that interviewing either in person through a 'panel' or over access grid results in better information being provided as the users can be questioned where issues are not clear. The requirements gatherers used had extensive technical experience so were able to articulate questions

to ensure that the team could document requirements from the scenarios.
Our expectations initially were that users could articulate what they did in terms of scenarios from which we could define detailed system requirements. This process did not result in scenarios that enabled us to perform this task but were more a detailed description of the constraints under which we could work for this initial stage. The language used in these scenarios was typically 'I have developed a software package which.'
The scenarios were more of a knowledge elicitation process, enabling the team to find out what the users wanted and the gaps in their capability. Moving these scenarios onto detailed systems requirements was deemed impossible for this stage of the project.
For this reason the technical team tasked with delivering the initial simulation framework architecture have taken the information provided by the users to

a) construct an initial design for the architecture to act as a straw man, and
b) define a set of usable demonstrators to help explain to the users what is possible.

The architecture design describes the key functions of the Integrative Biology infrastructure, based on our understanding to date. As part of building this understanding, an evaluation of existing technology was undertaken, and this highlighted some key e-Science deliverables that may be utilised to form a prototype or demonstrators. The demonstrators are based on real applications like the COR [2] and CMISS packages, adapted to show some of the potential (and problems) of the sort of environments proposed by the project. Coming from real systems, these demonstrators will themselves allow users to extend the range of science the original systems could tackle. Development of this architecture and the adapting of existing models to match the new architecture and will act as a basis for more detailed discussions with users and requirements definition. Once the prototype and demonstrators have been built and studied by the users, the scenarios will then be revisited.

The demonstrators and prototype will enable simple observational studies to be carried out with users working with the facilities provided.
This stage of requirements is expected to get us back onto the initial anticipated approach, defining detailed requirements linked to scenarios. We will also utilise the prototype to review the user interface needs.
The benefits of this approach are believed to be:

§ Earlier results
§ A better chance of standardisation
§ Hopefully, an increased buy-in from users who see real science being done, not just mock-ups.

The disadvantage of this approach is that there is a danger that we are seen to be imposing our solutions on users. We will have to ensure that the initial prototype is 'sold' to the users as a basis only and is only to be used to inform the real needs of the users. We will also have to make sure that the prototype can (or at the very least, clearly has the potential to) enable the users to carry out scientific investigations they could otherwise not have done. The initial scenarios will help us to shape the next requirements phase to ensure that we do not just get requests to refine the demonstrators.

The prototype will generate a set of definitive experiences that can be quantified and qualified in terms of what worked and what didn't work and why, which will be useful to this project and others.

Acknowledgements
The authors wish to thank other members of the Integrative Biology consortium for their input into, and comments on, the work described in this paper, in particular the role of the scientific collaborators in determining the requirements for this project. The authors wish to acknowledge the support provided by the funders of the Integrative Biology project: The EPSRC (ref no: GR/S72023/01) and IBM

References
[1]CMISS- http://www.bioeng.auckland.ac.nz/cmiss/cmiss.php
[2]COR- - http://www.hellix.com/People/AGarny/PhD/, http://cor.physiol.ox.ac.uk/