The Challenges of Clinical e-Science: Lessons Learned from PsyGrid

John Ainsworth, Robert Harper, Lucy Bridges, Pauline Whelan, William Vance, Iain Buchan
School of Medicine, The University of Manchester
{john.ainsworth, robert.s.harper, lucy.bridges, pauline.whelan, william.vance, iain.buchan}@manchester.ac.uk

Abstract

The PsyGrid project was established to design, build and operate a multi-centre, secure, clinical data capture system that could support clinical trials and cohort studies from any clinical domain. After two years of development and one year of operational experience we present an evaluation of the technological, operational and organisational choices we made and assess the consequences of these decisions. We conclude that the development of successful e-Science systems depends on consideration of the needs of all stakeholders, a user centred design approach and thorough systems analysis.

Introduction

The health informatics literature is littered with examples of systems that provided innovative solutions and yet failed to deliver to the clinical community [1][2][3][4]. Our concern was to ensure that PsyGrid [5] did not become an exercise in the application of the latest computing technologies to clinical research, but that we developed an e-Science system of direct use to our immediate user community and that this would subsequently foster the adoption of e-Science in the wider clinical research community. We used a human centred project methodology [13][14] that emphasised the importance of analysing the needs of all the stakeholders. We then used this analysis to inform the design and development of PsyGrid, the operation of the PsyGrid system, and also the way in which the PsyGrid project was organised. We determined that these key success factors were consideration of the needs of all stakeholders, a user centred design approach and thorough systems analysis.

In the remainder of this paper we present an overview of the PsyGrid project and describe our progress to date. We then provide an analysis of the success factors we identified, and report on how this influenced the design, development, and operation of the system and the organisation of the project. In each of these areas we describe the major decisions we made and discuss the consequences that ensued. We concluded that the stakeholder driven approach was very effective and enabled us to identify and address the key issues.

Background

PsyGrid is an e-Science project funded by the Medical Research Council and the Department of Health. The project had twin aims: one was clinical and one was technological. The clinical aim was to perform a longitudinal study into the aetiology of First Episode Psychosis, which drew subjects from all of the eight hubs of the Mental Health Research Network [6]. The scale and distribution of the study was unprecedented in mental health research in the UK. This served to provide concrete deliverables and a strong focus to the development of technology within the project. The second aim, the technological aim, was the design, development and operation of middleware and applications to enable the clinical arm of the project. The functions, architecture, implementation and deployment of the PsyGrid system has been described in detail in [7] and [8].

The initial requirement for PsyGrid was to support a single study with eight centres, twenty users and a projected subject study size after two years of the order of one thousand. At the time of writing, we have three studies currently deployed and two more will be deployed in the next month.

This will increase our user base to approximately two hundred, and the number of centres will be thirty. The duration of the studies ranges from two to ten years. Over the next eight months a further five studies will be deployed. Not all the studies are in mental health, as PsyGrid now also supports studies in diabetes research and the system is currently being evaluated for use in primary care and stroke research. There have been no significant changes required to support new clinical domains. From the above we conclude that:

1. PsyGrid's architecture is scalable.
2. PsyGrid is generic such that it can be applied across clinical topics.
3. PsyGrid is reliable enough to be used in the clinical environment.
4. PsyGrid delivers the functionality required.

The addition of new studies has been very helpful for two reasons. Firstly, it validated the existing functionally as being re-usable. Secondly new functionality that had not been anticipated but could also be re-used for future studies could emerge. A good example of this is the Type A Diabetes study which required us to extend the Electronic Screening Log to include family relationships between subjects. Future studies where genetic factors are a consideration will doubtless require this functionality.

Critical Success Factors

We identified many factors that could determine the success or otherwise of the project, but there were three that we considered critical.

1. Negative user attitude

Each multi-centre study or trial has its own user community but the project roles and organisation within in each is largely the same. Each one is lead by a clinical researcher (Chief Investigator, CI), who oversees the whole study, supported by other senior clinical researchers (Principal Investigators, PI), who oversee individual centres. This group of users are primarily involved during the initial inception and design of the study and they tend to have little involvement in the fine detail or day-to-day management of the study. Their typical interaction with the system is to receive by email weekly status reports. However, they are generally enthusiastic supporters of using e-Science. A Clinical Project Manager (CPM) is responsible for the day-to-day running of the study and detailed specification of the data set. They have overall responsibility for data quality and interact with the system to review and approve documents. We found CPMs to be enthusiastic advocates of the e-Science approach, for one principal reason: their administrative burden is greatly reduced through the automatic generation or reports; not having to physically manage paper and the ability to see data as soon as it is entered. The CPM manages a team of Research Assistants (RA). The RA's task is to collect the clinical data and input the data to the system. This is the group of users who will spend the most time using the system and is also the group who benefit the least from paper-less data collection. The attitude of this group of users to Information Technology (IT) in general ranges from indifferent to negative. We concluded that we would not be able to change this attitude overnight, but we could certainly reinforce it in this timescale. If this had occurred then the system would not be widely adopted. We identified the following mitigation for this risk:

- Ensure that the system is robust e.g. it does not crash unexpectedly causing loss of data. This had a direct bearing on the technology choices we made, favouring mature, proven implementations over new and experimental technologies. It also affected the design and test procedures we employed.
- Ensure that the system is reliable, such that there it is always available to the user when needed. This led us to design the system with a redundant hardware architecture that had no single point of failure [8].
- Ensure that the system responds to the user in a timely fashion. There are few more things off putting to a user than wondering how long it will be before a system completes their request.
- Ensure that the system does not require the users to attain a higher level of sophistication with information technology than they already possess. This has a direct bearing on the user interface presented to the user. For example we cannot expect them to manage a file-based X.509 certificate, or learn a new user interface paradigm. To ensure that the users were comfortable with the system we employed Joint Application Development workshops

[12] as the primary means of exchanging ideas with our user community.

- Respond to user feedback and implement good suggestions. This creates a positive feedback loop, establishing a communication between developer and user. Otherwise, if a user's ideas are not responded to they will view the time taken to make a contribution as wasted and they will not do it again.
- Provide a support service with immediate response.

2. Acceptable to NHS Trusts

The second factor we identified was that the system needed to be compatible with and acceptable to the NHS Trusts in which it would be deployed. In practice this translates into the following requirements, which were elicited from NHS IT managers and network engineers:

- The deployment and operation of the system must not require any changes to a Trust's existing infrastructure.
- The system must not deny resources to existing clinical systems or require any intervention from Trust IT personnel for its operation.

3. Ease of customisation and operation

Clinical trials and studies are expensive to run and any reduction in the cost of operation will be welcomed by both researchers and funding agencies. If the costs associated with electronic data capture systems could be greatly reduced or ideally eliminated entirely by PsyGrid, then this barrier to adoption would be removed. There were three different costs to consider:

- The cost of the software application itself. We eliminated this cost by releasing PsyGrid as open-source software.
- The cost of the hosting platform and its operation and maintenance. This cost cannot be eliminated entirely, but was reduced by designing PsyGrid such that it could operate across a wide range of deployment architectures from single box solutions to fully redundant, incrementally scaleable solutions. It can also be reduced through economies of scale – the greater the number of studies hosted on a single platform then the lower the cost per study.
- The cost of designing and implementing new studies. Our approach was to design the system such that this task could be performed by the clinical researchers themselves, without requiring any more advanced IT skills than would be required to use typical office applications. The implications of this for the design and development of PsyGrid are wide ranging. Any aspect of a trial or study, for example the security or the randomisation scheme, must be configurable and graphical tools must be provided for the user to enable them to do this.

Technology

In this section we discuss the technological choices we made and analyse the consequences of those choices.

Middleware

PsyGrid employs a Service Oriented Architecture using WS-I compliant web services. The decision to use plain web services rather grid middleware such as the Globus Toolkit [9] resulted from a cost-benefit analysis of the various technologies. In brief we reached the following conclusions:

1. The various middleware toolkits provided a lot of functionality that we did not require, and for the functionality that we did need we could construct alternatives quickly and easily.
2. The Grid middleware toolkits were perceived to be less mature than the web service stacks on which they build.
3. Using a Grid middleware stack would tie us into old versions of the web services container.

Above all we chose the approach which would give use the highest degree of flexibility, was the simplest to implement and operate and was the most stable and reliable given that our requirements could be implemented easily using plain web services.

Security

The key decision was whether to use an existing access control framework or to implement our own. The decision not to use Grid middleware and plain web services meant that we were not tied to any particular security framework. The key factors that drove our security decision were:

1. Simple for the end users.

2. Flexible and adaptable for us as the developers to cope with unforeseen requirements.
3. The availability or capability to develop simple management tools.

The last point was very important. Any security management tool must be understandable and useable by the clinical researchers, if they are to design and manage their own studies. We did not believe that any of the existing frameworks were able to provide this and so the decision was made to implement our own. This has given us a security framework that exactly meets our needs and which we can quickly and easily extend. The Security Manager application has been developed which the Clinical Project Managers use to manage user accounts and privileges, without needing to understand the underlying security technologies. However the development and validation effort has been high. The Policy Authority (PA), a component of the security system that maintains policy and makes access control decisions, was designed such that access control policy could be configured on a per-project basis. It was envisaged that different studies would have different roles, groups and access requirements. In practice this has not been required as each study uses the same policy with a common set of roles. The policy in all the studies has remained static except for the addition or removal of user groups. So instead of providing a full policy editor to the end users, we only enabled the addition and removal of user groups. Since the whole policy does not need to be exposed to end users, in hindsight we should have used XACML [11] as our policy language, rather than our own design. The reason we did not choose XACML in the first instance was the lack of simple management tools, and the complexity of developing one.

Repository

The PsyGrid data repository was designed using a model based approach, such that a standard set of objects could be used to model any data collection project. The alternative would have been to utilize an approach where custom database tables were designed for each project.

During the time that the PsyGrid system has been fully operational the number of hosted projects has increased from one to three, and during this period the original repository object model has only required very minor modifications to support the two additional projects, whose requirements were not known at the time when the object model was designed. Therefore it is fair to say that the model-based approach has been a success.

One area that has caused some concern is the performance of the repository. Initially it was thought that long download times for datasets and records from the repository were due to the network latency of performing web-service calls between the client and the repository over the NHS network. However, further investigation showed that the majority of the operation time was taken up by calls to the database. Database tuning has increased performance somewhat but several factors remain that limit how much performance may be increased by. Firstly the use of Hibernate [10] for an Object Relational Mapping (ORM) layer means that we do not have direct control over the SQL statements used to retrieve data from the database. Secondly, as records and datasets are being populate by the repository for use in an environment where they are totally disconnected from the database we do not have the opportunity to use technologies such as lazy-loading to increase performance; the whole object graph must be populated from the database in one go. Thirdly, with hindsight the class model of the repository could have been compromised somewhat to reduce the number of layers of class inheritance in order to simplify the SQL statements required to populate the objects.

One area that caused some difficulties was the modification of live datasets, a process we called “patching”. In a perfect world all data entry forms for a data collection project would be designed beforehand and then would be static for the duration of the project. However, in practice this was not the case and all three projects currently hosted on the PsyGrid system required modifications after they have gone live and data collection has begun.

For minor changes to the wording of questions, or adding additional options to multi-select lists, the process was relatively straightforward:

1. Download the dataset from the data repository.
2. Make the necessary changes to the dataset's object model.
3. Save the dataset back to the data repository.
4. Update the “last modified” date of the dataset.
5. When users next logon to the system

through the data entry application (named CoCoA), it checks for updated datasets, sees that the "last modified" date is later than the time the dataset was last downloaded and downloads the new dataset.

However the majority of dataset patches were more complex than this and, most significantly, require modifications to be made to all records associated with a dataset, as well as the dataset itself. For example consider a patch where a question was to be removed from a dataset. All existing records for the dataset must be checked to see if they contain an answer to the question, and, if so, delete it.

Whilst this is not overly challenging from a development point of view (although writing patches like this was significantly more time consuming) it caused more problems operationally. The only way that these patches may be safely applied was by scheduling a system shutdown and requesting that all users of the system commit all data stored by CoCoA in caches on the local file system to the data repository. This however, made the success of running a patch reliant upon the user-base all complying with the request to commit all their data. On several occasions this did not happen, even though notice of system maintenance periods was given typically one week in advance and reminders were sent in the intervening period. Thus far no major problems have been caused by data not being committed before a patch has been applied, but there is potential for incompatibility between records created before the patch was applied and the patched dataset and currently the only way to resolve this would involve a complete loss of the locally stored data for the incompatible record (and not just the specific incompatible data).

CoCoA

The major design choice that had to be made for the PsyGrid CoCoA data entry application was whether to use a thick-client (Java Swing) or thin-client (browser based) approach. In the end our hand was largely forced by the requirement for the application to be able to be used when not connected to a network; and so a thick-client application was developed.

That being so, the need to develop a thick-client application using Java Swing has, in all probability, allowed for a richer user interface than could have been provided by a web-based application, where certain functionality – such as real-time data validation and calculation of "derived" entries – would have been more challenging to implement (although one could argue that the recent development of Web 2.0/AJAX interfaces could have allowed a comparable user interface to be developed for a browser hosted application).

The necessity of offline operation has created some challenges, though. Any operation that, in an online environment, would normally require an access to the database must be designed with careful consideration for working offline. A case in point is the generation of unique identifiers for new records being added to the system. If the application could be guaranteed to be online this would cause no significant difficulties; the next available identifier would be requested from the repository and assigned to the record. However, a significantly more complex scheme had to be used to facilitate offline usage. When the user first logged on the system (at which point they must be online) CoCoA requested a block of identifiers from the data repository, which were then persisted locally. As new records were created they were assigned identifiers from this block. Whenever the application was online it checked to see whether the number of available identifiers in the block was running low, and if so requested some more. Whilst this scheme was generally successful it was not without problems. Firstly our user base were generally more used to a paper-based system, and could be confused that the identifiers were not allocated in contiguous number order. This was exacerbated by the system "losing" identifiers, which resulted in large gaps in the numbering of identifiers. Identifiers, which are stored locally in the users home area, could be "lost" in a number of ways, for example; a user running CoCoA on more than one computer, or running CoCoA on the same computer but via different operating system user accounts.

Offline operation also raised potential data integrity issues. All data was initially persisted on the local file system, and was only committed to the data repository via a separate commit process for which CoCoA had to be online. This meant that data may potentially have been stored on the users computer for an indefinite amount of time, where it was at far greater risk of loss or corruption than if it were stored in the data repository. Whilst attempts were made to mitigate this problem, by providing

prompts in CoCoA for the user to commit their data, there is no way to completely solve this problem.

Finally, distribution of the software has to be considered with a thick-client application. Java Web Start was used to distribute CoCoA via the internet and whilst this generally worked adequately it was not completely without problems. For instance, software updates were often not detected the first time the application was started after a new release was deployed.

Organisation

The PsyGrid Informatics team consisted of a project manager and four software developers. The whole team is located in the same office at the University of Manchester. Consequently there were no communication overheads associated with multi-site working and information could be rapidly disseminated and assistance was always on hand. A small compact team enabled us to use a flexible and informal development process. Our approach combines elements of both the agile [15][16] and the evolutionary prototyping [16] development paradigms.

As previously discussed our main source of requirements was from the Clinical Project Managers and the Chief/Principal Investigators. As this group of users in general has little IT knowledge, the requirements they provided were very high level and required interpretation. The users also had limited understanding of what was possible and what was infeasible. Given these preconditions, the only viable approach to development was an iterative process where prototypes were produced for approval and validation from the users.

The project was planned into releases that deliver significant new functionality. The content of these releases reflected the original PsyGrid project proposal and the work packages that were funded. For each release we followed the iterative model by combining requirements gathering and prototyping iteratively until the users and ourselves were in agreement, then we would commit to development of the final prototype. The development cycle for a release was between four and six months. However, new user requirements would come in all the time and we wanted to deal with these quickly and effectively. Typically these would be small features where the implementation effort was no more than two weeks. We prioritised these requests and dealt with them accordingly. As soon as the small features were completed we released them by upgrading the system software. Consequently the system was being continually upgraded at approximately two-week intervals. If a critical or serious error was discovered it was fixed and deployed immediately. This continuous cycle of fix-improve-upgrade was highly visible to the users as they were able to see their problems being resolved and their ideas incorporated. This was a self-reinforcing process.

For the development of a major release a developer was given responsibility for developing a particular component of the system. However to ensure that knowledge was distributed throughout the team and to guard against concentrating too much knowledge in any one individual, no component had a particular “owner” who worked on it continually for the lifetime of the project. This approach was also beneficial for the implementation of small features that generally cut across many different components of the system. In the first 18 months of the project we had 80% turnover in the development team, and because of the approach of distributing knowledge throughout the team we were able to absorb these changes with minimal disruption.

Our testing process prior to release of the software had three phases. The first was unit testing of individual components as they were developed. These unit tests were subsequently incorporated into a regression test suite. The next phase of the testing process was deployment of the new software on a staging server, which had exactly the same configuration as the NHS production system. The new software was validated on the staging server. This was black box testing using the system as the user would. Depending on the size and scale of the changes, the software may be subjected to ‘soak testing’ for several weeks on the staging server. The final phase of testing was post deployment on the NHS production system. Again this involved running a small set of tests to ensure that deployment was successful. A special test data set has been installed for this purpose, as the actual clinical data sets cannot be interfered with.

Operations

The PsyGrid Informatics team not only developed the system software but also maintained an operational system and supported the end users. There was no single dedicated system administrator or 1st line support contact in the team. Instead a support rota was employed whereby every member of the team took responsibility for these tasks for a week at a time. During their time on support a team member's priority was to carry out the weekly system administration tasks and respond to user support requests. In practice this did not take 100% of their time and so in the remainder they continued with system development. The rota system worked well because it exposed the developer to all aspects of the system and so built up their understanding of how functionality they did not develop was implemented; it enabled the developers to talk first hand to the users and develop relationships with them; and from a project management perspective we were not dependent on a single individual. One difficulty with this approach is that it took time to develop the required database administration and server administration skills.

User support was provided through a dedicate telephone line and a dedicated email address, and so response to users was almost immediate. We have developed and maintain a user guide but in practice we found that our users were not prepared to read documentation. The community of clinical researchers who use the system preferred to be given training on how to use the system and so we developed training materials, which could either be class based or self-study.

The user community was found to be inconsistent in reporting system failures and errors. Initially when CoCoA encountered an error it would open a dialog box reporting the error and prompted the user to email the log files produced to PsyGrid support. However, we found that some users would simply ignore this, and so the errors would be left unknown and unfixed. We simplified this for our users and ourselves by automatically sending error logs to the central data repository, whereby we received notification of all failures and our users no longer had to do anything.

Our software upgrade process took advantage of the redundant server hardware architecture that we used to maximise availability to minimise the downtime of the system during the process. For upgrades to the software of the web services (and not changes to data set definitions or the underlying database model) we could simply deploy the new software on the inactive server and then switch activity between servers. The only downtime in the process was the time it took to switch from the active to the standby server, which was about sixty seconds.

Conclusion and Future Work

We have reported on the progress of the PsyGrid project and have shown that we have been able to meet our goal of creating a secure clinical data capture system that is easy to use and to re-use. We have argued that we achieve our goals by identifying the needs of the stakeholders and ensuring that their needs were met; by using a user centred design approach; and by thorough systems analysis. We recommend that all e-Science projects should follow this model due to the complexity and diversity of the different individuals and organisations involved.

We failed completely to identify one of the needs of the clinical researchers, which would have been relatively easy to solve in the initial design phase, but at this point in the project has no simple solution. Prior to PsyGrid many researchers had their own single site, ad-hoc data collection systems, and they would like to migrate their data from these systems to PsyGrid. The use of a model driven data repository makes this very time consuming and tedious, as it is not possible to directly map a semi-structured flat file to a single relational database table.

Finally, one group of stakeholders that we did not consider were the patients themselves. At the most basic level this could be performing self-assessment and entering their own data. Is there some way in which they could usefully interact with PsyGrid such that their care might be improved?

Acknowledgements

The UK Medical Research Council and the UK Department of Health funded this work through the clinical e-Science program.

References

[1] House of Commons Committee of Public Accounts Department of Health: The National Programme for IT in the NHS HC 390 [Incorporating HC 1360-i of Session 2005-06] Published on 11 April 2007 by authority of the House of Commons London: The Stationery Office Limited Twentieth Report of Session 2006–07

[2] Heeks R. Health information systems: failure, success and improvisation. Int J Med Inform. 2006 Feb;75(2):125-37.

[3] Brender J, Ammenwerth E, Nykanen P, Talmon J. Related Factors influencing success and failure of health informatics systems--a pilot Delphi study. Methods Inf Med. 2006;45(1):125-36.

[4] Beynon-Davies P, Lloyd-Williams M When health information systems fail. Top Health Inf Manage. 1999 Aug;20(1):66-79.

[5] The PsyGrid Project. http://www.psygrid.org

[6] The Mental Health Research Network. http://www.mhrn.info

[7] Ainsworth J.D., Harper R.S., Juma I, Buchan, I.E. "PsyGrid: Applying e-Science to Epidemiology" Computer Based Medical Systems 2006. CBMS 2006. 19th IEEE International Symposium (pp. 727-732)

[8] Ainsworth J.D., Harper R.S. "The PsyGrid Experience: Using web services in the study of schizophrenia", Int. J. Healthcare Information Systems and Informatics, 2(2), 1-20 April-June 2007

[9] The Globus Toolkit. http://www.globus.org/toolkit/

[10] Hibernate Object Relational Mapping. http://www.hibernate.org

[11] eXtensible Access Control Markup Language (XACML) Ed. Tim Moses, OASIS Standard, 1 Feb 2005

[12] Wood J. and Silver D., "Joint Application Development", 2nd ed., New York : Wiley, 1995.

[13] ISO 13407:2000 Human-centred design processes for interactive systems, April 2004.

[14] ISO/TR 18529:2000 Human-centred lifecycle process descriptions, April 2004.

[15] Cohen D., Lindvall M., Costa P. "An introduction to agile methods." Advances in Computers 62: 2-67 (2004)

[16] Sommerville I. "Software Engineering", 8th ed., Harlow, England: Pearson Education. 2007 ISBN 0-321-31379-8.