Soapbox

Prevention of errors and user alienation in healthcare IT integration programmes

Tim Benson
Honorary Senior Research Fellow, Centre for Health Informatics (CHIME), University College London and Principal Consultant, Abies Limited, London, UK

ABSTRACT

The design, development and implementation stages of integrated computer projects require close collaboration between users and developers, but this is particularly difficult where there are multiple specialties, organisations and system suppliers. Users become alienated if they are not consulted, but consultation is meaningless if they cannot understand the specifications showing exactly what is proposed. We need stringent specifications that users and developers can review and check before most of the work is done. Avoidable errors lead to delays and cost over-runs. The number of errors is a function of the likelihood of misunderstanding any part of the specification, the number of individuals involved and the number of choices or options.

One way to reduce these problems is to provide a conceptual design specification, comprising detailed Unified Modelling Language (UML) class and activity diagrams, data definitions and terminology, in addition to conventional technology-specific specifications. A conceptual design specification needs to be straightforward to understand and use, transparent and unambiguous. People find structured diagrams, such as maps, charts and blueprints, easier to use than reports or tables. Other desirable properties include being technology-independent, comprehensive, stringent, consistent, composed from reusable elements and computer-readable (XML).

When users and developers share the same agreed conceptual design specification, this can be one of the master documents of a formal contract between the stakeholders. No extra meaning should be added during the later stages of the project life cycle.

Keywords: computerised medical record, conceptual design specification, interdisciplinary communication, systems integration

Introduction

Despite years of effort, we seem little closer than we were 20 years ago to delivering joined-up healthcare information systems, providing access for patients and clinicians to healthcare information whenever and wherever required, and reusing clinical data to improve safety and efficiency.

The difficulties of sharing data between patients and care providers are now better understood, and foremost among these is the human factor. Users need to be consulted at every stage of design, development and implementation. Consultation with users about complex information technology (IT) projects is difficult because users need to understand precisely what is proposed, at a level of detail that they can check, review and sign off, without being alienated by lengthy, complex and arcane technical specifications produced for developers.

A longitudinal survey of doctors’ views about the National Programme for IT in the English National Health Service (NHS) demonstrates the size of the gap between doctors’ wish to be consulted (92% think it is important) and their perception that consultation is inadequate (only 5% think it has been adequate).
Users and developers do not speak the same language

The term ‘user’ is intended to cover not simply end-users, but also business analysts, managers and others who understand the user domain better than the technology. Similarly, the term ‘developer’ includes analysts, programmers and integration engineers who understand the technology better than the user domain. Users and developers need a common shared vision of what they are trying to do, but they do not speak the same language. Without a common language we have a Tower of Babel. Problems lie on both sides. Users find it hard to visualise what can be achieved with an electronic system that is not constrained by the physical requirement to move pieces of paper. They do not understand the development life cycle and insist on new features late in the day. Developers try to shoehorn users’ needs to fit their existing system or pattern, thinking that it will be quicker and cheaper to reuse what already exists. They often lack the domain knowledge to understand the business process and what the user really requires.

Too often, both parties genuinely believe that the developers will deliver just what the users want until the moment comes when the user tries to use the product for the first time. Sometimes, users join the design teams, but then the risk is that they ‘go native’, working and thinking as developers, rather than representing their constituency.

Communication between one user and one developer is difficult enough, but both sides can sit round a table and educate each other; this usually results in a satisfactory solution. Unfortunately, this does not scale (see Figure 1).

Achieving shared understanding becomes much harder in large projects, where users across many specialties and locations need to work with developers from different suppliers. The communication paths increase exponentially with the number of participants. Figure 2 shows the number of paths between just three users and two developers. Each additional path introduces new opportunities for misunderstanding. Misunderstandings lead to errors; errors increase costs, create delays, reduce productivity and hit profits.

Errors multiply according to the relationship: $M = M 	imes I 	imes C$ where:

$M$ = Misunderstanding – errors increase with the probability of misunderstanding of any part of the specification, caused by the complexity of the language used or the real-world business process, relative to the participant’s technical or domain knowledge

$I$ = Individuals – errors increase with the number of individuals (users and developers) involved in implementing a specification

$C$ = Choices – errors increase with the number of choices that each individual needs to make – the total number of options allowed.

This explains why complex, lengthy specifications that have to be implemented many times are far more expensive to implement than simple short specifications, only implemented once. There is nothing new in this. These issues have been recognised in software development projects since Fred Brooks demonstrated in *The Mythical Man-Month* that ‘adding manpower to a late software project makes it later’ (Brooks’ Law); this was primarily a consequence of the need to increase person-to-person communication between members of the team and the time taken to learn about the project.

Also relevant to this discussion, Brooks suggests devoting one-third of the schedule for any software task to planning, half to testing, but only one-sixth for coding. These estimates were developed for software development. Integration needs even more planning and testing. He also contends that ‘conceptual integrity is the most important consideration in system design’ and that ‘ease of use dictates unity of design, conceptual integrity’. Systems need to be straightforward. Simplicity is good, but not at the expense of being awkward.

Figure 1 Communication between one user and one developer

Figure 2 Communication paths between multiple users and developers; more paths introduce more potential errors
People like using maps

People communicate in different ways, but semantic accuracy increases as follows:

- face-to-face discussions or telephone conversations
- free-text documents, such as email, meeting minutes and reports
- structured documents, such as templates, spreadsheets and tabular lists
- ad hoc diagrams and sketches
- formal diagrams, such as maps, engineering blueprints and UML diagrams.

People often use a combination of these methods, which improves semantic accuracy and ease of use.

A simple example may help to make the point. My son goes to school about three miles from my house. There are four road junctions on the route. From the internet, I have found two route plans: Figure 3a shows the route in tabular format, and 3b shows the route as a map.

The table format appears, at first sight, to contain more information than the map, but most people find the map to be far easier to use.

Unified Modelling Language

For integration projects we can use UML modelling tools to specify precisely each of the business processes and the data content of each message using a standard diagram notation. UML is the industry standard for modelling software systems. Reading UML diagrams is not difficult, but has to be learned, and UML should be part of any health informatics curriculum.

On the other hand, the production of UML models is a skilled task that requires the use of a specialised UML tool. Such tools support detailed definitions of each element, hyperlinked documentation (HTML) and XML representation using XML Metadata Interchange (XMI). For our purposes, the two most important types of UML diagram are activity diagrams, for representing business processes, and class diagrams, for representing data content.

Terminology

Terminology is the set of expressions used by people involved in a specialised activity. Traditional medical terminology contains many ways of saying the same thing (synonyms) and terms that mean different things depending on context (homonyms). Computers cannot cope with ambiguity, and so computerised terminologies or coding systems have been developed to provide explicit identifiers for each term and concept.

![Figure 3a Tabular route-map (Reproduced with permission of the AA)](image1)

![Figure 3b Graphical route-map (Reproduced with permission of Google Maps)](image2)
It can be argued that ordinary users do not need to become involved in the details of clinical terminology and that this should be left to the experts. This may be partly true, but shared terminology is an inescapable aspect of shared understanding. Users have to agree what terms to use.

A large clinical terminology such as SNOMED-CT is analogous to a large atlas or a scheme like Google Maps, which is larger than any one person could ever need. Each user simply uses that part that they require for the job in hand. The only precondition is to know how to map-read. Users need to have a basic understanding of clinical terminology.

Development life cycle

The development life cycle is shown in Figure 4 as a simple waterfall model. Our attention here is on the first three or four steps in the life cycle. These processes are iterative and might involve substantial revisions as the work proceeds. The final four stages (implementation, testing, deployment and support) are not discussed further.

2 Process analysis and design

I keep six honest serving men (they taught me all I knew); their names are what and where and when and how and why and who.6

The task in stage 2 of the life cycle is to specify the business processes and corresponding information flow(s) that take advantage of the features of electronic systems. This process includes the development of storyboards, use case descriptions, a glossary, activity diagrams and conceptual class diagrams.

This stage specifies how the system is to work in terms of the business processes, rules and information flows. Tools used include storyboards, use case descriptions, a glossary and UML activity diagrams.

3 Conceptual design specification

The conceptual design specification is the deliverable from stage 3. It is a precise specification of the information to be exchanged in each transaction, in technologically neutral form. It includes details of the terminology to be used. All stakeholders need to share and agree the conceptual design specification. This is the focus of this paper.

The conceptual design specification comprises a set of UML class diagrams, with each element defined, multiplicities specified and explicit data types. The conceptual design specification is the most detailed statement that users should, with help, be able to understand, comment on, review and sign off. It can be part of a contract for the technical work to be done later.

The conceptual design specification should meet the following criteria:

- comprehensive and complete within scope
- stringent, detailed, rigorous and precise
- coherent (fits together) and consistent throughout
- comprehensible – can be understood and reviewed by both users and developers
- composed from reusable elements
- represented in a computer-readable language (XML).

Each use case or transaction can be modelled as a single view into a larger, comprehensive UML model, which can represent a number of different use cases, and can be output as a set of consistent diagrams, hyperlinked documents or XML.

4 Technology-specific design

The same conceptual design can be implemented in different ways by different developers. Furthermore, technologies change and evolve with remarkable speed.
Conceptual designs change much more slowly. The mapping from the conceptual design specification to any specific technology should not involve any changes, either by addition or constraint, to the meaning of the specification. The technology-specific design is the deliverable from stage 4 of the development life cycle, and specifies precisely what is to be implemented, tested, deployed and supported.

Figure 5 shows the relationship between the conceptual design specification and the technology-specific design. Users and developers share the use of the conceptual design specification, but only developers use the technology-specific design. If developers have any doubt as to the meaning of any part of the specification then they need to consult the whole specification (both the technology-specific and conceptual design parts). The conceptual design is the ultimate authority, because this is what the users have approved.

Discussion

I became interested in these problems almost 20 years ago, after discovering first-hand that it was much harder to develop clinical systems for use in several different hospital specialties than for the single specialty of general practice. Initially, I thought that most of these difficulties could be explained by the absence of incentives and scalability problems. These play a part, but the problems of human-to-human communication between users and developers now seems to be more important.

Shared understanding is required at the conceptual level, not at the physical level, because different computer applications never hold data in precisely the same way. The Rosetta Stone, now in the British Museum, represents the same proclamation in three languages, used by the ancient Egyptian priests (hieroglyphic), the court (Greek) and the people (demotic). In our context, the languages are the two languages used internally by each of the systems that need to talk to each other and third, the lingua franca used on the network between them. The physical rendering of each language is different, as it is on the Rosetta Stone, but the meaning, at the conceptual level, needs to be precisely the same.

Others, such as Glushko and McGrath, working in the business sector, have identified much the same problems and a similar way forward, which they call document engineering, combining the techniques of business process, task, document and data analysis. They point out that:

The most basic requirement for two businesses to conduct business is that their business systems interoperate. Interoperability does not require that two systems be identical in design or implementation, only that they can exchange information and use the information they exchange. Interoperability requires that the information being exchanged is conceptually equivalent; once this equivalence is established, transforming different implementations to a common exchange format is a necessary but often trivial thing to do.

The approach set out in this paper is complementary to that of HL7 Version 3 and CEN 13606/OpenEHR, which provide sophisticated ways of handling almost any conceivable healthcare data. Both HL7 V3 and CEN 13606 may be regarded as lingua franca, used to support the interoperability of heterogeneous healthcare systems. Each is based on the principle of constraining or refining a generic reference model. However, HL7 V3 RMIMs (refined message information models) and CEN 13606 archetypes are difficult to understand unless the reader is already familiar with the underlying reference model. The HL7 V3 reference model (RIM) is inherently simpler but less straightforward to use than the CEN 13606 reference model.

Using the same system in more than one specialty is hard

Health care is complex. The UK government recognises more than 50 distinct specialties for doctors, and there are many more specialisations for nursing, scientific, therapeutic and administrative staff. Each group has its own way of looking at the world, reflected in
specialist terminology and procedures for professional certification and development.

Many IT suppliers have developed successful systems for use by one specialty but, despite their best endeavours, few have been able to replicate that success across other specialties. For example, suppliers of GP systems have found it difficult to replicate their success in hospitals. It has been suggested that this is due to more complex workflow, division of labour and patterns of information flow in the hospital,11 but that theory fails to explain why many successful hospital clinical systems, such as those used in renal medicine, maternity or cardiac surgery, have not been adopted by other specialties.

Exchanging records between different systems is hard

It is also difficult to share information between different computer applications within the same specialty because each computer application stores data in a different way and might use different internal codes. This is illustrated by the GP2GP project in England. Patients in England have a lifelong medical record, which follows them when they move from one GP to another. In an ideal world, each patient’s records would be sent electronically from their old practice to the new in a manner that minimises risk and avoids the need to re-enter information.

The GP2GP project set out to do just that. At the time the project was established in 2001, the project’s leaders recognised that it could be either the Holy Grail or a poisoned chalice.12 The jury is still out, but there is optimism that the first live exchanges between two different systems suppliers will be achieved in early 2007.

Document sharing has low barrier to entry

An alternative approach to interoperability is to exchange electronic documents. The central concept of HL7’s CDA (Clinical Document Architecture) is that each message shall include a human-readable representation of its content, which has persistence and can be authenticated. In addition, CDA Release 2 supports structured clinical data, using the HL7 V3 clinical statement model. The barrier to entry is low because all CDA documents can be rendered in a human-readable way, but coded data can also be included to provide a straightforward migration path towards new functionality, such as clinical decision support, audit and analysis.13

HL7 CDA can be used with IHE’s (Integrating the Healthcare Enterprise) Cross-Enterprise Document Sharing (XDS), which allows healthcare documents to be shared between different enterprises (and specialties). The key to XDS is standard metadata in a central registry; the registry keeps an index of each patient’s documents with a link back to each document’s repository. An XDS user creates a virtual patient record on the fly by retrieving documents (letters, results, images and folders) from their repositories in their original form (for example, CDA, PDF, image).14

Technologies such as CDA and XDS can deliver images and narrative documents when and where they are needed. But one of the main benefits of computers is to relieve humans of routine mental work, such as performing innumerable checks to ensure patient safety. This needs unambiguous structured data.

Reverse engineering

Ideally, the conceptual design specification should be prepared before any technology-specific design and implementation work. However, it is often desirable to reverse-engineer existing applications or designs to create a conceptual design specification where none exists, which can be explained to both users and implementers alike.

When reverse-engineering an existing design, particular care must be taken to avoid the temptation of following the implementation pattern. The conceptual design specification needs to use the language of the user, not that of the developer, and be part of a coherent conceptual model of the domain as a whole. Pointers can be provided to technology-specific artifacts to support traceability between deliverables from each stage of the life cycle.

Conclusions

Many of the difficulties met in developing, implementing and integrating healthcare computing systems stem ultimately from difficulties of human-to-human communication between users and developers. If users cannot understand technical specifications they cannot check them properly before most of the development work is done. They can become alienated.

Developers make avoidable errors, leading to delays and cost over-runs. The number of errors is a function of the complexity of the specification, its length and the number of stakeholders involved.
The technical specification needs to be supplemented by stringent conceptual design specifications, using UML diagrams that set out precisely what is required in a technologically neutral way. This must be easy to use and understand. People understand maps and diagrams more easily than long lists or reports.

The conceptual design specification is a set of detailed UML class diagrams, supporting data definitions and terminology. It is platform-independent, comprehensive, stringent, coherent, consistent, composed from reusable elements, and provides a computer-readable rendering in XML.

When the conceptual design specification has been reviewed, checked and signed off by the users and developers, it can be the basis of a contract between them. No extra meaning should be added or subtracted during the subsequent development and implementation stages.

The approach presented here is an extension of conventional software engineering best practice, and is relevant to a broad range of situations involving integration of federated systems, interoperability, data warehousing and data migration from legacy to new.

ACKNOWLEDGEMENT

I am grateful to Dr Ed Conley (The Welsh e-Science Centre, Cardiff University) for a number of suggestions, including the MIC/E acronym.

REFERENCES

2 Medix UK PLC. Survey (Q1066) of doctors' views about the National Programme for IT (NPfIT). November 2006. www.ixdata.com/reports/106620061121.pdf
5 Mead C. Data interchange standards in healthcare IT – computable semantic interoperability: now possible but still difficult, do we really need a better mousetrap?

CONFLICTS OF INTEREST

None.

ADDRESS FOR CORRESPONDENCE

Tim Benson
Centre for Health Informatics (CHIME)
University College London
Archway Campus
Highgate Hill
London N19 5LW
UK
Email: tim.benson@abies.co.uk

Accepted January 2007