Chapter

Open Source Health Systems

Dipak Kalra and David Forslund

The Requirement for Health Software

Computers are now necessary for the delivery of contemporary health care in most countries. Obvious examples of this include digital imaging studies in which a computer is directing the investigation itself and processing the measurement values in order to make them capable of display to a human expert.

Today, however, computer systems play much wider and mission-critical roles in patient-centered clinical care:

- Parts of a patient's health record are captured in a range of diverse clinical applications in many different care settings, increasingly without a duplicate paper record being made of the encounter, test result or prescription.
- The terminology systems and reference data that underpin manual coding activities and clinical applications are usually stored and accessed electronically.
- There are increasing numbers of alerts, electronic prompts and guidelines incorporated within medical systems that direct clinical workflow, warn against out-of-range readings and potential hazards, or advise on care management.
- Demographic systems and regional or national registers are relied upon to ensure consistent patient identification across multiple enterprises, and to operate population health screening and recall programs.
- Computers are used to manage personnel registries and to enforce access control policies within single enterprises and across health regions.

These and other roles for computers will increase considerably across the whole of health care in the near future. A vast set of information systems and components need to collaborate efficiently and reliably in order to deliver a coherent set of such information services to the healthcare workforce. This is a significant challenge for the design of both large and small systems, requiring pre-planned approaches to interoperability across development teams within and between vendors.

Interoperability between systems is important to ensure the following:

- Reliable communication of patient records and other data, to enable safe shared clinical care
- Consistent application functionality and workflow to support evidence-based clinical practice and efficient, joined-up, clinical services
- Access to common information services such as demographic, terminology, guideline and security components.

The challenge is greater still because the user requirements for health software are legitimately diverse across care settings, professions, and specialties, and they inevitably evolve as patterns of care delivery change and as a consequence of innovations in clinical practice. This places requirements on health systems and components to be highly configurable, adaptable, and, if necessary, easily upgraded or replaced.

It is recognized internationally that good (that is, well-liked and well-used) clinical systems often arise through iterative cycles of development and clinical piloting in small-scale settings, and are often driven by locally-recognized needs and championed both by the clinical and by the development teams. These "departmental" systems, of which any large hospital might have from 50 to 100, only occasionally evolve sufficiently to become suitable for widespread use. They are regarded as mixed blessings across the enterprise, for many well-understood reasons, but it cannot be denied that it is these kinds of innovations that have underpinned the global growth and acceptance of clinical computing and helped to shape the visions of larger-scale healthcare systems.

One of the major counter-challenges to adopting diverse and tailored clinical systems is the considerable training need within the clinical workforce in the use of computer applications. Almost universally, there are limited resources allocated to this activity on an ongoing basis, and efforts to build up well-trained teams are confounded by high staff turnover and the general mobility of clinical personnel. The commonly-advocated solution to this is the use of identical systems across a health service, but in fact the major difficulty faced by staff using systems is not about knowing which button to click. Rather, it is about how to interface their routine clinical workflow and spontaneous information needs with the systems that are available. What we really need is not identical but *intuitive* information systems.

Problems with the Current Marketplace

Although intuitive and interoperable clinical systems are what we need in health

care, procurement decisions are rarely based on those values. In particular, the decision-making purchasers of health software are rarely also the intended downstream users of the systems. Thus, the purchasers typically underestimate the need for and the value of interoperability within their enterprise and for communications outside it, lack knowledge of the existence of relevant standards that might be leveraged to ensure that interoperability is built-in, and do not provide adequately for their future needs for adaptability and systems evolution.

Procurement decisions tend to focus on:

- The perceived attraction of meeting global enterprise information system requirements in a single shot
- The particular support of efficient workflow, the inclusion of (statutory) standard data sets and reporting functions to meet internal and external accountability requirements
- The initial cost of the system, although the gross total cost of ownership in the medium term is surprisingly not considered to be so critical
- The perceived scalability of the system and the range of support services offered by the vendor (but often not the projected cost of such support)
- The perceived reliability of the supplier as judged by the company size and turnover, but not particularly by that company's strategic direction within health care
- The perceived safety and respectability of the decision—following conventional practice and avoiding personal risk, not going out on a limb.

A very clear and delightfully-written paper on these and other factors was published by Bleich (1998); it remains a pertinent commentary on the procurement process as it is today.

So, a whole series of market forces and concerns about accountability promote a tendency towards electing a single vendor solution set within any one enterprise or health region. The recent approach adopted in the United Kingdom by the National Health Service is a massive-scale example of this. (For an overview, go to http://www.doh.gov.uk/ipu/whatnew/prognoun.htm.)

In general, this approach to procurement does not encourage the adoption of standards:

- Large vendors do not need them, but can instead use private interfaces internally and leverage their size and position to dictate the interfaces they choose to offer to others.
- Standards compliance is sometimes claimed but in practice often partial,

and rarely verified by the purchaser; DICOM conformance has proved to be a good example of this problem internationally.

- There is no financial incentive for vendors to adopt standards, since the customer requirements for interoperability within a procurement specification are usually quite weak and are not key determinants within the selection process.
- The whole financial model leans toward vendor ownership of the software, and of the strategic direction of its evolution; the private finance initiative (PFI) approach adopted by several countries is perhaps an extreme example.

Limitations in a new enterprise system are often identified some time after its arrival, and the need for additional modules or the inclusion of third-party components has therefore to be retrofitted to the original contract. The need to communicate with other enterprises and external systems seems often to extend beyond the scope originally anticipated. Such post-hoc interoperability is often achieved through expensive custom-made interfaces, which are also expensive to maintain as each system evolves. The cost of these interfaces is typically based on single customer cost recovery, the argument being that every custom-made interface is unique. Any customization or local tailoring of the systems at either end of the interface adds to the costs incurred, creating a pressure to avoid significant localization and thereby rather defeating the object of having adaptable systems.

Vendor lock-in with respect to data is a well-known problem that surprisingly does not seem to be disappearing yet.

The importance of standards in the healthcare arena has been discussed numerous times with clear emphasis on interoperable data standards. However, besides data standards, there is great need for functional standards so that systems can functionally work together rather than just sending data from one system to another, hoping that they can figure out what to do with it. This importance has been treated well in the paper by Forslund and Kilman (Forslund 2000) as well as its impact on the global healthcare management infrastructure. The value of the health record as a way to collaborate on the treatment of a patient has also been discussed. (Kilman, 1997) Unfortunately, commercial pressures frequently work against this need for world-wide interoperable medical records.

As we discuss in the next section, open source has opened avenues of meeting the needs of interoperability and wide scale integration by changing the way that one approaches the software in healthcare.

What Is Open Source?

Open source software is neither new nor radical, but offers a different approach to the provision of information systems that has the potential to harness the best features of the departmental systems described earlier, combining this with some of the perceived value of large-vendor systems.

The term "open source" may have different meanings for different people; for some, it is thought to be synonymous with "public domain." However, open source actually has a formal definition, and software must meet a set of requirements and actually have a license of some type before it can properly be called open source. The motivation for open source is really quite simple. If the source of an application is readily accessible, many programmers can read it, evaluate it, and provide improvements so that the source code can evolve. Many people believe that, because of the large number of people reviewing such software, the quality of the software can be much better than is produced by the traditionally closed model.

The primary issues are the protection of intellectual property and the ability to make money in the process of developing and deploying the software. Contrary to popular views, the open source approach does not have to sacrifice either of these principles. In order to provide intellectual property protection, a formal license is normally required for open source. There are a wide variety of licenses that are used within the larger open source community, but they always involve these principles:

- Free Redistribution: There is no restriction on the distribution of the software in a royalty-free manner
- Source Code: The source code must be included in the distribution or readily made available in some form
- Derived Works: The license must allow for modifications and derived works to be distributed under the same terms as the original software
- Integrity of the Author's Source Code: There must be some provision to protect the integrity of the original authors' code

There are other important principles involved in open source licensing, not discussed here. For more information about the open source process, go to http://www.opensource.org.

There are many different licenses involved with open source, and they tend to fall into two primary camps. Perhaps the most well known is the General Public License (GPL) from GNU (a recursive acronym for GNU's Not Unix, pronounced

"guh-noo"). The GPL probably provides the greatest protection of the various licenses, but also restricts the ability of a user to commercialize the software without making the derived works also freely available. Another well known one is the Apache Software license (and derivatives) which allows for commercial products to be derived from an open source base without requiring giving back that software to the community. In fact, the Apache Software Foundation is probably one of the most successful and prolific contributors today to the open source effort.

Some times open source software is spoken of as "free software." However, there are two meanings to the term "free," which sometimes confuse the situation. The first is "free," as in no cost. The software is not charged for. This is normally a requirement as indicated above for open source. The more important term is the use of "free," in the sense of liberty, that is, what can be done with the software, in terms of redistribution, modification, and so on. "Free" does not mean non-commercial. There may be very commercial implementations of "free" software. Apache's web server is one of the well known examples. It is used heavily in commercial enterprises and commercial versions of the software can be purchased.

Advantages of Open Source Software

Open source software is frequently written initially within a small-scale and close working partnership of health care domain experts and systems developers. It is therefore very likely to be well tailored to the functional requirements of its specifying community. More importantly, because these are domain-specific specialized software components, they will have been developed on the assumption that they will in practice be used alongside other components that have not been developed by that group, and eventually in multiple and diverse settings. In contrast to the typical large-vendor approach, interoperability and standards compliance are therefore critical to the success of the open source software.

Open source software is usually not complemented by commercially-available support services (from the development team or from anyone else) to customise and adapt the software for individual settings. The normal assumption is that user sites will be able to adapt the components themselves and be able to plug the component into any reasonably anticipated computing environment. Open source components are therefore more likely to interoperate with other standards-based components with minimal interface configuration. They are often (but not invariably) quite readily adapted to local contexts—at least, it is normally

transparent to the local site how they can do it. The corollary, though, is that systems wanting turnkey solutions sometimes shy away from open source solutions, since they may require more customization. This could be viewed as an opportunity, however, for a system integrator.

Open source software is often a minority or niche player within its marketplace. Even Linux began that way. Functional excellence, innovation, and evolution have to be actively fostered within the development community to ensure that the component or system has credibility and remains useful in the medium and longer terms. Since open source teams rarely have a successor product they wish to "market," built-in obsolescence is largely unknown. This is particularly true at the moment, when open source systems have to prove themselves to be better than their nearest commercial rivals in order to be treated as an equal to them.

While usually not seeking to provide a comprehensive solution set, open source teams often make use of other open source components, creating a standards-based and highly flexible and adaptable deployment environment. However, there are some very large open source projects. For example, the U.S. Veterans Administration has historically adopted an open source approach to the systems used throughout its US hospitals and clinics.

As reflected by the definition of open source, there are two important characteristics of open source software that underpin the approach

First, the source code for the software is accessible:

- It is available for scrutiny, for peer review by the wider development community, providing a kind of quality assurance other than the black-box testing of software, which is the norm.
- It can be held by any user site as a form of protection against the original team folding: the user is empowered to take over the software's evolution or to commission this from another source.
- Vendor lock-in is practically impossible.
- Other component developers can understand in detail how they might best interface with it or incorporate it into other components and systems.

Second, the compiled, executable software is usually available at no cost or at a nominal cost:

- Potential user sites can pilot the software quite extensively before making a final decision on its adoption.

- It can be adopted for a lower capital cost and probably maintained at a lower ongoing cost of ownership.
- The costs are based on recurring services rather than on recovering the full development costs from each client.
- Since open source often makes use of and interoperates well with other open source software, the cost benefits of the approach tend to propagate across the overall enterprise information technology (IT) solutions.

Most importantly, the open source software is often nurtured within an active community of end users in a wide range of settings. Clearly, if the user community providing input on the requirements and feedback on the components is extensive, the component will evolve functional breadth and adaptability, and more comprehensive testing, making it suited to a diversity of settings.

The source code is often critiqued and contributed to by a number of voluntary teams and individuals, sometimes at an international level. An extensive and distributed development team can offer the benefits of a wider range of engineering experience, richer quality assurance and testing facilities and the capacity to extend the development beyond the potential of the originating team. Very often, a site will elect to adapt a component in order to make it more suited to their environment, offering such adaptation back to the community as a functional enhancement or improvement. Other groups might develop interfaces to other components, new database bindings, extensions to a user interface, etc. It is by this means that the component or system can evolve in the directions that are needed by its user communities, thereby remaining relevant and usable over long periods of time.

The technical capacity of a distributed set of participants, working voluntarily and often with no dedicated budget for the collaboration, does not inherently guarantee anything better than could be achieved by a single vendor employing staff with equivalent expertise. It is the enthusiasm, energy, and the close, almost pioneering, collaboration of users and developers that creates an environment that is very difficult to reproduce commercially. Rapid iterative cycles of development and *in-vivo* evaluation can replicate the best features of small-scale departmental systems but complement it with the technical rigour of source code peer-review and the potentially rapid broadening of the community by making it freely available.

Adopting an Open Source Solution

It would be wrong to imply that the case for adopting open source software is

overwhelming and without risk. There are a number of genuine concerns that healthcare services and organizations (or their IT departments) have about using such software. Some of these are common perceptions and misconceptions, but others reflect the weaknesses inherent in the open source approach as it is largely realized today.

Many open source projects have limited financial backing. The components might have been written by a team for their own enterprise; once they are completed to the extent required by that organization, the team might not be in a position to continue to evolve the software to incorporate requirements and contributions from other settings. Because of the initial small scale, the project might not have a well-documented architectural approach, but rather one created on-the-fly by the developers. The whole project might have arisen from a time-limited grant, such as funded research, with little assurance that the group will continue to exist after that project has finished. Since the work is largely fuelled by enthusiasm, there is always a chance that key members of the original team, or the team as a whole, will move on to new areas of interest. A rare but conceivable risk is that the group will decide to redirect their efforts in favour of a commercial successor to the open source software.

Vendors with a reasonably successful system and/or a positive branding image in the health sector will have sound business drivers to stay in the field, to recruit successor staff to maintain teams, to build against a well-defined architecture, and to ensure business continuity for their clients. Open source groups might have a moral commitment to this, but usually lack a business model that enables them to provide any long-term assurance of presence or dedication to the components they have developed. It would be wrong to imagine, however, that commerce is without equivalent risks: changes in market fortune, take-overs, or changes in high-level corporate strategy, can lead to products being discontinued or to companies folding. Major players in the healthcare computing market have been known to abandon this sector altogether, leaving significant problems for their user base.

Both proprietary and open source paradigms share a common risk for the customer: that the system provider might stop supporting or evolving software that has come to play an important role in the user's healthcare enterprise. Open source groups might appear at first sight to be "flakier" in this regard, usually being smaller teams with lower revenues deriving directly or indirectly from the developments. However, open source projects inherently offer some insurance against the disappearance of the originating team: the source code is available and there may be other groups already familiar with it and participating in its ongoing

development. At worst, migration to a successor product can be informed and thereby made safer by having the source code. The relative risks of each approach are therefore more difficult to quantify and to evaluate formally. As a result, the case for open source is partly a subjective one.

Open source projects have a moral commitment to quality, but frequently lack the kind of accountability that is commonplace in commercial contracts for health software. Open source components usually carry disclaimers and limitations that would prevent litigation should the team prove to have been negligent or the software not fit for purpose. For this reason, commercial vendors are sometimes reluctant to invest the person-time needed to understand code within an open source component or to acquire the in-house expertise necessary to reduce the risk they might incur in using that component. Still, most consumer shrink-wrapped software also carries a disclaimer that is quite extensive.

Another area of concern more open to evaluation is the engineering quality of the components. In particular, there may be doubts about the application of quality assurance measures that ought to be used during software development and testing. There are often concerns about the diversity, quality, and coherence of contributions that may be arising from a distributed (and somewhat unregulated) pool of development volunteers. This is countered by the open source movement with the argument that the developers involved are often "commercial-grade" (i.e., they also work or have worked for commercial software companies), and that the originating groups do regulate and eventually adjudicate the inclusion of code from other collaborators. The general experience of open source projects is that contributions are usually of very high quality and provide much-valued enhancement to the core component. Furthermore, the components are often tested quite widely in different settings soon after release—a kind of wide-area *beta* testing phase with full awareness of this role, while commercial customers sometimes find themselves hapless *beta* testers of software they thought was complete (i.e., an unintentional reference site). Basically, open source projects benefit from a broad, informal code review process. This code review may be far more comprehensive than is usually done even in the finest commercial software establishments. Although not formally certified as higher quality, this process can be very rigorous (although formal software engineering processes are by no means universal).

All of this presumes that the open source developments *are* being widely deployed, tested, or extended by collaborating teams across the globe. In the early stages, which might last for some years, there may be a significant dependence upon the original development team. There may be few external parties who can

genuinely support and extend the code base, because it is too complicated to understand without significant investment of time and skills, because it lacks good design documentation, or because in its early releases it may be too rudimentary for what is needed by most sites for anyone to invest in adopting the code and extending it to meet local functional requirements. There is a critical scale of interest and participation beyond which technical support will keep the component going, and ensure the benefits described earlier can be realized. Below this critical scale, the founding group may have to continue solo development at a pace they can afford and manage, even if a wider user community is impatient to see it develop more rapidly.

Limited overall resources for an open source activity may mean that there is a lack of advisory support for the software, and frequently paucity of trainers and of training materials. In their absence, however, such support as can be managed by the development teams and core user communities is usually freely given (for example, through discussion lists), and of high quality. While such support services cannot be assured or specified in a service-level agreement, the authors are very familiar with the alternative: of commercial suppliers providing expensive contracted 24-hour helpline support that is delivered by poorly-trained operatives and leaves its customers very frustrated.

Adopting an open source component does not necessarily imply that an enterprise has to go on to incorporate other open source components. However, in the present marketplace, vendor products and open source products do not always mix and match seamlessly. Polarized positions are more common to find than complementary ones. This means that few commercial health software systems directly interoperate with open source components. In fact, most commercial software only interacts with vendor-specified third party components, or requires an expensive interface. Inevitably, for financial and perhaps evangelistic reasons, open source components tend to have been tested alongside other open source products. This clearly brings several benefits, including the ability to design a solution set that is entirely free of license costs. However, this may at times also mean that it is difficult to adopt a particular well developed component, because the complementary open source components necessary to build an overall solution might not be available or exist only at a lower level of maturity.

When weighing the various concerns discussed within this section, it is clear that the open source approach has some counter-arguments against its major perceived weaknesses. In the end, a somewhat subjective value-judgement has to be made when attempting to evaluate the overall relative risks of adopting commercial versus open source software.

Considerations for the Principal Software Development Team

Thus far, we have focused on the strengths and weaknesses of the open source approach from the perspective of a potential user or adopter. Such views have generated the popular vision of open source software and projects. It is the kind of vision that will appeal to some development teams and not to others. Any team or organization needs to consider a range of factors when deciding if their activity should be made available as open source.

Is the philosophy appropriate?

The team has to be comfortable with code transparency: the source code will be open to peer review and feedback. The general approach to development should have been to adopt public domain interfaces, and to seek conformance to relevant standards. The component(s) will normally have free access, almost always not for profit (at least, not from the software itself).

The team needs to be prepared to host an open development process, using tools that enable collaborative software development such as SourceForge. The responses and contributions will initially be bug fixes, modifications, and enhancements, but will progressively include suggestions for the strategic direction of the software as a whole. The originating team has to be ready and willing to progressively adopt democratic software development and project processes, in essence, to share autonomy.

Is the software (component) suitable?

The team or organization needs to be satisfied that the component or system is fulfilling a role in an area that is not already well served by another open source component or good low-cost commercial products—in short, that it is really needed. It will also be more likely to generate interest if it is intended for use in an area of innovation or rapidly evolving requirements, which will probably be an area of healthcare computing that is less attractive to commercial developers. To stimulate a committed and contributing community (vital to realizing the benefits of open source development), the components should be relevant to a sufficient number of sites and settings, preferably in diverse settings and ideally at an international level.

On a technical level, it will be suitable for open source if it is appropriately componentized to fit within a range of different pre-existing software

environments in different settings. Ideally, sufficient standards should exist (or be in the pipeline) to offer confidence about the interoperability interfaces that have been incorporated.

Could a community be established?

The team needs to have a sense of the scale of community that might be generated around the open source project. At least a few other participating groups will eventually be needed to fulfil each of several different kinds of role:

- Interested users, to contribute ideas and requirements
- Active users to deploy and evaluate the software, and to enlarge the experience base
- Potential demonstrator sites, attracted by the prospect of active involvement in the development process (and perhaps by the low cost of participation)
- Developers offering a peer review of the code
- Developers interested in contributing new code for enhancements and extended functionality
- Other developers, particularly open source, wishing to integrate their components
- Other stakeholders, including research and teaching communities.

Is the code suitable?

In order to be accessible to other online code reviewers, the source code including historic versions needs to be held in a formal repository. Concurrent Versions System (CVS) is the most common tool for configuration management in open source projects. It provides support for a moderately large team to work together on the development of an application with complete preservation of the changes made. (For more information, see www.cvshome.org.) The efforts of SourceForge.net providing a home for open source development with rigorous support for full configuration management are a major help to open source projects. SourceForge provides CVS support as well as mailing lists, announcements, documentation and related tools for managing large projects. Not only is the open source code freely available on their web site at www.SourceForge.net, but the entire development process can be accessed by others in the community. This enables collaboration on a scale unprecedented in software engineering, and allows for people to see what ideas are being worked on by a wide community. In early 2004, SourceForge.net was hosting almost 70,000 open source applications and had over 700,000 registered users. The code, and the participatory framework offered to open source contributors, needs to be

structured in a way that accommodates and organizes feedback responses and also change-manages any new code contributions.

Is the team ready?

The original authoring team has to recognize that niche software will be relevant and comprehensible to a small number of third parties, specifically, those who can actually picture what the intended functionality of such a component ought to be. There needs to be enough documentation for a code reviewer to navigate and understand it (at least to make some sense of it). Accompanying documents and/or slides are essential to enable clinical or other kinds of end user to understand why this component has been produced, how it might best be deployed and used, what requirements it aims to meet, and importantly what the group's immediate plans and intentions are. Installation instructions might also be nice!

Once a decision has been taken to "go open source," some months may be needed to prepare materials and to ready the team members themselves for the launch. Steps to migrate toward open source are detailed in Table 1.

INSERT TABLE 1 ABOUT HERE

The group must be ready and committed to foster an e-community (often using discussion lists) and to respond to inquiries about the project, the software, and the code. For registered open source projects, SourceForge.net hosts such capabilities. Most importantly, the team members must be ready for the code to move forward in new and exciting directions.

Case Study: openEHR

Deciding to Launch an Open Source Endeavour

Two research teams, at the University College London (UCL) in the United Kingdom and Ocean Informatics in Australia, came together in an initiative to use the open source in electronic health record (EHR) development. Both teams had established strong pedigrees in the publication of requirements for EHR interoperability, the design of information architectures to meet these, and proof-of-concept implementations of EHR systems in small-scale demonstrator sites.

Members of the teams, which included one of the authors of this chapter, had worked together previously. In late 1999, we recognized that each team had

achieved valuable new results with some degree of overlap and some complementary strengths, which might helpfully be combined.

Research on generic information models to represent and communicate EHRs has been active within Europe since the early 1990s. The European Union sponsored several multi-national projects, including the Good European Health Record (GEHR) (Ingram, 1995), Synapses (Grimson, Grimson, et al., 1998), and Synergy on the Extranet (SynEx) (Sottile, Ferrara, et al., 1999). In addition, the European Committee for Standardization (or CEN, for Comité Europeen de Normalisation) produced two generations of standards for Electronic Healthcare Record Architecture, ENV12265 (Hurlen, 1995) and ENV13606 (Kay and Marley, 1999). However, uptake of such specifications commercially has been limited for several reasons including the perceived complexity of building robust and scalable systems underpinned by a generic hierarchical model and the apparent success of use-case-specific messages to communicate, for example, laboratory and billing information (using EDIFACT or Health Level Seven version 2).

Our two groups felt that coming together would allow a synthesis of the best features of each approach, including a harmonisation of the two archetype-like techniques that had been developed in Europe (Kalra, 1996, 2003) and Australia (Beale, 2000). A union of developmental efforts would enable more effective use of our scarce research grant resources. Promoting the need for generic and interoperable EHRs under a common banner, and with harmonised specifications, would make the efforts of each party more effective and better coordinated.

We all agreed that active engagement in implementation and clinical evaluation was essential to validate the integrity of the approach and to continue learning cycles to evolve the requirements and refine the specifications.

In particular, we all felt that, in the absence of readily available interoperable (standards-conformant) commercial EHR systems, a good quality open source reference implementation would:

- Have value as a highly-visible proof that the generic information architecture approach is valid and scalable
- Offer a means of seeding new EHR demonstrators and user communities internationally, to promote the approach, to widen the requirements input base and to extend the field of empirical evaluation
- Engage a wider group of developers to complement and supplement the capacity of our small teams
- Provide a software exemplar of how the documented specifications can be implemented, which might be used as another kind of specification,

namely, a kind of Implementation Technology Specification (ITS), to assist other developers making their own versions of an interoperable EHR.

Working Together

The joining together of our two groups entailed many threads of activity. We established a not-for-profit organization, which we agreed to call the openEHR Foundation, and defined its mission statement, vision, and objectives. We largely agreed on our intended approach on access to our future specifications, tools, components, and code, including if and how we want to distinguish non-commercial and commercial users. In principle, we were reluctant to propose any form of charged licensing for the eventual openEHR record server, but felt we might instead seek donations from national or other large-scale enterprises that elected to utilize it. We are still a long way from knowing whether this will prove to be a wise position to take.

The founding partners, UCL and Ocean, have each invested significantly in establishing openEHR. In 2002, we launched our new-look web site (www.openehr.org) and e-discussion forums. We continue working to extend the international community of interest through conferences and other meetings. In this regard, we have been very helpfully supported by the EuroRec Institute (www.eurorect.net) that has frequently promoted our work within its activities.

We worked with lawyers to draft a set of legal instruments, such as memoranda and articles of association, to establish and run the openEHR Foundation, which has been formally registered. To protect the name openEHR as we grow and become better known, we registered the name as a trade mark in several parts of the world. We believe we need to establish a sensible license for our current and anticipated materials, to protect their integrity and to limit our liability, while facilitating their use within the openEHR community and by others. We have produced a copyright notice that is now included on all of our documentation, and a data protection policy for the web site and discussion lists. We are considering adopting the Mozilla license for our open source code, to permit the deployment of openEHR components within other open source or commercial software products.

We actively critiqued our two sets of information models in the light of the implementation and demonstration experience we have each gained, in order to design a new converged comprehensive reference model, an archetype specification, and related models and services. Published as evolving releases on

our web site, these were widely downloaded, stimulating valuable feedback.

We began planning a program of implementation that can take advantage of the skills and existing tools of each site, taking account of the funding and capacity each team as it might vary over the coming years. We are also establishing change control systems to maintain integrity of the development process, in particular for when the contributing team expands beyond the founding pair of teams.

Ready for Take Off

If in early 2000, you had asked us how long it would take to achieve much of the above and to begin writing code, we might have guessed up to two years. With no dedicated funded posts committed full-time to this challenge, progress has been a little slower, made in bursts whenever gaps arose in our "day jobs" and also whenever our Australian colleagues were able to get time and funding to join us in London. Contrary to all the promises of e-working, we have always found that progress really accelerates when we are brainstorming together in the same room.

An active discussion community has been established with over 300 members, which has brought us some very rewarding positive feedback, and also lots of stimulating issues that have challenged us to extend and refine our scope. This has also brought us into contact with other health informatics communities such as clinical guidelines, terminology, ontology, images, and security.

The process of developing appropriately-worded licenses and disclaimers has taken us several iterations, despite good legal support. Trade marking the name openEHR is certainly quite an expensive affair.

Perhaps what has excited us the most has been the launch of important new standardization activities in Health Level Seven (HL7) and the European Committee for Standardization (Comité Européen de Normalisation, or CEN). In HL7, the Structured Documents Technical Committee, the EHR Special Interest Group (SIG) and the Templates SIG have all established rich interfaces with the openEHR work, and involved us as individuals. In Europe, there is currently a major revision of the CEN EHR Communications standard (ENV13606), with significant input from members of our groups. Reciprocally, we have found that the participation in these has provided us with fresh ideas, and a rationale to harmonize the openEHR specifications with those activities—the main reason why we have delayed some of our specification releases during 2003.

We are now at the stage when stability is being reached in these standardization

activities, and we have issued a formal release of the openEHR specifications as the basis for implementation. We have established an Archetecture Review Board to oversee further revision, through a formalized change control process. We are now ready to begin the exciting implementation path, and we are actively seeking collaboration from industry and participation in national EHR demonstrator projects.

While we will inevitably begin in-house, we hope soon to be announcing access to some initial source code, and to begin enticing participation from our many international friends.

Case Study: VistA

Probably the most successful and widespread open source effort in health care began in the U.S. Department of Veterans Affairs (VA) as the Decentralized Hospital Computer System (DHCP) more than 20 years ago. To automate their widely distributed medical centers, VA chose the language MUMPS, from the American National Standards Institute (ANSI), as their implementation framework and created a suite of applications that still sets the standard for managing electronic health records.

Renamed the Veterans Health Information Systems and Technology Architecture (VistA), it remains at the forefront of open healthcare software today. Over the years, versions of it have been adopted by the Indian Health Service as well as the Department of Defense. The software is in the public domain today. More information about it is available at http://hardhats.org and at http://worldvista.org. Recently, this effort has benefited with the release of an open source MUMPS platform known as GT.M, available through http://www.sanchez-gtm.com.

Although not intimately familiar with the details of VistA, we recommend it as probably the most important open source case study in healthcare software. VistA contains probably the largest volume of clinical patient information and thus would be a rich resource to engage in the providing of a medical record directly to patients as envisioned in the Health***e***People effort.

Case Study: OpenEMed

OpenEMed has had a fairly long history. It started as an example of the National Information Infrastructure in the fall of 1993 as part of the internal Sunrise project at Los Alamos, to demonstrate a common infrastructure that would support the use and value of distributed applications to a number of disciplines. The TeleMed

project developed out of this effort, with no initial intent to be open source. The project first succeeded in managing computerized tomography image sets for cases of multi-drug resistant tuberculosis, including queries for images that had the same features. After this work, done with the National Jewish Medical and Research Center in Denver, Colorado, the system evolved into a more general medical record system with the help of the Telemedicine and Advanced Technology Research Center (TATRC) at Fort Detrick, Maryland.

At that time, the system was completely rewritten in Java and participated in pioneering standards within the Object Management Group (OMG) for distributed medical records management set by its Healthcare Domain Taskforce. Java was chosen as the implementation language because it supported rapid prototyping and was an adequate object-oriented language to support the object-oriented approach of OpenEMed. The system was designed around services that were designed to be very flexible in the support of patient identification, terminology discovery, clinical observations, and distributed access control. The standards developed and implemented actually used the original GEHR model as one of its objectives and fully supports that model.

In doing so, we adopted and helped create an open architecture before we adopted the open source model. We wanted the system to be fully interoperable with any system that implemented the same standards. We contributed significantly to the standardization of these core services in the medical record. We had several companies show interest in licensing the software, but found that it really did not make sense for us to enter into a restrictive licensing agreement.

Because mostly public funds were used to develop the software, it made sense to make the software available freely to the public. Thus we adopted our current open source licensing procedure based on the open source license from Berkley Software Design (BSD). We changed the name from TeleMed to OpenEMed to avoid any possible trademark conflicts. We positioned our work as a type of "reference implementation" of the relevant OMG specifications, which we believe is crucial to ensuring that the standards are fully implementable, not just some paper standard. By following this strategy, we have attracted more funds than we could have obtained from licensing, and brought more people to contribute to the software base. We have been able to use OpenEMed far beyond its original medical domain and closer to the original vision we had for the Sunrise project as a whole. Through the open source mechanism we have invited others to participate with the Los Alamos team in developing the software. Most of that participation has occurred from partners in Europe, most notably from the University of Maastricht. The project continues to invite interested participants

both from the commercial and research sectors to participate.

The licensing strategy we use enables us to utilize much of the very significant Apache software project funded as part of the Apache Foundation. Our open source license is compatible with this. It has enabled us to adopt some of the best new Java technology coming out and allowing OpenEMed to continue to be a state of the art development effort. We have demonstrated the ability to create a system which quickly adapts to changes in requirements and in the technology.

An important observation we have made over the development of a number of versions of the OpenEMed software is that open source provides a very solid foundation for our software. We try to have multiple implementations of various services we use (such as databases, object-relational mapping, security, etc.) so that we are not dependent on a single vendor for any of our technology. We have used commercial products in OpenEMed, but they are not needed for most of its functionality. We have found, over and over again, that the support for an actively supported open source project is light-years ahead of the support provided by a commercial vendor for a similar technology. We have found numerous times that bug fixes have come in as little as 30 minutes to about 24 hours, where bug fixes on commercial products with support contracts frequently are never done, even on a next major revision of the software. This is not true of all open source projects, but, because there are so many, it is not hard to find those that provide that kind of support, even it is not from the original developers. Picking a mainstream programming language is a significant help in this area.

The OpenEMed software is primarily a framework rather than a turnkey system, although it has complete implementations of several important areas of health care including a full-blown medical surveillance system and pilots of an immunization registry and a National Electronic Disease Surveillance System (NEDSS) Integrated Data Repository (IDR). The component approach we have adopted has gained interest within the US Federal sector as an example of the value of components in building complex systems. By reconfiguring our components, we are able to solve a number of problems without having to a major rewrite of a lot of code. The components we are using basically compose a service-oriented-architecture which has gained in popularity in recent years. The whole concept of providing reusable medical services is what the entire project is about.

The architecture that was used in OpenEMed has been used in other projects including the large federal Government Computer-based Patient Record (GCPR) project which was seeking to integrate the Indian Health Service, the Department

of Veteran Affairs, and the Department of Defense healthcare systems. In addition, it has been used in Brazil and in the European project called Professionals and Citizens for Integrated Care (PICNIC) (http://picnic.euspirit.org). The service-oriented approach used in OpenEMed is gaining wide acceptance today in the Web Services community and provides a significant data integration framework for many applications.

Currently, OpenEMed is the core architecture of the FIRST project at the City of Hope which is seeking to create a distributed collaborative system for managing clinical protocols. It also is a core component in an effort to provide a comprehensive integration strategy for biosurveillance within the United States.

What Makes an Open Source Project a Success?

Open source has been proven in a lot of areas to be the most robust and economical approach to software development and deployment. This is now widely accepted outside the domain of health care because of the broad success of the Linux operating system and tools like the Apache web server, which is the most commonly used web server in the world today.

We find open source to be a viable business model. For the developer, it can provide a broad base for the code development, substantially lowering the cost of developing of quality software. For the end user, it can provide economies by reducing vendor lock-in and enabling costs to be placed in the service area where they are incurred.

Contrary to the widespread notion that open source reduces the protection of intellectual property, open source can actually provide greater protection for intellectual property by enforcing the acknowledgement of who developed the code by placing the entire process under public scrutiny. However, some open source licenses, such as the GPL, provide more intellectual property protection than others.

In the domain of healthcare informatics, domain knowledge is spread around an enormous number of enterprises, and systems need to meet a diverse set of needs. Open source provides an excellent basis for meeting those needs. It enables a local organization to customize an open source solution to meet their needs rather than relying on specialized commercial solutions, which can be very expensive. The model of physicians sharing knowledge is very similar to the open source software model. The several examples we have provided for case studies also demonstrate this suitability.

End users have quite reasonable protection of their software investment, especially with the high volatility of software companies these days. With open source, they are able to ensure that the capability they have today will still be around tomorrow and not lost in some commercial buy out, and the product removed from the marketplace. This does not totally protect them from rising costs of maintenance, but does enable them to continue their business with reduced risk of code abandonment.

The community nature of open sources generally provides excellent support for upgrades and bug fixes to open source software. If individuals cannot solve a problem, they frequently can find a solution in the broader community. In essence, open source increases the effective size of their development team and support base, and they no longer have to rely on a single vendor for support and maintenance. The cost benefit of this is enormous and, for some projects, is the major motivator.

We also have clearly demonstrated the emphasis on the adoption of open architecture and open standards by open source efforts. This effort to support open standards is clearly driven by the community approach of open source software and the desire to reduce the cost by adopting standards. The architecture typically is open and enables other applications to plug into an infrastructure, again, to reduce costs and risks of a particular software approach. The weakness of some open source efforts is in not adopting an overarching architecture, but rather building it as needed over time. This can lead to difficulties in maintaining the code. This “extreme programming” (or XP) approach can be quite valuable, however, in enabling a software system to adapt to new requirements, as long as it is combined with a fairly rigorous software methodology.

In terms of making healthcare information available to the consumer and the person of primary interest, open source has a major role, because it can enable the consumer to have the tools necessary to understand his healthcare information without being constrained by a vendor which may preclude his ability to track his own personal healthcare information. This should be a major motivator by all people involved in healthcare informatics.

References

Beale T. 2000. The GEHR Archetype System. The Good Electronic Health Record Project, Australia. Available at

http://titanium.dstc.edu.au/gehr/req-design-documents/artchtypes_3.1B.pdf. Last accessed 03/19/2004.

Bleich HL. 1998. Why good hospitals get bad computing. Medinfo 9(2):969-972.

Forslund DW, Kilman DG. 2000. The Impact of the Global, Extensible Electronic Health Record. In: Eder, LB, ed., Managing Healthcare Information Systems with Web-Enabled Technologies. Idea Group Publishing: Hershey, Pennsylvania.

Forslund DW, Joyce E, Burr T, Picard R,Wokoun D, Umland E, Brillman J, Froman P, Koster F. 2004.Setting standards for improved syndromic surveillance.IEEE Eng Med Biol 23(1):65-70.

Grimson J, Grimson W, Berry D, Stephens G, Felton E, Kalra D, Toussaint P, and Weier OW. 1998. A CORBA-based integration of distributed electronic healthcare records using the synapses approach. IEEE Trans Inf Technol Biomed 2(3):124-38.

Hurlen P, ed. 1995. Project Team 1-011. ENV 12265: Electronic Healthcare Record Architecture. CEN TC/251, Brussels.

Ingram D. 1995. The Good European Health Record Project. In: Laires MF, Laderia MJ, Christensen JP, editors. Health in the New Communications Age. Amsterdam: IOS Press, 66-74.

Kalra D. ed. 1996. The Synapses User Requirements and Functional Specification (Part A). In: EU Telematics Application Programme, Brussels; The Synapses Project: Deliverable USER 1.1.1a.

Kalra D. 2003. Clinical Foundations and Information Architecture for the Implementation of a Federated Health Record Service. PhD Thesis. University of London.

Kay S, Marley T, eds. 1999. Project Team 1-026. ENV 13606: EHCR Communications: Part 1 Electronic Healthcare Record Architecture. CEN TC/251, Stockholm.

Kilman DG, Forslund DW. 1997. An International Collaboratory Based on Virtual Patient Records. Communications of the ACM 40*:110*-117.

Sottile PA, Ferrara FM, Grimson W, Kalra D, Scherrer JR. 1999. The holistic healthcare information system. In: Brown P, editor. Proc. Toward an Electronic Health Record Europe '99. CAEHR (Centre for the Advancement of Electronic health Records), 14-17 November 1999, London, UK. p259-266.