A New Model for Open Source Software in Public Health

Evan Bauer, Chief Technology Officer, Collaborative Software Initiative
Version 0.4
19 May 2008

The Collaborative Software Initiative (CSI) brings together like-minded companies to build software applications at a fraction of the cost of traditional methods. CSI introduces a market-changing process that applies open source methodologies to building Collaborative Software. CSI engages the power of community to build project teams and provides the central project management function for developing Collaborative Software, including development, testing and ongoing support for the code. CSI delivers the software to a broader base of customers with commercially supported open source and Software as a Service (SaaS) models. For applications that don't enable competitive advantage or are associated with non-value added activities such as compliance, regulatory reporting, and industry standard communications, Collaborative Software allows customer core team members to provide control and direction over a project while leveraging the efficiencies of using the same software and reducing costs. This paper discusses the application of that model to the software needs of the public health community.

IT and Public Healthcare Delivery

It has been said that: "Health care is vital to all of us some of the time, but public health is vital to all of us all of the time." Public health is the best example of a a non-competitive activity – always win-win – never a zero-sum game. In an environment where all contributions are to the public good, efficiency and effectiveness of delivery are paramount; at the same time financial resources are always constrained. The needs of a community for public health delivery are not correlated with that community's budget for public health.

The demands on public healthcare providers (local, state, and federal) are increasingly complex and critical. Bio-terrorism or an influenza pandemic are threats for which public health agencies would need to respond in conjunction with law enforcement and potentially military counterparts. Increased national and international air travel and air transport make the local containment of many infectious diseases difficult and allow human (as well as animal and vegetable) infection vectors to disburse nearly immediately and make investigation, identification, treatment, and quarantine near-real-time activities. Epidemiology is increasingly working in the present and future, not just the past tense – and where timely, accurate, information and communications is critical.

Public health provider IT has been a sector where traditional proprietary commercial software, federal government provided software, and academic open source software projects all play a role, with no dominant model for the creation, enhancement, and support of application systems.

Challenges in Public Health Software Design and Development

Many of the requirements for software to support public healthcare delivery are common across providing agencies, but wide variations in

UT-NEDSS Project

With the goal to protect citizens' health, UT-NEDSS project's objectives include:

- Protect the health of people by detecting and preventing disease
- Detect disease outbreaks and bioterrorism attacks more rapidly
- Enhance the timeliness and quality of information
- Facilitate the electronic transfer of appropriate information more efficiently

The UT-NEDSS project is the localization of the federal project and promotes the efficient, integrated and interoperable disease surveillance systems at federal, state and local levels. The vision for the project is to move from DOS-based programs and fax machines to an integrated surveillance system that can transfer appropriate public health, laboratory and clinical data efficiently and securely over the Internet while gathering scope, scale, demographics, organizational structure, work-flow, and technical support resources provide challenges to the design of any system intended to meet the needs of the broad user base. The integration requirements for any single application will vary based on the existing or required installed base of related and supporting applications; at the same time new systems should not restrict or inhibit future choices of applications. The recurring problem in commercial public health software has been a function of the interaction of two relatively independent factors:

1. Most of the commercial packages are derived from an original custom development project for a large, well-funded public health agency.
2. There is a relatively small number of such agencies and most small to medium-sized agencies generally have very tight IT budgets.

Too often, the result is software whose design was created on a top-down basis – for example, meeting federal requirements while not meeting needs of delivery organizations. Those that do meet the needs of providers (versus regulators and policymakers) have features and deployment scale and costs appropriate only to large well-funded jurisdictions. The lack of a traditional commercial market has often meant that software too frequently becomes stale, dependent on technical platforms that become obsolete or lack updates to meet the rapidly changing requirements of public health agencies. User agencies are too often asked to individually fully fund needed feature development that is then resold to other agencies.

One recurring problem in most traditional software development for public health is that software design is done by developers, who may have some exposure or access to public health professionals (the doctors, epidemiologists, nurses, and administrators), but typically they will create requirements and specification documents that will then be the basis for implementation – the success of which won't be known until many months or years of development have passed. Even if the translation from detailed design to working code is faithful, it memorializes needs defined long before the software is written or used.

Challenges to Open Source Public Health

Open source software is, as a result an increasingly attractive alternative to commercial packages for use by public health agencies. An important and outstanding question is what makes a successful community to create and support large-scale open source software projects to meet the needs of public health agencies?

Historically the most successful open source projects have been in the creation of infrastructure software: operating systems (Linux, BSD), networking components (Apache, BIND, OpenSSH), and database management systems (MySQL and Postgresql), as well as development tools from editing environments (Eclipse, NetBeans), computer language runtimes and compilers (Perl, PHP, Ruby) and frameworks (Spring, Rails, Zend). This is the software that enables the

and analyzing information quickly and accurately in order to identify and track emerging infectious diseases and bioterrorism attacks. Additionally, the UT-NEDSS project reaches beyond disease surveillance to address epidemiological, biostatistical and health services issues.

Impact
When deployed, UT-NEDSS project will directly contribute to the prevention of sickness and death by effectively collecting, identifying, tracking and trending information gathered about infectious diseases and bioterrorism attacks.

CSI Collaboration Toolset

CSI's software communities interact using a combination of web tools implemented with our partner Collab.net, extending their commercial open source service offering. These include:

- Project Tracking
- Requirements Definition and Tracking
- Document Management
- Source Code Management
- Forums with integrated mailing lists
- Wiki for discussion and documentation

Internet and is used to build both in-house, commercial and open source applications across industries and public sector organizations from Google and Amazon to banking and CRM systems.

CSI Collaborative Project Approach

The Collaborative Software Initiative creates vertical application open source software built around a founding core team made up of both practicing professional subject matter experts (SMEs) with professional software developers. A project methodology that draws on Lean Software Development and open source best practices supported by web-based discussion, specification, project tracking, code management, build, and testing facilities provides for efficient, geographically dispersed development. The projects are later opened to broader participation, with CSI and its partners providing enterprise support and training services to those user organizations who want them.

The Core Team

The CSI approach to software communities and projects is proving to be particularly valuable in the creation of needed public health delivery software. First and foremost, it neither expects developers to know the needs of doctors, nurses, and epidemiologists better than those practitioners do; nor does it expect that health professionals have the skills to design great software. Instead, CSI works to create a core development team that partners practicing health professionals with a paid team of experienced open source developers with explicit joint ownership of the success of the project.

Using a combination of web-based collaboration tools (see sidebar), regular conference calls with focused agendas, and periodic (typically quarterly) on-site meetings, the core team is a diverse community that provides a uniquely effective means of creating very usable, very high quality software. CSI and the public health agencies each provide an experienced project or product manager 100% dedicated to the project who orchestrate a team of full-time and part-time developers working with practicing health professionals who dedicate a few hours per week each. There is an appropriate diversity of skills on both sides – development, database, user interface, and quality assurance among the developers and nurses, doctors, and epidemiologists from an appropriate range of agencies among the subject matter experts.

Steering Committee Input and Oversight

Along with the project core team, a steering committee can be used to provide oversight and perspective on larger goals of the sponsoring agencies and organizations. Senior management from CSI and the agencies, knowledgeable academics, as well as senior SMEs from related organizations and projects provide perspective and a sanity check to the project managers and CSI.

The Development Methodology

CSI employs variations on a combination of open source practices and Lean Software Development to produce high-quality application software both quickly and cost-effectively. Web-based discussion forums linked to mailing lists, a wiki for design discussions and documentation, peer-discussed division of labor, and an “early and often” release schedule all come from our open source heritage. Wherever possible CSI develops with open source tools and on open source platforms, contributing back to the communities producing the tools we use. The project starts with several visioning and end-point definition sessions to define the overall goals of the project (the high-level roadmap) and to select the business standards and industry specifications the project will implement. Detailed design and specification is deferred to the first iteration of each release – CSI projects produce working systems, not reams of obsolete requirements definitions.

Time is then spent forming the team, creating a common initial understanding of the processes and web tools, and visiting the SMEs in their work environments where the developers get an initial context of where the systems will be used.

SME team members are educated in the use of User Stories as the basis for creating a backlog of requirements that are then estimated, prioritized, designed, and implemented. CSI practices test-driven development and uses continuous integration to automatically execute unit, integration, and user acceptance tests to both continuously improve quality and to provide a place for the core team SMEs to see and try new features between releases and provide immediate feedback on correctness and usability of the implementation. Where design, tool, or implementation alternatives need to be explored, developers split-up and perform set-based design dives to explore alternatives, then present their results back to the core team for decision making. Software releases come every few months, iterations within those releases every few weeks – design and implementation errors and misconceptions don't get much time to live.

Results

The first CSI project in the area of public health delivery went from its initiation meeting to delivery of release 1 software for production parallel use in less than five months (this is in line with CSI's experience in other industries.) In tracking the dynamic project roadmap against the end-point schedule and budget, our core team members get the software they need when they expect it and at the agreed upon cost. The core team makes the decision as to when the project and software is sufficiently mature for open source release – opening the project to the wider community who can both benefit from and contribute to the work. CSI provides professional collaboration, support services and high quality development to that wider community of users, providing the assurance of a sustainable, software asset for the public health community as a whole.