

Why inCounter is Open Source and Sharing is Good!

By James McMillan and Tom Carlson

The Problem (and a Caveat)

If it would please the court, in our opening statement, we want to definitively state that we are not against capitalism. We have worked for the past decade at the NCSC to facilitate private solutions to court's problems. Automated case management systems, for example, are many times better now than they were in 1990 due to the investment and efforts of the private sector. Private corporations have been able to bring professional development capabilities and other valuable ideas from other industries. But we are at a crossroads in the automation of the courts. As has been stated many times it is clear that E-filing is the next big step. But E-filing has many barriers to adoption. Well-defined legal, procedural, and technical issues are all affected by E-filing because the industry is continually changing the method, format, and capabilities of information presentation. As repeated in the inCounter white paper (<http://www.court-tech.org/incounter/>), one key barrier for a court is the fear of vendor exclusivity or "lock-in". The fear is rooted in a core judicial value dating from the dawn of civilization that courts are equal and non-biased arbiters, open to all. If a court signs an exclusive contract, or otherwise locks itself into a proprietary system, how can that be perceived as being open no matter what provisions are enumerated?

The Solution (Just One of Many)

We think that there is a way (but certainly not the only one) that can help the courts to be perceived as open and equal while still allowing service providers to openly, and profitably, interact with the judiciary. The inCounter E-filing project provides an Open Source system that courts and private vendors can use and extend to meet their needs. We recognize that there are a considerable numbers of software developers working in the courts, government, and in the private sector. Today, in financially difficult times it would be wise to leverage the power of experts by working together to solve common problems. In this article we will explore the ideas behind Open Source software and our hope that we can support a collaborative development for the good of all.

The time for talk has passed. There are over 200 courts in the USA now accepting E-filed documents for specific case types. The time has come to write code and test the standards and theories that have been discussed and developed over the past decade (JEDDI, Legal-XML, and others). Courts and vendors also have the choice of working together or working separately. We certainly understand the attractiveness of working separately... after all, one does not have to ask for permission to design and write the code the way that you want to do it. But let's explore reasons why one might want to help us in a collaborative effort.

First, we acknowledge that there are much better designers and programmers in the courts and private sector. Excellent case management and electronic filing systems have been

developed for all types of courts. Some other good examples are the E-filing system that was created in Topeka, Kansas for the Shawnee County District Court (<http://www.shawneecourt.org/>); the “smart document” system developed in Toronto, Canada; and E-filing systems developed by Austria, Finland, and Singapore. But it should also be recognized that many courts do not have the resources to develop their own system. Therefore, collaboration of government, private, and even academic developers using free and Open Source software could facilitate the creation of key programs to enable standardized E-filing interfaces for all courts.

Unfortunately, at this time we do not anticipate that courts will be able to standardized data tags and values for legal documents across jurisdictions. Representatives of one prominent E-filing vendor, Lexis-Nexis CourtLink, told us that they currently have over 1.5 million legal forms collected in their databases. Clearly, mapping the data and values from this number of legal forms to the hundreds if not thousands of case and document management systems will have to be done on a court-by-court basis. Facing these realities, the best solution would be to develop tools to facilitate the creation of court jurisdiction, location, and process specific electronic forms and supporting XML schema to control and validate what types of information is put into those forms. (For a good technical explanation of XML schema go to: <http://www-106.ibm.com/developerworks/xml/library/x-sbsch.html?dwzone=xml>).

Open Source Can Solve

The Open Source approach has already been proven successful in the creation of the Internet and in other business and academic pursuits. Much has been written about the Open Source philosophy. In a vast, but accurate, oversimplification, Open Source can be encapsulated in the phrase “Sharing is good!” In the inCounter White Paper we noted Eric Raymond’s paper titled “The Cathedral and the Bazaar” (<http://tuxedo.org/~esr/writings/cathedral-bazaar/>). In that landmark paper Mr. Raymond makes many interesting points about Open Source software development and in particular the Linux operating system. Some noteworthy quotes are:

- “Linux is subversive. Who would have thought even five years ago (1991) that a world-class operating system could coalesce as if by magic out of part-time hacking by several thousand developers scattered all over the planet, connected only by the tenuous strands of the Internet?”
- “The source-sharing tradition of the Unix world has always been friendly to code reuse...” “The Linux world has taken this tradition nearly to its technological limit; it has terabytes of Open Sources generally available. So spending time looking for some else's almost-good-enough is more likely to give you good results in the Linux world than anywhere else.”
- “In fact, I think Linus's” (Torvalds the creator of Linux) “cleverest and most consequential hack was not the construction of the Linux kernel itself, but rather his invention of the Linux development model. When I expressed this opinion in his presence once, he smiled and quietly repeated something he has often said: ‘‘I'm basically a very lazy person who likes to get credit for things other people

- actually do. ‘Lazy like a fox. Or, as Robert Heinlein famously wrote of one of his characters, too lazy to fail.’”
- “Early and frequent releases are a critical part of the Linux development model. Most developers (including me) used to believe this was bad policy for larger than trivial projects, because early versions are almost by definition buggy versions and you don't want to wear out the patience of your users.”
 - “In speaking on program development complexity: ‘The underlying problem here is a mismatch between the tester's and the developer's mental models of the program; the tester, on the outside looking in, and the developer on the inside looking out. In closed-source development they're both stuck in these roles, and tend to talk past each other and find each other deeply frustrating. Open Source development breaks this bind, making it far easier for tester and developer to develop a shared representation grounded in the actual source code and to communicate effectively about it.’”
 - “Even at a higher level of design, it can be very valuable to have lots of co-developers random-walking through the design space near your product. Consider the way a puddle of water finds a drain, or better yet how ants find food: exploration essentially by diffusion, followed by exploitation mediated by a scalable communication mechanism.”
 - “It's fairly clear that one cannot code from the ground up in bazaar style. One can test, debug and improve in bazaar style, but it would be very hard to *originate* a project in bazaar mode. Linus didn't try it. I didn't either. Your nascent developer community needs to have something runnable and testable to play with.”

Where Does inCounter Fit In?

To summarize the ideas stated above, the Open Source development concept marshals resources; enables programming code sharing and reuse; addresses system and development complexity; and facilitates a better testing environment. The final point notes the importance of someone actually starting a project. Someone needs to create the initial code that others can then review, use, modify, and extend. That is the role that we hope to play by creating inCounter, a basic system that can start the shared development of critical E-filing elements.

Open Source development has proven that good communication, supported by Internet tools such as websites, listserves, bulletin boards and code repositories, is the key to solving the problem. It is important that E-filing systems allow for open communications with all types of data capture systems. Government programmers, commercial service providers, and even private parties may build some of these modules. Legal-XML has and continues to do excellent work on creating foundational data description standards. However, as Eric Raymond notes, one must “facilitate standards by testing”. To be effective, E-filing systems must concurrently support commercial mission critical programs for Document creation and production, Case Management, Database Management, and Document Management within the court and thus cannot have licensing restrictions on the use of the software with these systems. Open Source

empowers developers to work on true value added systems and services that deal with information creation, organization, and complexity. For additional insight into this concept Eric Raymond also wrote *Homesteading the Noosphere* (see definition below) describes various zealotry and philosophical views of Open Source and free software development <http://tuxedo.org/~esr/writings/cathedral-bazaar/homesteading/>

Another critical element of Open software development is the licensing provisions and intellectual property claims that are attached to the programming code. The inCounter system was given what is basically a free license for government and commercial developers to use. We just asked for credit if the software was useful. While everyone likes accolades, what we really want is for others to add to the software and share their programs if it will help improve the system.

But That's Not How We Do Things!

No, it's not. And why not create a comprehensive E-filing design and plan? Isn't this what we are taught to do? This is the "constructing a building" argument. That argument says that you can't start construction until the plans are drawn. To state the obvious, software is not a building. It is possible to create the electronic equivalent of the penthouse suite before constructing the rest of the building. Mr. Clay Shirkey in his paper "In Praise of Evolvable Systems" (<http://www.shirky.com/OpenSource/evolve.html>) states that "... had the Web been a strong and well-designed entity from its inception, it would have gone nowhere."

This concept may be useful to the courts. The courts jealously guard their independence and often this is manifested in different procedures being implemented in different courts in the same jurisdiction. Court administration has long been charitably characterized as organized disorganization. The basic structure of laws and rules are the same, but the individual court processes differ. Similarly, if software development for courts could reflect this organizational structure, it might better meet the needs of the courts. The better approach to general development and implementation of E-filing is to not create a single standardized design, but rather develop a set of tools to facilitate the courts adoption of E-filing. Even if inCounter is not ultimately one of those tools, we hope that the ideas behind its creation will succeed and a better successor will be developed to replace it.

N/ The term `noosphere' is an obscure term of art in philosophy. It is pronounced KNOW-uh-sfeer (two o-sounds, one long and stressed, one short and unstressed tending towards schwa). If one is being excruciatingly correct about one's orthography, the term is properly spelled with a dieresis over the second `o' to mark it as a separate vowel.

Rules for Collaboration....

Is it going to be messy? Will projects fail? The answers to these questions are “yes.” But that has been true of any systems development scheme. More importantly, many of us work in and with courts because we believe in the Rule of Law and also in the idea of public service. Open Source software collaborative development speaks to this belief. We think that software licensing and intellectual property claims that restrict the development of electronic services for the courts and legal system have not been and will not be helpful. But we also think that commercial value-added services are also an important component and that clear delineation between the commercial and public interests are possible.

In short, we should be sharing. And we'll go first. Here's inCounter, to do with what you will.

Sharing is good.

Eric Raymond's Laws from "*The Cathedral and the Bazaar*"
<http://www.tuxedo.org/~esr/writings/cathedral-bazaar/>

1. Every good work of software starts by scratching a developer's personal itch.
2. Good programmers know what to write. Great ones know what to rewrite (and reuse).
3. "Plan to throw one away; you will, anyhow." (Fred Brooks, *The Mythical Man-Month*, Chapter 11)
4. If you have the right attitude, interesting problems will find you.
5. When you lose interest in a program, your last duty to it is to hand it off to a competent successor.
6. Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging.
7. Release early. Release often. And listen to your customers.
8. Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone.
9. Smart data structures and dumb code works a lot better than the other way around.
10. If you treat your beta-testers as if they're your most valuable resource, they will respond by becoming your most valuable resource.
11. The next best thing to having good ideas is recognizing good ideas from your users. Sometimes the latter is better.
12. Often, the most striking and innovative solutions come from realizing that your concept of the problem was wrong.
13. "Perfection (in design) is achieved not when there is nothing more to add, but rather when there is nothing more to take away."
14. Any tool should be useful in the expected way, but a truly great tool lends itself to uses you never expected.
15. When writing gateway software of any kind, take pains to disturb the data stream as little as possible—and *never* throw away information unless the recipient forces you to!
16. When your language is nowhere near Turing-complete, syntactic sugar can be your friend.
17. A security system is only as secure as its secret. Beware of pseudo-secrets.
18. To solve an interesting problem, start by finding a problem that is interesting to you.
19. Provided the development coordinator has a communications medium at least as good as the Internet, and knows how to lead without coercion, many heads are inevitably better than one.