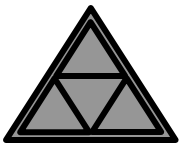
Three Pillars of Social Source

Connecting the Nonprofit Technology Sector



Gideon Rosenblatt
Executive Director, ONE/Northwest
March 2005

This document and related discussions are available at http://www.movementasnetwork.org

Introduction

In the world of scarce resources plaguing the nonprofit technology sector, we currently suffer from a conflation of roles. This paper outlines three functional roles that are essential for a vibrant nonprofit technology sector. These "three pillars" include the "application developer", the "application integrator" and the "application hoster." Drawing clearer distinctions between these roles will help nonprofit technology assistance providers clarify their organizational missions, which will reduce competitive overlap and pave the way for improved collaboration between organizations. These steps are absolutely necessary if we are to evolve the nonprofit technology sector into a more integrated "social source" movement dedicated to empowering the agents of social service and social change throughout our societies.

The ideas in this paper echo a similar analysis of functional roles from an earlier paper on the environmental movement called Movement as Network, which argued that:

> *"The environmental movement is not just some vague concept, but an actual entity. It is a network, made up of very real interconnections between people and organizations; a networked whole that is greater than the sum of its individual parts."*

In much the same way, the nonprofit technology sector must also come to see itself as something greater than the sum of its individual parts, for it too is a network - a network with the potential to become a movement. What holds it back from its potential as a movement is the lack of a unifying mission. Yes, the nonprofit technology sector does exist to serve the technology needs of the nonprofit community. But that in itself is not unique. Microsoft plays this same role every time a nonprofit organization uses Word to write a letter or Excel to create a spreadsheet. What is it that makes the nonprofit technology sector greater than the sum of its parts? What is its vision - its reason for existence? What, in short, would turn it from a sector into a movement?

As information technology evolves, it moves from the general to the specific; from generic infrastructure, operating systems and tools to general productivity applications and eventually to applications that are specific to the needs of particular classes of customers. It is in this last phase where the technology becomes deeply fused with strategy and has its most transformative effect on the organization. We in the nonprofit technology sector are just entering that era.

The case for open source software and the broader notion of a public commons for communications infrastructure have been well made by other authors. Much of the focus on open source within the nonprofit technology sector is still largely on broader, more generic tools. In September 2003, Jonathan Peizer at the Open Society Institute wrote an article called Realizing The Promise of Open Source in the

Non-Profit Sector, in which he made the case for using the open source model to build and deliver applications that are specific to nonprofit organizations and thus unlikely to be delivered by private sector software vendors. This idea of marrying open source software development with social service and social change applications is what is meant by the term "social source." As technology evolves and nonprofit organizations become more technology-savvy, the application-specific needs of the nonprofit sector are becoming increasingly clear. I believe that **building and delivering a social source commons aimed at addressing these needs could be the unifying vision that shifts the nonprofit technology sector into something more resembling an actual movement.**

Our practical experience implementing websites using Plone, a leading open-source content management system for our clients at ONE/Northwest (the organization I work for) has convinced me of the merits of Peizer's case for open source. In less than a year of utilizing this open source content management system, we have saved Pacific Northwest environmental organizations well over a half million dollars and are providing them with a powerful publishing tool that is piggy-backing on the innovations of hundreds of developers. There is something extremely compelling at the thought of leveraging these same forces to deliver a much wider range of applications specifically targeted at the requirements of social service and social change institutions.

Today, however, demand for a social source commons is simply too weak to bring it about. For the vast majority of nonprofit organizations, the issue of open source software – or for that matter any kind of software – is an extremely small part of their overall mission. **The social source commons, like any commons, involves investments that hugely benefit participating organizations, but that are beyond the reach of any one organization on its own.** Our nonprofit clients are unlikely to self-organize to make these investments. To fill this community need, the needs of individual organizations need to be aggregated into a center of gravity with the power to influence how software gets built. This is the rallying call that shifts the nonprofit technology sector into a social source movement. But to do so, will require significant restructuring of how technology is developed and how it is delivered to the nonprofit sector. That is the focus of this paper.

Introducing…the Three Pillars

In the article cited above, Jonathan Peizer made a case for a separation of duties within the nonprofit technology sector:

> *"One issue that people not working in the technology space often fail to appreciate is that the technical discipline consists of a number of vertical specialties just as other disciplines do. Technical support and training is a very different animal then software development requiring different skill and mindsets."*

Understanding the differences between these specialties is the first step in making a case for a clear separation of duties between them. From there the focus can shift to understanding how these entities might come together in a more integrated network of collaborators – a movement as network dedicated to building a social source commons.

Within the nonprofit technology sector, there are three core functional roles with distinct skills, personality types and sustainable business models:

Application Developers

Application developers focus on creating tools. Civic Space Labs is an example of this type of organization within the nonprofit technology sector. Writing good code is the core of what these organizations do, but that alone is insufficient. These organizations must know how to manage a software development process, which is part project management science and part creative inspiration. They must also be good at quality assurance, software architecture and communicating with others through clear technical specifications. And perhaps above all, these entities need mechanisms for understanding customer needs and translating those needs into functional requirements and ultimately into great products.

My experience working with really good developers is that they tend to be more like skilled artisans than stereotypical engineering geek. Writing code is a creative process and one crack coder is worth five average coders. This means that from an organizational perspective, an application development house places a high priority on recruiting, training and maintaining top talent. This is the job of the development manager, who is also charged with focusing this creative energy, making trade-offs between features, resources and time and, in the end, shipping a great product.

We tend to think of application developers in terms of companies like Microsoft or SAP that license their code either in the form of packaged product or for rent as an Application Service Provider (ASP). **It is important to keep in mind, however, that some of the most exciting applications development today is occurring outside corporate walls and within the loosely-knit networks of the open source movement.** These networks are challenging traditional business models and traditional assumptions about how software development is funded.

Application Integrators

Application integrators focus on delivering tools to clients. An application integrator is basically a fancy term for a technology consultant. These individuals and organizations specialize in listening to client needs and matching these needs with the right mix of technology and strategy through consulting processes that tend to be inherently labor-intensive. Integrators occasionally write code, but more as a means of extending or knitting together existing solutions than creating new applications "from scratch."

The types of personalities that excel in this work are usually the "right-brain/left-brain" types with big ears, good people skills and a mix of creativity and analytical skills. In the nonprofit technology sector, application integrators range from solitary technology consultants to larger-scale shops with a focus on a particular client sector (like ONE/Northwest which focuses on environmental groups), a particular geographical area (like NetCorps), or larger-scale networks of consulting organizations like NPower.

Truly mission-driven integrators distinguish themselves from private sector counterparts by measuring their work more in terms of client outcomes than strict profitability. For this reason, **mission-driven nonprofit technology integrators are the closest thing nonprofit organizations have to technology advocates and are likely to be the greatest single source of pressure for a social source commons**.

Application Hosters

Application hosters specialize in the operational expertise required to keep software services up and running with high degrees of reliability. Pure application hosting tends to be a more commoditized form of service than the other two pillars, so from a client perspective, the focus is on automation and streamlining process wherever possible in order to minimize operating costs.

The people who are really good in this field tend to have excellent process skills and are disciplined and detail-oriented. They also place a high premium on protecting client security. Nonprofit technology application hosters are under intense pressure when they compete head-to-head with private sector firms for highly commoditized hosting services, but excel when they focus on more specialized hosting that is targeted specifically at the nonprofit community. Nonprofit entities like Electric Embers and Riseup are making a strong case for this approach through hosting open source applications such as the mailing list manager, Sympa. **As the social source commons grows, dedicated nonprofit technology hosters will become increasingly essential to our ability to secure hosting for a whole range of very specialized applications that might not be attractive to private sector hosters.**

Conflating the Pillars

What happens when you start mixing and matching these functional roles? The skills required to be excellent at one are very different from those required to be excellent at others. Organizations that try to combine the three pillars invariably make subtle, but nonetheless very real, tradeoffs between where they invest resources. When an organization invests in two or more roles, it invariably drains its ability to excel at one. This is the notion of opportunity costs, and it's just the start of the problem.

The real pain caused by this conflation of roles centers on missed opportunities for collaboration between various segments of the nonprofit technology sector. This is the saddest part; for in a world of scarce resources, missed opportunities for collaboration hurt all the more.

The Developer/Integrator Dilemma: Customer Conflict

Let's say that you are an application integrator serving a particular client segment, such as the environmental community. One day you get a request from a client to develop an application that meets some particular need that they have. So you build the tool. Word gets out about how great the tool is and now clients from outside your mission start asking for the tool. What do you do?

You can deny out-of-mission organizations access to your tool, but then this great tool is underutilized and the costs of its development aren't being amortized against a very broad audience. That makes it more difficult to sustain and improve the tool over time. Plus, if the idea is good enough someone else is likely to duplicate your efforts and make their application more broadly available.

Or maybe you take the opposite tack and open the tool up to all comers. Now, you are running two businesses. In the first, you're providing consulting services to the core clients defined by your mission and on top of that you're now building software tools for a much broader client base. This sets up several destructive tensions. First, you will start feeling pressure to steer your in-mission consulting clients toward the tool regardless of whether it's *exactly* what they need. Then, your new "out-of-mission" customers for the tool will become frustrated that they aren't getting the consulting services they need to be able to really use it. Organizations beyond very small collections of volunteers tend to have distinct business rules and identities that require some level of customization and integration in order to mesh the tool into their day-to-day operations. Where will they turn?

Being an application integrator is labor-intensive work that requires hands-on consulting expertise. My last three years at Microsoft were spent learning this lesson the hard way while working on b-Central, the company's first, and I'm guessing last, foray into serving businesses directly through an ASP model. Application developers tend to underestimate what it takes to do this kind of labor-intensive work. (*"How hard could it be? We know the tool better than anyone".)* So they decide to compete

with the very people who are critical to performing this customization and integration work with any kind of real scale.

There *are* situations when mixing the developer and integrator roles can make business sense, but these tend to be reserved for narrow niches where clients are large and well-funded. When a consulting organization has significant software development resources on staff, it has higher overhead costs than those that do not. The easiest way to cover these costs is by focusing on larger, more profitable clients that can take advantage of the unique development capabilities. In many cases, large clients directly offset the cost of adding specific new functionality to an application. This ability to modify essential features of an application is one of the inherent advantages of working with a developer/integrator, but it doesn't come cheaply. This is one of the reasons **developer/integrators tend to creep up-market over time as they optimize consulting practices for larger, more demanding (and more profitable) clients.**

In the nonprofit sector, the broadest, most basic technology needs are currently being filled by developer-only entities like Microsoft and Intuit that have invested in broad integrator channels to support their tools. As suggested at the outset of this paper, we are now reaching a new phase in the evolution of technology, where development shifts to increasingly strategic, and more customized applications that are specific to particular client segments.

Today, these specialized applications tend to follow one of two paths in the nonprofit technology sector. One path is restricted to a smaller niche of large nonprofit clients for the reasons outlined above. The other path that is aimed at the broader base of smaller organizations routinely has trouble building critical mass because of the still anemic channel of application integrators that are dedicated to deploying nonprofit applications. To build the social source commons necessary to boost a broad base of the nonprofit sector's more strategic application of technology, we need a nonprofit technology sector that not only distinguishes between developer and integrator but works to build vibrant collaborations between them.

Developer/Hoster Dilemma: *"Faux-pen"* Source:

With the rise of the Internet, application development has shifted to the server. There are still exciting new applications where smart client code makes a lot of sense, but the benefits of communication, interconnection, and centralized storage make even these applications far more powerful when coupled with back-end servers. This is why hosting really matters for applications development in the nonprofit technology sector.

Let's shift gears and imagine that you are an application developer and you decide to get into the hosting business for your applications. There is a name for this model and it's called an Application Service Provider, or ASP. Now here's the catch: **the ASP model doesn't play nicely with the whole idea of open source** – and that is a real problem for the nonprofit technology sector.

A key element of open source is the ability for anyone to freely run the code wherever they like. But ASPs can take shortcuts in their coding when they don't have to worry about installation in third party hosting environments that don't completely replicate their own. Imagine the amount of work it would take eBay, Amazon or Salesforce.com to enable third parties to run these services in their own hosting environments. Theoretically, it might be possible, but the resources required would be huge.

A decent point, but do you want to know the real reason the ASP model and the open source model conflict? **The route to profit for an ASP (like any company) is differentiation - unique competitive advantage.** There are companies who call themselves ASPs that are really just hosting somebody else's application. I call these "application hosters" and lump them into the same business model as an Internet Service Provider. Real ASPs make money by having a unique asset that they "lease" to customers by providing it as a service. Open source code, almost by definition, is not unique. Everyone has access to it, which means it is not a unique asset that can command premium service payments. Host an open source tool and you're a hoster, not a real ASP in the "renting a unique tool sense" that really matters from a business model perspective. **This is why dedicated nonprofit technology hosters are essential if we are to build a thriving social source commons that is truly open.**

In their quest to maximize revenue, ASPs also frequently feel pressure to offer **integrated suites of solutions made up exclusively of their own in-house services**. The nonprofit technology community has its own examples of this, and it is unfortunate because this model prevents nonprofit clients from being able to pick and choose from the best of breed in each application category. This tendency springs more out of business model considerations (and occasionally legacy coding and architectural issues) than any inherent limitations of technology. When application integrators are truly independent of application developers, they are able to offer client organizations objective consultation around which applications really are best of breed for their specific needs. They have the skills to knit these applications together and put pressure on developers to build and maintain open interfaces for applications in the first place.

When an open source application's development is tied to a particular organization's hosting, it hinders the nonprofit technology integrator community's ability and willingness to contribute code back to the project. Even ignoring the logistical difficulties of managing external code contributions in a proprietarily hosted server farm, **business models that tie together application development and hosting**

create strong disincentives for developer participation. In a true open source project, I have complete control over whether and when to extend the code base. The project's core team of coders may or may not decide to incorporate my modifications into the trunk, but I always have the option of permanently or temporarily forking the code if *I* feel the reasons are warranted by *my* particular requirements. The obstacles to collaboration are even worse when the open source project's sponsoring organization is doing development, hosting and integration and my code contributions back to the project might be used by the sponsoring organization to compete with me in serving my own clients.

The Social Source Network

If the nonprofit technology movement were to separate these pillars, how might this help us build a more differentiated yet better integrated nonprofit technology network; one whose whole is greater than the sum of its parts?

Integrating the Pillars

Once integrators, developers and hosters are free to focus on what they do best, new opportunities for collaboration begin to surface.

The first and most obvious benefit is that integrators become a critical channel for hosters and developers to reach and serve nonprofit clients. Integrators specialize in building the kinds of client relationships that are difficult for more centralized hosters and developers to maintain. By relying on integrators for outreach and first-tier client support, hosters and developers can pare expensive sales and support teams down to smaller cores that focus on supporting integrators in the sales and support process. This cuts overhead costs and frees up more resources to focus on what the organization does best – developing great code.

Integrators gain an intimate understanding of client needs in the consulting processes that can be a valuable source of information to tool builders. When relations between integrator and developer are good, information flows more smoothly and can be a **critical resource for developing new features and finding software bugs**.

When integrators have incentive to build on top of the developer's code base, they can also **contribute useful extensions to the code back to the developer**. This is very important, because over the course of doing projects with clients, integrators often stumble onto new feature needs that will be hard for the core development team to justify prioritizing at any given moment in time.

Extending this logic further, it is very possible that **whole new applications might very well be pioneered by application integrators** on behalf of a particular client and then handed off to a dedicated application developer should the application

attract the interest of a broader set of developers. For example, ONE/Northwest recently used Plone to build *ClearVoices*, a collaborative application that is now helping Oregon's environmental community track and comment on bills in the state legislature. At some point, should there be demand for this application outside of our client base, it would be wiser for us to contribute that code to an open source project that could devote the resources to expanding the tool and making it available to a broader set of clients. **When integrators are free to extend applications and contribute them back, the nonprofit technology sector gains a critical engine for building the social source commons.**

The Social Source Commons Stack

The nonprofit technology community needs to take a hard look at its assumptions about how we should build a social source commons. Our work does not exist in a vacuum. We are part of a much larger information technology sector and this broader sector has access to funding that dwarfs what is available to the nonprofit technology sector. **Logic would dictate that wherever possible, we must leverage work that is being fueled by private sector dollars in order to save our limited philanthropic funding for the truly unique requirements of our sector**.

One way to think of this is as a software stack where the bottom layers have broader, more generic application and the higher up you go in the stack, the more specific and narrowly defined the functionality becomes. From this perspective, communications protocols would be down toward the bottom of the stack along with languages and operating systems. Somewhere in the middle, you would find general productivity applications like word processors and databases and at the top would be the very specific applications that only apply to certain customer bases (e.g. an application to help land trusts track their land purchases).

No one would argue for the nonprofit technology sector to reinvent the bottom of the stack. There is no compelling need, for example, to build an operating system that meets the specific needs of a nonprofit organization. There are certain applications at the top of the stack, however, that are so specific that they will simply not be met by the broader information technology sector. Where things sometimes get murky is in the middle of the stack.

A general rule of thumb is that wherever functionality can be leveraged from lower in the stack, it should be. Content management systems, for example, do not solve problems that are unique to nonprofit organizations. But a good content management system like Plone or Drupal can lend a great deal of functionality to applications that are specific to nonprofit organizations such as online advocacy systems. Why reinvent the wheel, when there are massive and ongoing investments in the broader information technology sector that can solve large portions of your problem?

Running a successful open source project like Plone is hard. It takes tremendous resources to build critical mass around an open source project and ensure it will be there for the long haul. The private sector is a valuable source of funding and labor to most, if not all, of the most successful open source projects. Wherever possible we need to ride on these larger waves of investment. **Wherever possible we should be looking for functionality in the middle (be it open source projects or development frameworks like Zope and Rails) that can be added to and built upon as we create the top layers of the social source stack.**

Who's Going to Pay for All This?

Open source software changes a lot of things - and business models are no exception. IBM is currently riding high on this wave. Microsoft is not. Application developers, integrators and hosters all need to understand how open source affects their assumptions about what they do and how they do it; who is collaborator and who is competitor. The nonprofit technology sector is different from the rest of the tech sector in certain respects - but not this one.

From a funding perspective, hosters are in a decent position to recover costs from client hosting fees and integrators from client consulting fees. In some cases, these entities provide community value that goes beyond what individual clients are willing or able to pay. For example, ONE/Northwest invests a great deal in movement building for the environmental movement and on research and development for integrating new strategies and tools. Our clients tend to have a hard time justifying paying for these community services directly, so we use contributed income from individual supporters and foundations to help offset these costs.

When application developers differentiate into just one of the three pillars, they lose the ability to offset costs through hosting fees and consulting income. For this reason, philanthropic backing from both individual supporters and foundations becomes absolutely critical. **The best way to make this philanthropic backing sustainable over time is to scale down the costs currently associated with these efforts.** By taking better advantage of middle layers in the software stack (application frameworks and other open source projects), nonprofit technology application development shifts away from monolithic standalone efforts toward extending existing projects that are getting big infusions from the private sector. As the overhead of hosting and maintaining sales and integration consultants is removed, development teams are also stripped down to their essence and become less expensive to maintain. What philanthropic backers pay for is code and the expertise surrounding it.

We need to look more closely at how the most successful open source projects are run outside of the nonprofit technology sector. The successful ones almost always have a core group of developers, often connected in a loosely structured, informal network. In many cases, the project's developers supplement income by doing consulting work on the side – as individuals, and not under the auspices of the

project itself. Most successful projects are also backed by one or more patrons – be it commercial, philanthropic or some combination thereof. The Plone Foundation is an interesting example of the kind of legal entity that is sometimes formed in order to support the network of developers working on an open source project.

The old days of dedicated software development firms are now giving way to something new; something that more resembles software development networks. We need to tap these and other new creative approaches to funding and managing software development today if we are to succeed in building the social source commons of tomorrow.

The Guilds of the Social Source Movement

The nonprofit technology sector needs to create centers of gravity around each of the three pillars. We need practitioners in each pillar that are large enough to be sustainable and who can model best practices to aspiring new entities to the field. **We also need the *network knitters* - those entities that can weave connections between organizations both within and between pillars.** In the private sector, this is the job of trade associations. Luckily, we have Aspiration beginning to play this role for the application developer pillar and N-TEN which is to some degree playing this role for the application integrator segment of nonprofit technology (though its focus is currently divided amongst tool builders, integrators and their nonprofit clients). **Over time, there is a huge opportunity for both of these organizations to become the modern age equivalent of *guilds* for each of their respective pillars**; encouraging the flow of best practices amongst members, lobbying on their behalf, facilitating the recruitment of new people into the profession and helping with their development through apprenticeships and other approaches.

Conclusion

This paper presents a framework for thinking about the nonprofit technology sector in terms of a network; a network with the potential to become a movement that is dedicated to the creation of a social source commons. As a first step, it suggests differentiating the nonprofit technology sector into clearly defined areas of core competence. These distinct functional roles then form the basis for building truly collaborative partnerships with other complementary organizations. Differentiated parts integrated into a greater whole – that is what makes for sustainable ecosystems and that is precisely what we need to create for this ecosystem of nonprofit technology to grow and sustain itself over the long-term.

This paper is simply a framework, or conceptual container, for rethinking the world that exists today. As such, it does not include all the answers. Indeed, it likely raises more questions than answers. It is put forward in the spirit of starting a conversation; an important conversation about how we will create the social source commons of our shared future.