TRANSIT AND OPEN SOURCE: IS IT AN OPTION?

Version 2.1

29 June 2005

Prepared by:
Paula Okunieff, ConSysTec
(formerly of Systems & Solutions, Inc.)
and
Nancy Neuerburg, N-Squared Associates, LLC

Preface

This paper was developed as part of the Transportation Research Board Transit IDEA Project 39 Dynamic Timetable Generator (DTG). The development of a Dynamic Timetable Generator was proposed to save transit agencies time and resources as they reformat their schedule information for presentation via the Web. Furthermore, transit expends these resources over and over again not only for each schedule change period but also when they translate the timetable into different languages or accessibility formats. There are hundreds of transit agencies that can benefit from a software application that efficiently pulls their schedule data from its raw state, reformats the data to a consistent form, and then applies the data to a presentation template of their design for any target population. Open standards support this vision, but they are not sufficient.

In the DTG project, we built code that scaled to small, medium and large transit agencies. Where possible, the project used Open Source Software (OSS) to develop the DTG prototype. In general, the reason that OSS was used was because the open source infrastructure code scaled better and supported the different requirements for each of the transit agency data sets.

From initiation of the project, the DTG Project Team, subcontractors and partnering Transit Agencies, agreed to explore a framework design that would ensure that the software remains freely available to the transit industry. To ensure interoperability as well as the custom features needed to support different transit agencies, access to the code is needed. In addition, the software needed to be trusted as reliable by skilled Transit information technology staff, and that innovations, modifications and enhancements be incorporated into the original code. To that end, the project team agreed to develop a white paper on OSS and propose a framework to support continued development of OSS by the transit industry.

In developing this paper, the documented extent of the worldwide acceptance and use of OSS was remarkably high. There is a large OSS user and development community, a wide range of resources available to open source developers, and positive impacts of OSS on business development. When using OSS in the Dynamic Timetable Generator project and developing this paper, we discovered that transit agencies must be better educated about OSS. Further, transit must develop policies on use and modification of OSS. Transit industry software developers and their funding bodies must facilitate the implementation of formal processes, institutional and organizational structures to support the effective use of OSS, while simultaneously enabling a grassroots technical organization to evolve. To implement these recommendations, it will require resources and commitment from public sector agencies for the full benefits of OSS to be realized.

Acknowledgements:
The authors would like to thank the many transit industry colleagues who generously shared their knowledge and opinions on these issues, including Bibiana McHugh (TriMet), Jim Davis (NYSDOT), and Dan Overgaard (King County Metro). In addition, we thank Brad Kittredge (KCM) for offering some early comments on this paper. The support of the Transportation Research Board, for the Transit IDEA Project 39 Dynamic Timetable Generator, is also gratefully appreciated.

You can reach the author at: Paula Okunieff 617-983-3364; paula.okunieff@consystec.com and Nancy Neuerburg 206-941-2106; nancy.nn@att.net.

Executive Summary

This paper was developed to provide an overview of Open Source Software for the transit industry, explore issues that might affect transit's use of open source software and propose that the transit industry should participate in both the use and development of Open Source Software (OSS).

Transit agencies can achieve improvements in service quality and performance efficiencies with the implementation of technology and software applications. There are an unlimited number of application *concepts* for managing information, providing services and tracking performance that may help agencies achieve their goals and service commitments. Expectations of customers, elected officials and transit managers are driving requirements for interoperability between the systems so data can be shared and integrated more readily. Demands for more integrated regional and multi-modal systems are also increasing the need for more integrated systems and data.

The need for these improvements in software available to transit is currently not being effectively met by most in-house software deployments, standards, or vendor products. Transit agencies cannot expect vendors to meet all their application needs, since the industry market is both relatively small and very diverse. Much in-house development occurs, but it is not done efficiently from an industry-wide perspective. Several transit agencies may develop similar applications, yet there is no forum or procedures for transit staff from several agencies to share or leverage their expertise to develop, debug or enhance an application.

Applications developed in-house and commercial off-the shelf software both pose a number of risks to transit. Staff turn-over, either in-house or within a vendor's organization, will impact product support. With commercial software, transit agencies are locked into a single-source service provider for the life of the product, subject to the vendor's update/upgrade schedule, their product obsolescence schedule, and the vendor's ability and desire to stay in business.

These and other factors drive the industry to seek a new paradigm to develop software. Open Source Software (OSS) and open standards provide a new approach to meet the challenges faced by transit agencies in procuring, developing, deploying, maintaining and servicing technology projects.

Common OSS Benefits
Open Source Software (OSS) can:

- Leverage multiple agency resources to develop and maintain software;
- Provide a forum to share best practices and develop tools based on those approaches;
- Build open standards-based applications that are interoperable;
- Identify and build key technologies and functionality that provide the most benefit to the most agencies;

- Build tools that are scaleable to serve small, medium and large transit agencies;
- Foster a community of transit IT professionals and consensus on industry best practices;
- Reorient the industry to a service-based commercial model, where vendors repackage, redistribute and compete on service and enhancements to industry developed software;
- Promote competition to improve software quality and lower costs.

OSS Background

OSS started with a simple license that allows free distribution and modification of software. It is now an industry with multiple license types. A significant and growing number of organizations and web sites now support OSS development and distribution. These OSS development web sites host tools to support application development and foster communication among a community of programmers, users, and observers. The tools provide transparency into the development and maintenance processes to meet key system engineering and software development best practices.

Today, there are OSS applications that command over 60% of the market in some infrastructure software business areas. Examples of well known OSS products include Apache Web Server, Linux, Firefox and Perl. Both Apache Web Server and PERL are currently used within the transit industry. Some OSS, such as Linux, is developed by a group of ad hoc software programmers who have an interest in reusing software that is offered on one of the OSS web sites. Other applications are developed by a company and put out on the web for redistribution and modification, for example, Firefox, developed by Netscape. The Apache Software Foundation (ASF), a non-profit organization, develops "branded," open source software.

The governance structure of ASF defines a process to ensure community interest, trains project managers, and ensures acceptable software maturity prior to its introduction on a project site. This formal, wholly volunteer effort supports over 25 projects producing several applications under each project umbrella. ASF is organized to pool individual and corporate operating funds. The funds are used to support the operations and the infrastructure only. No funds are used to procure software development support.

Approaches for a Transit OSS Initiative

The benefits of OSS to transit will occur in a phased approach. Already many transit agencies are using some OSS such as PERL and the Apache Web Server. The next wave of benefits will come after the transit industry becomes better educated about OSS and more comfortable with it. The most benefits to transit will come when a Transit OSS Initiative is implemented.

Because of its governance structure and obligatory processes, the ASF framework may provide the best approach or model for establishing a transit open source software initiative. Organized as a non-profit to pool funds and establish a governance structure, ASF has been very successful in developing and vetting projects that are interoperable, promote open standards, and redistribute open source applications that are now central to

the Internet. This model may be adapted by the transit industry to ensure the development of interoperable software, to identify key pooled development projects, prioritize enhancements and maintain the software to ensure a high degree of reliability over the product life.

Critical success factors and a three phased approach to implementing a Transit Open Source Software Initiative are presented in the paper.

- Stage 1: Preliminary Trial
- Stage 2: Interim Organizational Development
- Stage 3: Establish Transit OSS Initiative Consortium

The preliminary trial will introduce the industry to the development and use of transit-specific OSS. In the first stage, a transit software application will be assigned an OSS license and put on an OSS web site. Learning from the trial, in the second stage, it is proposed that the transit industry develop an interim forum in which to share, enhance and maintain transit OSS. At the same time, additional research and groundwork would be done to develop a governance structure for the third stage, the establishment of a Transit OSS Initiative Consortium. There are still some key legal, institutional and technical questions to answer prior to establishing a formal institution to develop software that meets transit industry needs.

As a part of the Transit OSS Initiative Consortium, a license would be developed that meets the needs of transit and ensures indemnification, protects participants from liability, fosters a community of developers and reorients the commercial industry to redistribute and provide service support to the public sector. One of the things that the Consortium would do for transit, is help identify software that is suitable for an OSS approach and that would provide widespread benefit to transit.

In closing, there appear to be clear opportunities for transit in the area of OSS, but given the newness of the topic to transit, education and outreach, lessons learned and a proof-of-concept need to be addressed before it will be viewed as a viable option on a wide-spread basis.

TABLE OF CONTENTS

ABSTRACT

This paper was developed to provide an overview of Open Source Software for the transit industry, explore issues that might affect transit's use of open source software and propose that the transit industry should participate in its development. There are technical, legal, institutional and management issues that are presented in this paper. The first part of the paper, Sections 1 through 5 discuss open source in general. The last two chapters describe the transit industry, its business environment, potential impacts of OSS, critical success factors in making OSS work, and an approach to moving it forward within transit.

1. What is Open Source?

Open source software (OSS) projects, where the use and modificaton of software is more freely shared than traditional commercial code or the privately developed code of an organization, were developed to fill a void or because programmers felt that the developed technologies would make a difference. OSS is making a significant difference by developing products that are improving productivity and speeding the deployment of new technologies. OSS has been credited by all sources with helping improve software reliability and lowering costs of commercial software, thus supporting the software industry (commercial and OSS) in general.

Open Source Definition

The term "open source" was copyrighted in 1997 with a very specific meaning[1]. An open source software license does not restrict a software product's use and distribution, rather it requires that the software source code be accessible, that derivative works also be open, that anyone can acquire the code and use the code without restriction. As stated by the formal open source definition, "Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with…" a set of criteria as follows: [see Appendix A for the complete definition]

- Freedom to redistribute source code and executable
- Make source code available to all
- Derived works are also open source
- Author may explicitly allow for derived works or modifications
- No discrimination against persons or groups
- No discrimination against use of application in business, non-profit, personal or any other field of endeavor
- License must not be tied to using another product
- License must not restrict other software that may be distributed with it. For example, the license cannot declare that a proprietary application that installs the software must also be open source.
- License must be technology-neutral

The license types have evolved since the first one, but the criteria for the open source brand are consistent. An OSS license must conform to the defined objectives.

Brief OSS History

Perhaps the impetus to develop "open-source" software came out of AT&T Bell Labs restricting the the Unix operating system license. In October 1993 Bell Labs offered free licenses of the UNIX source code when requested by universities and researchers, and later to the military and commercial organizations. Users (i.e., license holders) improved the software by adding utilities and porting it to many platforms. University of California at Berkeley developed related utilities and tools that were known as the Berkeley Software Distribution (BSD), and were distributed under the terms of a BSD license. The code for the BSD UNIX tools was also put into the public domain and improved by researchers and programmers. In the 1980s the [Defense] Advanced Research Projects Agency (ARPA) funded a research group in Berkeley, the Computer Systems Research Group (CSRG) to enhance UNIX and expand the BSD toolkit. The subsequent projects became the foundation of the Internet. In 1984, AT&T and Bell Labs were separated by a court injunction. Bell Labs needed to generate a revenue stream, and so they closed the UNIX license in order to commercialize it as a product[2].

Restrictions put on the UNIX license precipitated a group of researchers at MIT's Artificial Intelligence Lab to establish the Free Software Foundation (OSF). Leading this group, Richard Stallman published the GNU[3] Manifesto. The Manifesto promoted free software as the freedom to use the program for any purpose, to review, modify, and redistribute the code without restriction. To ensure compliance with the manifesto, Stallman established the General Public License (GPL) upon which most Open Source licenses are based. The GPL license insisted that derivative works be licensed with the GPL. The OSF was focused on programmers, expecting them to perform the following key roles: disseminating its message, assisting with recruitment, and selecting of developed tools. Without broader industry awareness and support, the movement did not gain much traction[4].

In 1990, Linus Torvalds, a computer science student at the University of Helsinki posted a primitive kernel of a Unix-like operating system on the Internet. He requested help to build out the code. The result was Linux. In 1992, Torvalds released a new version of Linux under the GPL license that required, in perpetuity, that all derivative works be open and freely redistributed.

In 1995, a group of researchers started to work on the Apache Web Server project, building on the efforts of a languishing National Center for Supercomputing Applications (NCSA) sponsored web server project. Eight people started the project and within three months over 150 people subscribed to the project's mailing list and participated in the discussions. Within the year, the first version was released under the BSD license. Today, the Apache Web Server runs close to 70% of all commercial web sites and is one of the many products that the Apache Software Foundation (ASF) develops in its family of web services related products. The ASF will be discussed in more detail in Section 4.

Open source software became recognized as a commercially viable alternative source of software when IBM and other hardware vendors embraced Linux and Apache as a way to differentiate their hardware and to compete against Microsoft. IBM, Silicon Graphics and other hardware vendors entered the OSS redistribution and support services industry.

The IBM move also helped changed the way business views open source software; OSS now competes against commercial software to support key IT infrastructure projects. OSS technical support may now be procured from a wide array of distribution and support vendors. Many IT market observers like Information Week, Wired, Gartner and Forrester predict that OSS will change the software market even further, with commercial IT businesses moving towards building their revenue streams from redistribution, service and support, and away from licensing fees. The impact on the consumer will be software with improved quality and lower costs for OSS and commercial software.

Common OSS Products

Today, many open source products hold a significant market share in their respective niches. The majority of mature technologies reside in the programming language, security and infrastructure markets. The most widely known applications include Apache Web Server[5] which has a market penetration of over 68% of web server applications, Linux[6] which is estimated to capture 20% of the desktop market by 2008, and Mozilla's Firefox released earlier this year which expects to attract 10% of the web browser market by the end of 2005[7].

Open source programming languages such as PHP, PERL and Python are now some of the most popular programming language for web applications. Some industry players such as Sun Microsystems and Apple are offering their operating systems (Solaris and MAC OS X / Darwin, respectively) as open source in order to enhance their hardware products and entice developers to build applications that run on their operating systems.

The number of sites that provide collaborative environments for developing, testing and maintaining open source code is growing. A site may support open source code for a single project, multiple unrelated projects, or in the case of Apache Software Foundation for multiple interoperable applications.

Open Source Related to Open Systems and Open Standards

Open systems and open standards are sometimes confused with open source. Open source software is associated with a class of license agreements that follow the requirements described in the Open Source definition (see Section 1 and Appendix A). Open standards are standards that are developed and promulgated in an open, consensus based environment by industry experts. Open systems are standards-based products that may be hardware or software, open source or proprietary. The Department of Defense defines open systems as:

> "A system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered components to be utilized

> across a wide range of systems with minimal changes, to interoperate with other components on local and remote systems, and to interact with users in a style that facilitates portability.[8]"

One may view open source software products as a "more pure and open" form of open systems software. Many open source and open systems products use open standards and in so doing help promote these standards and build markets for the standards.

Difference between proprietary and open source

> "It's partly a question of business models. Proprietary companies sell packages for money and see support as a cost. Open source companies give away packages and see support as the cash cow. Microsoft sees open source as meaning we give away Intellectual Property, while open source advocates call software you can't see a rip-off. There is not a lot of common ground.[9]"

This cynical quote summarizes, from the open source perspective, the differences between proprietary Commercial Off-The-Shelf (COTS) software and OSS. Yet, according to researchers, the majority of software used by businesses today is developed in-house; only about 30% of software is proprietary COTS. Some economists think that 30% is overstating the COTS market[10].

For businesses, there are three classes of software acquisition – proprietary COTS, open source and custom developed. Given the statistics, the majority of software is developed and maintained by in-house or contract resources. The public sector transit industry is no different than other industries; many of its applications are too small and too unique for large software companies to profitably develop and sell, so they are developed in-house or by contracted developers. Also, the transit-specific software market is not large enough to attract as many vendors as some other industries.

Small, medium and large transit agencies have all found it necessary to develop some of its software in-house. In-house, non-OSS software misses out on the benefits of sharing the development costs and the burdens of bug fixes and enhancements. The other benefits of a broader developer support group are also missed.

Each individual business in a vertical market acquires software that basically consists of the same functions with only a few differentiators. Vendors compete on the few differentiators. In a small, competitive environment a vendor may try to control the market by buying out its competitors until it monopolizes the market offerings[11]. Many special purpose Transit applications that are offered widely are controlled by a very limited number of vendors. When competition and/or market scope is limited, there is often less incentive for vendors to improve reliability and technical efficiencies, lower costs, or support technology transfer.

2. Open Source Development Models

An overview of the current development life cycle of open source software is provided in this section. The life cycle is characterized by many individuals providing feature enhancements, bug fixes and developer/user support in a controlled forum. Additional information about the organizations and infrastructure that have typically supported the development of OSS is included in Section 4, How OSS Works.

How is Open Source Software Developed?

OSS has a relatively short development history, yet successful open source products already exist. In all likelihood, the OSS development process will continue to be refined, facilitating better, more rapid development of OSS products. The community of open source software development appears to be a very large, amorphous phenomenon today. There are hundreds of applications, thousands of contributors, many consortia and support networks, and millions of observers on "open source" mailing lists all participating in this open source economy. In addition, there are peripheral markets that provide services for open source software or training materials on open source like O'Reilly publishing.

From a review of the most successful open source products, the current lifecycle of an open source application appears to proceed as follows[12]:

Early phase: technology preview
A bright person or organization publishes a kernel of an application to a web site, establishes an interactive editing and configuration management approach to develop the software, sets up a mailing list, publicizes the application, and adopts an open source license type. Programmers start contributing to the development. Commitment of a strong developer community will move this phase to the early deployment phase more quickly.

Characteristics of this early-product phase include:

- Technology changes rapidly and software updates are frequent;
- Testing is available to all, but few have the necessary skills to participate;
- Installation is subject to complicated procedures, documentation is minimal;
- There are few users and fewer resources for support.

Middle phase: early deployment
The middle phase is characterized by a version 1.0 release of the software. Apache Software Foundation (ASF) is trying to skip the early phase with its *Incubator* project (see Section 6 for more information). This phase is characterized by the following:

- Technology is easier to use with better documentation, automatic installation and packaging.
- Commercial vendors offer redistribution and technical support services for the product.
- Product is used in a production environment and is robust and trusted.

- Security and performance is enhanced through suggestions and incorporation of bug fixes and patches.

Maintenance phase: long-term support

At a certain point, the software offers the core features needed by users. At this stage, the product moves into the maintenance stage. An increased user base, demand for services and enhancements characterizes this phase. The basic attributes of this phase are:

- The software and related documentation are in a state of stability and maturity;
- Products are trusted because of the number of deployments and the number of years of performance and fault fixes;
- The users drive "demand for ongoing support and further development."

Open source applications drive the demand for more open source software and enhancements. Once an application is deployed, there is a desire to add functionality to it. Some open source development projects form organizations that support the development of a suite of projects. For example, the ASF discovered that in addition to the web server, there was a demand for related applications to interoperate with the core software. ASF[13] currently sponsors about 25 projects, each containing a set of subprojects. There are also about a dozen code sets in the Incubator, each waiting to be verified as "stable" before it becomes a formal Apache project or subproject.

With acceptance into the mainstream and promotion by corporations and governments, the open source industry has developed a number of mature institutions.

3. Benefits / Risks

In manufacturing industries, like automotive and aerospace, manufacturers demand that their Original Equipment Manufacturers (OEMs) share their technologies with competitors to minimize the number of single sole-sourced parts. Because the users of software have not leveraged their influence over software vendors, the commercial software applications industry is trending towards a monopolistic market within business segments. Open source has brought competition back into the market place, increasing choice, and driving commercial vendors to improve software quality, bring higher reliability and security, and lower costs.

With the endorsement of government agencies at the state and local levels in the US and at national government levels around the world, procurement of open source software is now a mainstream option for public sector organizations. In a recent document on guidelines for procuring information and communications technologies (ICT), the Australian government states:

> "Open source software development…can offer original solutions to problems not addressed by proprietary software and it has the potential to lead to significant savings in Government expenditure on information and communications technology (ICT).[14]"

The Australian Guidance document offers a valuable perspective into the risks and benefits of assessing the costs throughout the lifecycle of software – proprietary and open source. In addition, the U.S. Department of Defense (DOD) sponsored MITRE[15] to determine if they should consider implementing OSS. These two studies describe the benefits as follows:

Benefits

- Ability to choose among technical support contractors to implement and maintain the software
- Ability to customize the code to meet specific requirements
- Flexibility in customizing application to meet requirements
- Software support and maintenance may be extended indefinitely
- Ability to scale application to meet specific performance criteria

Both studies also considered the risks to deploy OSS and provided a check list of characteristics to explore prior to procuring OSS (see Table 1 below). The studies identified these risks as also occurring in the commercial industry. The risks consist of the following:

Risks

- Fragmentation or code "forking"
- Lack of compatible/interoperable applications
- Version control
- Building a viable community of programmers
- Trained staff for support and service
- Competition from proprietary vendors

Fragmentation: This is the possibility that source code will develop in different directions, that is, different flavors that are not compatible. For example, database vendors implement different SQL features that are not consistent with the ANSI SQL standard; this prevents some queries from being used by a database developed by a different vendor. In the long run, there may be multiple versions of the software available. To date this has not been a problem with the mature OSS products such as Apache and Linux. These groups developed non-profit institutions to control the input and formal release of the software versions.

Lack of compatibility/interoperable programs: Once mature, other programs tend to be based on the core functionality of open source code. In anticipation of the complexity of keeping related applications compatible, ASF and Open Source Application Foundation (OSAF) guarantee interoperability by building an infrastructure to manage projects and test applications of each version prior to its official release. The test suites and test environment is a key factor in ensuring interoperability. Open standards also facilitate compatibility. Many ASF projects use OASIS[16] or W3C[17] specifications as the basis for their development activities.

Version Control: Because in the early stages of application development, code is released early and often, there may be a lack of version control. Many of the institutions that support OSS development have built infrastructure to mitigate this risk. In particular, wiki[18] software enables an on-line, self-archiving record of each change to a document or software. Almost every OSS development activity uses wiki software to manage versioning.

Building a viable community of programmers: The skill level of a core group of programmers is critical to developing trusted, reliable code. The Australian guidance document advises that "[a]pplications without support from a strong developer community are best avoided.[19]"

Table 1: Due Diligence risk mitigation checklist[20]

Attribute	Yes/No
Does the software operate as described by the project's website and documentation?	
Does this operation match or exceed mandatory requirements for procurement?	
Is the software supported by a publicly accessible website that lists recent announcements?	
Does the software's website provide information on new versions?	
Does the software's website provide a running history of previous versions so you can check the frequency of releases?	
Does the software's website provide information on security fixes and feature updates?	
Does the software have a publicly accessible and independent user forum or mailing list?	
Does this forum have archives that can be searched either in-site or through a public search engine?	
Does the software have a published roadmap available on the project website?	
Does this roadmap meet the agency's requirements?	
If the project needs additional features to meet agency requirements, is there an obvious process for requesting new features or enhancements from the development team?	
Is there a general growth trend in the number of users?	
Are there local service providers listed on the project website?	
Have you performed a risk assessment of these service providers?	
Have you analyzed the requirements of any ancillary components, modules, libraries or systems the software needs to operate properly?	
Have you prepared a risk mitigation checklist for each of these subcomponents?	

Trained Staff for Support and Service: The Australian guidance document identifies three options for support and services: in-house, multi-sourcing through external provider

and single sourcing through an external provider[21]. Because the software is open, and can be maintained by skilled programmers with domain knowledge, there are more options for procuring support and services then for commercial software packages. The element of risk is driven by the OSS application maturity. In its early development phase, hiring a programmer to develop the code may be more effective then hiring a technical support contractor to serve as a program manager.

Competition from proprietary vendors: Threats of litigation and negative marketing strategies have not worked against the leading OSS development efforts. However, these are negative tools that may undermine an open source development effort. On the other hand, the negative publicity on the failings of commercial software such as the security holes in Internet Explorer have worked to the benefit of open source software programs such as Mozilla Firefox.

4. How OSS Works

When a business or organization decides to develop open source code, they must determine the rights and restrictions they want to impose on the software distribution and use. For example, do they want to distribute the software to anyone to use and modify; do they want to distribute the software and ensure that it is always available to anyone to use, modify *and redistribute freely*. The answer to these questions will impact what types of businesses participate it the development, and provide redistribution and support services.

Further, some vendors will offer OSS as a "loss leader" to lower the cost of procurement, while attracting ongoing support and customization services. The rights and restrictions on use and redistribution are based on the type of OSS license selected. The license will drive both what others can do with the software and how the initiating vendor or organization will benefit. The license also influences the types of satellite businesses that can emerge that depend on or support the OSS.

In this section, the major types of open source licenses are explained; business models and their relationship to license types are also described.

Open Source License Types

A hallmark of all open source software licenses is that the license allows the user to be granted "a world-wide, royalty-free, non-exclusive license.[22]" The license allows for free use, right to modify and redistribute the code. These are consistent with the Open Source Initiative (OSI) definition of "Open Source." All licenses assert that the code offers no implied warranty, and some state that the original license holder is indemnified against third party claims when code is modified.

OSI lists approved Open Source licenses, over 50 to date, on its web site. Table 2 summarizes the differences among the provisions of the major types of licenses and a summary of the number of applications that use each license type. These statistics are collected on a web site, SourceForge.net, which serves open source development

communities. The five licenses listed in Table 2 are the most widely used licenses as can be seen from the statistics collected by SourceForge.net.

Table 2: OSS License Provisions

License Provision	BSD/ MIT	Apache 2.0	GPL	LGPL	MPL
Grant Intellectual Property rights	√	√	√	√	√
Can modify and/or repackage under separate license		√			√
"In perpetuity" clause (all modifications are also subject to this license agreement)			√		
Allows proprietary libraries and APIs that are linked to OSS to remain proprietary	√	√		√	
No-Endorsement	√ (BSD)				
All subsequent work of contributors are also open and relinquish patent rights when incorporated		√			√
No warranty	√	√	√	√	√
Limitation of liability		√			
Cannot use name on revised versions (branding)		√			
Sample Distribution of OSS license types on SourceForge.net[23]	**4,552**	**389**	**43,638**	**7,110**	**788**

There are many variants of OSS licenses. Most licenses emerged from the Berkeley Software Distribution (BSD) License and the GNU General Public License (GPL). Soon after the initiation of the OSI, the Lesser GPL license was released. These are the three most popular licenses applied to OSS today.

Berkeley Software Distribution (BSD) is described in Section 1 Brief History. BSD and MIT License provisions allows for merging proprietary and open source code into a single application. Consequently, a proprietary software library that is needed for the proper execution of a BSD licensed application may need to be purchased. Most BSD and MIT License provisions are the same except the BSD includes a no-endorsement clause (that is, that Berkeley does not endorse the product that uses the license). BSD influenced many of the OSI approved licenses including Apache Software Foundation (Apache 2.0) and Mozilla Public License (MPL). Apache portal is repackaged and sold under the IBM Websphere Portal license, and Mozilla Firefox is repackaged and distributed by Netscape. Although different in style, the Lessor GPL contains similar provisions to the BSD; the key difference is that open source code keeps its "open source" character in perpetuity.

The Apache 2.0 License has a special provision that indicates that the Apache license can only be associated with products that are generated by the Apache Software Foundation. One can think of it as a way to brand the software that is issued by Apache. Apache also includes a "limits on liability" clause in their license. This is an added restriction from their original license.

The GNU General Public License was developed by the Free Software Foundation. This is by far the most widely used license type. It is applied by Linux and over 4,500 OSS projects. It is the only license that states that all modifications and all libraries used by the code are subject to the terms and conditions of the GPL license, that is, source and binary code should be freely available. This is sometimes referred to as the "viral" clause because the open source nature infects all the code it touches.

Developer Business Models

The biggest differences in OSS developer business models are in the areas of software ownership and operating procedures. The ownership issue, if parties decide that they want to own the OSS, may emerge as a complex legal arena in the near future. Unless specified differently, the coder or organization that hired the coder technically owns each person's contribution to a program. Some business model operating procedures deal with this issue to ensure that the OSS and software distributors are indemnified against third party claims, i.e., claims of copyright infringement.

This section will deal with the differences in operating procedures, including how qualified programmers are recruited, project managers are assigned responsibility, software quality is assured, enhancements implemented and other pertinent issues. In addition, ownership issues will also be discussed.

There are commercial businesses like Zope and Sun that offer open source code, but control the versioning. The software is owned by the business. The business controls who contributes to the base code. Anyone may download the code, submit bug reports and suggestions for patches, enhancements and modifications. However, only vetted programmers are allowed to participate on the site that supports the collaborative tools that manages the product enhancements. Specifically, Zope states on its site that "Zope Corporation still drives the primary vision and development of Zope, with each release being maintained and developed by more and more community members - either by submitting patches to the bug collector or by developers with access to check in new features and fixes.[24]" It is assumed that the vendor requires that any software contributors sign over their ownership rights or grant the corporation free, unlimited, worldwide distribution rights to the code. This model will hereafter be referred to as the *Commercial* OSS approach.

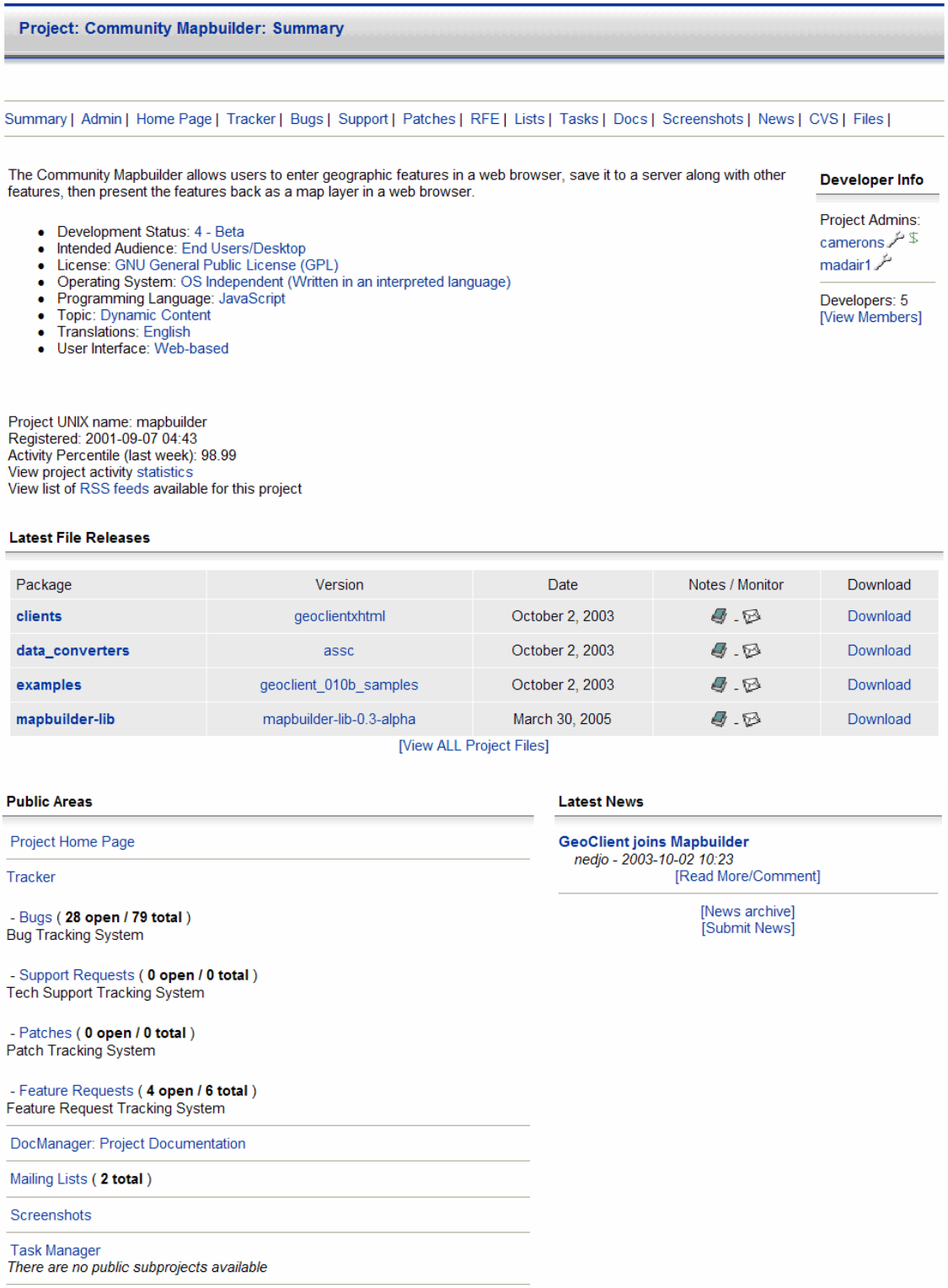
Alternatively, there are loosely knit communities where any individual may develop and post their software on open source community web sites like SourceForge.net (over one million registered users and close to 100,000 registered projects) or freshmeat.org, both part of the Open Source Technology Group (OSTG). The software is posted on an OSS site which offers a range of tools such as mailing lists, change reports, wiki software for

modifying the code, bug tracking and reporting, and other collaborative development applications. Figure 1 is an example of a project page from the SourceForge.net site. The tools that run behind the page provide project management support that includes mailing lists, change reports, and other collaborative development tools. Figure 1 shows a development page from the MapBuilder project which is based on the Open Geospatial Consortium web services standard. The page shows that there are five developers who have been working on the project for over three years. SourceForge collects statistics on the number of downloads, patches, bug tracking, feature requests, and more. The environment that supports OSS development projects is very robust and mature.

Project managers are assumed to be the original individual who posted the code, however, since the software is maintained on the web site, anyone may contribute to the code. When a project manager is not active, someone else may assume responsibility or start a new project using the core software. Enhancements to the code are not developed in a vacuum. There is a mailing list where issues and approaches are discussed, designs reviewed and testing results distributed.

In this environment, the ownership is complex. Each individual contributing to the application owns his or her code contribution. There may not be legal protection for the final application if a contributor decides to change his or her license criteria. This model will hereafter be referred to as the *Ad Hoc* OSS approach.

Figure 1: SourceForge.net OSS Development Page



Project: Community Mapbuilder: Summary

Summary | Admin | Home Page | Tracker | Bugs | Support | Patches | RFE | Lists | Tasks | Docs | Screenshots | News | CVS | Files |

The Community Mapbuilder allows users to enter geographic features in a web browser, save it to a server along with other features, then present the features back as a map layer in a web browser.

- Development Status: 4 - Beta
- Intended Audience: End Users/Desktop
- License: GNU General Public License (GPL)
- Operating System: OS Independent (Written in an interpreted language)
- Programming Language: JavaScript
- Topic: Dynamic Content
- Translations: English
- User Interface: Web-based

Developer Info

Project Admins:
camerons
madair1

Developers: 5
[View Members]

Project UNIX name: mapbuilder
Registered: 2001-09-07 04:43
Activity Percentile (last week): 98.99
View project activity statistics
View list of RSS feeds available for this project

Latest File Releases

Package	Version	Date	Notes / Monitor	Download
clients	geoclientxhtml	October 2, 2003		Download
data_converters	assc	October 2, 2003		Download
examples	geoclient_010b_samples	October 2, 2003		Download
mapbuilder-lib	mapbuilder-lib-0.3-alpha	March 30, 2005		Download

[View ALL Project Files]

Public Areas

Project Home Page

Tracker

- Bugs (**28 open / 79 total**)
Bug Tracking System

- Support Requests (**0 open / 0 total**)
Tech Support Tracking System

- Patches (**0 open / 0 total**)
Patch Tracking System

- Feature Requests (**4 open / 6 total**)
Feature Request Tracking System

DocManager: Project Documentation

Mailing Lists (**2 total**)

Screenshots

Task Manager
There are no public subprojects available

Latest News

GeoClient joins Mapbuilder
nedjo - 2003-10-02 10:23
[Read More/Comment]

[News archive]
[Submit News]

The Linux development environment is based on this *Ad Hoc* approach applying an informal distribution of control. The shear complexity of the code necessitates the decentralization of the effort. Linux has over 76 separate programs to develop core kernels and drivers. There are recognized managers for parts of the code, with a core group overseeing all of them. The development effort is hosted on the OSTG web site.

The Apache Software Foundation and the Open Source Application Foundation, on the other hand, are organized in a more centralized fashion, based on a "Meritocracy"[25] or technical merit. Both have a road map and formal (albeit evolving) specifications that drive the work. They are both organized under a non-profit or 501(C)3 corporate structure which assumes ownership over the software contributions or requires contributors to sign an indemnification agreement. OSAF is less formal than Apache. It was originally funded by Mitch Kapor who donated $5 million to start the project. The OSAF employs a core set of project managers and programmers to move the "Chandler" software suite into a mature state (middle phase).

The ASF has a different governance structure. Project managers are assigned based on merit and technical expertise. The non-profit corporation shields the Board of Directors, Project Management Committees and Committers from liability and indemnifies them against a third party bringing suit for patent or copyright infringement. The formal structure provides a vehicle to collect sponsorship funds from commercial vendors of which there are significant contributions. Many of the dedicated (volunteer) committers are assigned by these commercial vendors to work on or manage Apache projects. A set of bylaws spells out very explicitly how the organization is structured, how software is approved, who gets to approve, how volunteers are elevated to positions of increasing responsibility, etc.[26] Unlike OSAF, there are no members, officers or committers who are paid by the Foundation.

The "The Apache Way" is described as follows:

- "collaborative software development
- commercial-friendly standard license
- consistently high quality software
- respectful, honest, technical-based interaction
- faithful implementation of standards
- security as a mandatory feature[27]"

The organization has been said to be a forum where vendors can test interoperability of their applications in a standards-based, open infrastructure environment. A industry researcher[28] openly wondered whether this level of collegiality will continue when Apache begins development on applications that challenge commercial products. To date their main sponsorship has been by vendors to build tools that support their service offerings or hardware.

An innovation that Apache instituted is their "Incubator". The Incubator process ensures that any code issued under an Apache license has a committed set of volunteers and has a trusted code base. "The basic requirements for incubation are:

- A working codebase -- over the years and after several failures, the foundation came to understand that without an initial working codebase, it is generally hard to bootstrap a community. This is because merit is not well recognized by developers without a working code-base. Also, the friction that is developed during the initial design stage is likely to fragment the community.
- The intention to donate copyright of the software and the intellectual property that it may contain to the foundation -- this allows the foundation to obtain an irrevocable and permanent right to redistribute and work on the code, without fearing lock-in for itself or for its users.
- a sponsoring ASF member or officer -- this person will act as the main mentor, giving directions to the project, helping out in the day-to-day details and keeping contact with the incubator PMC.[29]"

These criteria catapult the effort into a "middle phase", and guarantee a certain level of success, a priori.

There are some industries that provide a web page of available business software that are developed under an open source license. Among these is the Health Care industry (http://www.openhealth.com/en/healthcare.html) and Remote Sensing and GIS (http://www.remotesensing.org/tiki-index.php and http://opensourcegis.org/).

The Commonwealth of Massachusetts has established a consortium for government organizations to share software. The Government Open Code Consortium (GOCC) allows members and observers who are government agencies to share code. The operating business model may not produce a knowledge base of industry best practices, commercial service support sector, or facilitate a community of skilled transit IT staff. These come only when an environment is established with the collaborative tools and dedicated people to support technical collaboration.

Many countries are providing guidance to their government agencies on acquiring, using and sharing OSS including Australia [Australia], Commonwealth of Massachusetts [Hamel], U.S. Department of Defense [Kenwood], Brazil, and the European Union[30]. The European Union, for example, hosts a web site[31] that provides links to open source software development projects and software. Many of the links are to existing open source development sites and software repositories like sourceforge.net and freshmeat.net

5. Business Impact

An interesting paradigm shift is observed with the widespread adoption of certain OSS. As stated earlier, the most direct effect observed by industry watchers is the improved software quality and lower costs. Reduction in risk has also occurred for some firms that replaced products produced by vendors that monopolize a marketplace. There are some clear impacts on business development, software market shifts, and procurement strategies that will affect any business entering the OSS project development arena including transit.

Software Market Shift

Linux, Apache and other OSS products have precipitated the growth of distribution and service industries devoted to management, enhancement, and installation services for OSS products. As a consequence, corporations may procure services based on value and competitive pricing. Industry watchers drives this point one step further, predicting that adoption of OSS will not put key software product vendors out of the business, rather, it will drive them to focus their revenue streams on service and support.

The Australian guidance document divides the OSS industry[32] as follows:

- Large vendors
- Established Small to Medium vendors who migrated into OSS
- Boutique consultancies and specialist who focus on specific market segment

The majority of the market is composed of the specialty consultancies that tend to be local, assembling around core client centers. Because of the recent rapid growth of OSS, most of these vendors have been in business for less than five years, although many have participated in open source development efforts.

User and Developer Relationship

Researchers also observe that the OSS community is far more invested in the quality and reliability of the product because the user and developer are more closely intertwined. Furthermore, the developer base is more widespread, global in scale because development is done as part of a virtual, internet community. Users have more access to the technical decisions and feel the immediate impact of feature decisions while under development. One might observe that the relationship drives a dynamic rapid prototyping methodology, that is, "design a little – build a little – test a little" then do it all over again.

Depending on the operating procedures of the business model, the OSS development community tends to reward technical merit, and assign "stewardship" over a project to a group of dedicated, skilled developers. Of course, this might be a self-selected group. If the group or project manager(s) do not perform, they may be replaced by others who are motivated.

Impact on Procurement Strategies

The Australian Guidance document [Australia] and Massachusetts Open Source Legal Toolkit [Hamel] are examples of how many government agencies are developing formal procurement strategies for acquiring software. These documents address open-source opportunities and risks. Many local and state agencies use open source or have developed strategies for procuring open source software. The Australian guidance document offers advice on procurement policies; issues focus on intended use, e.g., licensing differences effect how a modified open-source product can be shared. This

document and others describe when and how public sector software is eligible for open source software licensing.

As government agencies and private corporations begin to adopt OSS, they will find benefit in directly supporting these projects either through financial or skilled resources. Agencies will benefit from efficiencies derived from collaborative OSS development projects, for example, similar type government agencies must reinvent the same functionality to support their business processes. Already, the industry has seen unprecedented collaboration of industries, like the infrastructure, web development, and increasingly from the health care, to achieve efficiencies and share resources for research and development efforts. In particular, government business applications are identified, in a recent issue of *Government Technology*[33], as a potential leader in developing open source software. The reasons include: collegiality of developers, consistent business practices, and no motivation to compete for clients. The major risk is always sustaining interest over time in an OSS application.

6. Transit and Open Source

Overview of Transit Applications and Business Systems

Transit agencies must support a significant number of applications to support various Federal, state and local mandates and business processes to fulfill their mission. As a result, there is a wide range of software development areas for possible consideration as OSS candidates. In an organization like Washington Metropolitan Area Transit Authority (WMATA), the second largest rail and fifth largest bus operator in the US, there are more than 191 applications[34] that are supported by various users. Smaller, single mode agencies may have fewer, yet, there are still a significant number of applications that must be developed in-house by transit staff or hired contractors. King County Metro[35] has over 23 large, significant service oriented applications of which about 60% were developed in-house or have significant in-house components. TriMet[36] has 230 active applications of which about 150 were developed in-house.

Transit uses different types of software to fulfill its business needs such as the following:

Administrative and Accounting (General Ledger)	Paratransit Certification, Eligibility, Reservations, Dispatch and Billing	Employee and Labor Relations
Financial Planning and Budget	Capital Projects and Contract Tracking	Risk Management
Property Management	Asset and Resource Management	Vehicle Maintenance Management
Operations and Operator Payroll	Customer Information Systems and Customer Relations	Long Term Planning
On-board Systems	Service Planning	Human Resources

Commercial vendors provide many critical applications such as Scheduling, ERP, Maintenance Management and others, yet some of the software is prone to a range of risks for transit agencies. Applications vendors serving the industry in any given

business area are limited, a significant number every year are either going out of business or buying out competitors. The industry presents significant barriers to entry due to the complexity of its work rules, service industry organization, and spatial/temporal information structures. These complexities not only create major barriers to entry, but they present significant interoperability issues among existing vendor products.

For example, there is still no consensus and limited implementation experience on developing an operator interface device that supports a single point of sign-on, or real-time information software that works with a legacy Automatic Vehicle Location (AVL) System. Vendors are reluctant to solve these issues without Transit underwriting the non-recoverable engineering costs because there are small profit margins in investing and a relatively small market, as opposed to the computer industry. As a consequence, applications tend to be customized for each implementation.

Furthermore, Transit application vendors do not provide all the needed applications or required functionalities. There are some vendor products where competition is non-existent or whose functionalities do not match the expectations of their transit customers. For example, because a vendor product has not emerged that is close to being ready to use "out of the box," that supports most transit requirements for GIS-based planning and map-building, many transit agencies have built their own systems. For example, TriMet, DART, King County Metro, and OCTA have all built their own Transit GIS planning and analysis systems; all built on the same core application (ArcInfo) with significant duplicating of functionalities. Other Agencies have built functions that duplicate some of these planning applications including Miami-Dade County, Bi-State, among others.

Current statistics on buy versus build software procurements at transit agencies do not exist. However, experience has shown that there are few, if any, business applications that transit agencies can purchase and use right "off the shelf". There are always custom utilities, data interoperability issues and change orders that increase the costs, require in-house support or external service level agreements. As a consequence, more transit agencies choose to develop their own applications replacing vendor products that lock them into onerous service agreements, limit the technology transfer, restrict their data use, or run out their warranty even before the installation is complete.

Most maintenance of in-house software and custom components as well as commercial products is performed by internal Transit Agency staff. Yet, staff is overwhelmed with maintaining these systems, ensuring that they are upgraded properly, that data is transferred from one application to another, and that enhancements are done on time. Furthermore, many IT staff feel disconnected to transit as a business. Turn-over of talented programmers is high and tends to disrupt software development and enhancement projects, delaying or even ending a project. Furthermore, transit agencies are hampered by limits on new hires and salary caps that may be much lower then the regional market value. Dan Overgaard of King County Metro offers that "an OSS community might foster a connection to transit and transportation as a career path – at a mature stage, developers could have a sense of the broader community and, if they move

around, might look for another transit position rather than moving to another industry altogether, e.g., health care."

- OSS can leverage Transit Agency resources across agencies to develop applications that are currently developed by in-house IT resources at a single agency.
- OSS can leverage transit resources to develop interfaces with broader applicability among open source, in-house, or vendor applications.
- Modularized enhancements or special purpose functions and utilities can be developed and shared across agencies given an OSS development approach. For example, principles and practices from OSS can be used to spread and lessen the development burden for needed enhancements to existing software, including ad hoc reports and reporting templates and tools.
- Transit Agencies can pool their resources to provide in-house or contract maintenance and technical support services for critical business applications that are developed under an OSS process. Similar to developing OSS, agencies can share the costs of maintaining and upgrading critical business applications that are under an OSS license.
- Targeting of key resources to specific projects will support the development of applications that support interoperability.
- OSS will help improve competition in the industry.
- OSS will attract a larger community of software developers who will learn the transit business, and who then can provide service and support.

Even if only a third of transit software was developed in-house, the statistics should drive the public sector to consider open source software development projects. Many transit agencies have similar core functional requirements. OSS allows shared development of core functionality and facilitates additional agency-specific customization by a range of users. IBM has identified government business applications as a key area for open source software development[37] because of the collegiality of its software developers, and similarity of application needs. The Australian government has indicated in their guidelines that open source development whether performed in-house or through "in-house acquisition" should be considered as an alternative procurement strategy[38].

Smaller agencies may believe that they do not have the staff to support OSS development projects or maintain software in-house that is developed in an OSS environment. Yet, they often have individuals building desktop applications (without any back-up support) or they have the resources to procure, install, license, and purchase annual service contracts from a vendor. Because the OSS product is open, a number of transit organizations may share the expertise for the code, develop scripts to install the software, and maintain and modify the software to meet their specific requirements. Moreover, all agencies, small, medium and large, will benefit from the efforts of those agencies that do post their software, tools and reporting methods. A community of programmers will be available to support the agencies in dealing with the technical details and obstacles to deploying the application.

Finally, none of the studies reviewed believe that commercial off the shelf software vendors will be put out of business due to OSS. Rather, OSS competition will improve software reliability, lower costs, improve service, and grow an *open standards* environment in which all software is developed. Given these predictions, there is no reason not to pursue a transit industry open source software initiative.

Potential Benefits of Transit OSS Development Projects

Transit agency professionals have expressed a number of fears related to OSS products. Many of these fears are related to acquiring software that cannot be implemented or maintained over its expected life. Experience in developing OSS projects in other industries contests this concern. In addition, the risks are often less or not very different from those for homegrown and some COTS software. For example, one fear among transit professionals is that OSS programs will force the industry to rely on volunteers to develop and maintain critical operational code. This risk is probably less than for homegrown applications.

For homegrown applications, critical operational processes may be dependant on software developed by one or two individuals who can put a major software investment at serious risk the day they leave the organization. In fact, vendor products can go through the same staff turnover risks as in-house development projects. With OSS, the staffing risks can be spread across more organizations. Single agency, homegrown applications can also be put at greater risk than OSS, if the maintenance and enhancement resources are no longer included in the budget due to budget cuts.

In reality, if many agencies deem a project critical, they may apply their resources to develop, enhance and maintain the software. Currently, transit agencies pay significant license fees for upgrades and fixes to COTS. These resources may be applied to a pooled fund or in-kind resources to maintain and provide technical support on OSS software. Moreover, a transit agency will not be locked into a single service source to support their needs since anyone who understands the application, data and processes can "get up to speed" on the software.

There is considerable concern about whether the code can be "trusted." Has the application been subject to a formal software development process? Typically, the OSS environment provides the tools to ensure specification development, configuration control, design reviews, code reviews, unit testing, bug fixing and tracking, patching and versioning. The rigorousness of the process depends on project lead/administrator and volunteer programmers. Some OSS foundations have defined a formal process for software development. Certainly, the actual process is documented through the mailing list archives, tracking tool logs, and the design and code document wikis.

Other concerns include whether a vendor will take the code and reuse it. Use of the software, if developed using open standards, will benefit the industry by spreading these standards to proprietary applications.

Adoption of OSS in other industries demonstrates the benefit of Open Source Software Projects. Specifically, for transit agencies, OSS projects can provide significant cost savings and productivity gains:

- Increasing software quality and reducing costs through increased industry competition
- Opening service provision to multiple providers
- Opening competition in procuring software upgrades and enhancements
- Enabling interoperability through open standards and open source software (form, fit and function provisions)
- Enabling modular deployments
- Extending the life of applications
- Expanding application functionality
- Developing and sharing more tools to better meet transit business objectives

Many of these benefits may be realized as the industry invests in OSS. As is the cycle with all development projects, maturity comes over time and may not be realized for three to five years. Although, as many advanced technology deployments have demonstrated, sometimes the benefit of commercial applications is not realized for many years after deployment, and agencies do not always benefit from the success of other agency deployment innovations, reporting templates, and enhancements. A fundamental goal for developing a transit OSS initiative must be to leverage public resources from many agencies.

Critical Success Factors for a Transit OSS Initiative

There are a number of factors that will support or hurt the success of a Transit OSS Initiative. These factors impact the software design, infrastructure requirements, documentation and other areas of the system engineering process. Factors emerge from the variability of transit agencies themselves, or from nature of being a public agency. Some key characteristics that impact transit software development follow:

- Transit agencies vary in their staff skill level
    - Tools and documentation should be "accessible" to diverse skill levels
    - Applications should be built to operate in common operating environment typically deployed at transit properties
- With respect to procurement considerations, software application lifecycles are usually longer than hardware lifecycles
    - Application should be flexible to operate on multiple infrastructures and platforms
    - Software should be scaleable to hardware and other infrastructure software (e.g., server software, database)
    - Applications should be modular for ease of upgradeability and increased functionality.
- Transit agencies vary in size and service structure (e.g., service provisions, modes, etc.)

    - Software design should be scaleable both in its ability to add complexity to the data, and also to add storage and processing efficiencies
    - Software design should modularize areas that require customization based on individual, local, or regional differences.
- Transit applications function within an enterprise with a need to share information
    - Transit applications should adhere to open standards

Some researchers believe that open source software drove the use of open standards, for example had the OSS product -- Apache Web Server not been developed, then the internet would not be as ubiquitous and powerful as it is today. Open standard development efforts such as the World Wide Web Consortium and OASIS have provided a roadmap for OSS products to interoperate. Alternatively, OSS initiatives continue to drive the vision for open standards. Transit standards are not nearly at a state of maturity where they can provide a roadmap for the business needs of the transit industry. Rather, transit OSS products may provide a vision for what transit industry standards should be developed and how they can be effective.

The critical success factors that help ensure the effectiveness of transit's software and the success of the transit are as follows:

- Ensure interoperable suite of software tools
- Support development of robust and maintainable code
- Encourage business development of a support and warranty service sector
- Support a dedicated community of programmers competent in transit software

Interoperable suite of software tools. As described earlier, transit business systems are highly complex, interrelated, and dependent on data produced by other business systems. The tools developed for and by transit agencies must support the exchange of information among them.

> *A transit initiative to develop OSS should ensure that the tools are interoperable, work with existing (proprietary) applications, and implement both IT and transit open standards.*

Robust and maintainable code. Transit's primary mission is the provision of transportation services and information technology has become a critical infrastructure component. The software acquisition process, whether COTS procurement or OSS, should follow a systems engineering approach to ensure that the software is robust and the process is documented. Having robust and maintainable code is critical for ensuring successful systems. Currently, in-house IT staff is overwhelmed with their responsibilities which encompass the entire spectrum of activities including developing, upgrading, maintaining, operating and evaluating COTS and homegrown applications. Some agencies cannot hire the skilled staff that is needed to support their existing systems. Leveraging the resources needed to develop the core components across multiple organizations will reduce the pressure on existing staff while expanding the diversity of skills available to any one collaborating organizations.

A transit initiative to develop OSS should ensure that the development projects are open to all, supported by in-house IT staff and focus on building reliable, scalable, modular and high quality code. Differing functional approaches must be documented and archived to support the variety of needs required of the users. The OSS projects should be developed in an environment that supports the "best practices" in OSS.

A separate transit initiative may also be formed to supplement the OSS with (1) outreach and training and (2) demonstration projects for early-code releases.

Support and warranty services. Procuring technical support is a necessity. In this era of government downsizing, in-house staff may be supplemented by hiring contract staff or outsourcing. OSS initiatives may reduce some of the staffing burdens for support. Software warranties may not be all that they seem; some vendor applications are still being installed even as warranties expire.

In transit industry OSS guidance documents, public and private sector entities should jointly develop procurement language and level of service requirements to align expectations for a supportive business development environment.

Dedicated community of programmers. Open source software development projects become successful when there is a community of dedicated, skilled professionals contributing to developing, testing and deploying the code, and users who provide feedback on how well the code meets their requirements. Generating and sustaining this community of programmers will be the *major* critical success factor.

Transit agencies should dedicate a percentage of their IT staff resources or hire programmers to develop and maintain OSS projects. The public sector is the major beneficiary and the users of the software. Private companies and programmers will join the community when they see benefits for themselves, such as business opportunities or career fulfillment.

A transit open source initiative should be framed to encourage, support, and nurture these critical areas. Section 5 discusses differing business models that support open source development. Listed below are the key objectives that should be included in a transit OSS initiative:

- OBJ #1: Adherence to OSS "best practices"
  - OBJ #1.1: Develop product roadmap and specification
  - OBJ #1.2: Establish and use an OSS development infrastructure (includes wiki, change log, testing environment, bug tracking, mailing list, outreach and other collaborative tools)
  - OBJ #1.3: Build a strong, grassroots developer community that drives a formal software development process approach
- OBJ #2: Transit management support for OSS development and procurement

    - OBJ #2.1: Commitment of resources
    - OBJ #2.2: Policy support for OSS acquisition strategy
- OBJ #3: Consistent transit industry policy support for OSS acquisition and service provision
- OBJ #4: Framework to ensure technical requirements
    - OBJ #4.1: Incubator-type environment to ensure success of OSS programs
    - OBJ #4.2: Interoperability and adherence to open standards
- OBJ #5: Framework as a legal entity
    - OBJ #5.1: To pool resources, provide outreach, guidance, testing environment and certification
    - OBJ #5.2: Provide legal shelter for liability and patent infringement

7. Approach for a Transit Industry Open Source Initiative

There are many diverse goals for an OSS development initiative, such as the following:

- Improving the quality of operating software
- Increasing the effectiveness of deploying and maintaining the code
- Reducing costs of procuring and maintaining the application
- Meeting the goals of the decision makers who use these tools
- Reducing risk to the agency

These goals are similar to the goals for acquiring any software product. In fact, developing open source software has a number of similarities with developing in-house, commercial or proprietary software. Senior management retains its role of oversight. The significant differences are the contributions and collaborative efforts volunteers, and of course, the free use and redistribution of the source and executable code.

The benefits of OSS to transit will occur in a phased approach. Already many transit agencies are using some OSS such as PERL and the Apache Web Server. The next wave of benefits will come after the transit industry becomes better educated about OSS. Some of the education will come from the US government, other international and local government efforts and from transit specific education. The fullest benefits will be realized when there is a transit-specific OSS initiative in place, similar to what other industries are implementing. A phased approach is proposed below for discussion within the transit industry.

The critical success factors and key features described in Section 6 suggest a coherent OSS organizational approach for transit that is similar to the Apache Software Foundation. ASF was organized as a non-profit to pool funds and establish a governance structure. It has been very successful in developing and vetting projects that are interoperable, promote open standards, and redistribute open source applications that are now central to the Internet. Furthermore, the ASF structure may provide the legal liability and indemnification cover to corporate and individual participants. A model similar to the ASF model may be adapted by the transit industry to ensure the

development of interoperable software, to identify key pooled development projects, prioritize enhancements and maintain the software to ensure a high degree of reliability over the product life, and protect their legal interests.

Stage 1: Preliminary Trial

Objectives: There are two key objectives for the preliminary trial, (a) engage technical IT staff in adopting OSS that is targeted for transit agencies in demonstration projects, (b) educate a number of transit agencies about OSS and encourage them to use OSS that meets transit-specific business needs. Transit agencies already use OSS in their technology infrastructure. This trial should acquaint senior management to the extent and applicability of OSS to address their business needs. This stage is typical of the SourceForge communities of programmers discussed in "Developer Business Models".

The steps to achieve this stage consist of the following:

- Using existing software developed by a transit agency or multiple transit agencies (e.g., Dynamic Timetable Generator), assign an OSS license and set up the software on one of the OSS project web sites. Among the agencies, identify and assign a person (staff or consultant) to lead the effort.
- Develop a "roadmap" with the recommended enhancements.
- Publicize the project and recruit volunteers to help enhance the product through industry publications, distribution lists, personal contacts, and conference papers. Outreach should also include background and impact of OSS in the IT industry.
- The OSS project web site will provide the infrastructure to distribute and host the development process.
- After 6 to 12 months, evaluate the process. Determine, for example, how many programmers contributed to the development, how many people downloaded the software, who were they and how did they use the application, did they contribute enhancements back to the process, what issues were raised.
- Develop a checklist and guidance document for the industry on how to develop an OSS project.

Stage 2: Interim Organizational Development

Objectives: In this stage, establish a forum where transit agencies can share software without barriers. Understand the legal issues related to sharing, distributing and modifying applications in an open consortium. The Government Open Code Consortium (GOCC) established a forum for government agencies to share software in a membership organization. This organization may be an option to maintain an application code repository for software developed by the public sector.

The steps to achieve this stage consist of working with transit agencies and transportation organizations to publish their software under an OSS license or within a "members only" environment. A key part of this stage is to set up the software using the OSS project collaborative resources such as are available on open source software development sites. These projects may fall into multiple categories:

- Collaborative efforts of multiple transportation organizations that need to ensure interoperability and/or new technology not yet developed by commercial vendors (e.g., multimodal customer information).
- Unique products developed by a transit agency.
- A vendor that may offer a product as OSS as a marketing strategy (e.g., IBM's and Netscape's approach to compete with Microsoft).

Stage 3: Establish a Transit OSS Consortium

Objectives: Based on the early collaborative efforts, obtain consensus from a core group of organizations and individuals to form a consortium based on a common set of principles and critical success factors such as those described above. Table 3 describes some of the features of a Transit OSS Initiative that meet and/or support the Critical Success Factors and Objective statements of Section 6.

Table 3: Features of a Transit OSS Initiative

Features	Transit OSS Initiative (TOSSI)	Addresses OBJ #
Licensing	TOSSI developed software should remain OSS, recommend adopting LGPL license type	2.2
Legal entity type	501(C)3 Non-profit	5.1
Steward Organization / Staffing	University Lab (with equipment and graduate students in transportation); Programmers, board, members: no paid staff Policy governors may hire "maintainers" to maintain software in an OSS development environment.	1
Governance Structure	2 independent tiers; technical and policy; needs to be worked out. All volunteers	5
Membership	Technical: technical merit; Policy: institutional merit.	1.3, 2.1
Project Structure /Teams	Project managers come from membership. Teams include anyone who volunteers.	1, 2.1, 4
Technical Infrastructure	Initially use SourceForge or one of the other OSS development web sites to support development efforts. Incorporate best practices/processes for specification, design, code and installation documents.	1
New project review process	Create incubator process similar to ASF. Give up copyright to "Foundation."	4
Special Issues	Incorporate programs for (1) Outreach & Training; (2) Policy Guidance; (3) Best Practices for OSS Processes; (4) Guidebooks on OSS products.	3, 5.1
Pooled funds	Grants to support legal requirements, infrastructure and special issues (not programmers). Solicit funds from individuals, public and private sector.	5.1

The industry has several software projects that potentially could be used as a Stage 1 OSS development project. They include the Dynamic Timetable Generator, TriMet's Transit Tracker, and NYSDOT's Schedule Data Management System (SDMS).

While a proposed framework for implementing OSS in transit was presented above, the integration of OSS into transit is in the early phases. More lessons can be learned from OSS initiatives that are taking place in other industries. A number of open questions still remain that need to be resolved, including the following:

1. Can an agency apply an open source license to code developed internally? Are there State laws that impact the assignment of licenses to in-house software development activities?
2. Are there Federal restrictions on government funded software development projects that affect the ability to develop OSS?
3. Who will handle initial outreach and recruitment?
4. How is a two-tiered governance structure set up? What is fair?
5. What strategies should be established to mitigate commercial obstructionists?

In closing, there appear to be clear opportunities for transit in the area of OSS, but given the newness of the topic to transit, education and outreach, lessons learned and a proof-of-concept need to be addressed before it will be viewed as a viable option on a wide-spread basis.

8. Resources and References

[ASF] "FAQ: How the ASF works." www.apachesoftwarefoundation.org

[Australia] "A Guide to Open Source Software for Australian Government Agencies Developing and Executing ICT Sourcing Strategy." April 2005. http://www.sourceit.gov.au/sourceit/oss

[Castells] Manuel Castells, "Introduction: Open Source as Social Organization of Production and as a Form of Technological Innovation Based on a New Conception of Property Rights." From INNOVATION, INFORMATION TECHNOLOGY AND THE CULTURE OF FREEDOM: THE POLITICAL ECONOMY OF OPEN SOURCE. posted 31/01/05 11:22, Porto Alegre, 2005. http://openflows.org/article.pl?sid=05/01/31/2028221

[Hamel] Linda Hamel. "Nine Ways to Protect your State From the Legal Risks Posed by the Use of Open Source Software." Part of the Open Source Legal Toolkit. September, 2004.

[http://www.mass.gov/portal/index.jsp?pageID=itdsubtopic&L=5&L0=Home&L1=Policies%2c+Standards+%26+Legal&L2=Legal+Guidance+%26+Documents&L3=Software+Licensing+%26+Development&L4=Open+Source+Legal+Toolkit&sid=Aitd]

[Kenwood] Carolyn A. Kenwood, "A Business Case Study of Open Source Software." MITRE PRODUCT, July 2001. http://www.mitre.org/work/tech_papers/tech_papers_01/kenwood_software/kenwood_software.pdf.

[Mozilla] Paul Festa, "Mozilla releases Firefox 1.0." CNET News.com, Published on ZDNet News: November 9, 2004, 4:34 AM PT. URL: http://news.zdnet.com/2100-9588_22-5443931.html

[OSI] "The Open Source Definition". Version 1.9. www.opensource.org

[Perens] Perens, Bruce. "The Emerging Economic Paradigm of Open Source." February 16, 2005. http://perens.com/Articles/Economic.html

[Peterson] Peterson, Shane, "Proving its Mettle, Open source software has come a long way, baby." Government Technology, Volume 18, Issue 5, May 2005.

[Raymond] Raymond, Eric S. "The Magic Cauldron. " June 1999. [www.catb.org/~esr/writings/magic-caldron/magic-caldron.html]

Endnotes

[1] Although the Open Source Initiative (www.opensource.org) was created a year later in 1998.

[2] Castells, p., 3.

[3] Ibid. GNU as in "GNU's not UNIX", p., 4

[4] Ibid. p., 3.

[5] "Apache Web Server Captures Market Share With Open Source" By Charles Babcock, InformationWeek (March 29, 2004)

[6] Forecasted adoption rates for Linux range from a 6% market share by 2007, "Linux desktop market share to reach 6% in 2007" IT Facts (February 13, 2004) http://www.itfacts.biz/index.php?id=P723, to 20% by 2008, "Global IT firm predicts Linux will have 20% desktop market share by 2008" by Chris Gulker, NewsForge (August 14, 2003) http://www.newsforge.com/article.pl?sid=03/08/13/1424212. The growth is expected to be among enterprises rather than consumers.

[7] http://www.wired.com/news/infostructure/0,1377,64065,00.html

[8] http://www.sei.cmu.edu/opensystems/faq.html. This article provides a comprehensive description of open systems, standards and interoperability issues.

[9] *I don't want you to talk, Mr. Gates* ,-Posted by Dana Blankenhorn @ 10:27 am, General Development, http://blogs.zdnet.com/open-source/

[10] Perens, p., 2.

[11] Ibid.

[12] Adapted from Australia, p., 31.

[13] See http://www.apache.org/

[14] Australia, p., 7.

[15] Kenwood.

[16] "OASIS (Organization for the Advancement of Structured Information Standards) is a not-for-profit, international consortium that drives the development, convergence, and adoption of e-business standards." [http://www.oasis-open.org/home/index.php]

[17] "The World Wide Web Consortium (W3C) develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential. [http://www.w3.org/]

[18] "A Wiki or wiki (pronounced [w□ki□], [wi□ki□] or [vi□ki□]) is a web application that allows users to add content, as on an Internet forum, but also allows *anyone* to edit the content; "Wiki" also refers to the collaborative software used to create such a website (*see Wiki software*)." [http://en.wikipedia.org/wiki/Wiki]

[19] Australia, p., 16

[20] Ibid., p., 39]

[21] Ibid., p., 15]

[22] Mozilla Public License 1.1, Section 2.1

[23] based on a total of 63,188 projects. SourceForge.net is one of the largest web sites for OSS development projects [http://sourceforge.net/softwaremap/trove_list.php?form_cat=14] as of 10 May 2005.

[24] "Who maintains Zope?" [http://www.zope.org/WhatIsZope]

[25] http://www.apache.org/foundation/how-it-works.html#meritocracy

[26] See http://www.apache.org/foundation/how-it-works.html

[27] Ibid.

[28] "Apache Software Foundation Can be a Technology Powerhouse", N. Drakos. 11 August 2003. ID Number: T-19-1930.

[29] http://www.apache.org/foundation/how-it-works.html

[30] http://europa.eu.int/information_society/activities/opensource/index_en.htm

[31] http://www.euspirit.org/index.php

[32] Australia, pf., 12.

[33] Peterson, pf., 18.

[34] From their Enterprise Architecture Planning study performed in November 2001.

[35] Information provided by Dan Overgaard, King County Metro Transit on 12 May 2005.

[36] Information from TriMet Application Catalog on 16 May 2005.

[37] Peterson, p., 18

[38] Austrailia, p., 13.