



New York State Technology Enterprise Corporation

Open Source Code Review Requirements
And
Related Expenses

New York State
Board of Elections

Submitted to:

New York State Board of Elections
40 Steuben Place, Albany NY 12207

November 28, 2007
Version 2

Table of Contents

1.	THE TOPIC DEFINED.....	1
2.	THE SBOE PROPOSAL.....	1
3.	DEFINITION OF “OPEN SOURCE” CODE	1
4.	DEFINITION OF “OPEN SOURCE” CODE DEVELOPMENT AND SUPPORT	3
4.1	PRODUCT DEVELOPMENT	3
4.2	PRODUCT MAINTENANCE	3
4.3	PRODUCT SUPPORT	3
5.	SOURCE CODE TESTING PROCESS.....	3
5.1	DEFINITIONS	3
5.2	EXPECTED VOLUME OF SOURCE CODE FOR VOTING SYSTEMS	4
5.3	SOURCE CODE REVIEW PROCESS.....	4
5.3.1	<i>Definition of Static code analysis.....</i>	4
5.3.2	<i>NYSTEC recommendation for source code review approach.....</i>	5
6.	ESTIMATED EXPENSES.....	5

1. THE TOPIC DEFINED

During a recent Board of Elections commissioners meeting a proposal was submitted to provide some relief for testing expenses for voting systems that are based on “Open Source” code. The current estimate for complete certification testing of one voting system is approximately \$1,000,000 to be funded by the voting system vendors.

NYSTEC has been asked to provide a summary of what the source code testing would entail and an estimate of the cost burden that could be placed on the state.

It is not the intent of this document to discuss the pros and cons of “Open Source” vs. proprietary source code.

2. THE SBOE PROPOSAL

In order to provide some background information the following is NYSTEC’s summary interpretation of the proposal.

The proposal provides for an “Open Source” voting system vendor to not incur any expenses associated with source code testing and review. They would however still be accountable for expenses related to all other testing including functional, hardware, quality, security and usability testing.

We believe the proposal is pretty clear in its intent since it

- Uses the “Open Source” Initiative’s (OSI) accepted definition of “Open Source” to define the criteria for being considered as an “Open Source” voting system, and
- It clearly indicates that the relief from expenses only includes the source code review portion of testing and does not provide relief for any of the other testing.

3. DEFINITION OF “OPEN SOURCE” CODE

The following is an expanded definition of “Open Source” which hopefully adds some additional points of clarification.

Taken from the “Open Source” Initiatives (OSI) web site:

Open source doesn’t just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.

NOTE: It is the intent of the “Open Source” community to develop quality software for free distribution.

4. DEFINITION OF “OPEN SOURCE” CODE DEVELOPMENT AND SUPPORT

4.1 Product development

Open Source by definition is source code that is developed in an open and collaborative environment. Open source is typically created by a community of individuals with the common goal of developing a product to accomplish a specific task. A good example of open source development is the various versions of the Linux operating system which are developed by a group of individuals and free to use by anyone.

4.2 Product maintenance

Maintenance for the given product is provided by the same group of individuals. Users of the product are permitted and encouraged to provide input on problems (bugs) that are encountered. They are also encouraged to provide formal requests for features or enhancements via a defined process.

4.3 Product support

When a formal business provides financial support to develop a product they will follow the “Open Source” rules for development and maintenance of a product but generate revenue by providing a “fee for service” for user support. An example would be a business that funds the development of a Linux operating system will provide the operating system for free but offer a “fee for service” such as help desk support for users to call in for help.

Under the rules for “Open Source” anyone is permitted to use and modify a product but is not permitted to resell or openly distribute the modified product. In addition, unless the modifications to a product are done formally through the community of developers they are on their own for support.

5. SOURCE CODE TESTING PROCESS

5.1 Definitions

First some definitions from a prior paper on source code testing that still apply even in the “Open Source” arena.

Source code: A series of statements written in a human-readable computer programming language. The series of statements, often consisting of several files or modules are then converted to a computer executable format producing a computer program.

Code Inspection: A review to determine version, completeness, consistency, correctness, modifiability, structure, traceability, modularity, and construction.

Code Examination: A review of source to insure it is unmodified, contains no embedded or malicious code, contains no security vulnerabilities, and functionality to determine testing requirement.

Full Source Code Review: A review for full compliance to the 2005 EAC VVSG. This includes all standards dealing with formatting in Volume I Section 5.2.2 – 5.2.7 and Coding Conventions in Volume II Section 5.4.2 as well as any other standards violations.

Types of Reviews: There will be two independent source code reviews performed.

- **Security Review:** The security review will implement code inspection and code examination during the scope of this project. It will apply to all source code including third party source code and COTS code that is supplied for compilation or generated by another product.
- **Functional Review:** The functional review will implement a full source code review to all source code supplied by the vendor. The functional review will also implement code inspections and code examination as applied in this document for third –party software and COTS products.

5.2 Expected volume of source code for voting systems

Using examples of source code that had been submitted by voting system vendors for certification testing over the last year we were able to arrive at an estimate of the average size of the source code effort. Based on a review of the previously provided source code:

- The typical/average voting system includes approximately 800,000 lines of code
- The typical/average EMS system includes approximately 650,000 lines of code.

5.3 Source code review process

The approach the voting system test labs use for source code review is completed against text files that include the source code rather than against a running system. This approach is known as *Static Code Analysis*

5.3.1 Definition of Static code analysis

Static code analysis is the analysis of computer software that is performed without actually executing programs built from that software (analysis performed on executing programs is known as dynamic analysis) In most cases the analysis is performed on some version of the source code and in the other cases some form of the object code. The term is usually applied to the analysis performed by an automated tool, with human analysis being called program understanding or program comprehension.

The sophistication of the analysis performed by tools varies from those that only consider the behavior of individual statements and declarations, to those that include the complete source code of a program in their analysis. Uses of the information obtained from the analysis vary from highlighting possible coding errors (e.g., the lint tool) to formal methods that mathematically prove properties about a given program (e.g., its behavior matches that of its specification).

A growing commercial use of static analysis is in the verification of properties of software used in safety-critical computer systems and locating potentially vulnerable code.

5.3.2 NYSTEC recommendation for source code review approach

To provide the necessary source code review of “Open Source” voting system source code we recommend a static code analysis as the preferred method to accomplish a full and effective analysis of source code.

The static analysis includes two approaches.

- Use of automated tools to test coding practices such as indentation, commenting, variable names, and many other good coding practices. Other tools can be utilized to evaluate code for known coding vulnerabilities such as unused sections of code, bad branches within the code, back doors, etc.
- Manual review of source code to evaluate the outcome of the automated tools analysis and look for other coding vulnerabilities that tools can not uncover.

6. ESTIMATED EXPENSES

Based on a few studies such as the “State of California source code study” and other industry standard guidelines the average experienced code reviewer can evaluate anywhere from 100 to 1000 lines of code per hour. The low end assumes a full manual review of source code. The high end assumes a combination of manual review and the use of automated tools (as outlined above).

Using these assumptions and also assuming multiple people would be working together on the review it appears:

1. The review of the average voting system source code can be accomplished in approximately 4 to 6 weeks, and
2. The average Election Management System can be done in approximately 3 to 5 weeks.

NOTE: It is also assumed that this is one pass through the source code.

Using industry standard hourly rates for source code review the approximate cost per voting system could range between \$20,000 to \$75,000 depending on the complexity of the source code, the error rate (fatal flaws in the source code), and the amount of times the vendor is allowed to fix flaws.

Since there is no way to accurately predict the actual time required we suggest that you assume the high side for budgeting purposes.

DRAFT