IE URL Lock Team: Dongwhan Kim, Annie Zhao, Steven Lawrance **Nov 13th Report**

Our team presently knows which problems to solve and how to get there. Free and open access to the Internet permits users to access a theoretically unbounded amount of information, though in some settings, such freedom conflicts with acceptable use policies as well as parents' child-raising strategies. By restricting information access either to either a limited set of allowed sites or by blocking out only specific sites, information technology administrators and parents can control which web sites are permitted and which ones are not.

To fulfill the needs of this market for information technology administrators and parents, this project will extend the Internet Explorer (IE) URL Lock browser helper object (BHO). The improvements will increase its configurability beyond those who know Perl-compatible regular expressions and expand its browser coverage to include Mozilla Firefox. Other improvements will support the needs of the new configuration user interface as well as a new optional mode that permits all web sites while blocking only those that match a block list. The existing IE URL Lock BHO presently will only block all web sites except for those that are allowed. With these improvements, more information technology administrators and parents can effectively use the IE URL Lock to control web site access.

To plan our project, a high-level runtime architectural view was created, as depicted in Figure 1.

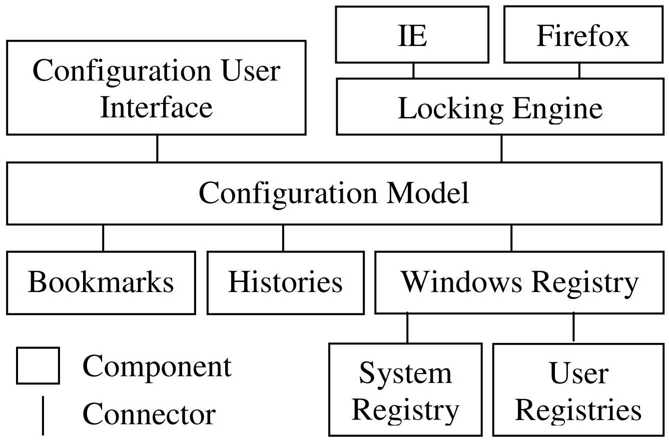

Figure 1: Runtime view of improved IE URL Lock

The configuration user interface follows a model-view-controller pattern with the controller and the view in the configuration user interface architectural component and the model in the configuration model. The view and controller are split into separate XML user interface language (XUL) and JavaScript files, respectively, to separate the user interface definitions from the user interface implementations. In the Mozilla Firefox version of the extension, the configuration user interface can run natively. The Internet Explorer version will launch the configuration user interface using Mozilla's XUL Runner. Both will use the same configuration model.

Both Internet Explorer and Mozilla Firefox will instantiate their respective versions of the locking engine, which is either a browser helper object or an extension. The locking engine will use the same configuration model that the configuration user interface uses to load the blocking configuration. Due to technical reasons, the Internet Explorer version of the locking engine will use its own read-only subset of the configuration model in C++ rather than the JavaScript version that we will build, though they will both reference the same Windows registry keys.

The configuration model will be able to read bookmarks and histories from both Internet Explorer and Mozilla Firefox, possibly including those from all users on the system. That data will be used by the user interface. Because the configuration model can load and save the IE URL Lock configurations for multiple users on the system, it will mount and unmount user registry hive files to modify and lock down other users' IE URL Lock configurations. This permits multiple-user configurations without requiring Microsoft's ActiveDirectory, which most home users do not use.

IE URL Lock Team: Dongwhan Kim, Annie Zhao, Steven Lawrance **Nov 13th Report**

To plan our project, our runtime architectural components were broken down into feature sets that were based on the functionalities that our group collaborated on. Using historical data from both the existing IE URL Lock C++ code and Steven Lawrance's Project 4 JavaScript files, estimates for lines of code (LOC) were devised for each feature, starting with each part that forms a feature. As this time, there are 6 medium, 1 huge, 1 large, and 1 small features with 4,600 estimated lines of code, which puts the LOC/person at about 1,530. The huge feature can be split up if needed due to its size. After analyzing the estimates, features were assigned to team members based on workload, skills, and interests. The current work breakdown is not finalized, though the team can proceed with it.

As a next step, our team will create the interfaces that sit at each connector in our runtime architectural view. After these interfaces are defined, a skeleton implementation can be created that the team can work from, enabling parallelism in our development process and, eventually, reducing the software integration time. Our team will consider whether or not it makes sense to employ other software engineering methodologies such as unit tests and integration tests.

The feature table appears below, which lists each of the nine features as well as the parts that compose each feature. The estimated LOC counts as well as feature sizes also appear in the table. The current owner of each feature appears below the each feature's name.

Feature / Person	Part	Est LOC	Size
1. Home user / easy interface	XUL: The "main/home" and "settings" tabs	330	
Dongwhan Kim	JavaScript: Controller code for the XUL, which updates the UI when settings change, loads settings using the underlying configuration reader, writes settings using the underlying configuration writer, populates UI lists such as user names and history/bookmark lists, and prevents editing if the user is not an administrator (saving won't work anyway as Windows enforces that security)	1200	
	Feature Total	**1530**	**Huge**
2. Configuration reader and writer (JavaScript)	Configuration reader, which reads in and parses the configuration from the registry, both system-wide and per-user	180	
Steven Lawrance	Configuration writer, which writes the configuration to the registry, both system-wide and per-user	180	
	Feature Total	**360**	**Medium**
3. Configuration native library (XPCOM using C or C++)	User registry loader and unloader, which might require a small XPCOM object to call Win32 API functions	50	
Steven Lawrance	Data source of user IDs, names, local administrator group membership (true or false), and their registry files	70	

	Flag that indicates whether or not the current user has administrative rights to the IE URL Lock registry (read-only)	30	
	Lock down the registry setting that permits users to disable the IE URL Lock so that they cannot do that, but do that only to non-administrative users	50	
	Interface to the native Perl-compatible regular expression library (PCRE) for maximum regular expression compatibility with what IE URL Lock uses (versus Firefox's built-in RegExp object, which differs from PCRE)	50	
	History data source for Internet Explorer history from all users using IEnumSTATURL via COM (this can be part of the XPCOM object as it's on the C++ side of things)	100	
	XPCOM overhead code	150	
	Feature total	**500**	**Medium**
4. Bookmarks and history data sources (JavaScript)	Bookmarks data source for Internet Explorer favorites from all users, which involves reading text files and retrieving the "URL=" lines	130	
Annie Zhao	Bookmarks data source for Firefox bookmarks from all users, which involves reading anchor tags from the bookmarks.html file of each user's default profile	150	
	History data source for Firefox history, which can be done in JavaScript using code using a Mork file format parser. more information exists at https://bugzilla.mozilla.org/show_bug.cgi?id=241438. The easiest implementation can skip the details of the format and only consider parenthesis groups that begin with "(*=http", where * can be any series of numbers and letters. Regular expressions can help. Time information isn't needed, so the Mork parser can be simple. Other Mork information exists at http://philwilson.org/blog/2005/01/how-to-export-firefoxs-history-to-text.html	130	
	Feature Total	**410**	**Medium**
5. Core engine for IE URL Lock (C++)	Add an "enable all web browsing by default" mode	30	
Annie Zhao	Add the ability to optionally show a dialog box when blocked instead of a web page	30	
	Feature Total	**60**	**Small**
6. User interface integration for IE URL Lock (C++)	Add a right-click menu to disabled links that can open the configuration window with the clicked-on link in the Add Location input box	100	

Steven Lawrance	Change the appearance of disabled links (should the cursor be different? Should the underlining be removed? I'm not sure..)	200	
	Add a menu item to Tools that launches our configuration editor via XUL Runner	70	
	Feature Total	**370**	**Medium**
7. Core engine for the Firefox version (JavaScript)	Write the blocking code as a Firefox extension, which reads in its configuration from the registry and makes blocking decisions using the Perl-compatible regular expression library (PCRE)	400	
Steven Lawrance	**Feature Total**	**400**	**Medium**
8. User interface integration for the Firefox version (JavaScript)	Add a right-click menu to disabled links that can open the configuration window with the clicked-on link in the Add Location input box	70	
Annie Zhao	Change the appearance of disabled links (should the cursor be different? Should the underlining be removed? I'm not sure..)	200	
	Launch the configuration window using either a menu item in Tools or through the "Options" button in the "Extensions" window	70	
	Feature Total	**340**	**Medium**
9. Expert user interface	XUL: The "main/expert" tab	230	
Annie Zhao	JavaScript: Controller code for the XUL	400	
	Feature Total	**630**	**Large**

The following table lists the work breakdown totals by team member. Note that the configuration user interface controller code might get split into multiple tasks.

Person	Features	LOC	Sizes
Dongwhan Kim	1	1530	1 Huge
Annie Zhao	4, 5, 8, 9	1440	1 Small, 2 Medium, 1 Large
Steven Lawrance	2, 3, 6, 7	1630	4 Medium

With these features and task assignments, our team plans to achieve our goal of enabling parents and information technology professionals to easily control web site access for their users and do so within our project's time constraints.