A Reference Architecture for Web Browsers

Alan Grosskurth and Michael W. Godfrey
School of Computer Science
University of Waterloo
Waterloo, ON N2L 3G1 Canada
{agrossku,migod}@uwaterloo.ca

Abstract

A reference architecture for a domain captures the fundamental subsystems common to systems of that domain as well as the relationships between these subsystems. Having a reference architecture available can aid both during maintenance and at design time: it can improve understanding of a given system, it can aid in analyzing trade-offs between different design options, and it can serve as a template for designing new systems and re-engineering existing ones. In this paper, we examine the history of the web browser domain and identify several underlying phenomena that have contributed to its evolution. We develop a reference architecture for web browsers based on two well known open source implementations, and we validate it against two additional implementations. Finally, we discuss our observations about this domain and its evolutionary history; in particular, we note that the significant reuse of open source components among different browsers and the emergence of extensive web standards have caused the browsers to exhibit "convergent evolution."

Keywords: *Software architecture, reference architecture, software evolution, component reuse, web browser.*

1 Introduction

A *reference architecture*[4] for a domain captures the fundamental subsystems and relationships that are common to the existing systems in that domain. It aids in the understanding of these systems, some of which may not have their own specific architectural documentation. It also serves as a template for designing new systems by identifying areas in which reuse can occur, both at the design level and the implementation level. While reference architectures exist for many mature software domains such as compilers and operating systems, no reference architecture has been proposed yet for web browsers.

The web browser is perhaps the most widely used software application in history and has evolved significantly over the past fifteen years; today, users run web browsers on diverse types of hardware, from cell phones and tablet PCs to regular desktop computers. A reference architecture for web browsers can help implementors to understand trade-offs when designing new systems, and can also assist maintainers in understanding legacy code.

In this paper, we present a reference architecture for web browsers that has been derived from the source code of two existing open source systems and validate our findings against two additional systems. We explain how the evolutionary history of the web browser domain has influenced this reference architecture, and we identify underlying phenomena that can help to explain current trends. Although we present these observations in the context of web browsers, we believe many of our findings represent more general evolutionary patterns which apply to other domains.

2 The Web Browser Domain

The World Wide Web (WWW) is a shared information system operating on top of the Internet. Web browsers retrieve content and display from remote web servers using a stateless and anonymous protocol called HyperText Transfer Protocol (HTTP). Web pages are written using a simple language called HyperText Markup Language (HTML). They may be augmented with other technologies such as Cascading Style Sheets (CSS), which adds additional layout and style information, and JavaScript, which allows client-side computation. Plugins are invoked for content that the browser cannot handle natively, such as Java applets. Browsers typically provide other useful features such as bookmarking, history, password management, and accessibility features to accommodate users with disabilities.

Tim Berners-Lee wrote the first web browser in 1991; it was text-only and also served as an HTML editor. Soon after, another text-only browser called Lynx was adapted for use with the WWW. In 1993, an easy-to-use, graph-

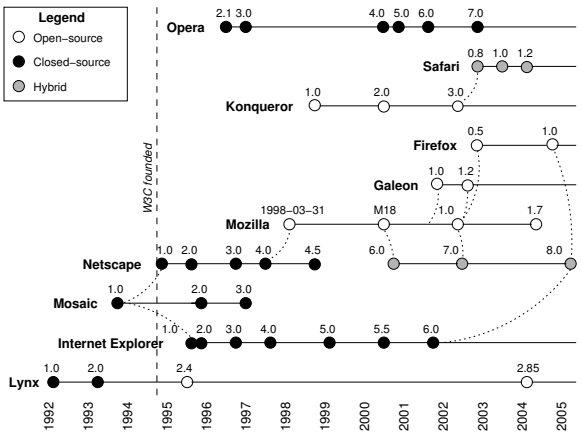

Figure 1. Web browser timeline

ical browser called Mosaic was released, and a company called Spyglass was created to commercialize its technologies. The program's author, however, left to co-found his own company, Netscape. In 1994, the World Wide Web Consortium (W3C) was founded to promote interoperability among web technologies. In 1995, Microsoft released Internet Explorer (IE), based on code licensed from Spyglass, igniting a period of intense competition known as the "browser wars." Microsoft eventually dominated the market, and Netscape open-sourced their browser under the name Mozilla in 1998. The closed source browser Opera also appeared in the mid-1990s. Figure 1 shows a timeline of the various releases of several popular web browsers.

A large number of Mozilla variations have appeared, reusing the browser core but offering alternative design decisions for user-level features. Firefox is a standalone browser that provides a more streamlined alternative to Mozilla's complex and integrated user interface. The open source Konqueror browser has also been reused; Apple has integrated its core subsystems into their OS X web browser, Safari, and Apple's modifications have in turn been reused by other browsers. Internet Explorer's closed source engine has also seen reuse: Maxthon, Avant, and NetCaptor each provide additional features such as tabbed browsing and ad-blocking. Although each browser engine produces similar results, there can be subtle differences as to how web pages look and behave. Netscape 8, based on Firefox, takes advantage of this: the user may switch between IE-based rendering and Mozilla-based rendering on the fly.

3 Deriving the Reference Architecture

Using the source code and available documentation for two different web browsers, we derived a reference architecture for the web browser domain. We used a process similar to that which is described by Hassan and Holt in [8]. Two mature browser implementations were selected

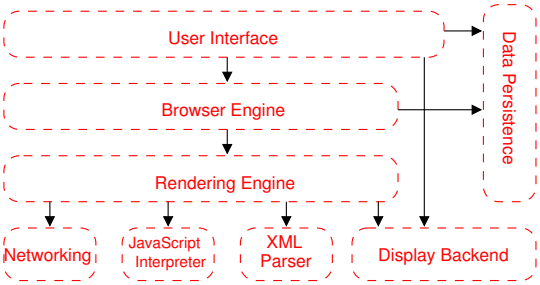

Figure 2. Reference architecture for web browsers

and, for each browser, a conceptual architecture was proposed based on domain knowledge and available documentation. The concrete architecture of each system was then extracted from its source code, using the QLDX[1] reverse engineering toolkit, and used to refine the conceptual architecture. A reference architecture was then proposed based on the common structure of these refined architectures, and it was validated against two other browser implementations.

The reference architecture we derived is shown in Figure 2; it comprises eight major subsystems plus the dependencies between them: (1) the *User Interface*; (2) the *Browser Engine*, an embeddable component that provides a high-level interface for querying and manipulating the *Rendering Engine*; (3) the *Rendering Engine*, which performs parsing and layout for HTML documents, optionally styled with CSS; (4) the *Networking* subsystem; (5) the *JavaScript Interpreter*; (6) the *XML Parser*; (7) the *Display Backend*, which provides drawing and windowing primitives, user interface widgets, and fonts; and (8) the *Data Persistence* subsystem, which stores various data associated with the browsing session on disk, including bookmarks, cookies, and cache. Mozilla and Konqueror were used to derive the reference architecture because they are mature systems, have reasonably large developer communities and user bases, provide good support for web standards, and are entirely open source. Due to space constraints, we will not show Konqueror's architecture nor discuss it in detail.

The Mozilla Suite was open-sourced by Netscape in 1998, and since then most of the system has been completely redesigned or rewritten and a large number of new features have been added. Mozilla's key design goals are strong support for web standards, support for multiple platforms, and fast page rendering. The mapping of Mozilla's conceptual architecture onto the reference architecture is shown in Figure 3. The *User Interface* is split over two subsystems, allowing for parts of it to be reused in other applications in the Mozilla suite such as the mail/news client. All data persistence is provided by Mozilla's profile mechanism, which stores both high-level data such as bookmarks and low-level data such as a page cache. Mozilla's *Render-*

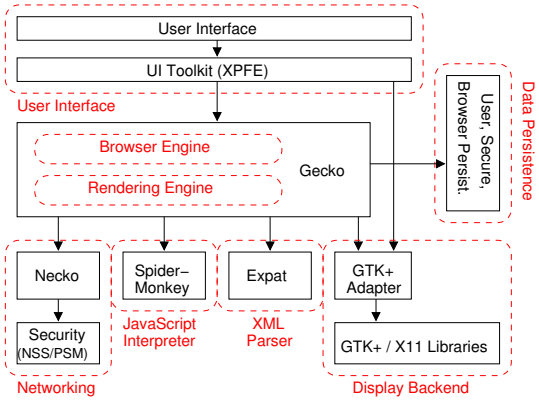

Figure 3. Architecture of Mozilla

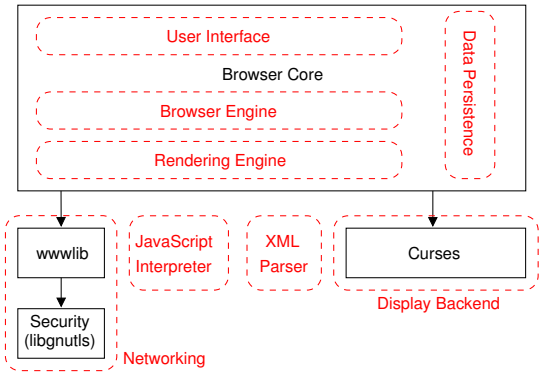

Figure 4. Architecture of Lynx

ing Engine is larger and more complex than that of other browsers. One reason for this is Mozilla's outstanding ability to parse and render malformed or broken HTML. Another reason is that the *Rendering Engine* also renders the application's cross-platform user interface. The UI is specified in platform-neutral Extensible User Interface Language (XUL), which in turn is mapped onto platform-specific libraries using specially written adapter components. This architecture distinguishes Mozilla from other browsers in which the platform-specific display and widget libraries are used directly, and it minimizes the effort required to support multiple, diverse platforms.

4 Validating the Reference Architecture

Two additional implementations were chosen against which to validate the reference architecture: Lynx and Safari. Lynx was chosen because it is the oldest web browser still regularly used and maintained. Safari was chosen because it represents an interesting mix of open and closed source technology—Apple has adapted Konqueror's core subsystems to use OS X libraries and added a proprietary user interface. Due to space constraints, we will not show Safari's architecture nor discuss it in further detail.

Lynx is a one of the most popular text-only browsers in use today. It predates the WWW, first serving as an interface for an "organization-wide information system." Custom hypertext capabilities were then added, followed by support for the Gopher protocol. Finally, support for WWW protocols was grafted on, making Lynx into a true web browser. This incremental development process has resulted in a system composed of small fragments of code with no coherent overall structure. Furthermore, much of the code is low-level and platform-specific, increasing its complexity.

The mapping of Lynx's conceptual architecture onto the reference architecture is shown in Figure 4. The `libwww` library provides a wide variety of functionality such as HTML parsing and support for both the HTTP and FTP protocols. The `libgnutls` library provides optional support for secure protocols. The `curses` library is used to display and navigate information on character-cell terminals. Lynx's conceptual architecture shows a clear separation between its three main subsystems: browser core, networking, and display backend; however, there is no clear separation between the *User Interface*, *Browsing Engine*, *Rendering Engine*, and *Data Persistence* subsystems. This is likely because they are less complex due to Lynx's text-only nature—the rendering engine outputs web pages in linear form rather than attempting to layout elements at appropriate coordinates, and the user interface relies solely on keyboard input rather than dealing with menus, widgets, and mouse events. Lynx does not contain a *JavaScript Interpreter* or an *XML Parser* because these are relatively modern features which are not supported yet. The lack of modularity and the text-only nature of Lynx make its conceptual architecture much simpler than our reference architecture; nevertheless, it still has some common elements with the reference architecture.

5 Summary and Related Work

There are several reasons why a web browser's architecture would differ from our reference architecture. Some of the subsystems in the reference architecture may be implemented as a single subsystem for simplicity, while others may be spread across multiple subsystems in the web browser for greater flexibility. New subsystems may be added to provide additional capabilities, while others may be omitted to make the browser more lightweight.

Table 1 shows various statistics about the different web browsers studied. Konqueror achieves nearly the same degree of standards-compliance as Mozilla with one-quarter of the amount of code. Lynx, while smaller than the other browsers, is still five times larger than Links, a more recent text-only browser with a comparable feature set. We are unable to obtain complete size information for Safari because of its closed source components; the numbers shown corre-

Table 1. Approximate web browser statistics

Project	Vers.	Language	Files	kLOC	MB*	Start
Mozilla	1.7.3	C++, C	10,500	2,400	29	1998
Konq.	3.3.2	C++	3,145	600	17	1996
Lynx	2.8.5	C	200	122	2.1	1989
Safari	1.2	C++, Obj C	>750	>136	>2.1	2003

*Represents the compressed tarball size in megabytes.

spond only to the open source parts. We are currently investigating how the Mosaic, Dillo, and Galeon browsers correspond to our reference architecture and hope to examine a web browser designed specifically for embedded devices.

Reference architectures have been proposed for other domains, including real-time train control systems[4], avionics[2], and web servers[8]. Product line architectures[3] are similar to reference architectures, although they generally represent a group of systems intended to be produced by a single organization, while reference architectures represent the entire spectrum of systems in a domain. Various aspects of Mozilla's architecture and development process have been studied.[7][9][5].

6 Conclusions

We have examined the history and evolution of the web browser domain, developed a reference architecture for web browsers based on two existing implementations, and validated this reference architecture by mapping it onto two additional implementations. We have also observed several interesting evolutionary phenomena; namely, emergent domain boundaries, convergent evolution, and tension between open and closed source development approaches.

As the web browser domain has evolved, its conceptual boundaries—both external and internal—have become increasingly more defined. However, there are still discrepancies as to the nature of these boundaries. Microsoft has claimed that Internet Explorer is a fundamental part of the Windows operating systems, providing rendering functionality to other applications. This has posed a problem for third-party browsers such as Netscape that sought to compete with IE. Similarly, email, usenet, and ftp client functionalities have been integrated with the Netscape and Mozilla browsers, discouraging competition from external clients. It will be interesting to observe how the web browser domain adapts to support embedded devices such as cell phones and PDAs, where limited memory makes it undesirable to deploy multiple competing applications.

The large amount of effort devoted to creating high-quality open source browser implementations has had an interesting influence on the domain. During the "browser wars," proprietary extensions were added to core components in order to attract customers. Today, increased pressure to comply with standards has led to reuse of core components; browsers instead differentiate themselves with user-level features. However, these features seem to be easily duplicated; for example, tabbed browsing and pop-up blocking were once innovative features but are now commonplace. These observations suggest that the web browser domain is exhibiting a form of *convergent evolution*[6].

The availability of mature browser components has also resulted in tension between open and closed source development approaches. Mozilla's open source engine has been reused in numerous applications, both open and closed source. Similarly, Konqueror's open source engine has been used as the basis for Safari. Although not required by the licence, Apple has contributed its changes to open source components back to the community. Conversely, Internet Explorer represents a closed source engine that can potentially be embedded in an otherwise open source product. Netscape 8 strikes a balance by embedding both the Mozilla and IE engines, allowing users to switch on the fly.

While we have seen applications composed of both open and closed source components before, the interaction usually takes place on the perimeter, as is the case with closed source binary modules for the Linux kernel. We believe the heterogeneous combination of core open and closed source software components within individual systems makes the web browser domain unique and interesting.

Acknowledgements We thank Ali Echihabi for his contributions to an earlier project out of which this paper has grown, as well as Ric Holt for his feedback and advice.

References

[1] QLDX reverse engineering toolkit home page. `http://swag.uwaterloo.ca/qldx`.

[2] D. Batory, L. Coglianese, M. Goodwin, and S. Shafer. Creating reference architectures: An example from avionics. In *Proceedings of the 1995 Symposium on Software Reusability (SSR '95)*, pages 27–37, 1995.

[3] P. Clements and L. M. Northrop. *Software product lines: practices and patterns*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[4] W. Eixelsberger, M. Ogris, H. Gall, and B. Bellay. Software architecture recovery of a program family. In *Proceedings of the 20th International Conference on Software Engineering (ICSE '98)*, pages 508–511, 1998.

[5] M. Fischer, M. Pinzger, and H. Gall. Analyzing and relating bug report data for feature tracking. In *Proceedings of the 10th Working Conference on Reverse Engineering (WCRE '03)*, pages 90–99, 2003.

[6] D. J. Futuyma. *Evolutionary Biology*. Sinauer Associates, Sunderland, MA, USA, 3rd edition, 1998.

[7] M. Godfrey and E. H. S. Lee. Secrets from the monster: Extracting Mozilla's software architecture. In *Second International Symposium on Constructing Software Engineering Tools (CoSET '00)*, June 2000.

[8] A. E. Hassan and R. C. Holt. A reference architecture for web servers. In *Proceedings of 7th the Working Conference on Reverse Engineering (WCRE '00)*, pages 150–160, 2000.

[9] A. Mockus, R. T. Fielding, and J. Herbsleb. Two case studies of open source software development: Apache and Mozilla. In *ACM Trans. Software Engineering and Methodology*, pages 11(3), 309–346, 2002.