Apps vs. Open Web: The Battle of the Decade

Tommi Mikkonen
Department of Software Systems
Tampere University of Technology
Tampere, Finland
tommi.mikkonen@tut.fi

Antero Taivalsaari
Advanced Development & Technology
Nokia Corporation
Tampere, Finland
antero.taivalsaari@nokia.com

Abstract—**Today, both desktop and mobile software systems are usually built to leverage resources available on the World Wide Web. However, in recent years desktop and mobile software have evolved in different directions. On desktop computers, the most popular application for accessing content and applications on the Web is the web browser. In mobile devices, in contrast, the majority of web content is consumed via custom-built native web apps. This divergence will not continue indefinitely. We anticipate that in the 2010's we will witness a major battle between two types of technologies: (1) native web apps and (2) Open Web applications that run in a web browser or some other standards-compliant web runtime environment. This "*Battle of the Decade*" will determine the future of the software industry – as well as the future of software engineering research – for years to come.**

Keywords–web applications, web programming, web-based software development, software engineering, web engineering, open web

I. INTRODUCTION

Although the World Wide Web has existed less than twenty years, the Web has already transformed our lives in countless ways. These days, everyday artifacts and services such as documents, photos, music, videos and newspapers are widely available on the Web. Online banking and stock trading have become commonplace. Various documents that used to be difficult to access, such as municipal zoning documents, government budget documents or tax records, are now readily available on the Web. Entire industries such as banking, financial services, electronics and book retailing, photography, and music distribution have undergone dramatic transformations. Web-based services such as Facebook and Twitter have altered the meaning of social life. The Web is even having a profound impact on politics and democracy, shaping the future of nations all over the planet.

The World Wide Web has also had a considerable impact on the software industry. These days, both desktop and mobile software systems are usually built to leverage resources available on the Web, with the objective that the same content can be accessed effortlessly from different types of terminals. However, in recent years desktop and mobile software systems have evolved in rather different directions. On desktop computers, the most popular application for accessing content and applications on the Web is the web browser. In mobile devices, in contrast, the majority of web content today is consumed via custom-built native web applications, or "apps" for short.

In this paper, we anticipate that in the 2010's we will witness a major battle between two types of technologies: (1) native web apps and (2) Open Web applications that run in a web browser or some other standards-compliant web runtime environment. The former approach implies the use of binary software and traditional software engineering practices, while the latter approach implies that conventional software engineering methods and practices will be replaced by technologies created for web development. This "*Battle of the Decade*", as we call it, will determine not only the future of the software industry, but the future of software engineering research as well.

This paper builds on a number of earlier papers [4, 5, 6, 7, 11, 12, 13, 14]. Many of the topics in this short paper have been covered more extensively in those earlier papers.

The rest of this paper is structured as follows. In Section II we provide a brief discussion on the evolution of the Web as a software platform, and then focus on the ongoing battle between native apps and Open Web applications in Section III. In Section IV we outline the research challenges that arise from the two divergent paths. In Section V we draw some final conclusions.

II. EVOLUTION OF THE WEB AS A SOFTWARE PLATFORM

Over the past twenty years, the World Wide Web has evolved from a document sharing system to a massively popular, general purpose application and content distribution environment – in short, the most powerful information distribution environment in the history of humankind. This evolution has taken place in a number of evolutionary phases or eras [14]. Note that here we intentionally focus on the evolution of the Web *as a software platform*. When viewed from other angles – e.g., from the viewpoint of online banking or music or video distribution – the history of the Web would look somewhat different.

In the first era – *the Web as a document environment* – the programming capabilities of the Web were very limited, reflecting the origins of the Web as a document sharing and distribution environment. In the second era – *the Web as an application environment* – the software development capabilities of the Web started emerging, with different technologies competing with each other vigorously. In the third era that is unfolding currently – *the Web as the application environment* – we believe that the landscape of the software industry will change dramatically, as the balance shifts irrevocably from binary end user software to web-based software. Note that these three eras are by no

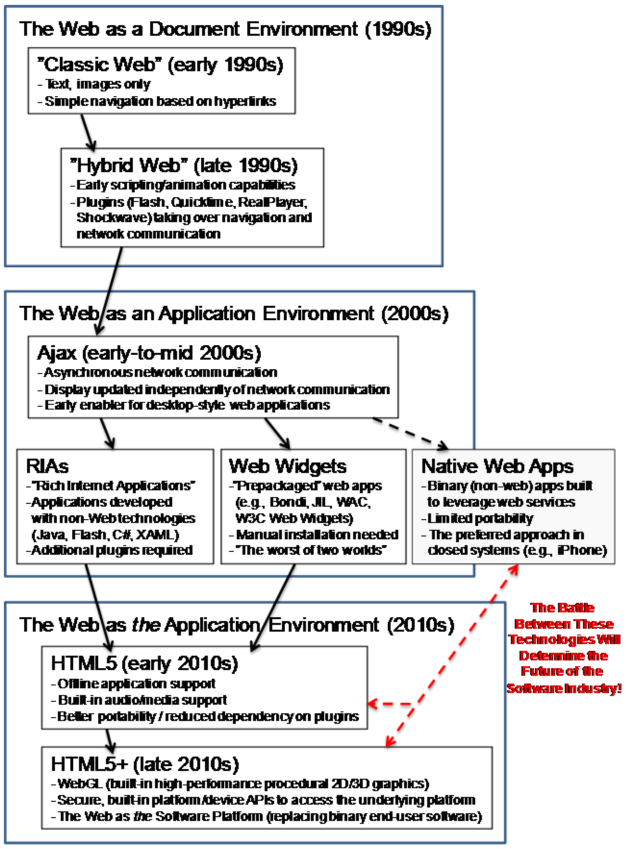

The Web as a Document Environment (1990s)
"Classic Web" (early 1990s)
- Text, images only
- Simple navigation based on hyperlinks
"Hybrid Web" (late 1990s)
- Early scripting/animation capabilities
- Plugins (Flash, Quicktime, RealPlayer, Shockwave) taking over navigation and network communication
The Web as an Application Environment (2000s)
Ajax (early-to-mid 2000s)
- Asynchronous network communication
- Display updated independently of network communication
- Early enabler for desktop-style web applications
RIAs
- "Rich Internet Applications"
- Applications developed with non-Web technologies (Java, Flash, C#, XAML)
- Additional plugins required
Web Widgets
- "Prepackaged" web apps (e.g., Bondi, JIL, WAC, W3C Web Widgets)
- Manual installation needed
- "The worst of two worlds"
Native Web Apps
- Binary (non-web) apps built to leverage web services
- Limited portability
- The preferred approach in closed systems (e.g., iPhone)
The Web as *the* Application Environment (2010s)
HTML5 (early 2010s)
- Offline application support
- Built-in audio/media support
- Better portability / reduced dependency on plugins
HTML5+ (late 2010s)
- WebGL (built-in high-performance procedural 2D/3D graphics)
- Secure, built-in platform/device APIs to access the underlying platform
- The Web as *the* Software Platform (replacing binary end-user software)
The Battle Between These Technologies Will Determine the Future of the Software Industry!

Figure 1. Evolution of the Web as a Software Platform
(for high-quality image, see http://lively.cs.tut.fi/WebEvolution.png)

means mutually exclusive. Rather, web pages and applications representing all three eras coexist on the Web today. A visual summary of the different eras is provided in Figure 1. We will ignore many of the details in this paper, since they are not relevant for the main theme of the paper.

III. THE BATTLE OF THE DECADE

The key point about the evolution of the Web presented in Figure 1 is the current, ongoing battle between native web apps and HTML5-based Open Web applications.

A. *Native Apps*

Custom-built native apps have become one of the dominant ways people use web services. For instance, on Apple's highly successful iPhone and iPad devices, as well as on Google's Android devices, the users typically access Facebook, Twitter, and many other popular web services using custom-built native apps rather than with the web browser. Such native apps are not really web applications at all; however, they use the same network protocols to access the backend services as the web browser does.

There are good reasons for native web apps to exist. While the underlying needs to communicate and access information are the same in desktop and mobile environments, the way people consume content and use applications with different types of terminals and devices are fundamentally different. In the mobile space, the time span of the users' actions is usually significantly shorter than in the desktop space; the users wish to perform rapid, focused actions instead of long-lasting sessions; actions must be simple yet focused, and they must accomplished with ease, using only a minimal number of keystrokes or finger presses, often while the user is walking, driving a car or is somehow otherwise distracted by other activities. The different usage modalities and smaller screen sizes have a significant impact on application design; generic web pages geared towards laptop or desktop computer users are not usually ideal for mobile use. In addition, performance issues or network connectivity issues can make web applications nearly unusable in mobile devices. The conventional web browser simply was not designed for such use.

By using the native graphics libraries, the look-and-feel of apps can be customized specifically to the needs of the application and the device; the applications can also leverage device-specific features much more comprehensively than a pure web application could. The downside of such apps is that they are *strictly platform-specific*. Apps developed for the iPhone run only on Apple devices, so several different implementations – composed with different platform-specific tools – are needed if the app is to run also on Android, Blackberry, Symbian, or other commonly used target platforms [15]. In many cases a separate app is needed for each of the different versions of the target device. Such fragmentation is what effectively killed Sun's (now Oracle's) once highly successful Java ME platform [10].

Another source of fragmentation is that different apps, developed by different parties, commonly assume different ways of interaction. For instance, gestures that work in a certain fashion in one application may imply totally different functions in other applications. This can be confusing for the user, and in the end lead to additional requirements on how applications should be defined, together with associated style guides and so forth.

Finally, unlike pure web applications, a native app requires conventional installation. The user must usually download the application binary from a specific location, such as Apple's App Store (*http://store.apple.com/*). In order to introduce new features, the user must typically download and install a totally new version or upgrade the application explicitly by device-specific means. This is clumsy and inconvenient for the user, e.g., since the application or the entire device may be partially unavailable while the download and upgrade is in progress.

B. *Open Web*

Following the Open Web principles laid out in the *Mozilla Manifesto* [8], web applications should be built on technologies that are open, accessible and as interoperable as possible, and should run in a standards compatible web browser without plugins, extensions or custom runtimes. In December 2010, Tim Berners-Lee – the inventor and founder of the World Wide Web – published an article in which he called the current trend towards custom-built native web apps "disturbing", because that trend is dividing information into separate content silos that are isolated from each other [2]. Such content is off the Open Web, and usually under the control of an individual company. Typically, you cannot bookmark, tweet or e-mail a link to such a page using a standard browser. Rather, you must explicitly download, install and use (and later upgrade) a vendor-specific app from a vendor-specific app store for each device platform in order to access such content.

Open Web applications have various benefits. For instance, they require no installation or manual upgrades, and they can be deployed instantly worldwide. A web application published in Tampere (Finland), say, is instantly and equally available in Tallahassee (Florida, USA), Tandragee (Ireland) or Taree (New South Wales, Australia) without explicit installation. The Open Web principles will allow application development and instant worldwide deployment without middlemen or distributors. Conventional binary applications are at a major disadvantage when compared to web-based software that can be deployed instantly across the planet.

So far, a number of obstacles have hindered the development and deployment of full-fledged, truly interactive web applications. The obstacles have been especially apparent in the mobile device space. We have analyzed the problems in earlier papers [4, 5, 6, 12]. However, new standards such as HTML5 and WebGL will eliminate many of the limitations in this area.

The forthcoming *HTML5* standard [16] complements the capabilities of the existing HTML standards with numerous new features. Although HTML5 is a general-purpose web standard, many of the new features are aimed squarely at making the Web a better place for desktop-style web applications. Examples of features that support desktop-style applications include offline applications that can be run even when an active network connection is not available (http://www.w3.org/TR/offlinewebapps/), a simple storage mechanism that behaves like a simple key-value database, allowing textual data to be stored locally in the computer/device, Canvas API that provides a 2D drawing canvas for procedural, interactive graphics, and built-in audio and video support.

WebGL (http://www.khronos.org/webgl/) [3] is a cross-platform web standard for hardware accelerated 3D graphics API developed by Mozilla (http://www.mozilla.org) and Khronos Group (http://www.khronos.org/), and a consortium of additional companies including Apple, Google and Opera. The main feature that WebGL brings to the Web is the ability to display 3D graphics natively in the web browser without any plug-in components. Unlike with earlier technologies such as Flash, O3D, VRML and X3D, with WebGL the 3D capabilities are integrated directly in the web browser, meaning that 3D content can run smoothly and portably in any standards-compliant browser. The possibility to display 3D graphics natively in a web browser is one of the most exciting things happening on the Web recently.

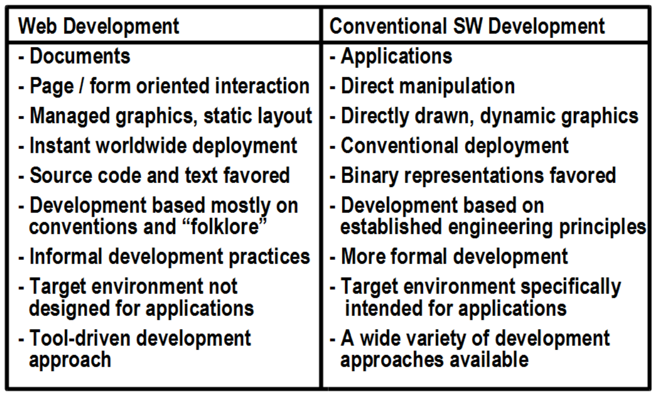
While HTML5 and the related W3C standard activities play a critical role in turning the Web into a serious application platform, it is important to note that the feature set offered by an HTML5-compliant web browser is still somewhat incomplete for real-world applications. As depicted in Figure 1, our prediction is that another major round of standardization will be necessary in mid-to-late 2010s to establish a more complete web application platform. We refer to such standard work informally as "*HTML5+*", that is, the next major version/successor of the HTML5 Specification. A critical goal in that work will be to more comprehensively "virtualize" the underlying operating system and device capabilities, as well as ensure that the necessary security mechanisms are in place to access the platform and device capabilities securely.

IV. Impact on the Software Industry and Software Engineering Research

The document-oriented origins of the Web have led to an *impedance mismatch* between web development and conventional software engineering. In this section we take a brief look at this impedance mismatch and its implications for the software industry and software engineering research.

A. Web vs. Conventional Software Development: The Impedance Mismatch

As we have discussed in earlier papers, a historical impedance mismatch exists between web development and software engineering. This impedance mismatch reflects the fact that the World Wide Web was originally designed to be a document distribution environment – not a software platform. The differences are highlighted in Figure 2.



Web Development	Conventional SW Development
- Documents	- Applications
- Page / form oriented interaction	- Direct manipulation
- Managed graphics, static layout	- Directly drawn, dynamic graphics
- Instant worldwide deployment	- Conventional deployment
- Source code and text favored	- Binary representations favored
- Development based mostly on conventions and "folklore"	- Development based on established engineering principles
- Informal development practices	- More formal development
- Target environment not designed for applications	- Target environment specifically intended for applications
- Tool-driven development approach	- A wide variety of development approaches available

Figure 2. Impedance Mismatch Between Web Development and Conventional Software Development

In the remaining parts of the paper, we will consider the two divergent paths that the evolution of the Web may take as a result of the Battle of the Decade. The implications for the software industry and software engineering research are entirely different depending whether the balance tilts towards native apps or the Open Web.

B. Scenario 1: Native Apps Will Dominate

Many people seem to take it for granted that especially in the mobile industry native apps will continue to dominate. For instance, in a September 2010 Wired magazine article Chris Anderson and Michael Wolff claimed that the Web is already dead [1], because for the vast majority of web services such as e-mail, news, Facebook and Twitter, users will prefer custom-built native applications (e.g., Flipboard for iPad) over open, unfettered web browser access.

The success of native apps is not entirely unexpected. Native apps enjoy considerable success partly because of commercial reasons (e.g., because it tends to be easier to monetize closed rather than open platforms) and partly because of technical reasons (e.g., because it is easier to define new APIs and optimize overall system behavior in world in which the platform is owned and controlled by a single vendor.

Superficially, from the viewpoint of software engineering, the native apps scenario is business as usual. Since the development model in this scenario revolves around the creation of rather conventional binary applications that are written, installed and run in a well-known fashion, existing design, integration and testing practices and methods can be used without major changes.

However, under the surface there are numerous things that need attention. To begin with, mobile devices are subject to significantly more variations and fragmentation than conventional desktop computers. For instance, screen size differences, different interaction and input mechanisms, memory and processing power limitations/differences and intermittent network connections create additional challenges for developers. In the area of Java ME development – the once dominant mobile application platform – some game companies reported that they had to create over a thousand different variants of their applications for different devices!

These days, the mobile industry seems to be headed to an equilibrium in which two or three native platforms will dominate the industry. The companies controlling those platforms place a lot stricter restrictions on the device capabilities than the Java ME specifications ever did. Nevertheless, the application developers will still have to create a large number of variants of their applications if they expect their applications to be available on all the major platforms, devices and countries; even if the developer is targeting only one major platform such as Apple's iPhone, internationalization and localization may still require effort.

In general, the successful creation of commercial native web apps places a lot of requirements on product family management. In order to offer an attractive app portfolio that covers all the different platforms, tools for managing fragmentation in massive scale are needed. Those tools must be able to provide cross-platform support that enables the use of the same code in different platforms, and is capable of recognizing and handling the micro-level fragmentation issues (bugs and "features") between different devices that use the same platform. The tools must also be able to take into account the different installation practices for different target platforms. For instance, for Apple's iPhone and iPad devices, application installation can only take place via Apple's Web Store.

The topics discussed above are just a tip of the iceberg for a proper research agenda for Scenario 1.

C. *Scenario 2: Open Web Will Dominate*

The starting point for Scenario 2 is that the transition towards web-based software development will continue and will eventually have a profound impact not only for desktop software but mobile software development as well.

The victory of Open Web applications is by no means guaranteed, though. There are still numerous issues that plague the development of web applications, and for mobile devices especially. In our earlier papers, we have divided those problems broadly into the following categories:

(1) software engineering principle violations,
(2) usability and user interaction issues,
(3) networking and security issues,
(4) browser interoperability and compatibility issues,
(5) development style and testing issues,
(6) deployment model issues, and
(7) performance issues.

We will not revisit all the categories in this brief paper. Rather, we highlight a number of topics that we believe should be high on the research agenda for Scenario 2.

First, the transition from binary applications to pure web applications will result in a shift away from static programming languages such as C, C++ or C# towards dynamic programming languages such as JavaScript, PHP or Python [9]. Since mainstream software developers are often unaware of the fundamental development style differences between static and dynamic programming languages, there is a need for education in this area. Developers need to be educated about the evolutionary, exploratory programming style associated with dynamic languages, as well as agile development methods and techniques that are available for facilitating such development.

Second, the software deployment practices for web applications are entirely different from conventional binary software. Web applications are distributed primarily in the form of source code, not binaries. Any application updates that are posted on the Web are immediately accessible to anybody anywhere on the planet. This "instant gratification" dimension will revolutionize the deployment and distribution of software applications, and will enable "nano releases", i.e., software releases that may occur multiple times per day or even every few minutes. For instance, recently Netflix (http://www.netflix.com/) reported that they commonly publish updates to their web applications up to six times per day! One of the main challenges in the deployment area is to define a model that addresses the fundamental changes in the nature of applications: applications that remain "always on", the ever-shortening nano release cycles, and the "perpetual beta syndrome", i.e., applications that will stay in continuous development mode indefinitely [11].

Third, in the testing area there is an increased need for code coverage testing methods to ensure that all the parts and execution paths of the applications are tested appropriately. Since web applications consist of pieces that are loaded dynamically without any static compilation, type checking or linking, it is quite possible for significant pieces of the applications to be missing at runtime. This feature, when combined with the lack of well-defined interfaces and general fragility that characterize web-based software [6, 12], leads to many interesting research topics and challenges, especially when developing mashups and mashware, i.e., software that dynamically combines content and components published in different sites all over the world.

In general, many of the development and deployment practices that are common in web-based software development go against the grain or even obliterate many of the established software engineering principles. So far, there has not been enough discourse between the software engineering and web engineering communities; this is definitely an area for future improvement.

V. Conclusions

In this paper, we have argued that the ongoing "*Battle of the Decade*" between native web apps and HTML5-based Open Web applications will determine the future of the software industry and software engineering research. We started the paper by summarizing the evolution of the Web as a software platform, followed by an overview of native web apps vs. Open Web applications that run in a web browser or some other standards-compliant web runtime environment. We then presented two alternative scenarios for the future of the industry based on the possible outcomes of the battle, as well as highlighted interesting areas for future research.

References

[1] C. Anderson, and M. Wolff, "The Web is Dead: Long Live the Internet," *Wired*, Sep 2010, pp.118-127, 164-166.

[2] T. Berners-Lee, "Long Live the Web: a Call for Continued Open Standards and Neutrality," *Scientific American*, vol 303, nr 4 (Dec), 2010, pp.56-61.

[3] Khronos Group, *WebGL Specification*, Editor's Draft, August 8, 2011. URL: http://www.khronos.org/registry/webgl/specs/latest/

[4] T. Mikkonen and A. Taivalsaari, "Web Applications – Spagetti Code for the 21st Century," Proc. 6th ACIS International Conference on Software Engineering Research, Management, and Applications (SERA'08), IEEE Computer Society Press, 2008, pp. 319-328.

[5] T. Mikkonen and A. Taivalsaari, "Creating a Mobile Web Application Platform: The Lively Kernel Experiences," Proc. 24th Annual ACM Symposium on Applied Computing (SAC'09), ACM Press, 2009, pp. 177-184.

[6] T. Mikkonen and A. Taivalsaari, "The Mashware Challenge: Bridging the Gap Between Web Development and Software Engineering," Proc. 2010 Workshop on Future of Software Engineering Research (FoSER'10), ACM Press, 2010, pp. 245-249.

[7] T. Mikkonen and A. Taivalsaari, "Reports of the Web's Death Are Greatly Exaggerated," *IEEE Computer*, vol 44, nr 5, 2011, pp.30-36.

[8] Mozilla, *The Mozilla Manifesto*, 2011. URL: http://www.mozilla.org/about/manifesto.en.html

[9] L. D. Paulson, "Developers Shift to Dynamic Programming languages," *IEEE Computer*, vol 40, nr 2 (Feb) 2007, pp. 12-15.

[10] R. Riggs, A. Taivalsaari, J. Van Peursem, J. Huopaniemi, M. Patel, and A. Uotila, *Programming Wireless Devices with the Java™ 2 Platform, Micro Edition* (2nd Edition). Addison-Wesley (Java Series), 2003.

[11] A. Taivalsaari, T. Mikkonen, D. Ingalls, and K. Palacz, "Web Browser as an Application Platform," Proc. 34th Euromicro Conference on Software Engineering and Advanced Applications (SEAA'08), IEEE Computer Society, 2008, pp. 293-302.

[12] A. Taivalsaari and T. Mikkonen, "Mashups and Modularity: Towards Secure and Reusable Web Applications," Proc. 1st International Workshop on Social Software Engineering (SoSEA'08, L'Aquila, Italy), Department of Software Systems, Tampere University of Technology, Report 1, 2008, pp 21-28.

[13] A. Taivalsaari, T. Mikkonen, M. Anttonen, and A. Salminen, "The Death of Binary Software: End User Software Moves to the Web," Proc. 9th Conference on Creating, Connecting and Collaborating through Computing (C5'11), IEEE Computer Society, 2011, pp. 17-23.

[14] A. Taivalsaari and T. Mikkonen, "The Web as a Platform: The Saga Continues", Proc. 37th Euromicro Conference on Software Engineering and Advanced Applications (SEAA'11), IEEE Computer Society, 2011, pp. 170-174.

[15] T. Wasserman, "Engineering Issues in Mobile Application Development," *Invited talk at MobiCase'10*, October 25-27, 2010, Santa Clara, California, USA.

[16] World Wide Web Consortium, *HTML5 Specification*, W3C Editor's Draft, September 10, 2011. URL: http://www.w3.org/TR/html5/

About the Authors

Dr. Tommi Mikkonen is a Professor of Computer Science at Tampere University of Technology, Finland. Tommi has pioneered the education of mobile software development in Finland, and he has arranged numerous courses on software engineering and mobile computing. Tommi's current research interests include cloud computing, web programming, embedded systems, and mashup development.

Dr. Antero Taivalsaari is a Distinguished Engineer at Nokia. Antero is best known for his seminal role in the design of the Java™ Platform, Micro Edition (Java ME platform) – one of the most successful commercial mobile software platforms in the world, with over three billion devices deployed so far. Since 2006, Antero's research has focused on web application technologies and web-based software development especially for mobile devices.

Together, Tommi and Antero lead the Lively Web Programming Research Team at Tampere University of Technology. For further information and a full list of research team publications, refer to http://lively.cs.tut.fi/.