Lessons learned from long-term micro-browser software architecture evolution

Toshihiko Yamakami
Research and Development Division, ACCESS
2-8-16 Sarugaku-cho, Chiyoda-ku, Tokyo, Japan
yam@access.co.jp
tel:+81-3-5259-3534 fax:+81-3-5259-3599

Abstract— **With recent advances in devices, networking infrastructure and content, the mobile Internet is becoming a solid reality. The embedded software engineering needs to cope with this emerging demand. With the rapid changes in the mobile Internet landscape, it is difficult to identify how the software design methods are managed in this domain. There are multiple design factors in the embedded software architecture: portability, alignment to CPU and memory constraints, performance, extensibility, reliability and time to market. In order to identify the fits between these factors and requirements, the author describes characteristics and design focus of four different micro-browser software architectures, which evolves for a decade to meet the digital appliance requirements. The author summarizes the design focus shift from the early stage to the recent stage. From this retrospective analysis, the author describes the relationship among software architecture, underlying environments and mobile-specific content, which is unique in the mobile micro-browser software engineering.**

I. Introduction

The mobile Internet keeps its penetration into every facet of life and every corner of the globe. In Japan, the mobile Internet users reached 78 millions at the end of 2005, out of the 90 million wireless telephony users. 40 million users shifted to IMT-2000 in last 3 years, highly aimed at high-speed data services. This rapid pace forces the embedded software engineers to cope with the two-fold difficult challenges: convergence to the PC Internet and resource-critical embedded software development. This two-fold challenge needs extensive consideration for a wide range of trade-offs in software development. The author discusses how the micro-browser software architecture evolves in the last 10 years. From the retrospective analysis of the architecture changes, the author presents a design factor evolution model to highlight the ground design of portable micro-browsers.

II. Purpose of Research and Backgrounds

A. Purpose of the Research

The emergence of the mobile Internet gives a wide range of challenges in the micro-browser software engineering. It was a small piece of software in the early stage. However, it becomes one of the most complicated software part for network-enabled digital appliances, like mobile handsets. It is extremely important to construct a stable and extensible software framework to cope with the challenges like a) variety of execution environments and b) multimedia enabler extensions. Embedded software engineering started decades ago, however, the focus on the portable mutlimedia software was not highlighed until very recently. There are not many cases for studying design principles for portable and extensible multimedia software platforms in the mobile Internet. Using the time-dimension based retrospective view on multiple software architectures, the author aims to identify the design evolution.

B. Related Works

The typical embedded software does not need to consider extensibility because the platform designer knows about applications running in the specific domain. The mobile Internet software opens the new challenge that the embedded software engineering to cope with the open challenges of the downloadable third-party intellectual properties. No one can design the whole set of the Internet converged software. Therefore, the platform software needs to cope with the wide variety of the extensibility existing in the PC Internet. The author described the software engineering challenges in the open and constraint embedded Internet [1].

The embedded network software engineering started in 1970's. The portability, design productivity, compactness were the key features in the past days. The advances in the mobile computing accelerate the energy consciousness in this domain. The variety of the underlying environment prevented the integrated architecture design studies. Energy-aware OS research is studied in a range of fields recently. OS-directed power management is done for energy-aware scheduling by [2]. Dynamic voltage scaling systems were studied for the mobile environment in [3]. Due to the resource constraint, the application research explored DSP and system-on-chip issues e.g. [4]. The mobile portable device emergence stimulates the various energy efficient system designs like [5]. The multimedia application like MP3 is a target of energy-efficient profile methodology [6]. The compiler approach for code size constraint was pursued by Naik [7]. The code size reduction by the modular approach was discussed by Yam [8]. The past research focuses on the low-level system design because the

1-4244-0340-5/06/$20.00 ©2006 IEEE

traditional embedded software is closed and components are under control of a single task manager. It was based on the monolithic assumption in the embedded software engineering. The monolithic and extensible nature of the micro-browser software engineering was discussed by Yam [1]. The architecture trade-off design to cope with emerging open multi-media application demands is rarely studied in the research literature.

III. Design Factors

Software engineering needs to consider various constraints inherited in the embedded environment. The following aspects need to be addressed:

- Portability
- Constraints in memory size and CPU power
- Performance
- Extensibility
- Reliability
- Short time to market

In the digital appliances including mobile handsets, the underlying environment varies from device to device. To cope with the cost-performance demands, a wide range of different execution environments are used. With the memory capacity restrictions, the underlying operating systems (OS) varies. Memory size and CPU power are also restricted to meet the cost-performance demands. Performance is critical in low-CPU-power environments. Extensibility is important when the various types of multimedia applications are introduced. Reliability of software is important when embedded software grows and becomes complicated. Real-time factors like catching an incoming call also requires reliable real-time processing to ensure the mobile handset usability. When the market is very competitive, short time to market is important like the cases in mobile handsets.

IV. Architecture Evolution

The author shows the past software architecture evolution in the examples of micro-browsers in mobile handsets in this section. NetFront is a micro-browser ported to a wide range of digital appliances from Internet TV to mobile handsets. The earliest micro-browser architecture for digital appliance was depicted in Fig. 1. It was implemented a decade ago. The design was focused on portability and parallel execution of decoding. In this early example, the porting layer and how to interface with it was the architectural focus. The first generation of micro-browser was designed for Internet TV sets. The software architecture is depicted in Fig. 1. The characteristics and focus of architecture are shown in Table. I. In this architecture, there were two major design factors:

- portability, and
- performance.

In order to pursue portability, the abstraction layer for porting was created at the bottom of the architecture. Also, the TCP/IP module and rendering module were integrated into one because it guaranteed the single interface to the porting layer. Multiple entry points for porting layer makes the porting layer structure complicated and porting process difficult. For

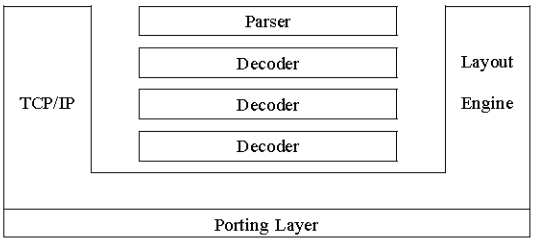

Fig. 1. NetFront 1.0 Architecture

TABLE I
Characteristics and design focus of NetFront 1.0

issues	description
characteristics	Porting layer enabled abstraction of underlying execution layers
focus of design	performance, portability

performance, the parallelism needed to progressive rendering was explicitly introduced into architecture. One parser and three decoders ran in parallel, which caused any decoder completed first could be displayed without delay. Due to the poor CPU power in the Internet TV, this performance issue was solved in the architecture level.

In the early evolution of the micro-browsers in the mobile Internet, the priority was resource efficiency, especially the minimum use of memory. When it uses much memory, it directly means need for more processing power and need for more battery power. These were all significant constraint factors in early days. At this time, modularity had less priority in the architecture design. Everything was packed into a small footprint. An example of the footprint optimized software architecture for the second generation micro-browser is shown in Fig. 2, which was used for mobile handsets, game consoles, and PDAs. In the early stage of the embedded software engineering, e.g. in the mobile Internet, the compactness was the top priority and network software was hard-coded to fit into the limited resource. In this architecture, there was no more explicit implementation issues. The number of processes to realize functions was hidden in architecture. The communication function like TCP/IP and rendering function were separate. A new abstract window layer was created to cope with poor window capability in the underlying environment. UI module was separate to ease user interface development. All UI issues were implemented in HTML, which reduced the UI implementation cost and enabled easy UI modification on the device.

Generally, the expansion of mobile multimedia applications started to feed more memory, processing power and battery power from the market demands. The today's micro browsers

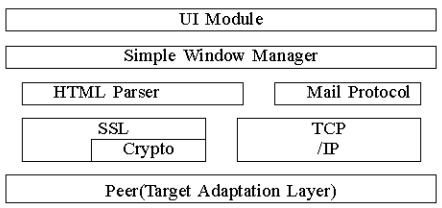

Fig. 2. NetFront 2.0 Architecture

TABLE II

CHARACTERISTICS AND DESIGN FOCUS OF NETFRONT 2.0

issues	description
characteristics	Window abstraction for porting Layered network architecture
focus of design	Portability, layered modularity

TABLE III

CHARACTERISTICS AND DESIGN FOCUS OF NETFRONT 3.0

issues	description
characteristics	Single core source for multiple profiles Customized Clib for performance and portability Modularity for language extension
focus of design	Extensibility, portability, maintenance-ability

in the mobile handsets increase software complexity as they need to pass approximately 100,000 test items by final installation. The mobile rich media needs further incorporation of the de fact standard platforms like FLASH or PDF in the mobile handset environment. These third-party intellectual properties need extensibility and modularity in the software architecture. An example of the extensible and modular micro-browser architecture is shown in Fig. 3. The characteristics and design focus are depicted in Table III. This software architecture has porting experience with Linux, iTRON [9], Palm OS, Symbian OS [10], Pocket PC, Windows CE and BREW [11]. In this architecture, all core modules are shared among different profiles. Before that, the core modules for game consoles and ones for mobile handsets were separately implemented. Using configurable content language module architecture, each content language module needed for each profile can be configurable by software. This eases software maintenance. With this extension, the additions and deletions of content language become easy. The porting layer was renamed as Peer (abstraction layer). It also enables the selection of the underlying environment features. When there are no OS nor window systems, WAVE (NetFront-based abstract window system) and uIRTON can be integrated. When there is OS, WAVE can be integrated to the target OS. When there are OS and OS-bundled high level window systems, NetFront 3.0 can be ported to those environments. Also, Clib (C library) was introduced. This is a complete rewrite of C library in order to minimize the footprint and needed work area size. Common C library uses code to duplicate string at string operation. In order to ensure the code size and work area size, Clib is rewritten. It also facilitates easy porting because it does not cause any porting problems due to the library mismatch. With extensible architecture, the third-party intellectual properties (many multimedia features) can be easily incorporated. In order to achieve this extensibility, the footprint size increased. In NetFront 2.0, a micro-browser in mobile handsets could be implemented by 300Kbytes program code with 150Kbytes work area size. In NetFront 3.0, a micro-browser in mobile handsets needs 2 Megabytes memory in the minimum configuration.

It enabled the modular architecture design which separates an underlying processing engine, a rendering engine, an XML parsing engine, and a set of content language processors.

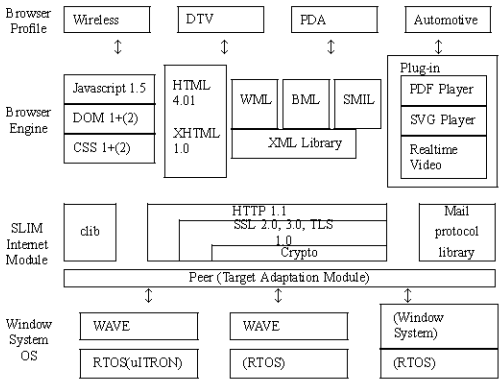

Fig. 3. NetFront 3.0 Architecture

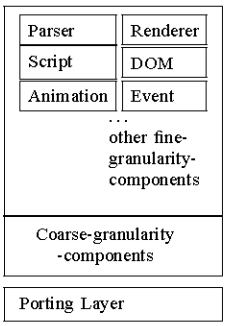

Fig. 4. Post-NF3 fine-granularity architecture

In the laboratory environment, we experiment post-NetFront 3 architecture, so-called micro-granularity architecture [8]. A new micro-component was introduced in order to construct multiple applications other than browsers. A micro-component is a fine-granularity shared components enabling networked applications in a mobile handset. It can be used to construct mail clients, game clients, compound document processing clients on a mobile handset. The architecture is illustrated in Fig. 4. The characteristics and design focus are illustrated in Table IV.

V. A FACTOR EVOLUTION MODEL

In order to highlight the design focus shift, the author summarizes the comparison of five design factors in NetFront 1.0, 2.0, 3.0 and Post-NetFront architecture in Table V. The performance factor has been covered by the underlying clock rate increase. The compactness factor has been covered by the underlying memory capacity increase. The drive from the Internet converged content drives the extensibility. The portability factor persists over a decade due to variety of the underlying environments and different evolutions in different device classes (low-end or high-end). The complexity of the software increases, therefore, the code-stability has been highlighed in the development. Abstraction of underlying environments has

TABLE IV

CHARACTERISTICS AND DESIGN FOCUS OF POST-NETFRONT 3.0

issues	description
characteristics	Fine-granularity architecture for multiple applications
focus of design	Reusability, maintenance-ability, extensibility

TABLE V

FOCUS ON SOFTWARE DESIGN

	NF1	NF2	NF3	Post-NF3
Performance	High	Low	Fair	Fair
Compactness	High	High	Low	Fair
Extensibility	Low	Low	High	Very High
Portability	High	High	Very High	Very High
Code-stability	Low	Low	High	Very High

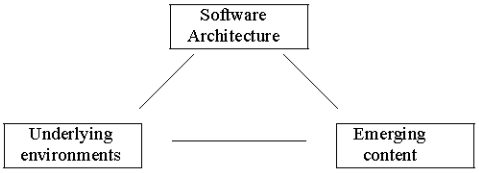

Fig. 5. An impact model on micro-browser software architecture

trade-off with performance and footprint size compactness. When the architecture is independent from underlying environments, it loses opportunity to explore environment-specific features in performance or foot-print size.

This reflects the relations among the embedded software architecture, underlying execution environments and emerging content as depicted in Fig. 5. The varieties of the underlying execution environments and emerging content determine the design focuses of software architecture. It is straight-forward, however, it is not common in PC computing or server-side computing.

VI. DISCUSSION

With the micro-browser evolution, one of the issues to be covered is portability. The embedded software engineering copes with the variety of the underlying environment. In the early stage, software needed to cope with the various issues like communication interfaces and display interfaces due to the limited capability of underlying OS. In recent years, some of the underlying environments are equipped with high level OS functions like window systems and virtual memory systems. Symbian OS provides a virtual memory system and a window system on its own. In this case, the software design needs to cope with these functionalities in mind. Sometimes, it reduces some of the detailed programming techniques required in the early stage. Sometimes, the low-end mobile handsets inherit legacy environments and the high-end mobile handsets inherit the state-of-art environment. BREW runs both in static binding and dynamic binding environments. Even the chipsets have embedded memory management unit (MMU), there are cases which MMU is not used in order to reduce the performance or to reuse the legacy software assets. The portable embedded software design copes with these dynamism in the mobile handsets as well as the dynamism in the mobile Internet content evolution. Some of the existing mobile handsets are capable of processing SMIL [12] content. Video clips written in SMIL for handsets are 3-10 Mega bytes. Such a handset is equipped with 50-100 Mega bytes memory. This is equivalent to the middle-90 personal computers (PC). There is a handset with Windows Mobile, and bundled with Skype. Some of the handsets apparently approach the direction of the PC. Also, there are handsets with Linux, in order to utilize the large amount of software assets for Linux. When handsets are dominated by Windows Mobile or Linux, the landscape of the embedded software engineering may change drastically. The hard-coded compact code library for portable and compact software may lose its significance. As the software development cost grows, it is apparent that we see the Achilles' heel in the software development in the mobile Internet. In order to solve the software development bottlenecks in this part, the software engineering needs more systematic approaches to replace the past day's state-of-art techniques. The dynamism in the underlying environment heavily impacts embedded software architecture because it determines the demands how flexible the software architecture should be. There are still conflicting two factors to drive more variety or less variety in the mobile handset underlying environments. The software architecture still need to cope with potential richer variety in the execution environment.

VII. CONCLUSION

The number of word-wide mobile handset shipment reaches 700 million, which increases every year. From this perspective, network software embedded in the handsets is the best sold software in the world. The importance of quality improvement of embedded software engineering increases. The technology to make software compact is a pile of state-of-art programming techniques. The design principles of such techniques were not well studied in the past. In addition, the use cases to study the design principles were uncommon. This study highlights the micro-browser evolution in a decade in order to capture the design principle shifts in the past decade. The author attempts to capture the long-term design shifts in the micro-browsers over a decade. It shows the design factors required in each stage. Each software architecture corresponds to the demands at each stage. The long-term focus shift analysis shows the general framework among architecture, underlying environments, and content. This is unique for the resource-constraint mobile handset environment. There are some efforts to reduce the variety of the underlying environments in the mobile handset to reduce the software development cost. It should be noted that the embedded software design still copes with the variety of the underlying environment and mobile content for the near future. The past resource-constraint in the mobile handsets made the portable software development hard and complicated. It leads to the difficulty in the use case studies for this domain. This research uses widely-ported embedded micro-browser architectures for long-term design focus shift studies. It shows the software design patterns in different design focuses in different stages. The author describes the relationship among software architecture, underlying environments and mobile-specific content, which is unique in the mobile micro-browser software engineering. The ubiquity of computer communication leads to extensive software design demands in the near future. It is difficult to change the software architecture after it is ported to multiple platforms. In the past, there was little guidance to design such portable software architecture. In order to cope with the different focus of the design factors, this empirical study gives a useful guidance in the embedded software design.

REFERENCES

[1] T. Yamakami, "An extensible monolithic software design approach for internet-convergent embedded applications," in *Proceedings of CIT2004*. September 2004, pp. 568–574, IEEE Computer Society Press.

[2] F. Bellosa, "The benefits of event: driven energy accounting in power-sensitive systems," in *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, September 2000, pp. 27–42.

[3] W. Yuan and K. Nahrstedt, "Mobile and wireless system: Integration of dynamic voltage scaling and soft real-time scheduling for open mobile systems," in *Proceeding of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, May 2002, pp. 105–114.

[4] G. Qu and M. Potkonjak, "System synthesis of synchronous multimedia applications," *ACM Transactions on Embedded Computing Systems*, vol. 2, no. 1, pp. 74–97, February 2003.

[5] T. Simunic, H. Vikalo, P. Glynn, and G. De Micheli, "Energy efficient design of portable wireless systems," in *Proceedings of the 2000 international symposium on Low power electronics and design*, August 2000, pp. 49–54.

[6] T. Simunic, L. Benini, G. De Micheli, and M. Hans, "Source code optimization and profiling of energy consumption in embedded systems," in *Proceedings of the 13th international symposium on System synthesis*. September 2000, pp. 193–198, IEEE Computer Society, Invited Paper.

[7] M. Naik and J. Palsberg, "Compiling with code-size constraints," *ACM Transactions on Embedded Computing Systems*, vol. 3, no. 1, pp. 161–183, February 2004.

[8] T. Yamakami, "A micro-component architecture approach for next generation embedded browsers," in *Proceedings of ICESS 2005*. December 2005, pp. 102–108, IEEE Computer Society Press.

[9] H. Takada and K. Sakamura, "μitron for small-scale embedded systems," *IEEE Micro*, vol. 15, no. 6, pp. 46–54, December 1995.

[10] Symbian, "Symbian os technology web page," Available at: http://www.symbian.com/technology/technology.html, 2005.

[11] QUALCOMM, "BREW," Available at: http://www.qualcomm.com/brew/, 2002.

[12] D. Bulterman et al , "Synchronized MultimediaIntegration Language(SMIL 2.1)," W3C Candidate Recommendation 13 May2005, Available at: http://www.w3.org/TR/2005/CR-SMIL2-20050513/, May 2005.