

Runtime Environments – the Future

Roy Ben Hayun

Published by the Symbian Developer Network

Version: 1.0 – September 2007

1	INTRODUCTION	2
2	THE BUSINESS CASE FOR FUTURE DEVELOPMENTS.....	2
3	THE CONCEPTS ACT AS SHAPING FORCES	2
4	THE IMMEDIATE FUTURE OF MOBILE RUNTIME ENVIRONMENTS.....	3
5	VISIONS OF A DISTANT FUTURE.....	4
6	CONCLUSION.....	5
7	FAREWELL TO THE SERIES.....	5

1 Introduction

Welcome to the fourth and final instalment of Mobile Runtime Environments, the series of papers that gives developers, technical leaders and technical decision-makers an insight into the various Runtime Environments available for Symbian OS.

A hosted Runtime Environment can be defined as an execution platform that is hosted by, and receives services from, the underlying native Operating System. Well known examples of mobile Runtime Environments are Java, Flash, Ruby and Python, amongst others.

The focus of this month's article is the future of mobile Runtime Environments – what should the mobile Runtime Environment of the future be able to do for the phone user and how will the developer make that happen.

As in all discussions on future events, this is a speculative work that will try to identify leading trends, scope the potential changes and refer to some recently released, mobile Runtime Environments.

It is recommended that you first read the previous three articles in the series to learn about the business requirements of Runtime Environments, the theoretical aspects of the Runtime Environment concept and the current phenomenon of mobile Runtime Environments available on Symbian OS.

2 The business case for future developments

The series started by looking at the business case for Runtime Environments, where we discussed how a Runtime Environment that is to be chosen as the development platform for a software product needs to satisfy requirements arising from solid business-oriented thinking. The technical aspects may well play a part in the choice but the decision makers, who will determine which Runtime Environment to select, are unlikely to be swayed by theoretical beauty, richness of language features or other computer language philosophies. They have a business to run, products to develop and customers to satisfy.

But we also need to consider what phone users are looking for. The obvious popularity of graphically-rich interfaces has made it clear not just to users but to OEMs, network operators and to the entire mobile industry, that the user experience is what it is all about. A rich, smooth, impressive user experience is required, with all manner of sound and visual effects. (Need to be convinced? Just look at the list of best selling phones.) This is one force that drives mobile sales, and the Runtime Environments of today are gearing up to being able to deliver those rich interactive applications on limited handsets.

Another force which strongly affects the advance of mobile Runtime Environment is the need for network operators to provide new, compelling services to sell to users i.e. when looking at a new feature of a Runtime Environment, a network operator would first want to know what services this package enables.

3 The concepts act as shaping forces

The technology to deliver both those services and the rich user experience is influenced by the traditional concepts of mobile Runtime Environments as we discussed in the second article of the series. In that context, the concepts act as shaping forces.

Future Runtime Environments will evolve from existing ones, improve performance and system integration, but will also be influenced by trends in software engineering.

The trend for languages considered suitable for Runtime Environments seem to be going in the direction of high level, productive and dynamic languages (e.g. JavaFX script and other dynamic scripting languages). Coupled with this, the libraries for mobile Runtime Environments will probably be designed as higher level APIs for mobile technologies that enable services (e.g. MSA libraries for location, payment etc.).

Language features and libraries that are only used by complex processing and computations could be omitted, as long as they are not required for the rich user experience or for enabling services. For example, XML libraries are required but need only be powerful enough for enabling mobile services; not for fine-grained large scale data processing. The same is true for server side Remote Method Invocation, which can be omitted.

The force of consolidation will drive Runtime Environments to multi-paradigm programming, in which different development technologies would be consolidated together into a single richer, multi-paradigm Runtime Environment. In this scenario, the power of the developer would grow by being able to leverage and mix web development technologies with scripting and the standard programming languages in a single Runtime Environment. From "we can do A with X development technology and we can do B with Y development technology" it will drift towards "we can mix A and B with a single Runtime Environment that consolidates X and Y development technologies". (The increased size of the runtime executables and libraries will be balanced by the increase in available ROM memory and the potential reuse of the runtime libraries for several VMs on a single device).

There are several trends that are evident in software engineering today but not all apply to mobile Runtime Environments. For example, there are talks about Functional Programming being a possible next step in computing, so we need to ask the primary questions about which services will Functional Programming enable and which user experiences will it help to achieve on a handheld device. Such a trend will probably remain on stronger computational devices, such as PCs, that are used for other purposes than mobile phones. In the same way, not every trend will apply to or be adopted by the mobile space.

4 The immediate future of mobile Runtime Environments

Let's start with the immediate and known future - Java and MSA. Java Micro Edition (ME) is moving towards the next generation of mass-market mobile Java applications - the Mobile Service Architecture (MSA - JSR 248), which will create an even more powerful environment. MSA is raising the bar for the standard Java application environment by re-defining the set of mandatory JSRs on Java-enabled mobile phones and clarifying implementation details of the supported JSRs, where required, to reduce fragmentation and to improve platform portability.

If we look further ahead, Java Standard Edition (SE), geared for desktop computers, will gradually supplant Java ME as technology improvements let more computing power be packed into smaller devices. Sun, for example, is trying to combine phones, TV set-top boxes and other embedded devices within the Java SE specification. The convergence to Java SE won't happen overnight but is going to take years; possibly up to a decade.

Sun have also announced a new scripting language geared specifically to providing a rich user interface, with fancy effects that are difficult to develop with the existing mobile Runtime Environments. As such, JavaFX Script is geared toward designers rather than engineers.

Adobe Integrated Runtime (AIR) is Adobe's new Runtime Environment that helps developers to leverage existing web development skills (Flash, ActionScript, Flex, JavaScript, HTML, Ajax) to create rich Internet applications (RIAs) for desktops.

Although AIR 1.0 will not be available on mobile devices, it uses technologies that are deployed on mobile devices. Indeed, the Adobe AIR roadmap includes plans for a mobile version, which will integrate with phone specific features such as voice, messaging and existing content playing applications.

As for other dynamic and scripting languages, in the near future they will still be exploring where to go. They can be good for many purposes but they will need to compete with other platforms which have been on mobile devices longer and which have a bigger developer community. We will try to find a valuable usage for them in section 6.

The scope of mobile Runtime Environments will remain in the application space and will not leak into system development, where the power of C, C++ and assembly languages is essentially irreplaceable. We can expect Symbian C++ to remain the primary development language for using the full power of Symbian OS APIs and for the tightest integration with the services available on a phone. In that sense, Symbian C++ can be described as the Porsche which allows you to move the fastest, and Java is the Ford which drives you and your family at a more sedate but protected pace. Each has its own usages and they have complementing roles in development for mobiles. And the race for the Runtime Environment that will deliver a smooth user experience has only started; JavaFX script? Adobe AIR? Both? Some other technology? We shall have to wait and see.

5 Visions of a distant future

Let's try to look far ahead, a decade from now, remove all technical barriers and imagine how things would look if a software engineer's wish list was coming true:

A powerful Multilingual Runtime Environment would allow developers to use Java (which will still be prevailing, but getting closer to Java SE with MIDP APIs) for computations and core application services. Rich and visually compelling applications would be implemented using a multitude of web development technologies such as JavaFX script, Flash or Adobe's AIR, all inside a sandbox of a single Runtime Environment, or on top of a JVM that can host other languages. If required, it would be possible to install an add-on virtual machine (VM) that would interoperate with other scripting languages such as Python, Ruby or the latest trendy scripting language.

Combining native code would still not be recommended because of the fragmentation and the number of mobile platforms.

During development, it would be possible to interleave declarative code snippets (e.g. declarative UI or event handling). And, of course, all profiling, debugging, logging and monitoring on real devices would be an integral part of any SDK and IDE.

A bit far fetched? Maybe. There is a possibility that this utopian ideal could happen in a decade. But even the world of open source and open innovations, trends and advances in software engineering, hardware optimisations - all these together cannot account for such a big leap without a strong business case for the network operators, OEMs, device manufacturers, technology producers and content providers.

Technological innovations will not exist without a sound business case for enabling a profitable mobile data service. Therefore the above vision is nothing more than a crystal ball mirage if it cannot satisfy a common set of business requirements from a long list of industry players.

An alternative vision

Since there is no certainty as to which direction things will go in, we could easily explore another route.

Let us see how a common design pattern, the Model-View-Controller (MVC), could be influence Runtime Environments several years from now, assuming innovations will take place.

The MVC is one of the most successful architecture patterns and is used to implement numerous mobile applications. The MVC separates between the user interface (View), the domain where data is kept (Model) and the use-cases logic (Controller).

The ruling paradigm of a single development technology implies that the MVC is implemented exclusively in a single language e.g. Java, Python, C++ or other languages.

Assuming those innovations will take place, each of the three building blocks of the MVC, namely the Model, the View and the Controller, could each be implemented in the technology most suitable for its role, as previously hinted to.

For example, the View component would be implemented using AIR, JavaFX script or SVG, all of which are fit for the purpose of rich UIs and could be used by skilled visual effects designers.

The Model component, which is responsible for managing data and data structures, could benefit from using dynamic languages (e.g. Ruby, Python etc) that give flexibility and are amenable to change without being tied to statically typed data structures.

The logic in business applications could also benefit from dynamic languages such as Ruby, whose loose syntax enables easy generation of Domain Specific Languages (DSL).

The controller would probably be implemented using Java, which will remain the platform with the widest scope of support for mobile APIs (e.g. payment, connectivity, security etc.).

Sounds like fun for developers. But again we must chant the usual mantra question – where does the money come from, so that developers can have fun? And which revenue-generating mobile data services could quickly and reliably be developed using such a challenging combination of development technologies?

6 Conclusion

There is no definitive conclusion to this article. Applying principles and/or trying to follow previous events has not yet given anyone a real answer on what will take place in the long run, ahead of the event horizon.

The business requirements will still define and shape. The concepts and the state of the art developments will influence and mould the existing Runtime Environments into more evolved and powerful technologies. But there are too many variables in the equation to predict if either of the two possibilities outlined above, which do indeed overlook the business needs and focus on technical and slightly academic perspectives, will actually come to exist.

However, there is still a good chance we will see similar technologies running on our handsets. Evolved and more powerful VM technologies, more development options, additional consolidation of technologies, easier and faster development – all of these will happen, given time.

7 Farewell to the series

This is the time to look back at the four articles in the series and summarize. Each article stands on its own and analyses a certain view on runtime environments running on mobile handsets.

The first article discusses the business arena in which mobile technologies, and mobile runtime environments specifically, live.

The second article gives a more academic, and less practical look, into programming languages, technological libraries and the software engineering related to mobile runtime environments.

The third article gives a pragmatic overview into a chosen selection of the existing runtimes, from Java to OPL.

This final article puts those three views on to a time line, seeing what is already know to be happening, what may happen in the not so distant future; and the possibilities for 5 years and beyond.

It is my hope that this series of articles has managed to give an interesting and structured way of looking into runtime environments, considering various aspects of them and showing their interdependencies. Hopefully, after breaking the term “runtime environment” in smaller parts and reconstructing them inside the relevant context, the bigger picture seems much clearer.

I would like to thank all my colleagues who helped me in researching and writing this series.

Want to be kept informed of new articles being made available on the Symbian Developer Network?

[Subscribe to the Symbian Community Newsletter.](#)

The Symbian Community Newsletter brings you, every month, the latest news and resources for Symbian OS.