

What's New in the System Model for Symbian OS v9.4?

Ben Morris

Published by the Symbian Developer Network

Version: 1.0 – November 2008

1	INTRODUCTION	2
2	A LITTLE BIT OF HISTORY	2
3	THE SYSTEM MODEL AT V9.4	3
	3.1 CORE OS.....	3
	3.2 NEW AND DEPRECATED COMPONENTS	4
4	OTHER FEATURES OF THE MODEL	6
5	THE SYSTEM MODEL BEYOND V9.4	7
6	FROM V9.4 TO V9.5	7
7	CONCLUSION	8
8	AUTHOR PROFILE	9

1 Introduction

The latest version of the Symbian OS System Model to be published is for Symbian OS v9.4.¹ A great deal has changed since the model first appeared for Symbian OS v7.0 back in 2003. The operating system itself has evolved greatly in that time, and the v9 releases have marked something of a watershed:

- Symbian OS v9 completes the evolution to the real-time kernel architecture of the EKA2 kernel. This in turn enables new communications and graphics architectures that are needed to handle the real-time and high data bandwidth requirements of the latest generations of applications and services, for example, real-time audio and video streaming and mobile broadcast TV.
- Symbian OS v9 introduces platform security, enabled by the EKA2 kernel.
- Symbian OS v9 supports the latest hardware technologies, like NAND Flash, as well as new software technologies such as LBS (Location-Based Services) that pave the way for next-generation software services.

Out in the marketplace, Symbian OS has shipped in over 250 phone models, with total sales of Symbian OS-based devices approaching 250 million.

It may be a little less 'headline grabbing' than all of this, but the System Model has evolved significantly since its first release, too.

This paper introduces the latest version of the System Model and looks at what's new in the model, what's new in the operating system and at how the model is being used as a tool for understanding and managing the operating system itself.

2 A Little Bit of History

The first version of the System Model created a simple, layers-and-blocks representation of the operating system. For insiders (but not for third parties), it was augmented with a component listing that was not quite exhaustive, but that did a reasonable job of dividing the system into blocks of fine but meaningful granularity.

The resulting model was not perfect, but it was a necessary move away from the impenetrable view of the system that Symbian developers had learned to live with.

In particular, the System Model applied some top-down principles and definitions to a representation that was derived bottom-up, from the system itself. This turned out to be a successfully pragmatic approach:

- It introduced a layered model based on a simple stack-like abstraction, from bottom to top, and provided a higher level structure that followed the basic vertical flow of provider/user dependencies.
- The logical block structure within layers was based on general technology criteria (for example, graphics, networking and telephony).
- Within blocks, the layering of component collections was based approximately on provider/user dependencies.
- Individual components were the discrete, buildable parts of the system.

¹ See developer.symbian.com/main/documentation/technologies/system_models/index.jsp for the Symbian OS v9.4 System Model and the previous versions.

3 The System Model at v9.4

Bob Rosenberg is the system architect who has been leading the System Model work and driving its evolution for the last few releases of the system. He's responsible for both the model and the tools that sit behind it and, as he explains, the basic principles of the model remain unchanged in v9.4.

'Layers contain blocks (and in some cases sub-blocks). Blocks (and sub-blocks) contain component collections, which contain components. Components are, in effect, build packages.'

Behind the scenes, many of the evolutionary goals that were set for the model have now been met. For example, while early versions were purely descriptive, the model is now used to build the actual software system.

However, the biggest change is the arrival of Core OS, and with it the first major reorganization of the model since it was first released.

3.1 Core OS

Compared to previous versions, the v9.4 System Model is radically restructured:

- A Core OS 'super layer' is introduced, consisting of the Kernel Services and OS Services layers.
- A new HAL layer is introduced, below the Core OS, accommodating the lowest level hardware-dependent components, which are reorganized in terms of Board Support Packages.
- In place of the Application Services and UI Framework layers of previous models, and the slightly awkward placement of Java, the UI Framework components move down into a new Generic Middleware layer and are reorganized, and Java moves into the Application Services layer, which is now the highest layer of the operating system. The UI Framework layer disappears.

The Core OS is defined as a hardware-independent (at the bottom) and GUI-independent (at the top) base platform for mobile phones, which can be brought up on new hardware in a two-step process:

- Step one: bringing up the kernel and basic user services, enabling processes to be created and run.
- Step two: bringing up complete cellular phone, networking, graphics and multimedia stacks supported by a fast communications framework and comprehensive base services.

The Core OS, in other words, is intended to provide a complete, basic phone platform.

The Generic Middleware and Application Services layers complete the operating system offering, providing a complete platform for the S60, UIQ and MOAP UI frameworks.

Most importantly, the Core OS is free of any dependencies on components in the layers above it, making it not just a logical platform but a truly buildable platform. To put it another way, the Core OS is guaranteed to be free of any dependencies on licensee components.²

² In the anticipated new open source context, this guarantees that it is 'IP free' and genuinely open, as licensed.

3.2 New and Deprecated Components

In terms of functionality, v9.4 is a significant incremental step in the v9 evolution of Symbian OS.

The most important functional additions are all targeted squarely at improved performance:

- Demand Paging is extended from being applicable to ROM only to all internal fixed storage.

This improves startup times and responsiveness on NAND Flash-based systems by reducing the granularity at which code and data are loaded into RAM.

- RAM defragmentation.

This improves memory utilization and enables the powering down of unused RAM, which reduces power consumption (and therefore improves battery life).

- File caching.

Read ahead and lazy write strategies improve the performance of streaming applications and reduce the need for local buffering, which in turn improves memory utilization and therefore potentially reduces power consumption (thereby improving battery life).

In addition, new and extended APIs include:

- The Open Environment, previously available as an add-on package, is now integrated into the operating system release.
- MDF extensions to enable plug-in audio and video codecs, aligning with the OpenMAX 1.0 standard.
- MTP over USB.
- URI Black List/White List support.
- New EGL APIs and bindings (Khronos EGL graphics) that extend OpenGL ES support.
- SQL Server performance improvements, especially for batched transactions and table Delete From and Drop.
- Some enhancements to OMA SyncML are introduced in the higher layers of the system, and support for ARM 1100 is introduced at the bottom of the system.

As well, there is improved debug tools support:

- Core Dump Viewer and Memory Reporter are both introduced.

This functional evolution is reflected in specific changes in the model, which for the first time clearly marks deprecated components. For v9.4, the deprecated components are as follows:

- Application Services: PIM

PIM Application Services	Agenda Model is deprecated and is replaced by a new Calendar component.
Office Application Engines	Data, Sheet, Chart and Word Engines are deprecated.

- Application Services: Remote Management

Sync and Remote Management Framework	New components include OMA SyncML and Config. Connectivity Framework is deprecated.
Sync and Remote Management Services	Connectivity Services is deprecated.

- Generic Middleware: Generic Application Support

File Handling	Other File Converter Plug-ins are deprecated.
Time Zone Services	World Server is deprecated.

- Generic Middleware: Application Protocols

Connectivity Transports	PLP Remote Link, PLP Variant and Event Broadcast are all deprecated.
Application Layer Plug-ins and Utils	URI Permission Services is a new component.
MTP Transports	MTP USB Transport is a new component.

- Generic Middleware: Multimedia Middleware

Multimedia Protocols	MTP Framework is a new component.
----------------------	-----------------------------------

- OS Services: Base Services: Generic OS Services

Generic Open Libs	There is a new Open Environment Core.
Resource Management	There is a new Hardware Resources Manager.

- OS Services: Comms Services: Cellular Baseband Services

Telephony Utilities	Dial component is deprecated.
Telephony Reference Platform	TRP Agent is a new component.

- OS Services: Comms Services: Networking Services

TCP/IP Utilities	Network Address and Port Translation is a new component.
------------------	--

- OS Services: Comms Services: Networking Services

Serial Comms Server Plug-ins	MUX CSY is a new component.
------------------------------	-----------------------------

- OS Services: Multimedia

Multimedia Device Framework	OpenMAX is a new component.
-----------------------------	-----------------------------

- OS Services: Graphics

Khronos APIs	EGL API is a new component.
Khronos Implementation	EGL Implementation is a new component.
Location-Based Services	There is a complete new block.

- Kernel Services: Kernel-side Services

User Library and File Server	NAND Flash Translation Layer component deprecated.
------------------------------	--

- HAL: Board Support Packages

Assabet	Assabet BSP is deprecated.
---------	----------------------------

4 Other Features of the Model

Some useful features of the model are now available to the external user, including the dependency maps that are displayed when mousing over components. Based on the internal tools that support cross-referencing and other dependency reporting across the operating system, incorporating this data into the model gives an easy-to-use and immediate indication of the flow of dependencies within the system, and is hopefully useful to ISVs to support debugging.

As well as dependencies, interface access, licensing and component type information is provided.

Beyond the Core OS, much work has been done behind the scenes so that for the first time the model really does represent a true map of the system as it is shipped and, most significantly, as it is built. The System Definition XML file that defines the model now drives the software build. The graphical version of the model is generated (in SVG graphics) as a representation of the built system, derived from the definitions which also build the system. To enable this, wherever possible, components are defined as concrete, discrete and unambiguous parts of the system that correspond to the system as it is delivered to customers.

This approach is applied with a very few exceptions, of which Java is the most glaring, although there are also a handful of 'placeholder' components representing things which at the time of release are not yet part of the product (for whatever reason).

Components, as shown in the model, map directly to the packages delivered by the CBR (Component Based Release) tools, which underlie the licensee delivery model.

In effect, a component represents the finest granularity at which a complete piece of functionality can be included or excluded from a build of the system. (Feature selection at a finer grained level within components is provided by the variability mechanisms, and determines which behavior is compiled into the system in a build, rather than what components are built.)

5 The System Model Beyond v9.4

With each operating system release, plans for the System Model become more ambitious. The so-called 'roadmap' model goes beyond current releases to represent where the system is expected to go in the future, serving as an interface between the system architecture and product design and strategy organizations within Symbian.

Increasingly, the model reflects the future end goals of modularity, not just in terms of system architecture, but all the way to engineering organization and delivery. This puts some interesting stresses on the model, because (ultimately) it represents individuals sitting at desks, not just the architecture of the system.

There are consequences too in the architecture, and therefore in the model. For example, test code in the v9.5 model is being moved into individual components, rather than being represented in a separate 'test' model.

Part of the fallout from these changes is a solution to the longstanding TechView 'question,' i.e., the fact that it sits awkwardly with other parts of the delivery, as a non-product quality part of the delivery which nonetheless is required to successfully create products, and which is necessarily exposed to developers (including third-party developers) in training SDKs and similar materials. TechView in the future is recast as a suite of test applications, which includes the test UI as well as system-wide deliverables like validation suites and smoke tests.

6 From v9.4 to v9.5

From the perspective of the System Model, v9.5 is an incremental but significant improvement on v9.4. As Bob Rosenberg says:

'v9.4 reflects the assets we've got. v9.5 is better organized to show what's coming.'

There are some specific movements of components in v9.5, though relatively few:

- HTTP utilities move down into the Core OS (although the framework and protocols remain in the Generic Middleware layer).
- Some drivers move out of the kernel and into their technology blocks.

- Application Provisioning and Installation merges what were UI Framework and Security components that deal with application install and launch, and includes the view server.
- PIM and Messaging are more layered.
- The Multimedia protocols SIP and RTP have been merged with the HTTP protocols to make High-level Internet Protocols.

Meanwhile, as the model evolves into v9.5 and beyond, the message is to look out for more device state management components that manage both what's installed on the device as well as what is actually running at any time. However, since these components will reside above the Core OS, there will not be many changes to the Core OS itself in v9.5.

Less visibly, but perhaps more significantly, in v9.5 there is some important tightening of the rules that underlie the model:

- Blocks and sub-blocks are generically considered modules.
- Non-published partner dependencies between modules are now forbidden. In other words, if a team has an internal API it cannot be used outside that team's technology. This in effect treats other teams in the company as partners rather than allowing them privileged access to another team's software.

Such dependencies were considered unhealthy, although there was no actual rule enforced to forbid them. This means that although the interface access categories of 'Internal Component' and 'Internal Technology' remain, in effect there is no longer the notion of 'Internal All.'

Also, within the model, the rules for representing and positioning plug-ins have been clarified:

- plug-ins that are hidden beneath a framework appear below it
- plug-ins that are used to access a framework appear at the same level or above it.

Meanwhile, the headline ambition for the Core OS is clear enough: the Core OS should build independently. As currently defined, everything required to build it is included (and some things that are not required are also included). Similar principles are applied to all modules, i.e., to blocks too, in an attempt to enforce the discipline that independent modules should be decoupled as far as possible so that the system can be built without specific modules.

As Rosenberg says, 'It's sort of a dream but it's what we are aiming for, to make the operating system more customizable for each actual device and avoid including things that certain devices will never need.'

The goal of modularity goes hand in hand with the complementary goal of variability. As Rosenberg puts it, 'Modularity and variability are related, but modularity is not a requirement of variability, and variability works on a slightly lower level than modularity. So, modularity is concerned with "work without such and such a technology as a block," whereas variability says, "can we work without this specific component of such and such technology?"'

7 Conclusion

The Symbian OS System Model has evolved as the operating system itself has evolved. The basic ideas behind the model remain the same, with the basic structure derived from the dependencies in the system.

The most significant change in the v9.4 model is the introduction of the Core OS 'super layer,' which reflects the significant engineering work to break unnecessary dependencies in the system and improve modularity.

At the same time, the model has to evolve as a tool for managing and building Symbian OS, and not just representing it. In v9.4 the model really does 'define the system,' which was the original goal. For system builders, including licensees, the model can be used as a tool for investigating and understanding the system, and mapping and managing its dependencies. The tools that sit behind it can be used to create custom licensee models and to help manage extensions, customizations and other architectural work, including building the system.

Meanwhile, it continues to provide all developers with a high level view of the operating system as it evolves, supported by analysis tools, for example, dependency tools and component listings, which turn it into a useful and effective interface with the system.

8 Author Profile



Ben Morris freelances as a writer and software architect specializing in Symbian OS. He is the author of *The Symbian OS Architecture Sourcebook: Design and Evolution of a Mobile Phone OS*, published by Symbian Press and John Wiley & Sons, Ltd. He can be contacted through www.benmorris.eu.