NSI Overview

1 About NexWave products

Software component infrastructures are widely recognized as a major productivity enhancement approach, decreasing time to market, increasing quality and easing maintenance, thanks to the reuse of already developed, flexible and re-configurable components. Business, desktop and server software use asynchronous, message-passing component models such as CORBA, which allow application-level componentization and run-time upgrades at the expense of a non- real-time, resource intensive approach. Some Consumer Electronics (CE) manufacturers have pioneered internal source code based, build-time component models, which allow embedded applications to be componentized; however, commercial component systems for the embedded market have been surprisingly rare.

NexWave Solutions have developed a portable, non-intrusive component infrastructure designed for embedded systems. The NexWave Software Infrastructure (NSI) defines how to model components and how they interact, independently from the host operating system. NSI component binaries can be produced for several embedded operating systems and will load and operate on all these platforms. Unlike business component infrastructures, NSI uses a passive model in which components call each other by means of synchronous function (method) calls. This model is not only economical in terms of processor and memory resources, it also enables the component infrastructure to guarantee real-time operation and extends the typical synchronous programming paradigm of C and C++ languages, with which embedded software developers are familiar. NSI is non-intrusive as it does not degrade real-time performance nor introduces a completely new programming model. Unless live upgrades are desirable, software systems can be built entirely from NSI components on a development system, without the need for any run-time software.

The binary, self-contained and non-intrusive nature of NSI extends the normal benefits of component-based software development and offers the following:

- Software development productivity and quality improvements through the reuse of well-isolated components and by operating on individual components.
- Seamless integration in production processes, early prototyping, and quicker time to market, thanks to the build-time and run-time configuration management and flexible assembly of existing components.
- Return on investment on constantly increasing software development

costs, by the introduction of a component binary standard that allows exchange and reselling between manufacturers, while simplifying IPR and licensing issues.

- Software updates and extensions to equipments in customer premises, using NexWave Solutions' run-time dependency management and software infrastructure.
- Repurposing of software across a wide range of consumer electronics, communication, networking or other products, thanks to a solution that supports both single- and multiple-application equipment, and both compile-time and run-time component assembly.

NexWave Solutions market a product that offers a complete solution to equipment manufacturers:

- NSI (NexWave Software Infrastructure): a software management and deployment solution consisting of a component model, binary format, a component creation, assembly and configuration management tool, with an optional run-time infrastructure, and tools for software updates and extensions.

2 NSI

The NexWave Software Infrastructure (NSI) is a complete software management and deployment solution based on a non-intrusive component architecture. With NSI, embedded software is developed as independent components, which can be assembled in multiple configurations corresponding to a product line and independently reused and updated.

One of the unique benefits of NSI is that it splits software into components at the binary level (rather than the source code level) and allows components to be assembled statically (off-line) on a development system or dynamically (on-line) on an embedded target.

2.1 Features

- Total life cycle software management

Unlike a development environment or a productivity enhancement tool such as an object-oriented language, NSI manages the complete life cycle of embedded software components. While the reach of a traditional tool stops after the compilation and debugging stages, NSI allows compiled, binary components to be assembled later, into a working system, according to flexible configurations for a particular product. NSI enables the resulting software system to connect back to the manufacturer for updating the individual components as well as new functionalities, remote diagnostics and remote support. NSI follows the life of the component from its compilation, to

assembly into a larger system, debugging, deployment, remote diagnostics and replacement by a newer version.

NSI offers a complete set of tools for software development, software deployment, quality assurance and factory tests.

- Self-contained binary components

NSI assembles complete software systems from independently compiled, binary components. This feature removes the need for software developers to supply source code for software systems that need to be reconfigured. Components are compiled for each processor architecture and executed natively, without a virtual machine.

NSI component management recognizes different stages and responsibilities. The compilation process produces self-contained binary components. At this stage the components are completely independent from their target environment. The preparation for the appropriate target environment takes place in the subsequent assembly process.

- Complete component assembly and configuration support

By allowing components to express their dependencies, NSI enables a flexible assembly of components from various suppliers into a working software system. Components can either use any available implementation of the external functionality they depend on or bind explicitly to a particular implementation. NSI enables components to bind to different implementations of a particular functionality, depending on external configuration parameters, which can even be determined at run-time. This is a major feature in allowing the reuse of already developed components, in completely different products, saving both time and resources.

NSI allows engineers to take self-contained, binary components from their software teams or from external suppliers and specify configuration details of the embedded system in a file that is separate from the components that are used.

- Online management of "live" components

Thanks to run-time software supplied by NexWave Solutions and that the manufacturer has the option of adding to the target device, NSI manages components on a "live" target, at run-time. NSI provides Quality Assurance teams and Factory Testers with tools that carry out live diagnosis, browsing and unit testing of software running live on the target. NSI enables targets in customer's premises to identify themselves to the manufacturer, provide diagnostics, receive remote assistance and download secure upgrades of individual software components, cryptographically signed by the manufacturer.

NexWave Solutions have developed a set of proprietary components that

handle the import of new components, dependencies and dynamically access to run-time components. These "run-time support components" are a standard part of the NSI package.

2.2 Benefits

In addition to the generally accepted benefits of component-based software development, the above features generate a range of benefits that are specific to NSI:

- A portable, non-intrusive component infrastructure

NSI is non-intrusive by nature as it uses a synchronous model, independent from the host operating system. The synchronous model ties into the traditional programming paradigms, offering important benefits for the adoption of NSI in an existing software environment:

- Legacy systems can be "componentized" and assembled with new components.
- Performance is equivalent to non-componentized code.
- Footprint can be managed and has no overhead in the case of build-time assembly.
- The independence from the host operating system allows system architects and developers to retain their existing environment and protects them from being locked into a particular platform or vendor.

The benefits of the binary format are the intellectual property protection and reduced liability risk for component suppliers. It is a key enabler for the exchange of components between different teams, business units and manufacturers to exchange binaries, increasing their return on investment for their software. NSI's binary format and non-intrusiveness provide much greater economy of scale than simply standardizing products on one operating system API.

- Swapping and probing

NSI is capable of run-time dependency management, as well as the dynamic loading and unloading of independent components. Components that were built into a system on the development environment can be managed at run-time, without having to be modified or even recompiled. When deployed on a device with networking capabilities, components that are managed by the run-time infrastructure are individually identifiable and replaceable while the device is running. These capabilities generate a wide range of benefits for product development and product management: software can be both fixed remotely and fully transparent for the users, reducing product recalls. Customer support becomes more effective with the remote probing facilities.

Existing systems can be extended or even completely replaced on the fly, creating an after-market for the devices. Depending on the marketing strategy of the manufacturer, the life of the device can be extended and support for new protocols or formats added.

·Early prototyping (assembled from configurable, binary components)

NSI is an infrastructure that allows manufacturers to easily build libraries of readily available, reusable binary components, explicitly designed to be flexibly assembled in many different configurations. Reuse and configurability are by nature beneficial for prototyping. NSI facilitates early prototyping by creating a software system from existing components, creating only the minimum set of new software for the prototype of the product being considered.

3 Product scope

3.1 What is NSI?

NSI is a complete solution for software deployment on embedded systems, which consists of a packaged and integrated release of several software basic blocks.

- Component architecture

The component architecture specifies how developers describe software components, including what properties and features are common to all components, how to express functionality provided to other components, and how to express functionality needed from other components. The component architecture also describes how components are stored in binary form, how to express the assembly of binary components for a particular system, and how binary components are physically packaged together.

The component architecture is defined by the following:

  - The ".dep" language, used to model a component and its interfaces.
  - The component interface, a description of the component necessary for the dependency management and component assembly tasks.
  - The standard data types used to define the component interface.
  - The XCOM binary format, a specification of how to store the code, data and interface of components as binaries, and a standard for the exchange of binary components.

- Build tools

NSI provides a complete tool chain of command-line tools that transform components in source form into independent binaries, and then assemble

component binaries into a software system according to a supplied configuration:

- NexDev, a standard suite of Makefiles and scripts that drive the compilation of a set of NSI components, supporting free as well as commercial compilers. It is easily extendable by customers to support their own internal tools.
- depc, the ".dep" interface compiler.
- GNU gcc, the default compiler for all processor architectures supported by NSI, replaceable by commercial or internal compilers on an architecture basis.
- GNU binutils and ld, the linker that produces binary components in the XCOM format.
- xcomstrip and xcompostld, production code optimizers for binary components in the XCOM format.
- mkimage, which packages independent components into an executable image. Components remain independent binaries handled by the NSI run-time, and can be individually swapped and probed on the target device.
- mksimplefs, a file system image creation tool

- Deployment tools

The command-line tools are driven by a fully graphical deployment environment that enables developers and architects to assemble components into software systems, and modify them on the target device:

NexBuilder, the build-time (off-line) environment. Features component cataloguing and browsing, system configuration management (templates, selection of policies and implementations), component assembly, dependency checking and creation of a software system, either fully resolved or designed to be upgraded and extended at run-time.

- Run-time infrastructure

NSI's optional run-time infrastructure provides run-time dependency management and allows for swapping (updates), probing and extending the software on the target device as follows:

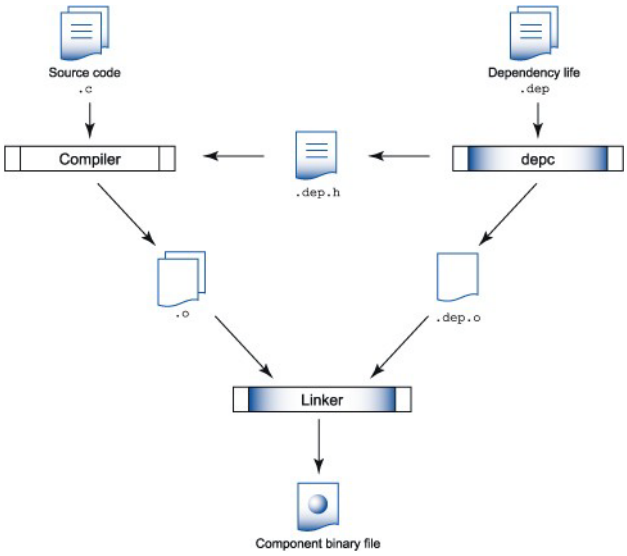
- The on-line dependency manager (depmgr), responsible for identifying and managing available components and interfaces on the target device
- Target identification management
- Updates (cold- and warm-swapping, extension to new components) management
- Debug messages logging

3.2 Build chain

NSI provides developers and architects with a complete chain of specifications and tools to create, build and assemble software components into integrated software for embedded products. The total build chain is broken down into two phases: first the development of single components, then the assembly and deployment of these components into configurable systems.

3.2.1 Component development

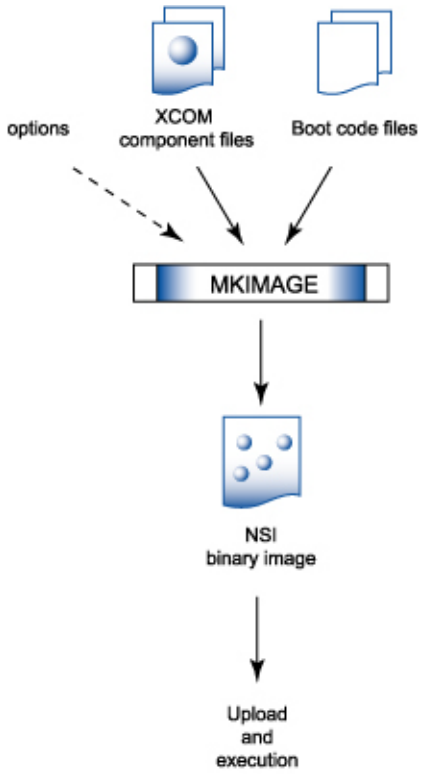
In this part of the process, the actual code and the interface definitions are created. Both code and interfaces are expressed in binary format. The result is a single, self-contained binary component.



3.2.2 Assembly and Deployment

At this stage, the dependencies between the components are defined and managed, and all selected components are assembled into the embedded system for the selected target environment.

On-line dependencies are handled by a set of NSI run-time components that are included in the final image. These components handle dependencies dynamically and support on-line swapping and probing of the components in the final bootable image.



3.3 Target environments and target markets

NSI has been designed for the embedded software market from the ground up. As a product, it is inherently operating system independent. NSI is available on platforms such as the µITRON and Linux operating systems. Making NSI available on other platforms enables developers to use their specific services, such as a large library of pre-existing middleware. The host platform makes no difference to NSI's capabilities.

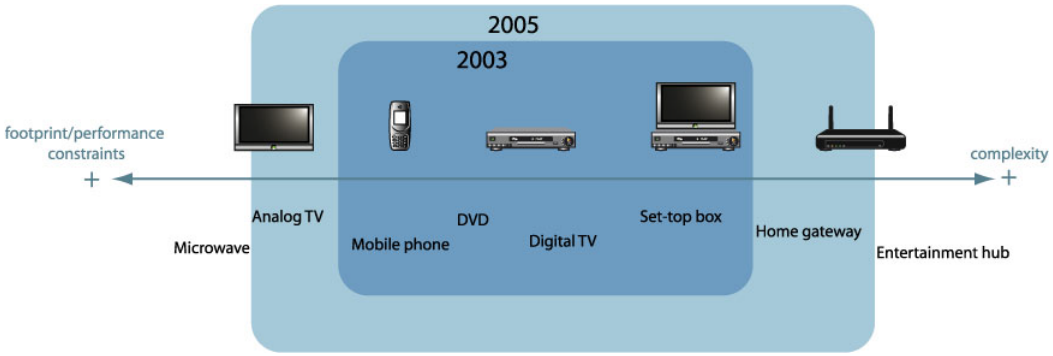
Even though there are no real technological limitations that limit the use of NSI to certain environments, the features of NSI make it a more natural fit for specific markets. The binary format is particularly useful for the semiconductor market, which is increasingly packaging software together with silicon products. The rich component configuration and assembly support add significant value to the Consumer Electronics market with its large quantities of new consumer products manufactured annually. The on-line swapping of live components is clearly important for network and telematics equipment.

After careful market analysis, NexWave Solutions have selected Consumer Electronics (CE) and Communication and Networking equipment as the primary target markets. NexWave Solutions have tailored their products internal organization to these specific markets.

3.3.1 Consumer Electronics

The CE market is characterized by tight margins and a high turnover of consumer products. As a result, CE manufacturers have been encouraged to optimize their production processes. Considering the ever-increasing size and complexity of software in CE products, these tight production constraints need to be applied to software as well. NSI is a perfect enabler for the optimization of the software management process along with the integration of software deployment in the existing CE production process.

To support the full range of CE products, some further tailoring of NSI is required. The diagram below shows the coverage of CE products with the current version of NSI.



4 NSI in Summary

NSI FEATURES

Cross platform (Linux, Windows, TRON) with a realtime API for embedded applications

Light-weight framework for CBSE

Cutting-edge technology, easy to implement on legacy software (no or few source code changes)

Location transparency

Visual assembly of binary components made possible with explicit "Provides" and "Requires"

Code re-use with the exchange of re-usable binary components instead of source code

NSI ALLOWS

Simplification: Removing one or more steps from an unnecessarily complicated process, or reducing unnecessary variety in the process. This can often be done by replacing components to produce stepwise improvements.

Integration: Joining two or more previously unconnected or uncoordinated processes into a larger coordinated process. This can often be done by inserting additional components to create new links.

Transformation: Creating a radically new process. Disassembling the components and putting them back together in a new way.

Portability: The specification of a component is platform-independent. Therefore, a component can be quickly rebuilt for a new platform, without affecting any other components.

NSI BENEFITS

Functionality: Using pre-existing components allows faster delivery of greater functionality.

Maintainability: The modular structure of a component-based solution allows individual components to be replaced easily.

Usability: Use of standard components supports commonality of GUI.

Efficiency: Performance bottlenecks can be identified, and the need for performance tuning can usually be localized in a small number of performance-critical components. Components can be internally optimized to improve performance, without affecting their specification; components can be moved between platforms to improve performance, without affecting the functionality or usability of the application.

Reliability: Given the formal and complete specification of a component, its reliability comes down to the simple question: does the component provide a correct and complete implementation of its specification? The reliability of the application as a whole is a separate issue, but is clearly enhanced when the application is constructed from reliable components.

Portability: The specification of a component is platform-independent. Therefore, a component can be quickly rebuilt for a new platform, without affecting any other components.

NexWave Solutions
1068, rue de la Vieille Poste - 34967 Montpellier Cedex 2, France
Tel: +33 4 99 52 89 89 - Fax: +33 4 99 52 89 88
www.nexwave-solutions.com - contact@nexwave-solutions.com