T-76.115 Requirement specification

29.11.2001

Confuse

State

Version:	2.0
State:	Accepted
Distribution:	Public
Created:	03.10.2001 Antti Haapakoski
Last modified:	29.11.2001 Antti Haapakoski

Change log

Version	Date	Author	Description
1.0	28.10.2001	Tuomo Koskenvaara	Public release, accepted by the customer
1.02	7.11.2001	Antti Haapakoski	Corrected the figure 2 in Section 4 (xml file was in wrong place). Made document suitable for pdf
2.0	29.11.2001	Antti Haapakoski	Incremented the version to indicate the end of T1.

Contents

1 Introduction **5**
- 1.1 Background . . . 5
- 1.2 Goals . . . 5
- 1.3 Current solutions . . . 5
- 1.4 Glossary . . . 5

2 Overview **7**
- 2.1 Environment . . . 7
- 2.2 User group . . . 8

3 About the requirements **8**
- 3.1 Priorities . . . 8
- 3.2 Identifiers . . . 8
- 3.3 Fit criteria . . . 8

4 Architectural requirements **8**
- 4.1 UR-01 - Main components and their interconnections U . . . 8
- 4.2 UR-02 - The system must use the Configuration Engine M . . . 10

5 Internal interfaces **10**
- 5.1 UR-03 - The system reads a text file listing the available packages M . . . 10
- 5.2 UR-04 - The system produces a XML file listing the packages in the configuration M . . . 10

6 Functional requirements **10**
- 6.1 UR-05 - User can add a package to the configuration M . . . 11
- 6.2 UR-06 - User can remove a package from the configuration M . . . 12
- 6.3 UR-07 - User can test if a configuration is valid M . . . 12
- 6.4 UR-08 - Administrator can add new packages to the system M . . . 13
- 6.5 UR-09 - System shows a list of available packages M . . . 13
- 6.6 UR-10 - System shows a hierarchical view of available packages U . . . 13
- 6.7 UR-11 - System shows error message if configuration is invalid M . . . 13
- 6.8 UR-12 - System shows conflicting packages if configuration is invalid U . . . 14
- 6.9 UR-13 - System suggests a solution if configuration is invalid N . . . 14
- 6.10 UR-14 - User can choose to install the configuration to PDA M . . . 14
- 6.11 UR-15 - The system knows how the installation succeeded M . . . 14
- 6.12 UR-16 - The system can remove packages from PDA N . . . 15
- 6.13 UR-17 - There is a default configuration U . . . 15
- 6.14 UR-19 - User can save a configuration M . . . 15
- 6.15 UR-20 - User can load a configuration M . . . 16
- 6.16 UR-21 - Incremental installation of packages U . . . 16

7 Quality requirements **16**
- 7.1 UR-22 - Source code is well commented in English M . . . 16
- 7.2 UR-23 - The system components are loosely connected and modular M . . . 17

8 Efficiency requirements **17**
- 8.1 UR-24 - The memory use must be known and predictable N . . . 17
- 8.2 UR-25 - Minimal temporary storage when installing to PDA N . . . 17

9 Portability requirements **17**
- 9.1 UR-26 - All of the system can be ported to PDA N . . . 17

10 Usability requirements **18**
- 10.1 UR-27 - The configurator must have a graphical user interface. M . . . 18
- 10.2 UR-28 - Efficiency of use N . . . 18
- 10.3 UR-29 - Ease of learning N . . . 18

11 External interfaces **18**
11.1 Connection between the PC and the PDA 18
11.2 Configurator engine 18
11.2.1 Configurator's requirements for interfaces 19
11.2.2 Preferable environment 19

12 Requirements Traceability Matrix **19**

1 Introduction

Today many software system have thousands of different configurations. Some of them work and some of them don't. One configuration consists of several components and there can be complex dependencies between those components. Making such system work by trial-and-error could be impossible.

The problem is emphasized in mobile devices. They usually have an expensive, low bandwidth connection. The configuration has to work when downloaded.

This is where automatic configurators come in. Configurator makes sure the user selects a working configuration with all its dependencies satisfied. The configuration is then guaranteed to work when it is installed to the target device.

1.1 Background

This project is a part of the SARCOUS research project. SARCOUS studies static and dynamic configuration of product families and their components. Here are the goals of the SARCOUS project according to the SARCOUS Research Plan:

- Increased reusability and configurability of components, product architectures and other relevant knowledge
- Method for formal modeling of the product knowledge relevant for software configurability
- Methods for interactive or automatic configuration of software on the basis of the product models
- Methods for managing the evolution of the product models
- Methods for dynamic reconfiguration of systems according to user needs when the product models, and th product individuals are developed and managed by one/separate organization(s).
- Methods for dynamic real-time reconfiguration of systems in response to the changes in the environment in which the system finds itself

1.2 Goals

The goal of this project is to produce a simple configuration environment for mobile devices. The environment will work as a practical configurator example for the SARCOUS research project.

1.3 Current solutions

There are many different configurators available. For example when installing (deb, rpm) packages in Linux, the configurator makes sure all the packages needed by the new package are already installed. Also many companies offer WWW-based configurators for their products, like cars and computers.

The current configurators have several problems. The logic that checks the dependencies could be clearer and it should be separated from other parts of the program. Sometimes the installer does "hacks" to make a configuration work.

Some configurators try automatically to fix the configuration if it is not valid. This may lead to irrational and unwanted changes to the configuration.

On mobile devices it would be nice to know if a configuration works **before** downloading all the packages to the device. For example one could download a large package to a PDA just to find out it does not work with the current system library versions.

1.4 Glossary

BLUETOOTH	A Technology that implements small range radio link between computers, celluler phones, PDAs etc. Teknologia joka toteutta radioyhteyden tietokoneiden, matkapuhelimien, taskutietokoneiden yms. välille lyhyillä matkoilla.
Bootloader	Program that boots computer. Tietokoneen käynnistävä ohjelma.
Burana	Bug Report And Nag Application. Bugiraportointi ja versionhallintajärjestelmä.
CCCC	C and C++ Code Counter, A free software tool for measurement of source code related metrics by Tim Littlefair. Ilmainen koodirivien laskenta aohjelma.

CF-card	Compact Flash Card, Memory card for handhelds, mp3 players and digital cameras. Muistikortti kämmentietokoneille, mp3 soittimille ja digitaalikameroille
configuration	A set of components or packages forming a system. In this context a list of packages to be installed in the PDA. Systeemin muodostava joukko komponentteja tai paketteja. Tässä yhteydessa lista paketeista jotka asennetaan PDA:n
configurator	A program that helps the user to make a valid configuration and possibly use that configuration. Ohjelma joka helpottaa kelvollisen konfiguraation luomisessa.
deb	extension for Debian Linux packages. Debian Linuxin pakettien tarkenne.
Debian GNU/Linux	Free UNIX like operating system. Vapaa UNIX tyyppinen käyttöjärjestelmä.
Delphi method	A method for combining several estimations. Menetelmä useiden arvioiden yhdistämiseksi.
Ethernet	Typical method of implementation for local LANs. Tyypillinen paikallisverkon toteutustapa.
Ethernet frame	Packet that is directed trough ethernet network. This packet can carry for exaple IP packets. Ethernet verkossa kuljetettava paketti. Tämä paketti voi kuljettaa esim. IP paketteja.
Familiar Linux	Linux distribution for handheld computers using StrongArm 110 Processor. Kämmentietokoneelle tarkoitettu Linux-jakelu.
firewall	Limits accessibility between local and public network. Rajoittaa liikennettä paikallisen ja julkisen verkon välillä.
Flash	Non-volatile Random Access Memory. Haihtumaton luku ja kirjoitusoperaatiot salliva muisti.
Gb	Gigabit. Gigabitti
GB	Gigatavu. Gigatavu
GPRS	General Packet Radio System. A new nonvoice value added service that allows information to be sent and received across a mobile telephone network. Uusi palvelu datan siirtoon matkapuhelin verkossa.
HUT	see TKK
IP	Internet Protocol. A connectionless network level protocol layer of the TCP/IP. Yhteydetön TCP/IP:n verkkokerros.
iPAQ	A hand held pen operated computer by Compaq. Compaqin tekemä kynäohjattu taskutietokone.
iPKG	Itsy Package Management System. A lightweight configuration system for Familiar Linux. Kevyt konfiguraationhallintasysteemi Familiar Linuxille.
LAN	Local Area Network, infrastructure of physical connections between computers. Allows data transfers. Tiedonsiirron mahdollistava infrastruktuuri tietokoneiden välillä.
lparse	Front end for Smodels. Käyttöliittymä Smoldssiin.
Mb	Megabit. Megabitti.
MB	Megabyte. Megatavu.
NFS	Network File System, Filesystem that allow use remote harddisks. Tiedostojärjestelmä joka sallii kovalevyjen etäkäytön
package	One component of a configuration. Package can be for example a file containing some program or library. Konfiguraation osa joka voi sisältää ohjelman tai kirjaston.
PC	Personal Computer. Henkilökohtainen tietokone.
PC-Card	See PCMCIA
PCMCIA	Personal Computer Memory Card International Association, An accessory bus used in laptops and handhelds. Salkku- ja kämmenmikrojen yleinen oheislaiteliitäntä
PDA	Personal Digital Assistant, A mobile, handheld computer with software like calendar, contacts, calculator and more. Kannettava tietokone jossa on ohjelmia kuten kalenteri, yhteystiedot, laskin yms.
porting	Modifying the code to work in some other environment. Porttaaminen, Koodin muokkaaminen toisessa ympäristössä toimivaksi.
PPP	Point to Point Protocol, Protocol for serial lines. Protokolla sarjayhteyksille
Processor	An integrated chip that makes arethmetic operations. Integroitu piiri joka suorittaa aritmeettisia operaatioita.

RAM	Random Access Memory, usually volatile. Luku ja kirjoitusoperaatiot salliva muisti, ei pysyvä.
RL	Rule based Language. A language for representing configuration knowledge. Kieli konfiguraatiotiedon esittämiseen.
ROM	Read Only Memory. Vain luettavissa oleva muisti.
RPM	Extension for RedHat Linux packages. RedHat Linuxin pakettien tarkenne.
SoberIT	Software Business and Engineering Institute (in HUT). Ohjelmistoliiketoiminnan ja -tuotannon instituutti (TKK:lla).
SSH	Secure Shell, Secure replacement for Telnet. Turvallinen Telnetin korvike
SCP	Secure CoPy, Secure replacement for FTP. Turvallinen FTPn korvike
SSHD	Secure Shell Daemon, SSH Server. SSH Palvelin
Smodels	An implementation of the stable model semantics for logic programs.
TCP/IP	Transmission Control Protocol, a connection-oriented internet protocol. Yhteydellinen internetprotokolla.
Tirana	Work raporting system. Tuntiraportointijärjestelmä.
TKK	Helsinki University of Technology, Teknillinen Korkeakoulu.
UMTS	Universal Mobile Telecommunications System, Third-generation (3G) mobile communications system. Kolmannen sukupolven matkapuhelinjärjestelmä.
USB	Universal Serial Bus, Serial interface that is used in computers and accessories. Sarjaväylä jota käytetään tietokoneissa ja oheislaitteissa.
USDP	Unified Software Development Process, Generic context for a software project. Ohjelmistoprosessin geneerinen viitekehys.
UML	Unified Modeling Language, a standard for visualization and specification of a software system. Standardi ohjelmiston visualisointiin ja määrittelyyn.
ViCa	Visualization Client Application. Visualisointi ohjelma.
WinCE	Windows CE, Microsofts operating system for handhelds. Microsoftin käyttöjärjestelmä käsimikroille
Wireless Ethernet	802.11b Wireless Ethernet, See WLAN.
WLAN	Wireless Local Area Network. Computer network that uses radio waves to transmit data. Langaton verkko tietokoneiden välillä.
XML	Extensible Markup Language. A markup language for documents containing structured information. Kieli rakenteisten dokumenttien kirjoittamiseen.
xmodem	A file transfer protocol for serial connections. Tiedostonsiirtoprotokolla sarjayhteyksille.
X environment	Graphical window system for UNIX. Graafinen ikkunajärjestelmä UNIXille.

2 Overview

2.1 Environment

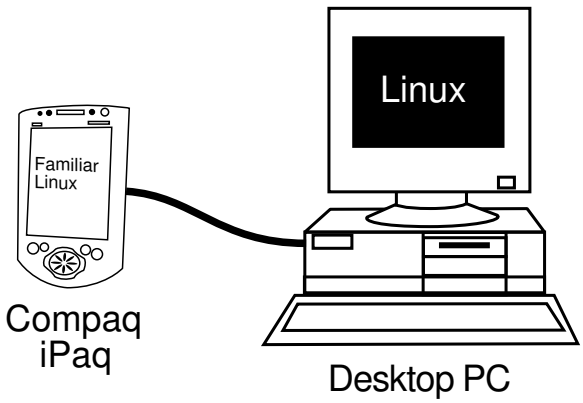

Figure 1: System environment consists of a PC connected to a PDA

The environment consists of a PDA device and a PC workstation. The PDA is connected to the PC with a serial cable, infrared, ethernet or WLAN. The LAN connection will be mostly used during the project.

The PDA is Compaq iPaq Pocket PC H3600 Series with Familiar Linux 0.4 or better installed. It also has a PCMCIA network interface card.

The PC workstation has Linux operating system and the X environment installed. The processor is at least 500MHz. The PC has minimum of 5 Gb hard disk space. The PC also has a 10/100 Mb/s ethernet card.

The PDA will be configured with the PC workstation and all the available package files are stored on the hard disk of the PC.

2.2 User group

The system will be mostly used by the researchers in HUT SoberIT laboratory (the customer). It can be assumed that they have excellent computer skills.

Perhaps later the system will be used by other people with weaker computer skills. But it can be assumed that they know how to use normal GUIs, like Windows applications and WWW-pages.

In this document **User** or **End user** refers to the most common user of the system. She uses the configurator to modify the package set installed in the PDA. This group includes the researchers and possibly other people with weaker computer skills.

Administrators don't use the system so frequently. They do advanced things like making new packages available to the **end users**. Administrators are researchers in the SoberIT and they can use for example a command line interface.

3 About the requirements

3.1 Priorities

The requirements are prioritized using three different priorities: **Must**, **Useful** and **Nice to have**.

Must-priority requirements are necessary properties of the system and they have to be implemented.

Useful requirements are not mandatory but they should be satisfied if there is enough time.

Nice to have requirements are additional and optional features of the system. They will be implemented if there is still time after implementing the **Useful** features.

3.2 Identifiers

Every user requirement is identified by a unique code of the form *UR-XX* where UR means User Requirement and XX is the number of the requirement. This makes the requirement traceability easier.

3.3 Fit criteria

Every user requirement has also a **fit criterion**. It is a metric such that it is possible to test if an implementation satisfies the original requirement. Fit criteria make the requirements testable.

4 Architectural requirements

This project creates some parts to a larger software system. There are existing components that will be used. Therefore the main architecture of the system is defined by the customer.

4.1 UR-01 - Main components and their interconnections U

Identifier:	UR-01
Priority:	Useful
Fit criterion:	Each component can be compiled into a separate executable file.
Created:	09.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

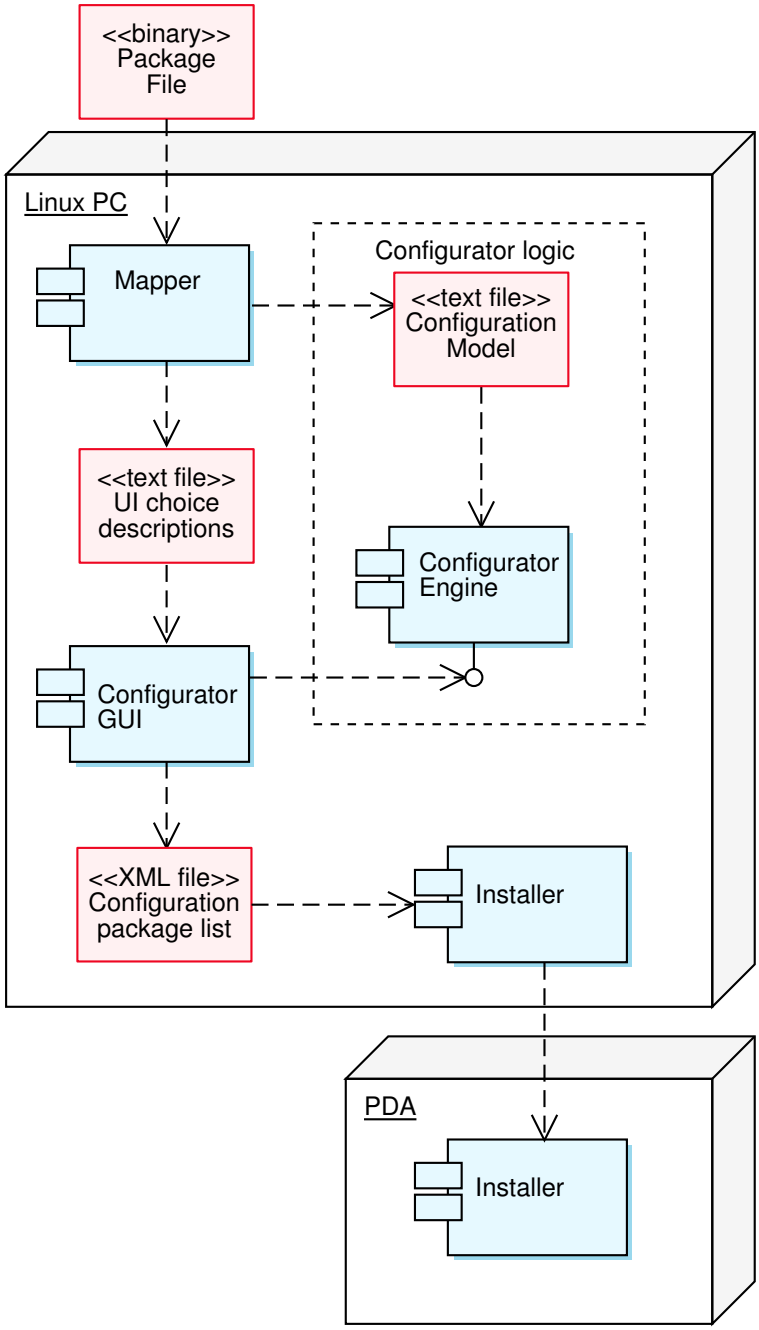

Figure 2: UML Component diagram about the suggested architecture of the system.

The customer suggested an architecture shown in Figure 2. The system consists of a **Mapper** that imports iPKG package files, a **Configurator GUI** that receives the user commands and **Installer**s on both the PDA and the PC. The installers transfer and install the configuration to the PDA.

The **Configuration logic** part of the system has already been implemented in a previous HUT project. The **Configuration Engine** can be asked if a certain configuration is valid. It makes the decision based on the package dependencies in the **Configuration Model**.

4.2 UR-02 - The system must use the Configuration Engine M

Identifier:	UR-02
Priority:	Must
Fit criterion:	The system calls the configuration engine and uses its results to determine the configuration validity. The system source code does not contain any configuration logic.
Created:	22.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

The configuration logic is already implemented by Tommi Syrjänen in the **Configuration Engine**. The system must use that engine to determine the configuration validity.

5 Internal interfaces

5.1 UR-03 - The system reads a text file listing the available packages M

Identifier:	UR-03
Priority:	Must
Fit criterion:	The text file must exist on the file system and only its content affects the list of available packages shown on the UI.
Created:	09.10.2001 Antti Haapakoski
Last modified:	26.10.2001 Jussi Vainionpää

The **UI choice descriptions** text file contains at least the names of the packages to be shown in the UI. It may also contain information on the hierarchy of the packages.

5.2 UR-04 - The system produces a XML file listing the packages in the configuration M

Identifier:	UR-04
Priority:	Must
Fit criterion:	A text file must be generated on the file system and that file solely determines all the packages in the configuration. The file must be in XML format specified by the customer.
Created:	09.10.2001 Antti Haapakoski
Last modified:	26.10.2001 Jussi Vainionpää

The text file contains a list of packages in the configuration. The file is in XML format. The XML objects are specified separately by the customer. The packages are listed in installation order. The file is read by other components of the system to actually do the installation. If required extra XML parameters or tags may be used to describe whether package is already installed.

6 Functional requirements

Most of the functional requirements are derived from the use cases of the system shown in Figure 3. Rest of the functional requirements define more detailed properties of the system.

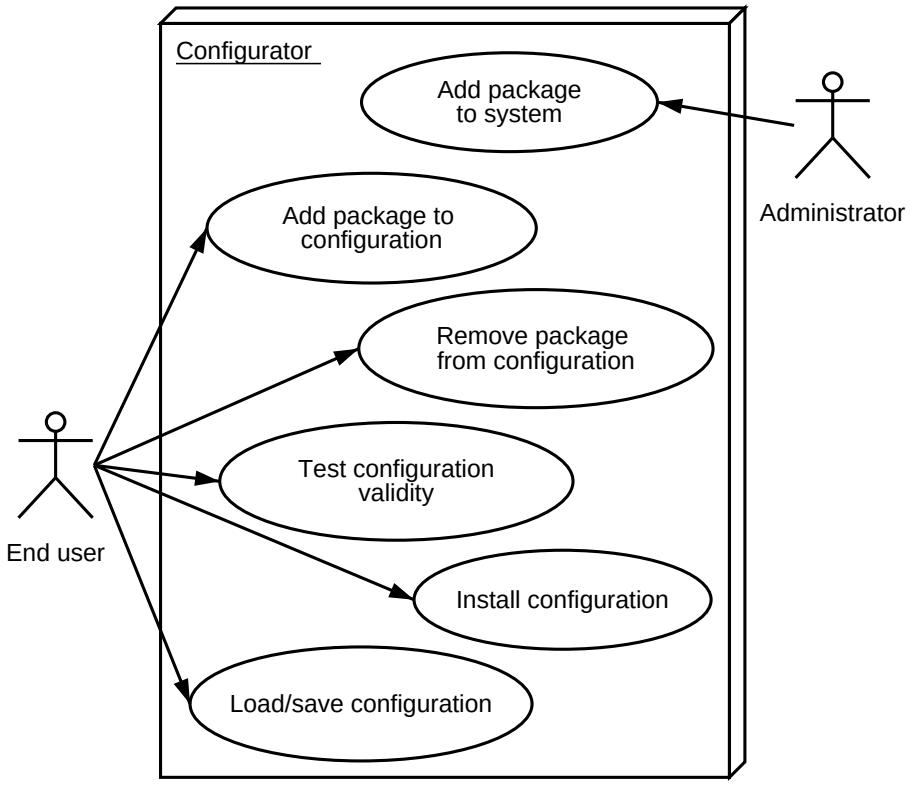

Figure 3: Use cases that have human actors

6.1 UR-05 - User can add a package to the configuration M

Use case name:	Add package to configuration (see Fig 3)
Actors:	End user
Frequency:	When configuring: Once in 10 seconds
Preconditions:	The configurator UI has been started. A configuration is shown.
Postconditions:	The new package is in the configuration
Basic course of action:	1. User finds a nice package not in the configuration 2. User visually selects the package 3. The system adds the package to the configuration
Exceptions:	If real-time validity checking is used, see UR-07 in 6.3.

Identifier:	UR-05
Priority:	Must
Fit criterion:	When the user creates a certain UI event, a new package is included in the configuration under construction: This means three things: 1. The new package is visually shown to be in the configuration. 2. The package is included in validity testing. 3. If the configuration is installed, the new package will be installed.
Created:	09.10.2001 Antti Haapakoski
Last modified:	26.10.2001 Jussi Vainionpää

The user must be able to make a configuration using a graphical user interface on the PC workstation. The user makes the configuration by selecting packages shown on the screen.

Minimal case is that the user can select from two working configurations. But usually there are numerous configurations that can be made by selecting packages.

6.2 UR-06 - User can remove a package from the configuration M

Use case name:	Remove package from configuration (see Fig 3)
Actors:	End user
Frequency:	When configuring: Once in 30 seconds
Preconditions:	The configurator UI has been started. A configuration is shown.
Postconditions:	The package is not in the configuration
Basic course of action:	1. User finds a package in the configuration 2. User visually unselects the package 3. The system removes the package from the configuration
Exceptions:	If real-time validity checking is used, see UR-07 in 6.3.

Identifier:	UR-06
Priority:	Must
Fit criterion:	When the user creates a certain UI event, a package is removed from the configuration: This means three things: 1. The new package is visually shown NOT to be in the configuration anymore. 2. The package is NOT included in validity testing anymore. 3. If the configuration is installed, the package will NOT be installed.
Created:	09.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

The user can remove a previously added package from the configuration.

6.3 UR-07 - User can test if a configuration is valid M

Use case name:	Test configuration validity (see Fig 3)
Actors:	End user
Frequency:	When configuring: Once in 60 seconds (or once in 10s if real-time checking)
Preconditions:	The configurator UI has been started. A configuration is shown.
Postconditions:	The configuration remains the same. (or is fixed, see UR-13 in 6.9)
Basic course of action:	1. User has added/removed some package(s) 2. User indicates he/she wants to check the validity. (or the real-time validity checking is on) 3. The system checks the configuration validity 4. The system tells the configuration was valid
Exceptions:	System takes some action if the configuration is not valid (see UR-11 in 6.7)

Identifier:	UR-07
Priority:	Must
Fit criterion:	When the user creates a certain UI event, the system calls the configurator engine to test if the configuration is valid and tells the result to the user. Valid configurations are valid in terms of the configuration model.
Created:	09.10.2001 Antti Haapakoski
Last modified:	26.10.2001 Jussi Vainionpää

Minimal requirement is that the user can explicitly test the validity of the configuration. But it would be **useful** if the validity of the configuration would be tested **real-time** as the user adds or removes the packages.

6.4 UR-08 - Administrator can add new packages to the system M

Use case name:	Add package to system (see Fig 3)
Actors:	Administrator
Frequency:	Once a week
Preconditions:	The package is not in the system yet
Postconditions:	The package is now available for user selection
Basic course of action:	1. Administrator has a package not in the system 2. Administrator asks the system to import the package 3. The system adds the package to its internal structures
Exceptions:	If the package is not valid, an error is reported.

Identifier:	UR-08
Priority:	Must
Fit criterion:	The administrator can make the system add a package. After addition, the new package becomes available to the configurator UI and in the Configuration Engine.
Created:	09.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

The system can read an iPKG package of Familiar Linux and import its dependencies to the **Configuration Model** and other package information to the internal data structures of the configurator.

6.5 UR-09 - System shows a list of available packages M

Identifier:	UR-09
Priority:	Must
Fit criterion:	The user can tell what packages are available by looking at the UI.
Created:	09.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

When the user is making a configuration, the system shows a list of available packages. This requirement is satisfied if UR-10 is satisfied because UR-10 is a more demanding requirement.

6.6 UR-10 - System shows a hierarchical view of available packages U

Identifier:	UR-10
Priority:	Useful
Fit criterion:	Packages are visually arranged as a tree, not a pure list.
Created:	09.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

The available packages are shown as a tree. This helps the user to manage a large set of packages and to understand their relationships.

6.7 UR-11 - System shows error message if configuration is invalid M

Identifier:	UR-11
Priority:	Must
Fit criterion:	After validation, if the configuration is invalid, the user will always know it.
Created:	09.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

When checking the validity of a configuration (see UR-07 in 6.3), the system can find out that the configuration is invalid. In this case at least an error message coming from the **Configuration Engine** must be shown.

UR-12 and UR-13 are stronger requirements than this. If one of them is satisfied then is also this requirement.

6.8 UR-12 - System shows conflicting packages if configuration is invalid U

Identifier:	UR-12
Priority:	Useful
Fit criterion:	After validation, if the configuration is invalid, the user will always know why the configuration is invalid.
Created:	09.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

This is a more demanding version of requirement UR-11. In addition to showing an error message, the system shows also the dependency conflict that caused the error.

UR-13 is stronger requirement than this. If UR-13 is satisfied then is also this.

6.9 UR-13 - System suggests a solution if configuration is invalid N

Identifier:	UR-13
Priority:	Nice to have
Fit criterion:	After validation, if the configuration is invalid, the system shows a suggestion that will make the configuration correct.
Created:	09.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

This is a more demanding version of requirements UR-11 and UR-12. In addition of showing the conflicting packages, the system also suggests a solution to the conflict.

6.10 UR-14 - User can choose to install the configuration to PDA M

Use case name:	Install configuration (see Fig 3)
Actors:	End user
Frequency:	Once a week
Preconditions:	The configurator UI has been started. The configuration is valid
Postconditions:	The PDA contains the new configuration
Basic course of action:	1. User is happy with the configuration 2. User commands the system to install the configuration 3. The system installs the configuration 4. The system tells that installation was successful
Exceptions:	If an error occurs during installation, see UR-15 in 6.11.

Identifier:	UR-14
Priority:	Must
Fit criterion:	There is a control on the UI that activates the installation. After the installation, the configuration is on the PDA.
Created:	09.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

When the user has made a valid configuration he/she may want to install it to the PDA. The system transfers the new packages to the PDA and installs them.

6.11 UR-15 - The system knows how the installation succeeded M

Identifier:	UR-15
Priority:	Must
Fit criterion:	If an error occurs during the installation, it is reported to the user or at least to some log file or console. Successful installation is also reported.
Created:	09.10.2001 Antti Haapakoski
Last modified:	26.10.2001 Jussi Vainionpää

When installing to the PDA, an error could occur. An error should be reported to the PC side of the system. Errors are not analyzed and the user is instead instructed to go back to the previous configuration. It would be useful have a list of packages that were successfully installed reported to the PC side.

6.12 UR-16 - The system can remove packages from PDA N

Identifier:	UR-16
Priority:	Nice to have
Fit criterion:	When a package is removed from the configuration and installation is started, the system adds no packages to the PDA and removes only the selected one. No packages are sent to the PDA.
Created:	09.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

Usually additional packages are installed to the PDA but sometimes the user may want to remove some packages. It would be nice if the user did not have to "clear" the PDA and reinstall all but those removed packages. This means that the PDA-side of the system must be able to uninstall single packages. See also UR-21 in 6.16.

6.13 UR-17 - There is a default configuration U

Identifier:	UR-17
Priority:	Useful
Fit criterion:	User does not have to build any configuration from scratch.
Created:	09.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

The user does not have to build his/her configuration from scratch. There is a basic configuration the user can start to modify.

6.14 UR-19 - User can save a configuration M

Use case name:	Save configuration (see Fig 3)
Actors:	End user
Frequency:	Once in an hour
Preconditions:	The configurator UI has been started.
Postconditions:	The current configuration is copied to the file
Basic course of action:	1. User is happy with the configuration 2. User commands the system to save the configuration 3. Possibly the system asks for a file name 4. The system saves the configuration to a file
Exceptions:	If the file exists, user is asked for confirmation. If an error occurs, the system shows an error message.

Identifier:	UR-19
Priority:	Must
Fit criterion:	There is a control on the UI that activates the configuration saving. After the save, the current configuration is copied to the selected file. The configuration is saved in XML format (see UR-04 in 5.2.)
Created:	09.10.2001 Antti Haapakoski
Last modified:	26.10.2001 Jussi Vainionpää

The user can explicitly save configurations. This makes possible to remember the configurations of several PDAs.

6.15 UR-20 - User can load a configuration M

Use case name:	Load configuration (see Fig 3)
Actors:	End user
Frequency:	Once in a day
Preconditions:	The configurator UI has been started.
Postconditions:	The current configuration is replaced by the one in the file
Basic course of action:	1. User wants to look or change an old configuration 2. User commands the system to load the configuration 3. Possibly the system asks for a file name 4. The system loads a configuration from a file
Exceptions:	If the file does not exists or is invalid, an error is reported.

Identifier:	UR-20
Priority:	Must
Fit criterion:	There is a control on the UI that activates the configuration loading. After the load, the current configuration is replaced by the one in the selected file. The file is in XML format (see UR-04 in 5.2)
Created:	09.10.2001 Antti Haapakoski
Last modified:	26.10.2001 Jussi Vainionpää

The user can explicitly load configurations. This makes possible to remember the configurations of several PDAs.

6.16 UR-21 - Incremental installation of packages U

Identifier:	UR-21
Priority:	Useful
Fit criterion:	When a new package is added to the configuration and installation is started, the system removes no packages from the PDA but only installs the new one. Only the new package is sent to the PDA.
Created:	09.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

First the user has an empty PDA where the system installs all the packages in the configuration. But later the PDA already contains some packages. One solution would be to “clear” the PDA before installation and then install again all packages of the configuration.

Incremental installation means that only the missing packages are transferred and installed to the PDA. See also UR-16 in 6.12.

7 Quality requirements

7.1 UR-22 - Source code is well commented in English M

Identifier:	UR-22
Priority:	Must
Fit criterion:	Every source file must have a start comment describing the file contents. Every class/interface has a comment about its meaning. Every comment must be in English.
Created:	09.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

Source file comments are written in English and each source file must have a comment about its contents in the beginning of the file. Important classes, methods and functions and their parameters must also be commented.

7.2 UR-23 - The system components are loosely connected and modular M

Identifier:	UR-23
Priority:	Must
Fit criterion:	This cannot be precisely defined. The customer decides this by studying the system design and implementation.
Created:	09.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

The tasks of the configuration should be divided into different system components. The components should have a clear and minimal interface. There should be as little dependencies as possible between the components. The customer has suggested an architecture to satisfy this requirement, see UR-01 in 4.1.

8 Efficiency requirements

8.1 UR-24 - The memory use must be known and predictable N

Identifier:	UR-24
Priority:	Nice to have
Fit criterion:	The memory usage of the installation must be known (in kb). The memory requirements of each package is stored in the XML file describing the installation as size in the same way as price in the examples.
Created:	09.10.2001 Antti Haapakoski
Last modified:	26.10.2001 Jussi Vainionpää

The memory usage of the **installation** in the PDA should be known in all situations.

8.2 UR-25 - Minimal temporary storage when installing to PDA N

Identifier:	UR-25
Priority:	Nice to have
Fit criterion:	All packages of the configuration to be installed are not concurrently in the PDA.
Created:	09.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

There are several ways to install the packages to PDA: transfer and install all packages at once, transfer and install one package at a time or install the packages directly from the PC. Different ways of installation consume different amounts of storage space. This requirement means that memory usage of packages should be minimized.

9 Portability requirements

9.1 UR-26 - All of the system can be ported to PDA N

Identifier:	UR-26
Priority:	Nice to have
Fit criterion:	The whole system can be run with the processor of the PDA
Created:	09.10.2001 Antti Haapakoski
Last modified:	22.10.2001 Antti Haapakoski

Most parts of the system will run on the Linux PC. In the future, some or all those parts may be ported to the PDA. This sets restrictions to the available programming languages and libraries. For example advanced features and libraries of C++ may not be supported on PDAs.

10 Usability requirements

10.1 UR-27 - The configurator must have a graphical user interface. M

Identifier:	UR-27
Priority:	Must
Fit criterion:	UI can be used with the mouse cursor.
Created:	20.10.2001 Ari Haapaniemi
Last modified:	22.10.2001 Antti Haapakoski

The basic requirement is that the user interface is graphical.

But it would be **Useful** if the UI was a dynamic, web-based application enabling communication via HTTP protocol.

10.2 UR-28 - Efficiency of use N

Identifier:	UR-28
Priority:	Nice to have
Fit criterion:	An experienced user can make a normal configuration in 1 minute.
Created:	09.10.2001 Antti Haapakoski
Last modified:	25.10.2001 Antti Haapakoski

This requirement sets the minimum speed of use of the system for an experienced user.

10.3 UR-29 - Ease of learning N

Identifier:	UR-29
Priority:	Nice to have
Fit criterion:	A new user who has never seen the system can learn to select and install a configuration in 15 minutes. A help file is not required.
Created:	09.10.2001 Antti Haapakoski
Last modified:	26.10.2001 Jussi Vainionpää

This requirement means the system is easy to use for a new inexperienced user.

11 External interfaces

11.1 Connection between the PC and the PDA

As the section 2.1 describes, the PDA may be connected to the PC in many different ways. So to support all of them, a high level (OSI network level) protocol is needed.

So probably **TCP/IP** will be used in the PC<->PDA communication.

11.2 Configurator engine

Configurator engine contains several tools which are used to convert a Debian GNU/Linux package description file to the rule-based language RL [1]. RL is based on stable model semantics and represents configuration knowledge, more precisely Debian GNU/Linux configuration model and user requirements. It improves current configuration model providing new packages (software, aggregated) and more importantly formal semantics for package relation descriptions.

Smodels [2] is an engine that tries to find a proper and optimized configuration for the required packages. We will use smodels via lparse [3] front-end, input of which is a logic program containing the user requirements and the package configuration compiled by translator from filtered configuration model. Filtered configuration model is created by change_list_format-script removing inconsistencies from Debian GNU/Linux Packages.gz. Output from smodels is also filtered with extract_configuration-script to create a description file containing only packages (name, version) to be included the new valid configuration.

Lparse, translator, scripts and the formal model are made by Tommi Syrjänen, smodels originates from Patrik Simons.

11.2.1 Configurator's requirements for interfaces

- Input for configuration engine should be in Debian GNU/Linux Packages.gz format, ascii file or alike. Lparse needs a precompiled logic-program (examples [6]) from packages description file.
- Output is an ascii file where required packages (name, version) are in readable format.
- User requirements are in an ascii file using logic programming format as input for lparse, examples [6].
- Smodels [5] requires a unix shell, c++ compiler.
- Lparse [4] requires a unix shell and c++ compiler.
- Translator [7] requires a unix shell and c++ compiler.
- Scripts [7] made to parse input for translator and output from smodels are made by Perl requiring Perl interpreter and a shell.

11.2.2 Preferable environment

- Debian GNU/Linux 2.0 or greater (other Linux distributions may be suitable).
- Bourne Shell or equivalent.
- Perl 5.005 or greater.
- GCC 2.8.1 or greater.

12 Requirements Traceability Matrix

A requirements traceability matrix is used to allow requirements to be easily traced to the components implementing them. The matrix is also used for backwards tracing, for example to see what requirements are affected if some component falls behind schedule and gets reconsidered.

Only such requirements that can be mapped to distinct components are included — it would not make sense to have something like "UR-23 - The system components are loosely interconnected and modular" in the matrix as it would apply to every column.

The matrix will be included at least in the design specification.

References

[1] Syrjänen Tommi, *Research Report A55: A Rule-Based Formal Method for Software Configuration*, 74 pages, December, 1999. <http://www.tcs.hut.fi/Publications/reports/A55abstract.html>

[2] Simons Patrik, *Research Report A58: Extending and Implementing the Stable Model Semantics*, 109 pages, April, 2000. <http://www.tcs.hut.fi/Publications/reports/A58abstract.html>

[3] Syrjänen Tommi, *Lparse 1.0 User's Manual*, 2000. <http://www.tcs.hut.fi/Software/smodels/lparse.ps>

[4] Syrjänen Tommi, *lparse-1.0.9.tar.gz source package*. <http://www.tcs.hut.fi/Software/smodels/src/lparse-1.0.9.tar.gz>

[5] Simons Patrik, *smodels-2.26.tar.gz source package*. <http://www.tcs.hut.fi/Software/smodels/src/smodels-2.26.tar.gz>

[6] Syrjänen Tommi, *smodels-debian-model-1.0.tar.gz source package*. <http://www.tcs.hut.fi/~tssyrjan/configuration/smodels-debian-model-1.0.tar.gz>

[7] Syrjänen Tommi, *translator and scripts.*