ABout Time Linux - dependency engine (v0.1)

Roel Arents
Roel.Arents@student.ru.nl

Sander van Hooft
info@sandervanhooft.nl

Matthijs Smith
MatthijsSmith@student.ru.nl

Sharon Wezer
S.Wezer@student.ru.nl

December 15, 2005

Contents

1 Introduction **4**

2 Problem Statement **4**

3 Stakeholder Analysis **4**

4 Mission, Vision and Values **4**

5 Statement of Work **5**

6 Risk Analysis **6**

7 Use Case Survey **8**
7.1 Provide Depends Tree Up 8
7.2 Provide Depends Tree Down 8
7.3 Check Version Compatibility 8
7.4 Provide the Number of Dependencies of a Package 9
7.5 Provide the Number of Dependants on a Package 9
7.6 Provide Proper Build Order 9
7.7 Provide the List of Packages Without Dependencies 10
7.8 Use Case Diagram 10

8 Use Cases **11**
8.1 Provide Depends Tree Up 11
8.2 Provide Depends Tree Down 12
8.3 Check Version Compatibility 13
8.4 Provide the Number of Dependencies of a Package 14
8.5 Provide the Number of Dependants on a Package 15
8.6 Provide Proper Build Order 16
8.7 Provide the List of Packages Without Dependencies 17

9 Scenarios **18**
9.1 Use Case 1: Provide Depends Tree Up 18
9.2 Use Case 2: Provide Depends Tree Down 20
9.3 Use Case 3: Check Version Compatibility 22
9.4 Use Case 4: Provide the Number of Dependencies of a Package 24
9.5 Use Case 5: Provide the Number of Dependants on a Package . 24
9.6 Use Case 6: Provide Proper Build Order 25
9.7 Use Case 7: Provide the List of Packages Without Dependencies 26

10 Domain Models **27**
10.1 ORM Diagram Use Case 1 and 2 27
10.2 ORM Diagram Use Case 3 28
10.3 ORM Diagram Use Case 4 and 5 28
10.4 ORM Diagram Use Case 6 29
10.5 ORM Diagram Use Case 7 29

11 Business Rules **30**

12 Non-functional Requirements **30**

13 Terminological Definitions **32**

14 Example Trees **33**

14.1 Tree 1 33

14.2 Tree 2 34

14.3 Tree 3 34

1 Introduction

ABout Time Linux is a project that plans to provide a source-based Linux distribution based on a tool set that has been clearly documented from the beginning of the design cycle. The project we are working on is about the design of AbTLinux, our main goal will be to deliver a clear design document. Software packages, installed on AbTLinux, may have one or (many) more dependencies between themselves. There are two main types of dependencies: ”Relies On”-dependencies (RO) and ”Depends On”-dependencies (DO). These dependencies are either required or optional. The dependencies implicate that specific conditions for a coherent consistency are met. If the consistency is lost due to changes in the system (made by user), then we speak of broken dependencies, which leads to a not properly working system.

2 Problem Statement

After every change in the system (that may be initiated by a user) a tool should perform a dependency check to find out if the requirements for the dependencies between software components are fulfilled. At this moment there is no such tool. Absence of this tool is the main reason for the development of a new tool. The goal of this project is to develop a tool to help check if the requirements for the dependencies are fulfilled. Possible actions can be taken based on the results of these checks.

3 Stakeholder Analysis

a. Project leader and Executive Sponsor: Eric Schabell Mr Schabell is the developer of the desired tool. Therefore, he has great interest in what the tool should do exactly. Furthermore, for him proper documentation and a precise description of the new tool are especially important. If the requirements of this new tool are well described, the developer should be able to build the program exactly as the customer wanted. He is also responsible for the funding of the development of this new tool.

b. Initiators: Laurent Wandrebeck, David Demonchy, Eric Schabell These initiators were the first to recognise the problem as described in the problem statement. They will have a part in developing the tool, once the requirements are finished. Also they will be able to help with problems concerning the requirements and will have a supervising role.

4 Mission, Vision and Values

Mission
The aim of this projectgroup is to develop the requirements for a necessary tool that will find dependencies between software packages on a computer system. We aspire to describe the requirements in such a way that the future developers of this tool will have no questions about what this tool should do

and how it should do it. As a result, future users of Linux will see that working with several software packages in any way will be much less complicated as it is today

Vision
The depEngine of the abt tool will keep track of package dependencies. Based on that knowledge, it will provide smartly sorted installation orders for the abt tool, given a certain wanted change (like install or remove a package). For example when you want to update some installed package, the depEngine will figure out what other packages depend on the to-be-updated package and whether they need to be changed too. Also it will look for packages that the to-be-updated package depends on and if they are available.

Values
- The most important guideline for the building of the AbT-linux distribution is documenting everything clearly, and having made requirements of the to-be-built software before starting to code. This also applies to the depEngine component.
- Another important value is to keep it free (as in beer) and licensed under the GPL.
- Also Keeping It Simple (silly) is a value which is kept in mind for the whole project.

5 Statement of Work

1. **Scope**
The dependency engine is going to be a part of the software package manager. This application will be used by ABTLinux users, who are in need of an easy-to-use application to manage the dependencies between software packages. Most of the requirements are listed by Eric Schabell, initiator of the project.

2. **Objectives**
To investigate and list the requirements for a correct working DepEngine. We will answer the questions "what is the DepEninge and what should it do?".

3. **Application Overview**
The application will help manage (and specify) several types of build dependencies and with this list it will be able to prevent any loss of necessary packages.

4. **User Demography**
The depEngine will be used by the software package manager. Which can be used by the AbtLinux users for several actions on packages. Also developers and future administrators will use the depEngine in the development process and in the maintainance process.

5. **Constraints**
Lack of time due to other activities related to Information Science and personal activities.

Some of the stakeholders live abroad, it makes it more difficult to communicate.

6. **Assumptions**
We assume to get enough feedback in time on the document we handed in.
We assume that the stakeholders are cooperative.
We assume that the stakeholders have enough knowledge to answer our questions.

7. **Staffing**
As a group everything concerning the project will be discussed and later on parts will be divided among the teammembers to work on it themselves. These parts will then be joined together and again discussed by the group.

8. **Deliverable Outlines**
We will deliver the Requirements LaTeX file following the given template on AbTlinux.org, to enable Eric Schabell to develop the depEngine tool.

9. **Expected Duration**
Deliverable dates:
Thursday 13 Oct: Facade Phase
Thursday 17 Nov: Filled Phase
Thursday 15 Dec: Focused Phase

6 Risk Analysis

Definition

No.	= Number
RNB	= Resolution Needed By
DLIIO	= Days Lost If It Occurs
LIWH	= Likelihood It Will Happen
RR	= Risk Rating

No.	Risk	Category	RNB	Status	DLIIO	LIWH	RR
1	Since the developers have to work on the project in their own time, a release date for the project cannot be given. In the worst case, it will never finish.	Time	na	Cannot be resolved	na	na	na
2	Because the stakeholders are pretty much the same people as the developers, time could be an issue here too, for the same reason as risk no 1.	Time	na	Cannot be resolved	na	na	na
3	Since the requirements engineers keep their created documents on their own computers, there's a small chance that all data could get lost. But there's a big(ger) chance that a little bit of the documents could get lost.	Comm. / Data	asap	Resolved by creating a forum on which the documents will be posted. Also e-mailing implicitly creates backups.	1	50%	0.5
4	Almost all of the off-site communication between the stakeholders and the REs, as well as between the REs themselves, is with the help of the internet. If at a time, no internet connection is at hand, it's very difficult to work on the project, because also a lot of information won't be available.	Comm. / Data	na	Will not be resolved	1	50% (either there's no (computer with an) internet connection available, or the internet connection is down.	0.5

7 Use Case Survey

7.1 Provide Depends Tree Up

Use Case Number:	**1**
Use Case Name:	**Provide Depends Tree Up**
Initiating Actor:	AbT-tool
Description:	The depEngine receives a package name and version to sort out the dependencies of this package and return a dependency tree.
Completeness:	Complete
Maturity:	Mature
Dependency:	None
Source:	Eric Schabell
Comments:	None

7.2 Provide Depends Tree Down

Use Case Number:	**2**
Use Case Name:	**Provide Depends Tree Down**
Initiating Actor:	AbT-tool
Description:	The depEngine receives a package name and version to sort out the dependants on this package and return a dependency tree.
Completeness:	Complete
Maturity:	Mature
Dependency:	None
Source:	Eric Schabell
Comments:	None

7.3 Check Version Compatibility

Use Case Number:	**3**
Use Case Name:	**Check Version Compatibility**
Initiating Actor:	AbT-tool
Description:	The depEngine will check the given dependency tree for version compatibility between packages that depend on each other.
Completeness:	Complete
Maturity:	Mature
Dependency:	None
Source:	Eric Schabell
Comments:	None

7.4 Provide the Number of Dependencies of a Package

Use Case Number:	**4**
Use Case Name:	**Provide the Number of Dependencies of a Package**
Initiating Actor:	AbT-tool
Description:	The depEngine receives a package name and version to find out the number of dependencies of this package.
Completeness:	Complete
Maturity:	Mature
Dependency:	None
Source:	Eric Schabell
Comments:	None

7.5 Provide the Number of Dependants on a Package

Use Case Number:	**5**
Use Case Name:	**Provide the Number of Dependants on a Package**
Initiating Actor:	AbT-tool
Description:	The depEngine receives a package name and version to sort out the number of dependants on this package.
Completeness:	Complete
Maturity:	Mature
Dependency:	None
Source:	Eric Schabell
Comments:	None

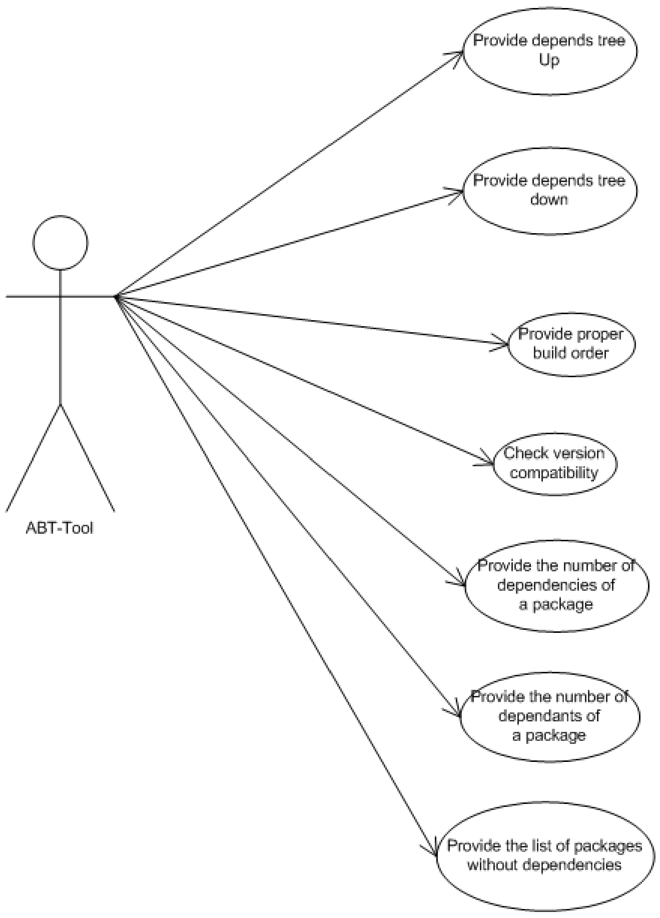
7.6 Provide Proper Build Order

Use Case Number:	**6**
Use Case Name:	**Provide Proper Build Order**
Initiating Actor:	AbT-tool
Description:	The depEngine receives a package name and version to sort out the proper build order, using the package up and down tree.
Completeness:	Complete
Maturity:	Mature
Dependency:	None
Source:	Eric Schabell
Comments:	None

7.7 Provide the List of Packages Without Dependencies

Use Case Number:	**7**
Use Case Name:	**Provide the List of Packages Without Dependencies**
Initiating Actor:	AbT-tool
Description:	The depEngine provides the AbT-tool with the list of packages which dont have any dependencies.
Completeness:	Complete
Maturity:	Mature
Dependency:	None
Source:	Eric Schabell
Comments:	None

7.8 Use Case Diagram


Provide depends tree Up
Provide depends tree down
Provide proper build order
Check version compatibility
Provide the number of dependencies of a package
Provide the number of dependants of a package
Provide the list of packages without dependencies
ABT-Tool

8 Use Cases

8.1 Provide Depends Tree Up

Use Case Name:	**Provide Depends Tree Up**
Authors:	Roel Arents, Sander van Hooft, Matthijs Smith, Sharon Wezer
Dates:	December 14, 2005
Iteration:	Focused
Description:	The depEngine receives a package name and version to sort out the dependencies of this package and return a dependency tree.
Preconditions:	None
Triggers:	Request from the AbT-tool for a dependency tree of dependencies of the given package.
Basic Course of Events:	1. Receive package name and version from the AbT-tool 2. Read package description 3. Find the packages which the given package depends on 4. Check the found packages for other packages which they depend on. Repeat this until there are no new packages found 5. Create the dependency tree with the found packages 6. Write these depEngine actions to a log 7. Return the tree to the AbT-tool
Alternatives:	4. If a package is already checked for its dependencies do not check it again.
Exceptions:	None
Postconditions:	- The AbT-tool has a dependency tree of packages which the given package depends on.
Related business rules:	Business Rule 1, 2 and 3

8.2 Provide Depends Tree Down

Use Case Name:	**Provide Depends Tree Down**
Authors:	Roel Arents, Sander van Hooft, Matthijs Smith, Sharon Wezer
Dates:	December 14, 2005
Iteration:	Focused
Description:	The depEngine receives a package name and version to sort out the dependants on this package and return a dependency tree.
Preconditions:	None
Triggers:	Request from the AbT-tool for a dependency tree of dependants on the given package.
Basic Course of Events:	1. Receive package name and version from the AbT-tool 2. Read package description 3. Find the packages which depend on given package 4. Check for other packages which depend on the found packages. Repeat this until there are no new packages found. 5. Create the dependency tree with the found packages. 6. Write these depEngine actions to a log 7. Return the tree to the AbT-tool
Alternatives:	4. If a package is already checked for its dependants do not check it again.
Exceptions:	None
Postconditions:	- The AbT-tool has a dependency tree of packages which depend on the given package.
Related business rules:	Business Rule 1, 2 and 3

8.3 Check Version Compatibility

Use Case Name:	**Check Version Compatibility**
Authors:	Roel Arents, Sander van Hooft, Matthijs Smith, Sharon Wezer
Dates:	December 14, 2005
Iteration:	Focused
Description:	The depEngine will check the given dependency tree for version compatibility between packages that depend on each other.
Preconditions:	- The depEngine has a dependency tree of packages.
Triggers:	The AbT-tool sends a request to check for version compatibility.
Basic Course of Events:	1. Receive the dependency tree from the AbT-tool 2. Compare the versions in the package description with the actual versions of the packages in the tree 3. Write these depEngine actions to a log 4. Return a message to the AbT-tool that every package version is compatible
Alternatives:	4. If the comparison is false, return a message that the package versions are incompatible.
Exceptions:	None
Postconditions:	- The AbT-tool will receive a message whether the package versions are compatible or incompatible.
Related business rules:	Business Rule 1, 2 and 3

8.4 Provide the Number of Dependencies of a Package

Use Case Name:	**Provide the Number of Dependencies of a Package**
Authors:	Roel Arents, Sander van Hooft, Matthijs Smith, Sharon Wezer
Dates:	December 14, 2005
Iteration:	Focused
Description:	The depEngine receives a package name and version to find out the number of dependencies of this package.
Preconditions:	None
Triggers:	Request from the AbT-tool for the number of dependencies of the given package.
Basic Course of Events:	1. Receive the dependency tree for the package from the AbT-tool 2. Count the packages in the tree 3. Log the count actions 4. Return the number to the AbT-tool
Alternatives:	None
Exceptions:	None
Postconditions:	- The AbT-tool has the number of packages which the given package depends on.
Related business rules:	Business Rule 1, 2 and 3

8.5 Provide the Number of Dependants on a Package

Use Case Name:	**Provide the Number of Dependants on a Package**
Authors:	Roel Arents, Sander van Hooft, Matthijs Smith, Sharon Wezer
Dates:	December 14, 2005
Iteration:	Focused
Description:	The depEngine receives a package name and version to sort out the number of dependants on this package.
Preconditions:	None
Triggers:	Request from the AbT-tool for the number of dependants on the given package.
Basic Course of Events:	1. Receive the dependants tree for the package from the AbT-tool 2. Count the packages in the tree 3. Log the count actions 4. Return the number to the AbT-tool
Alternatives:	None
Exceptions:	None
Postconditions:	- The AbT-tool has the number of packages which depend on the given package
Related business rules:	Business Rule 1, 2 and 3

8.6 Provide Proper Build Order

Use Case Name:	**Provide Proper Build Order**
Authors:	Roel Arents, Sander van Hooft, Matthijs Smith, Sharon Wezer
Dates:	December 14, 2005
Iteration:	Focused
Description:	The depEngine receives a package name and version to sort out the proper build order, using the package up and down tree.
Preconditions:	- The depEngine has an up and down dependency tree of packages.
Triggers:	Request of the AbT-tool for the proper build order.
Basic Course of Events:	1. Receive the package up tree from the AbT-tool 2. Sort packages descending on depth 3. Put the packages on the build order list 4. Place the package on build order list 5. Receive the package down tree from the AbT-tool 6. Sort the packages ascending on depth 7. Put the packages on the build order list 8. Log sort action 9. Return the build order list to the AbT-tool
Alternatives:	None
Exceptions:	None
Postconditions:	The AbT-tool receives a list with the proper build order for the package.
Related business rules:	Business Rule 1 and 2

8.7 Provide the List of Packages Without Dependencies

Use Case Name:	**Provide the List of Packages Without Dependencies**
Authors:	Roel Arents, Sander van Hooft, Matthijs Smith, Sharon Wezer
Dates:	December 14, 2005
Iteration:	Focused
Description:	The depEngine provides the AbT-tool with the list of packages which dont have any dependencies.
Preconditions:	- The depEngine has a dependency tree of packages.
Triggers:	Request from the AbT-tool for a list of packages without dependencies.
Basic Course of Events:	1. Receive the package tree from the AbT-tool 2. Check all packages descriptions if they don't have any dependencies 3. Add those packages to a list 4. Write a log 5. Return this list to the AbT-tool
Alternatives:	2. If a package is already checked for its dependencies do not check it again.
Exceptions:	None
Postconditions:	- The AbT-tool will have received a list containing the packages without dependencies.
Related business rules:	Business Rule 1, 2 and 3

9 Scenarios

9.1 Use Case 1: Provide Depends Tree Up

Use Case Name:	**Provide Depends Tree Up**
Scenario Number:	**1**
Use Case Steps:	1. depEngine receives FOO4 ver 1 to be investigated 2. depEngine reads the description of the package 3. FOO4 ver 1 DO FOO5 ver 9 and FOO4 ver 1 DO FOO6 ver 3 4. depEngine reads the descriptions of FOO5 ver9 and FOO6 ver 3 and finds that they have no more dependencies 5. depEngine creates this tree: FOO4 ver 1 * * ** DO ** FOO6 ver 3 * ** DO ** FOO5 ver 9 6. depEngine writes to the log 7. depEngine returns the tree to AbT-tool
Alternative Path:	None

Use Case Name:	**Provide Depends Tree Up**
Scenario Number:	**2**
Use Case Steps:	1. depEngine receives FOO8 ver 2 to be investigated 2. depEngine reads the description of the package 3. FOO8 ver 2 oRO FOO9 ver 3 4. depEngine reads the description of FOO9 ver 3 and finds that it oRO FOO8 ver 2 FOO8 ver 2 has already been checked for its dependencies and will not be checked again 5. depEngine creates this tree: FOO8 ver 2 * * ** oRO ** FOO9 ver 3 6. depEngine writes to the log 7. depEngine returns the tree to AbT-tool
Alternative Path:	None

9.2 Use Case 2: Provide Depends Tree Down

Use Case Name:	**Provide Depends Tree Down**
Scenario Number:	**1**
Use Case Steps:	1. depEngine receives FOO6 ver 3 to be investigated 2. depEngine reads the description of the package 3. FOO6 ver 3 DO FOO4 ver 1 4. depEngine reads the description of FOO4 ver 1 and finds that it RO FOO1 ver 6. depEngine reads the description of FOO1 ver 6 and finds that it has no more dependencies 5. depEngine creates this tree: FOO6 ver 3 * * ** DO ** FOO4 ver 1 ....* ....* ** RO ** FOO1 ver 6 6. depEngine writes to the log 7. depEngine returns the tree to AbT-tool
Alternative Path:	None

Use Case Name:	Provide Depends Tree Down
Scenario Number:	**2**
Use Case Steps:	1. depEngine receives FOO2 ver 2 to be investigated 2. depEngine reads the description of the package 3. FOO2 ver 2 oRO FOO1 ver 6 4. depEngine reads the description of FOO1 ver 6 and finds that it oRO FOO2 ver 2. FOO2 ver 2 has already been checked for its dependencies and will not be checked again 5. depEngine creates this tree: FOO2 ver 2 * * ** oRO ** FOO1 ver 6 6. depEngine writes to the log 7. depEngine returns the tree to AbT-tool
Alternative Path:	None

9.3 Use Case 3: Check Version Compatibility

Use Case Name:	**Check Version Compatibility**
Scenario Number:	**1**
Use Case Steps:	1. depEngine receives this tree: FOO4 ver 1 * * ** DO ** FOO6 ver 3 * ** DO ** FOO5 ver 9 2. depEngine reads package description and finds: FOO4 ver 1 DO FOO6 ver 3 FOO4 ver 1 DO FOO5 ver 9 depEngine compares them to the actual version numbers. Description - Actual Version Number FOO4 ver 1 FOO4 ver 1 FOO6 ver 3 - FOO6 ver 3 FOO5 ver 9 - FOO5 ver 9 3. depEngine writes to the log 4. depEngine returns the message that the comparison succeeded
Alternative Path:	None

Use Case Name:	Check Version Compatibility
Scenario Number:	**2**
Use Case Steps:	1. depEngine receives this tree: FOO6 ver 3 * * ** DO ** FOO4 ver 9 ....* ....* ** RO ** FOO7 ver 3 Note: This example contains the wrong dependency package (two packages with different versions are not the same) and this package contains different dependencies then the correct package. 2. depEngine reads package description and finds: FOO6 ver 3 DO FOO4 ver 1 FOO4 ver 9 RO FOO7 ver 3 depEngine compares them to the actual version numbers. Description - Actual Version Number FOO6 ver 3 - FOO6 ver 3 FOO4 ver 1 - FOO4 ver 9 FOO7 ver 3 - FOO7 ver 3 3. depEngine writes to the log 4. depEngine returns the message that the comparison failed
Alternative Path:	None

9.4 Use Case 4: Provide the Number of Dependencies of a Package

Use Case Name:	**Provide the Number of Dependencies of a Package**
Scenario Number:	**1**
Use Case Steps:	1. depEngine receives the dependency tree for the FOO8 ver 2 package from the AbT-tool 2. depEngine counts that there are 2 packages in the tree 3. depEngine writes to the log 4. depEngine returns that number to the AbT-tool
Alternative Path:	None

Use Case Name:	**Provide the Number of Dependencies of a Package**
Scenario Number:	**2**
Use Case Steps:	1. depEngine receives the dependency tree for the FOO1 ver 6 package from the AbT-tool 2. depEngine counts that there are 6 packages in the tree 3. depEngine writes to the log 4. depEngine returns that number to the AbT-tool
Alternative Path:	None

9.5 Use Case 5: Provide the Number of Dependants on a Package

Use Case Name:	**Provide the Number of Dependants on a Package**
Scenario Number:	**1**
Use Case Steps:	1. The depEngine receives the dependants tree for the FOO10 ver 3 package from the AbT-tool 2. The depEngine counts that there are 6 packages in the tree 3. depEngine writes to the log 4. The depEngine returns that number to the AbT-tool
Alternative Path:	None

Use Case Name:	Provide the Number of Dependants on a Package
Scenario Number:	**2**
Use Case Steps:	1. The depEngine receives the dependency tree for the FOO6 ver 3 package from the AbT-tool 2. The depEngine counts that there are 3 packages in the tree 3. depEngine writes to the log 4. The depEngine returns that number to the AbT-tool
Alternative Path:	None

9.6 Use Case 6: Provide Proper Build Order

Use Case Name:	Request to Downgrade Package
Scenario Number:	**1**
Use Case Steps:	1. depEngine receives up tree of FOO7 ver 1 from the AbT-tool 2. depEngine sorts packages in up tree descending on depth 3. depEngine adds packages to list The list is now: ”FOO9 ver 3, FOO5 ver 9, FOO6 ver 3, FOO8 ver 2, FOO10 ver 3” 4. depEngine adds FOO7 ver 3 to the building list The list is now: ”FOO9 ver 3, FOO5 ver 9, FOO6 ver 3, FOO8 ver 2, FOO10 ver 3, FOO7 ver 1” 5. depEngine receives down tree of FOO7 ver 1 6. depEngine sorts packages in down tree ascending on depth 7. depEngine adds packages to list The list is now: ”FOO9 ver 3, FOO5 ver 9, FOO6 ver 3, FOO8 ver 2, FOO10 ver 3, FOO7 ver 1, FOO11 ver 2, FOO12 ver 1, FOO13 ver 2, FOO14 ver 4” 8. depEngine logs sort action 9. depEngine returns the build order list to the AbT-tool
Alternative Path:	None

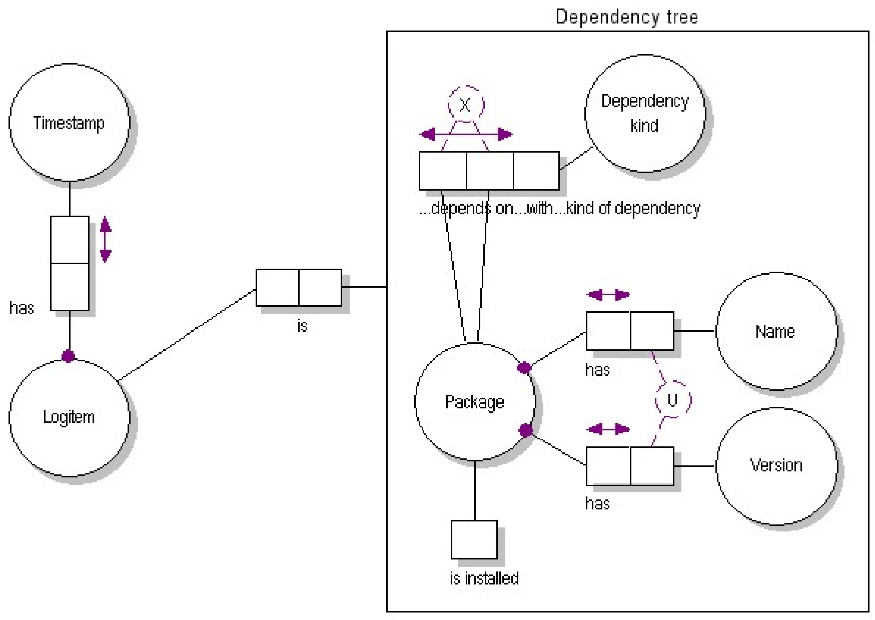
9.7 Use Case 7: Provide the List of Packages Without Dependencies

Use Case Name:	**Provide the List of Packages Without Dependencies**
Scenario Number:	**1**
Use Case Steps:	1. depEngine receives tree of FOO7 ver 1 from the AbT-tool 2. depEngine finds that FOO5 ver 9 and FOO6 ver 3 do not have any dependencies 3. depEngine adds FOO5 ver 9 and FOO6 ver 3 in a list 4. depEngine writes a log 5. depEngine returns the list to the AbT-tool
Alternative Path:	None

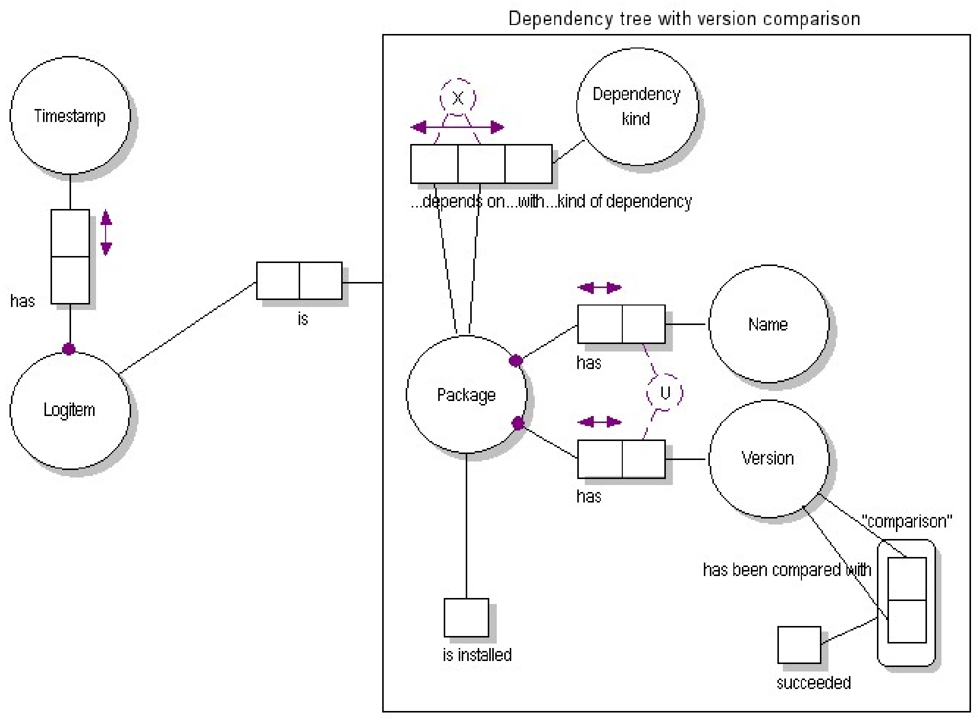
Use Case Name:	**Provide the List of Packages Without Dependencies**
Scenario Number:	**2**
Use Case Steps:	1. depEngine receives tree of FOO1 ver 6 from the AbT-tool 2. depEngine finds that FOO3 ver 3, FOO5 ver 9 and FOO6 ver 3 do not have any dependencies 3. depEngine adds FOO3 ver 3, FOO5 ver 9 and FOO6 ver 3 in a list 4. depEngine writes a log 5. depEngine returns the list to the AbT-tool
Alternative Path:	None

10 Domain Models

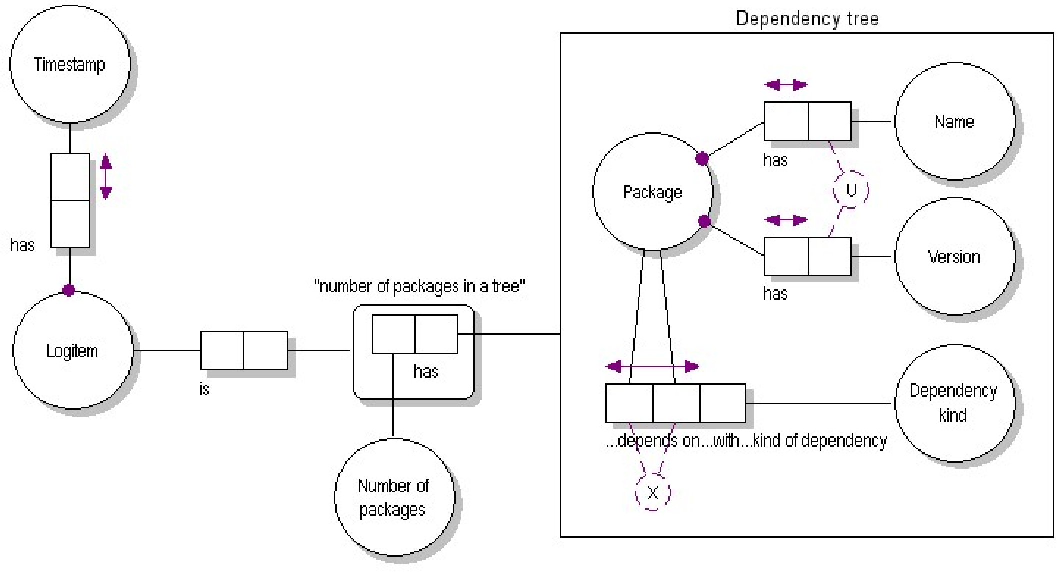
10.1 ORM Diagram Use Case 1 and 2



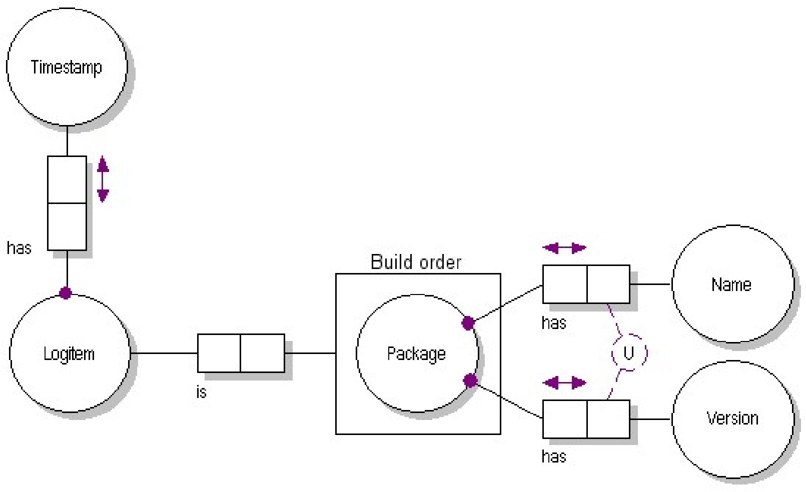
10.2 ORM Diagram Use Case 3



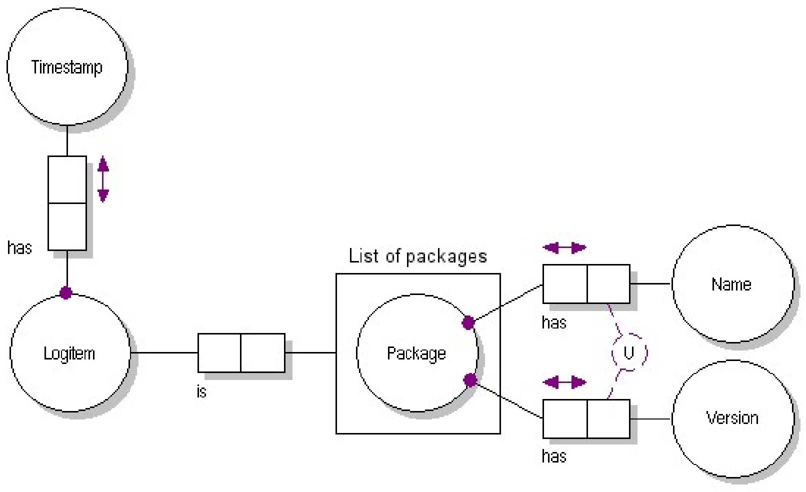
10.3 ORM Diagram Use Case 4 and 5



10.4 ORM Diagram Use Case 6



10.5 ORM Diagram Use Case 7



11 Business Rules

No.	Rule Definition	Type of Rule	Static or Dynamic	Source
1	All the requests made by the AbT-tool and every action that's done by the depEngine have to be logged	Structural facts	Static	Eric Schabell's case briefing and feedback from Arnoud Vermeij and Stijn Hoppenbrouwers
2	All packages have a name and version number	Structural facts	Static	Eric Schabell's case briefing and feedback from Arnoud Vermeij and Stijn Hoppenbrouwers
3	All packages have a proper description	Structural facts	Static	Feedback from Eric Schabell

12 Non-functional Requirements

Definition
No. = Number
NFC = Nonfunctional Category
NFR = Nonfunctional Requirement
Mat = Maturity
Com = Comments
OI = Outstanding Issues

No.	NFC	NFR Name	Description	Completeness	Mat	Source	Com	OI
1	Security	Auditability	The depEngine system has to write a log	Complete	Mature	Abtlinux.org	None	None
2	Compatible	Integratability	The depEngine system is a subsystem of the AbT-tool	Complete	Mature	Abtlinux.org	None	None
3	Compatible	Interoperability	It should be easy for the AbT-tool to ”talk” to the depEngine system	Complete	Mature	Abtlinux.org	None	None
4	Performance	Maintability	It should be easy to maintain and even upgrade/update the system	Complete	Mature	Abtlinux.org	None	None
5	Performance	Performance	The depEngine has to have a high level of performance, otherwise its possible the system will fail to function properly	Complete	Mature	Abtlinux.org	None	None
6	Performance	Reliability	The AbT-tool relies on the depEngine	Complete	Mature	Abtlinux.org	None	None

13 Terminological Definitions

Term	**Definition**
Package	A complete set of data that can be used to manage all aspects of using this single piece of software.
Package Manager	A software application that manages all aspects of handling software such as installation, upgrading and uninstalling packages. Different versions of a package can exist.
Linux Distribution	A combination of the Linux kernel with a specified collection of packages.
AbTLinux	A certain Linux distribution, (to be) created by Eric & co.
AbT-tool	The package manager on AbTLinux.
Dependencies	Links to packages that rely or depend on the current package.
Dependency Tree	States the order of linked packages.
(computer) system	A computer with AbTLinux installed on it.
depEngine	Abbreviation for "dependency engine". The depEngine is the underlying "motor" of the AbT-tool. It will check on packages' dependencies when a change in the system's configuration is desired.
Install	Copy built binaries to specified locations.
(re)build	(Re)compile and link source.
(re)configure	Change applications configuration.
up-/downgrade	Installing a newer or older version of a package.
Build Order	States the order in which the linked packages must be build.
source(code)	Program code of a package.
LaTeX	Word processor especially suited for writing scientific documents.
Proper description	A list with all the dependencies of a package denoted.
Depends on (DO)	If one package depends on another package, this one package will not be able to function (correctly) if the other package is not installed on the system. Either if the other package gets rebuild or reconfigured, the one package has to be rebuild.

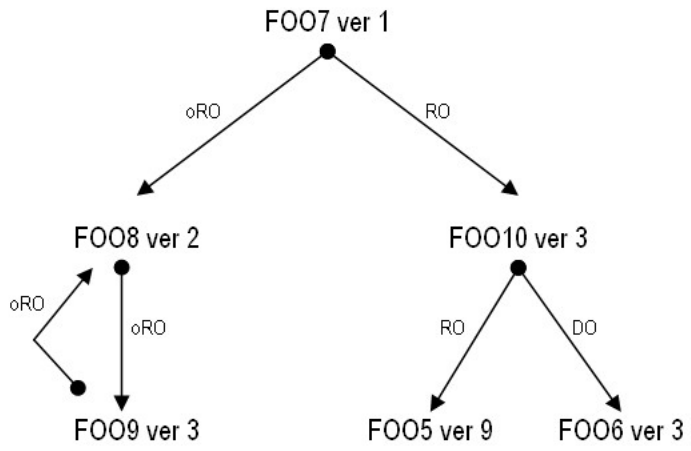
Term	Definition
Relies on (RO)	If one package relies on another package, this one package also will not be able to function (correctly) if the other package is not installed on the system. But it only has to be rebuild when the other package gets reconfigured.
Optional dependencies (oDO or oRO)	If one package optionally depends or relies on another package, this one package will miss some functionality if the other package is not installed but it will still function normally.

Specification of the maturity and completeness degrees

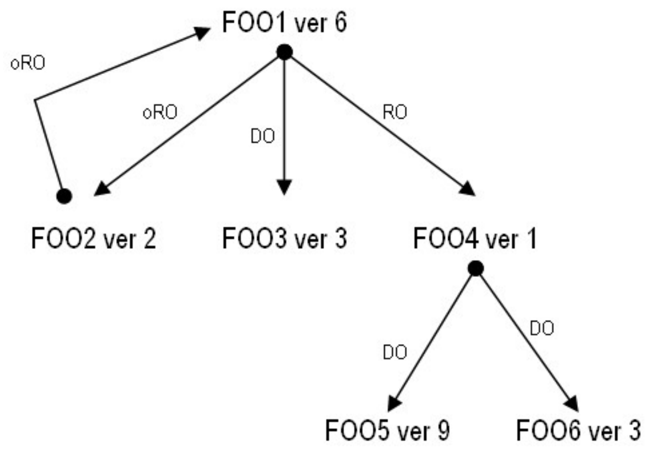
Uncomplete	Immature	0% - 20%
Nearly Uncomplete	Nearly Immature	20% - 40%
Halfway Complete	Halfway Mature	40% - 60%
Nearly Complete	Nearly Mature	60% - 80%
Complete	Mature	80% - 100%

14 Example Trees

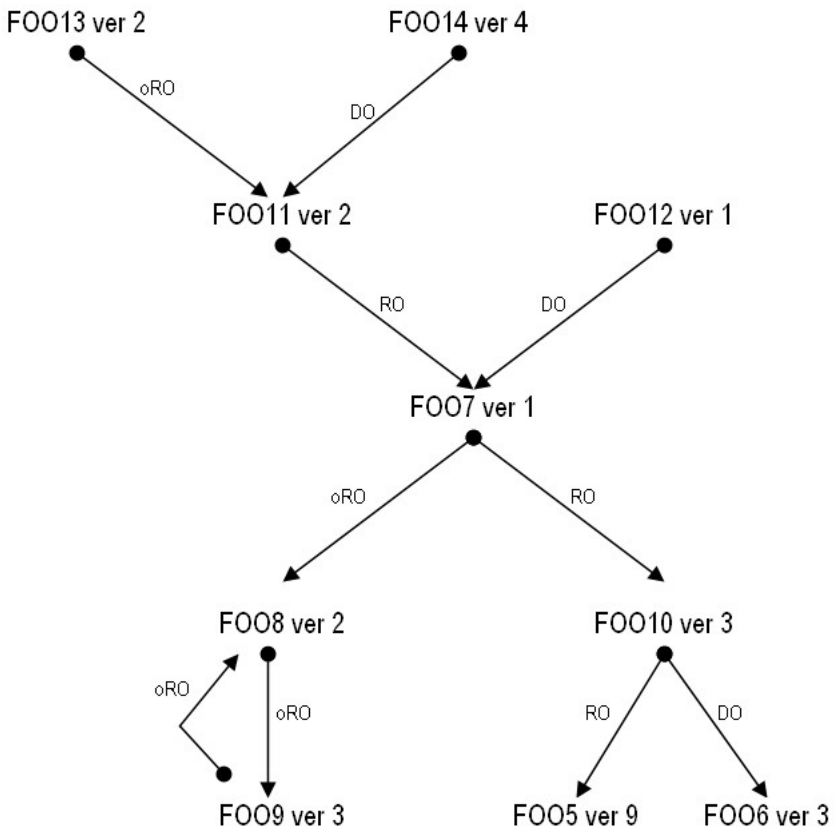
14.1 Tree 1



14.2 Tree 2



14.3 Tree 3


FOO13 ver 2
FOO14 ver 4
oRO
DO
FOO11 ver 2
FOO12 ver 1
RO
DO
FOO7 ver 1
oRO
RO
FOO8 ver 2
FOO10 ver 3
oRO
oRO
RO
DO
FOO9 ver 3
FOO5 ver 9
FOO6 ver 3