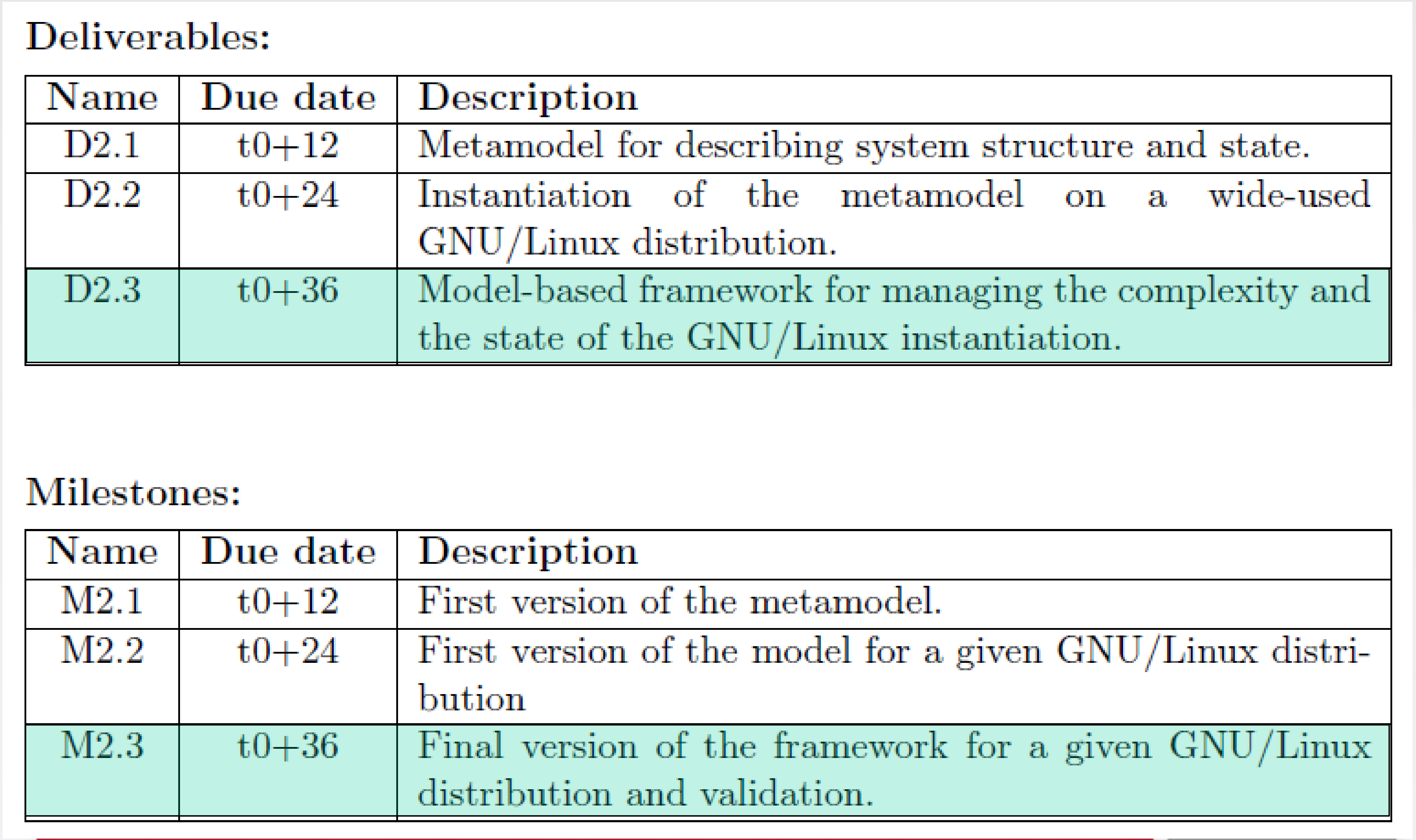
WP2: Upgrade simulator and Failure detector

Davide Di Ruscio
University of L'Aquila (UDA)
davide.diruscio@univaq.it

Outline

- What is due at T0+40
- Upgrade simulator
- Failure detector
- WP2/WP3 integration points
- WP2/WP4 integration points
- Discussion
- Next steps

What is due at T0+40



Deliverables:

Name	Due date	Description
D2.1	t0+12	Metamodel for describing system structure and state.
D2.2	t0+24	Instantiation of the metamodel on a wide-used GNU/Linux distribution.
D2.3	t0+36	Model-based framework for managing the complexity and the state of the GNU/Linux instantiation.

Milestones:

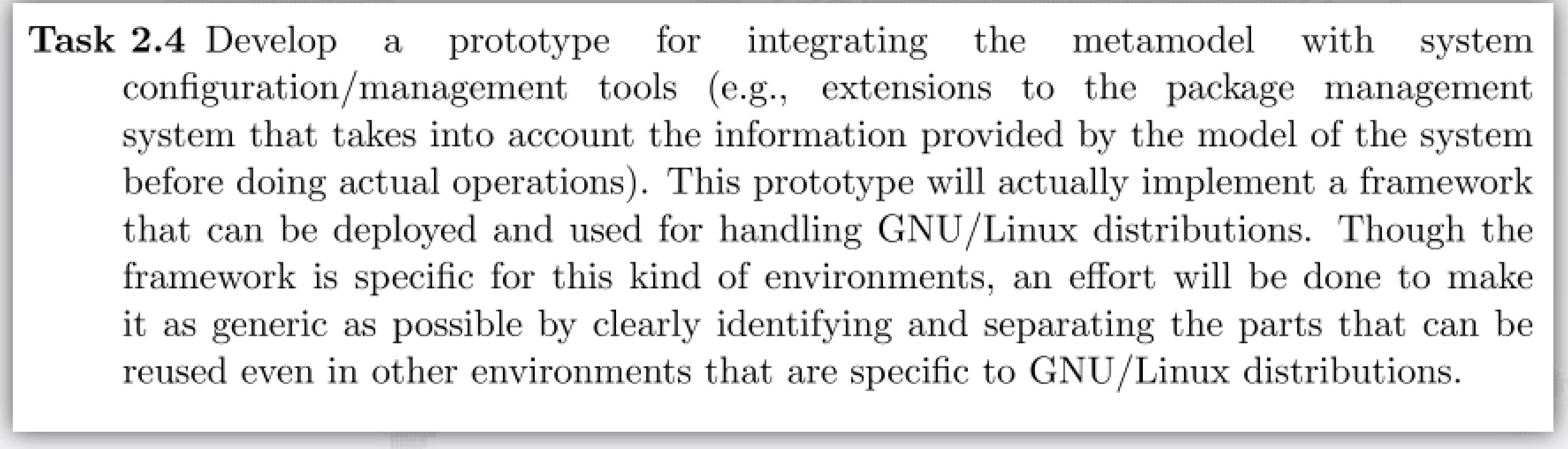
Name	Due date	Description
M2.1	t0+12	First version of the metamodel.
M2.2	t0+24	First version of the model for a given GNU/Linux distribution
M2.3	t0+36	Final version of the framework for a given GNU/Linux distribution and validation.

Deliverable D2.3

Title of D2.3: Model-based framework for managing the complexity and state of the GUN/Linux instantiation

Due at: T0+40

From the DoW:



Task 2.4 Develop a prototype for integrating the metamodel with system configuration/management tools (e.g., extensions to the package management system that takes into account the information provided by the model of the system before doing actual operations). This prototype will actually implement a framework that can be deployed and used for handling GNU/Linux distributions. Though the framework is specific for this kind of environments, an effort will be done to make it as generic as possible by clearly identifying and separating the parts that can be reused even in other environments that are specific to GNU/Linux distributions.

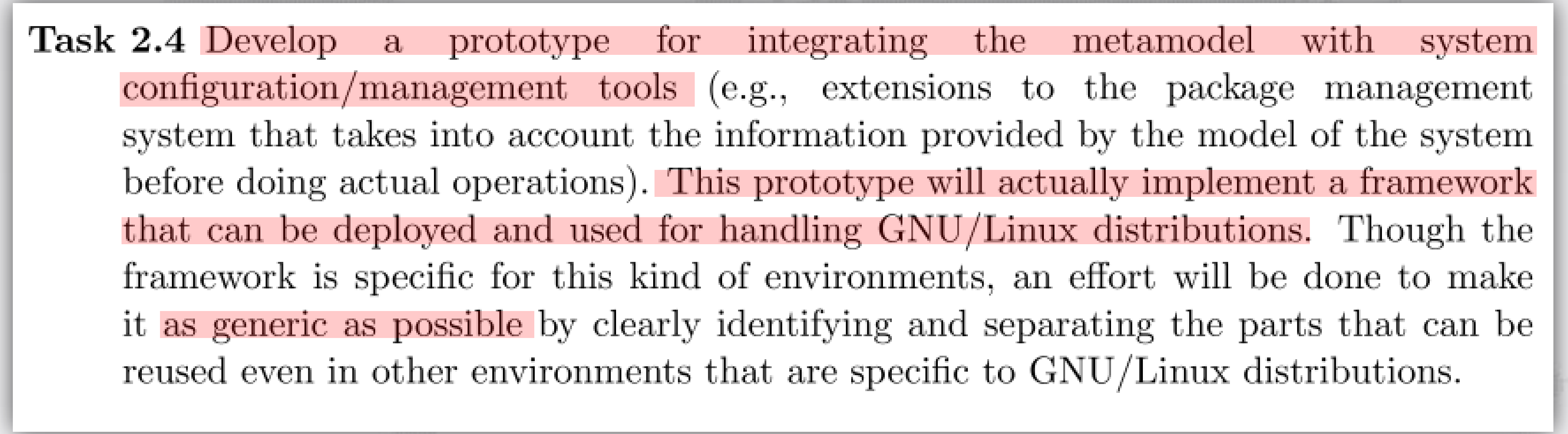
mancoosi

Deliverable D2.3

Title of D2.3: Model-based framework for managing the complexity and state of the GUN/Linux instantiation

Due at: T0+36

From the DoW:



Task 2.4 Develop a prototype for integrating the metamodel with system configuration/management tools (e.g., extensions to the package management system that takes into account the information provided by the model of the system before doing actual operations). This prototype will actually implement a framework that can be deployed and used for handling GNU/Linux distributions. Though the framework is specific for this kind of environments, an effort will be done to make it as generic as possible by clearly identifying and separating the parts that can be reused even in other environments that are specific to GNU/Linux distributions.

Upgrade failures

- Current tools are able to predict a very limited set of upgrade failures before deployment

- When trying to predict upgrade failures, existing tools only consider static package metadata and the behaviour of the maintainer scripts is completely ignored
  - This leaves a wide range of failures unpredicted

Upgrade failures classification

» ***Static deploy-time***, occur when a static requirement is violated during the upgrade.

- The low-level package manager fails at deploy-time, aborting the upgrade process

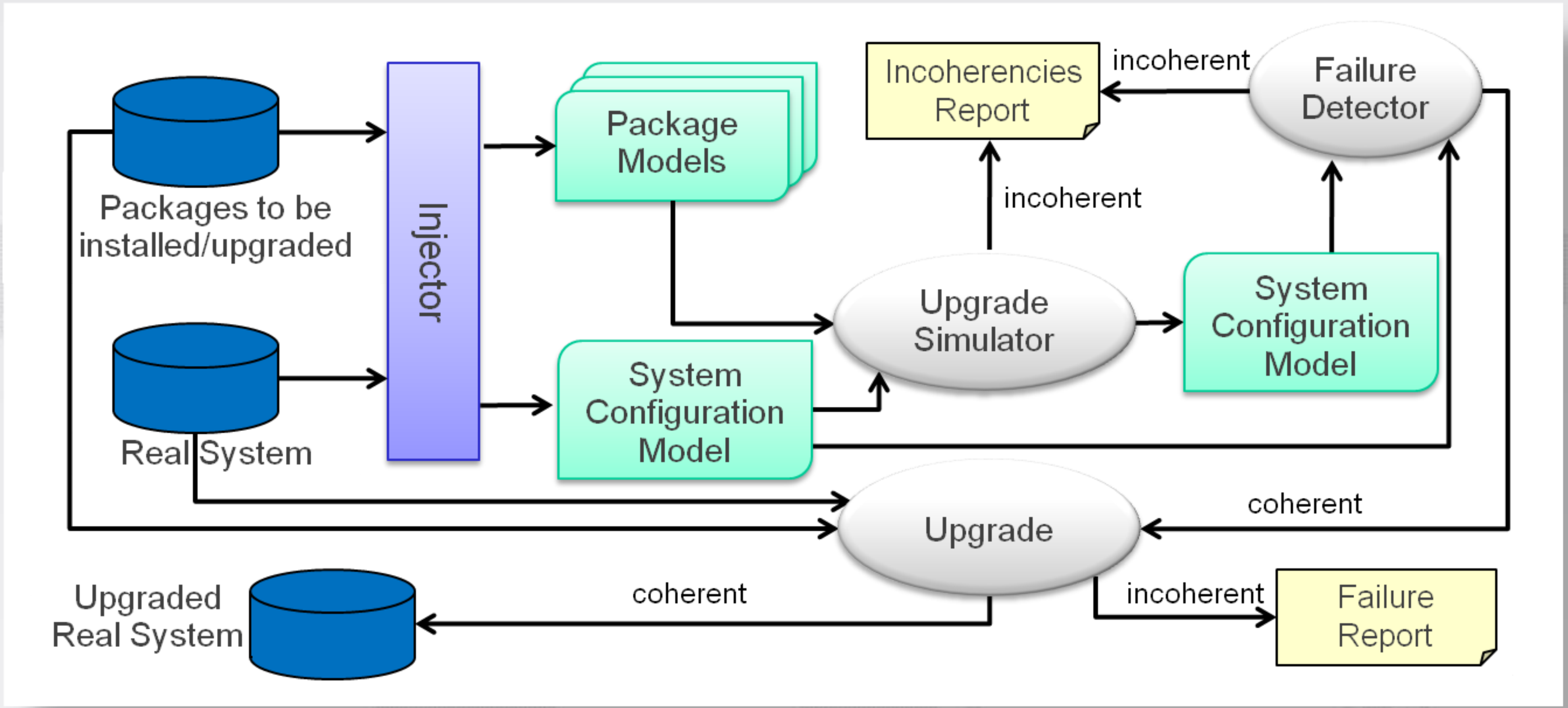
» ***Dynamic deploy-time,*** occur when a maintainer script fails

- They are tricky to deal with, given that shell script failures can originate from a wide range of errors, ranging from syntax errors to failures in the invocation of external tools
- They are not addressed by state of the art package managers

» ***Undetected failures,*** remain undetected through upgrade deployment

- According to all involved tools, the upgrade has been completed successfully, but the obtained system configuration contains incoherences

Using models to enhance package upgrades

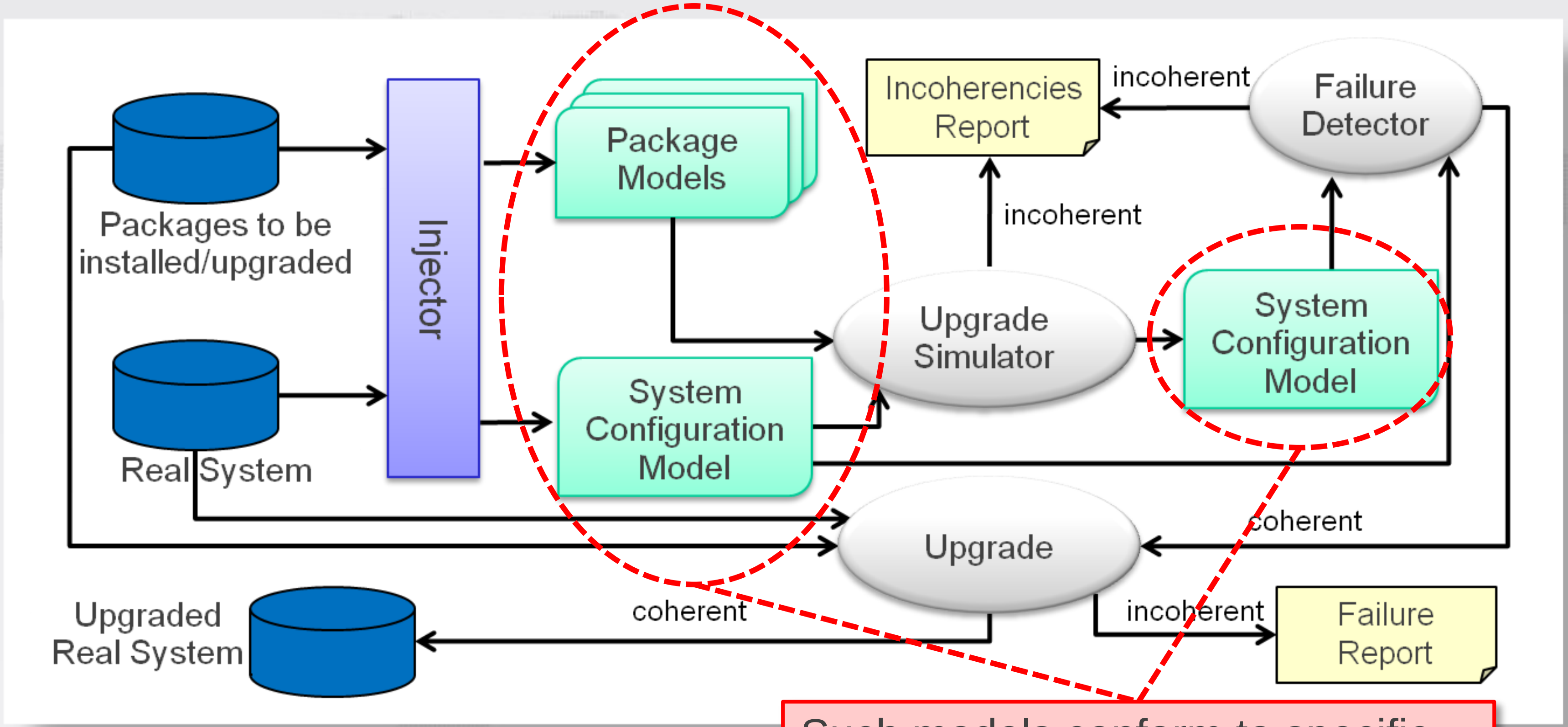


» A model-based approach is introduced to support the package upgrades and enhance the failure detection possibilities:

- A simulator is used to predice the effect of maintainer script executions (*deploy-time failures*)
- A failure detector is used to deal with *undetected failures*

mancoosi

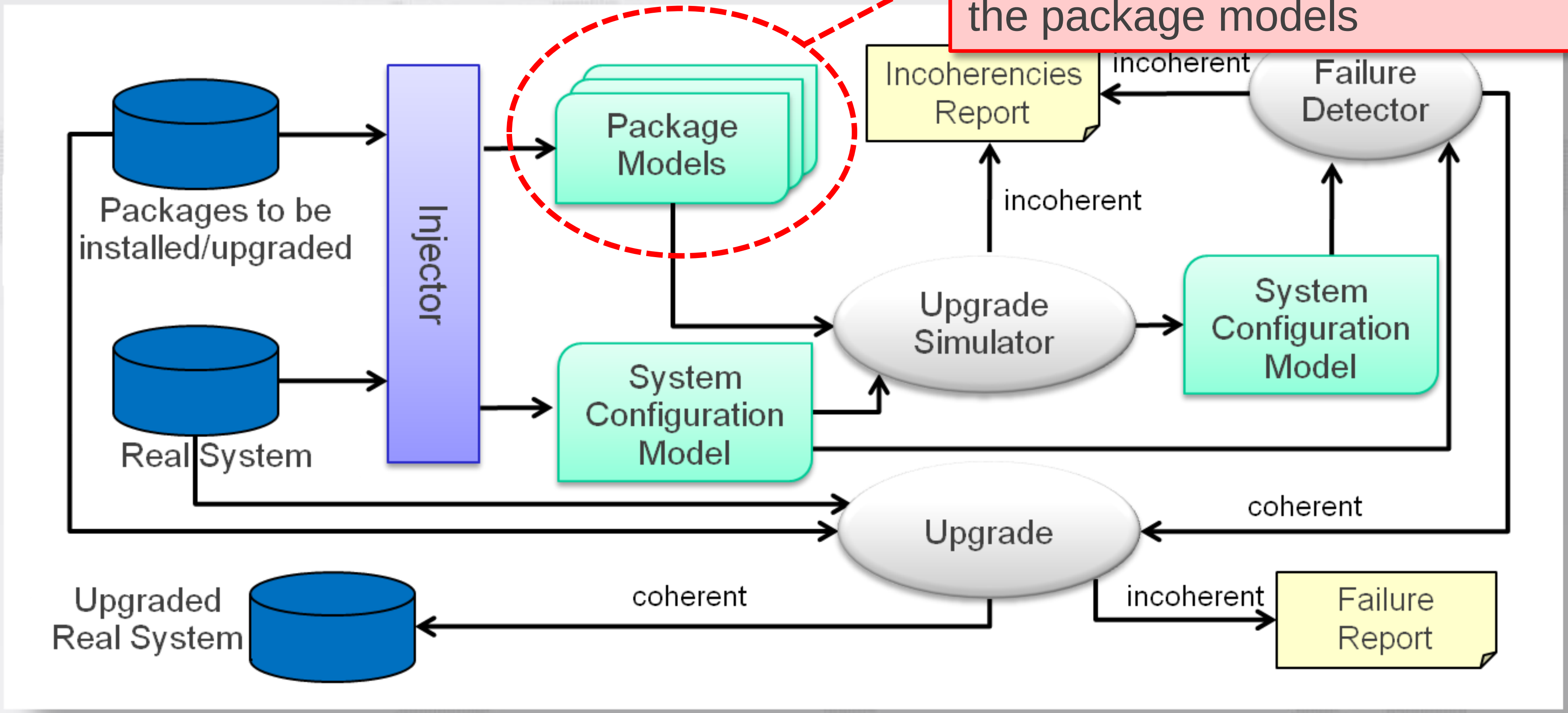
Using models to enhance package upgrades



Such models conform to specific metamodels which have been specified in the D2.1 and refined during D3.2 writing

Using models to enhance package upgrades

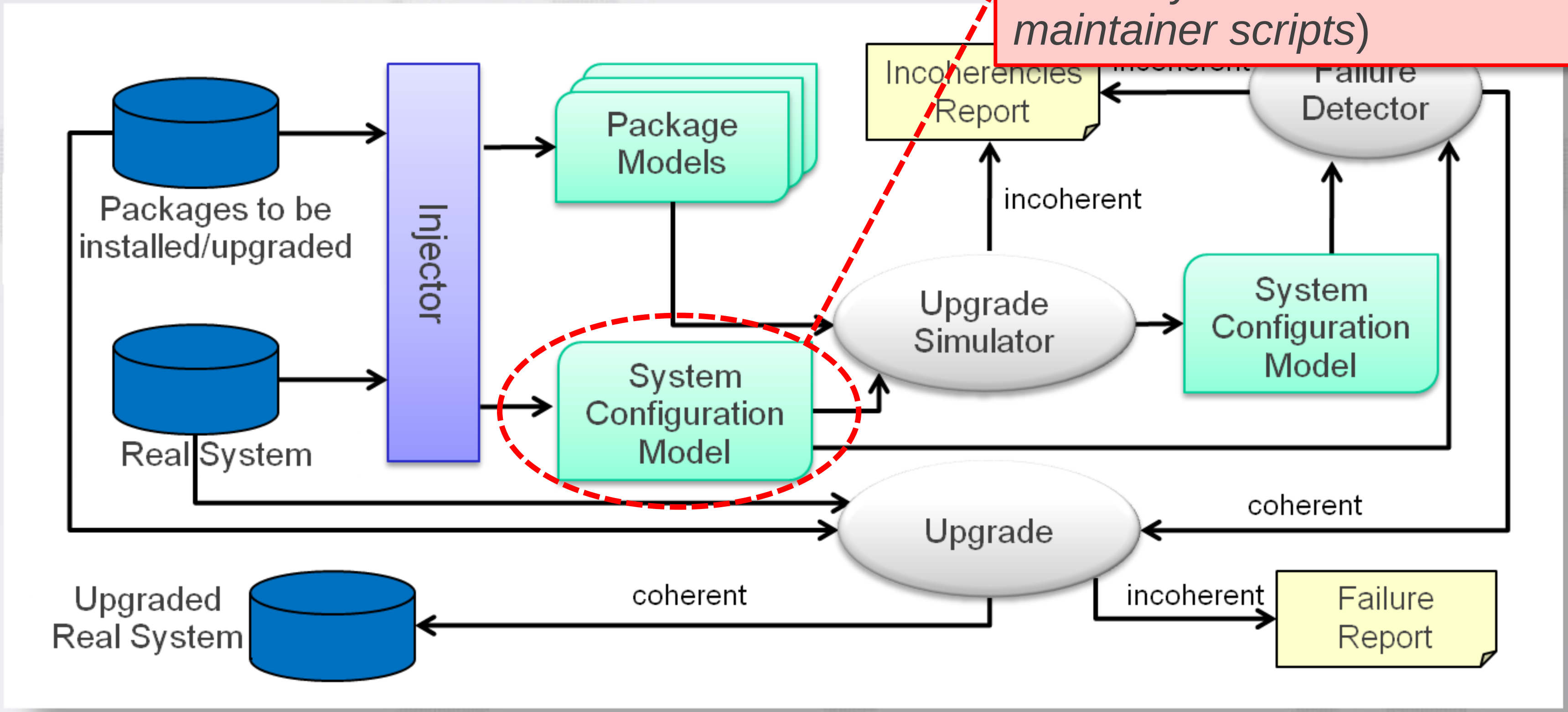
Maintainer scripts defined with the DSL conceived in WP3 are part of the package models



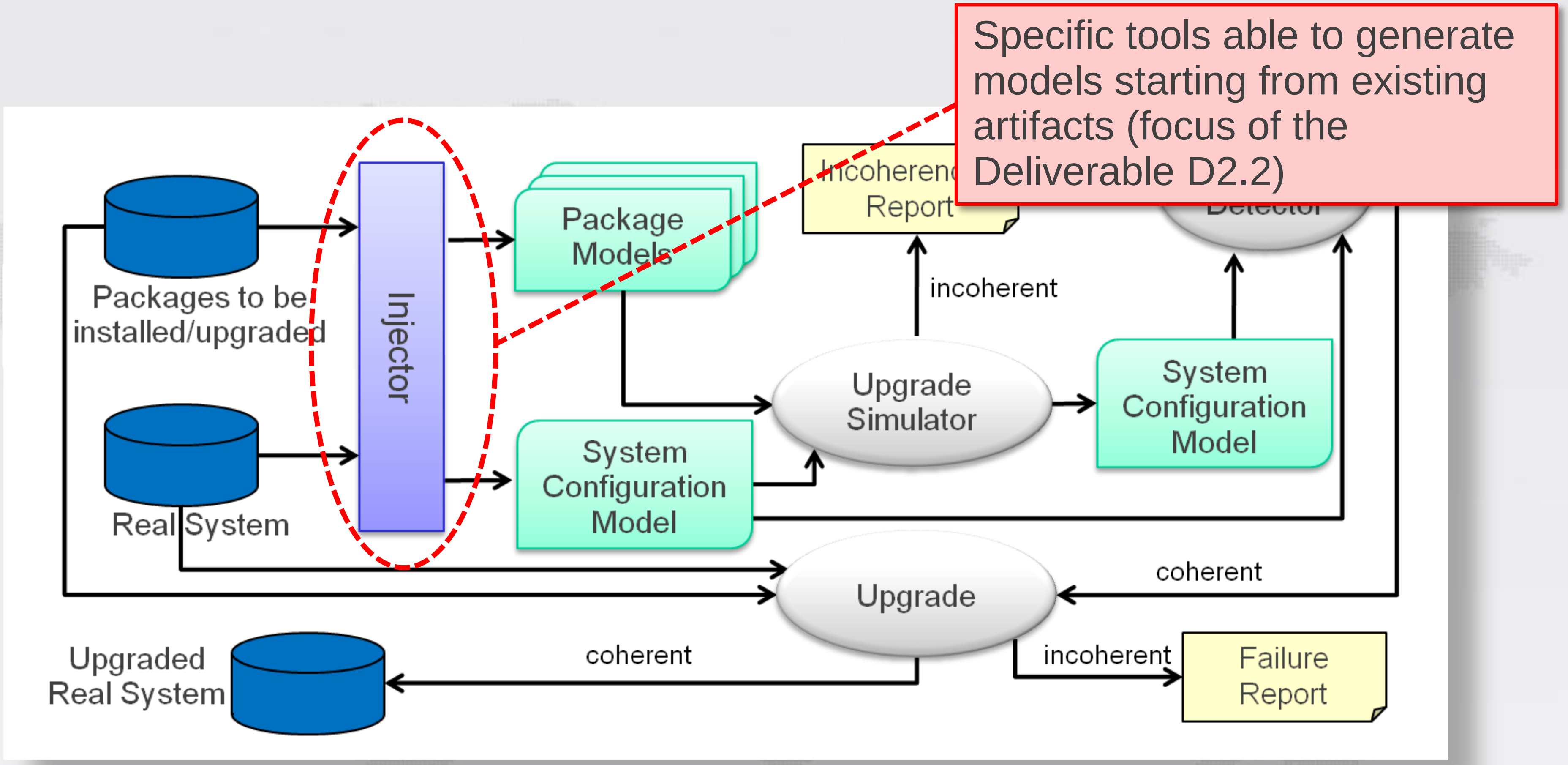
mancoosi

Using models to enhance package upgrades

System configuration model which is affected by the upgrade (and hence by the execution of the *maintainer scripts*)

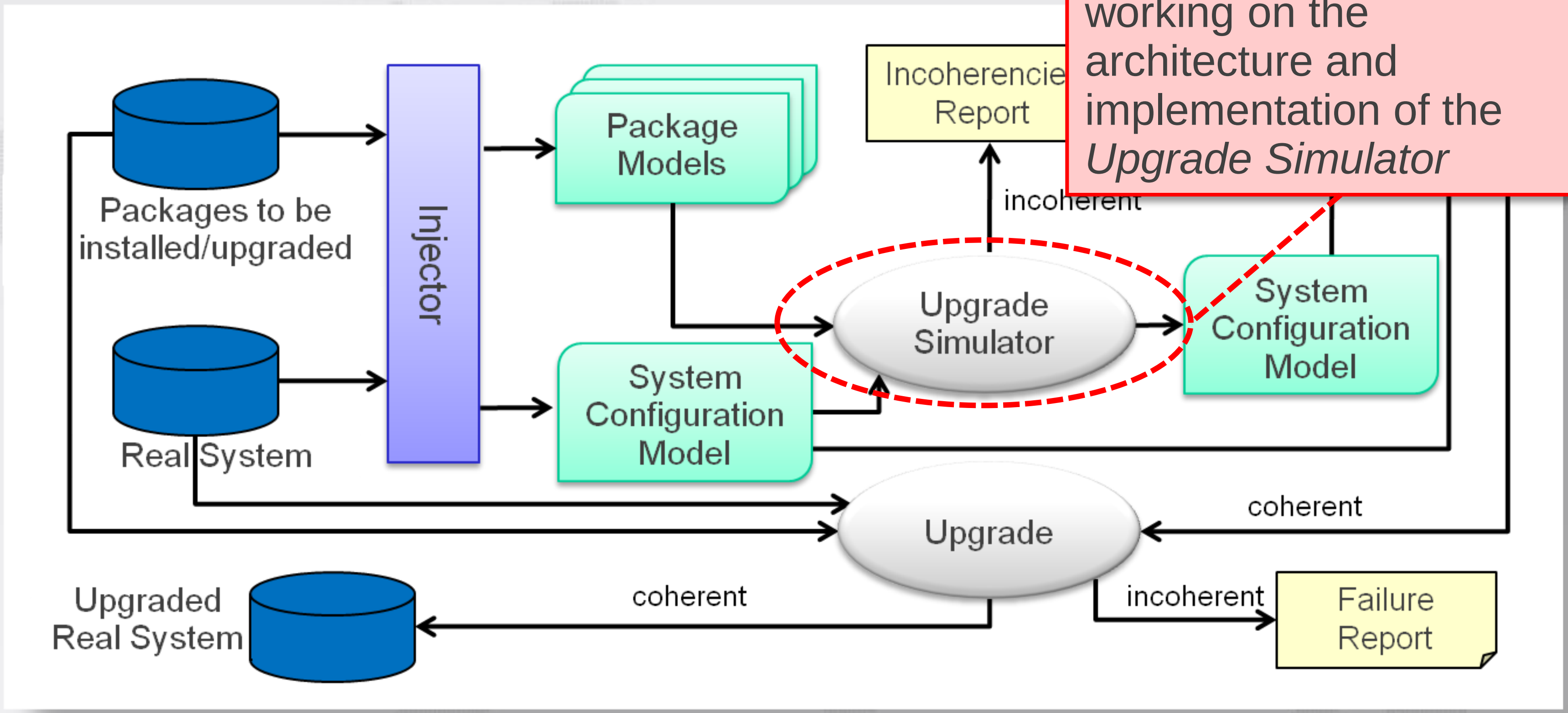


Using models to enhance package upgrades



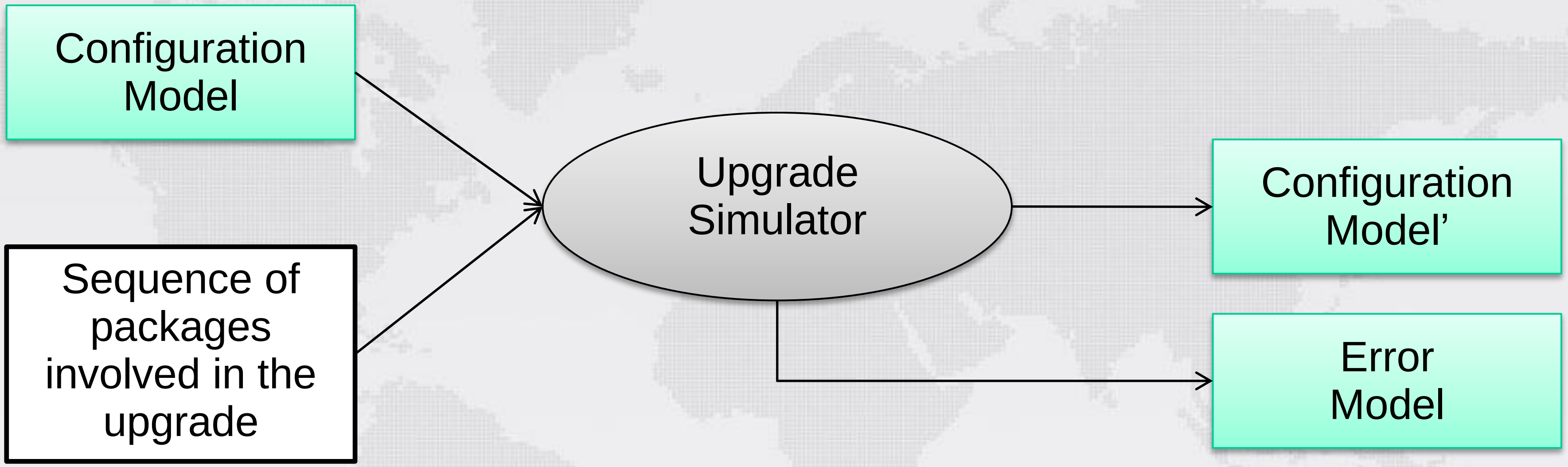
Specific tools able to generate models starting from existing artifacts (focus of the Deliverable D2.2)

Using models to enhance package upgrades

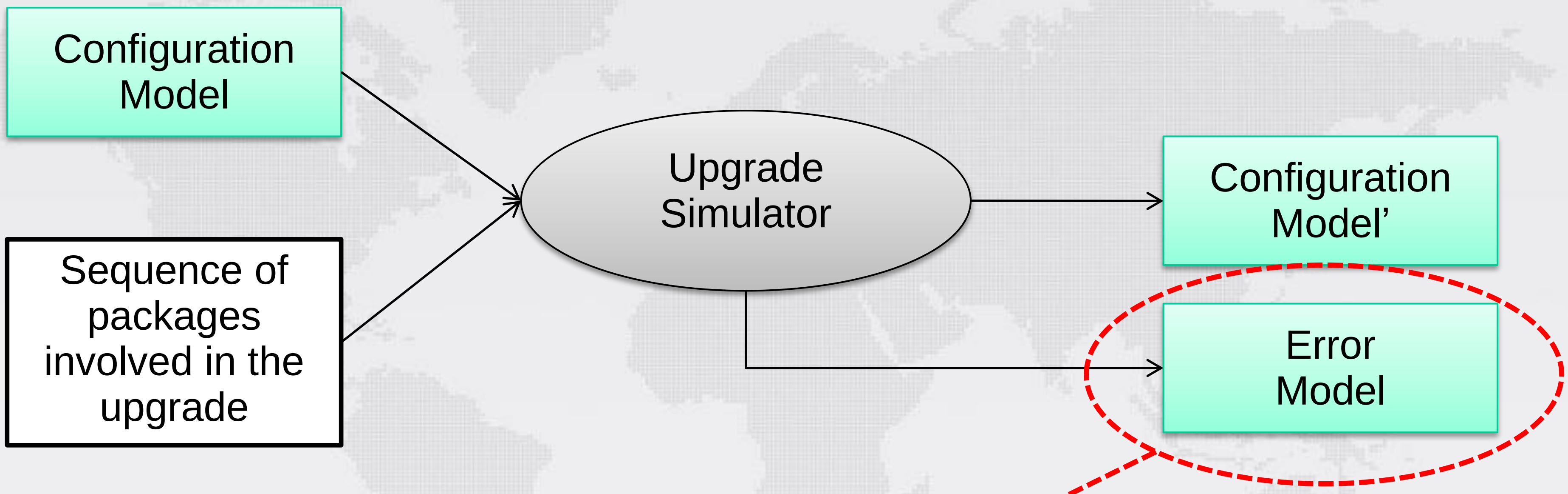


Currently WP2 is mainly working on the architecture and implementation of the *Upgrade Simulator*

Upgrade Simulator



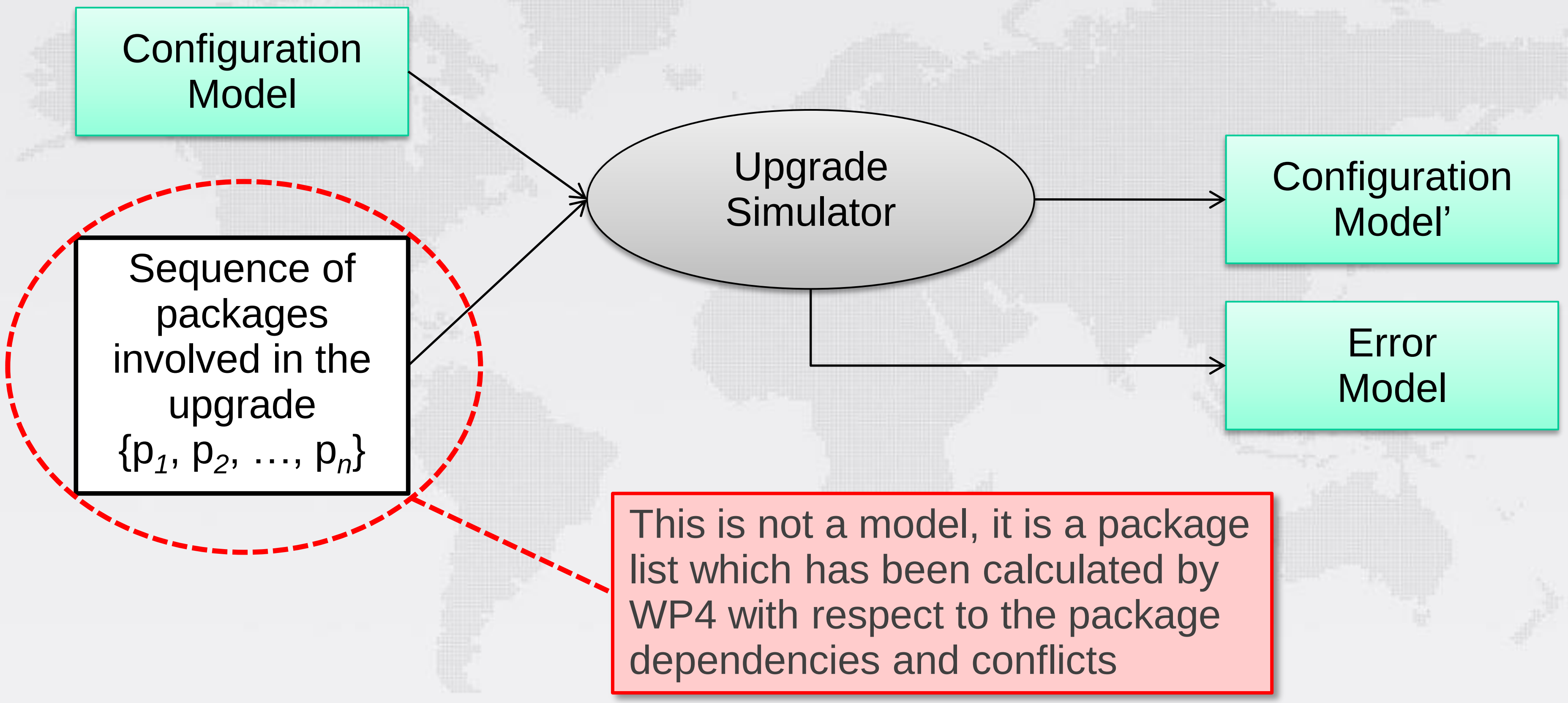
Upgrade Simulator



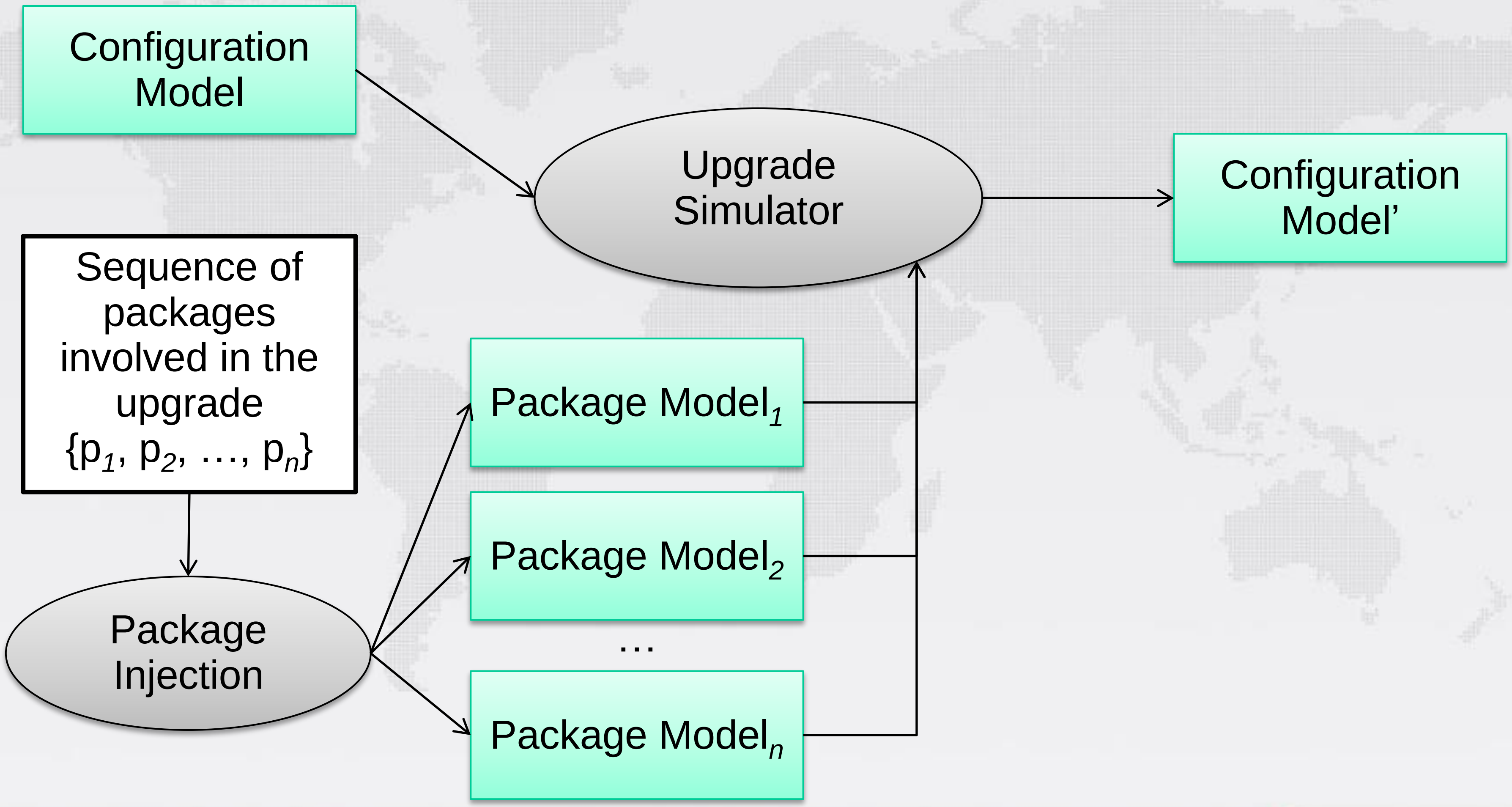
This model stores errors and warnings that can occur during the simulation

mancoosi

Upgrade Simulator

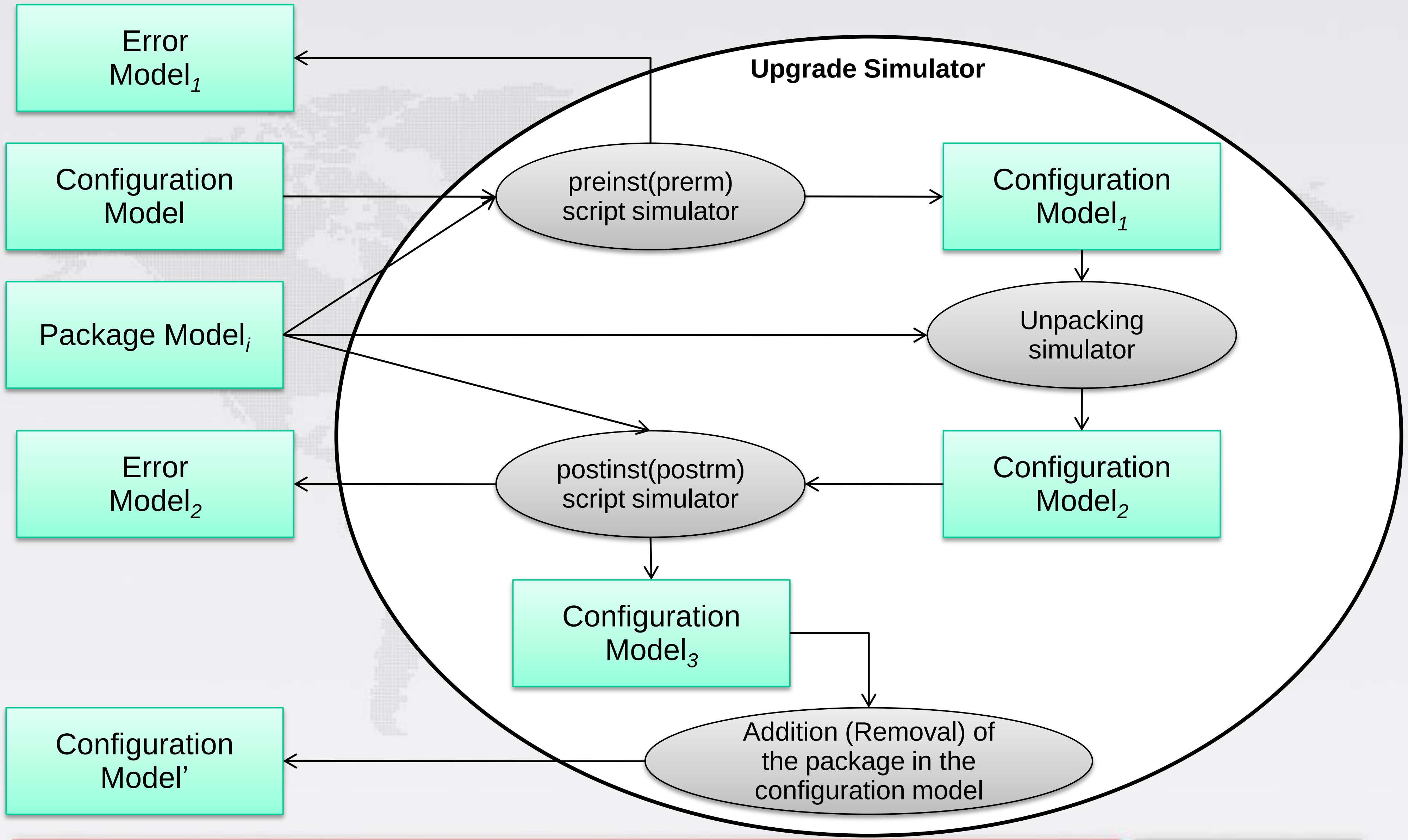

Configuration Model
Sequence of packages involved in the upgrade $\{p_1, p_2, \ldots, p_n\}$
Upgrade Simulator
Configuration Model'
Error Model
This is not a model, it is a package list which has been calculated by WP4 with respect to the package dependencies and conflicts

Upgrade Simulator

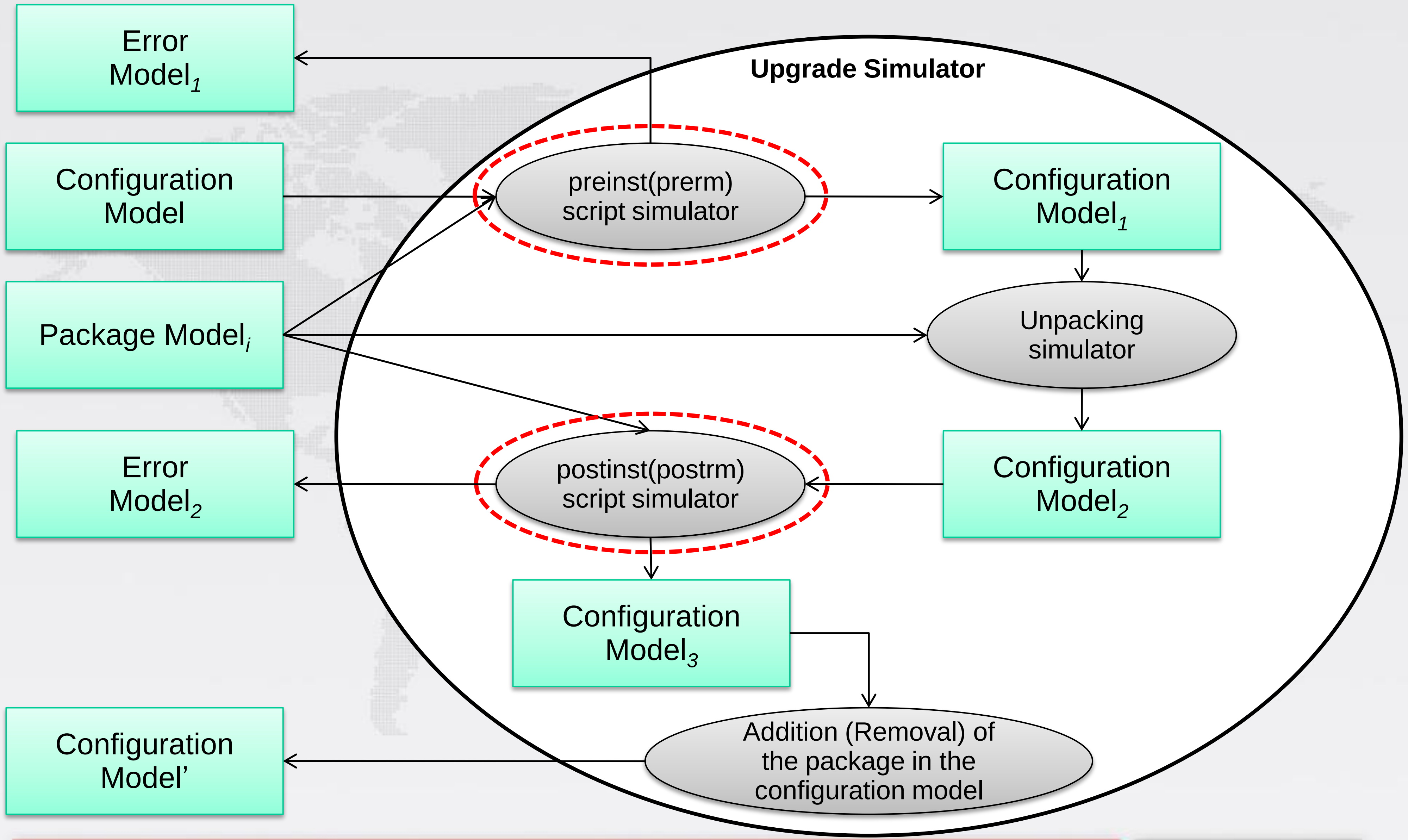


mancoosi

Upgrade Simulator



Upgrade Simulator

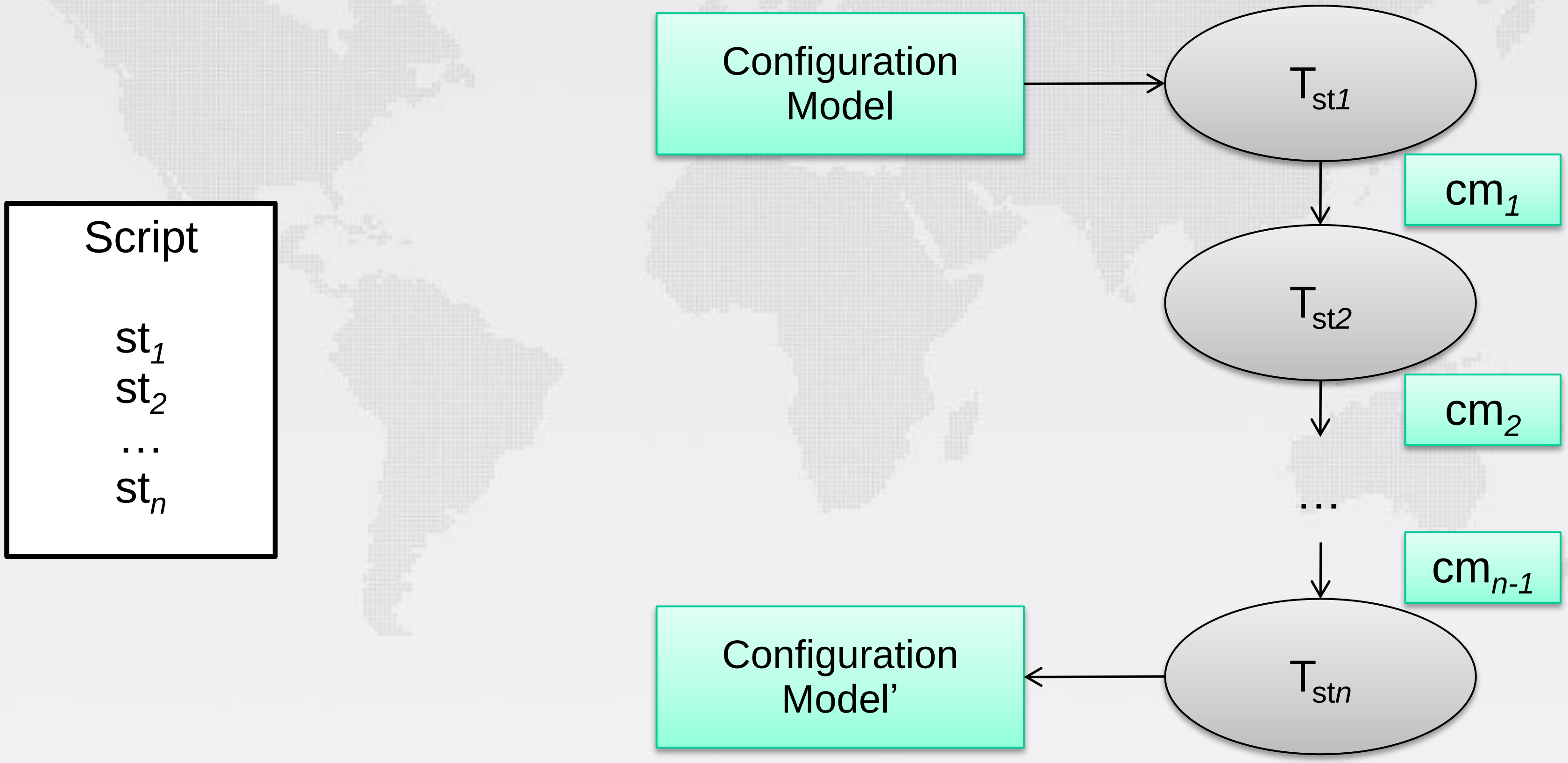


Upgrade Simulator: Script simulator

- The scripts are specified by means of the Mancoosi DSL
- For each DSL statement (*st*) a corresponding model transformation (T_{st}) is defined in order to specify how the execution of st affects a source configuration model (*cm*)
- Such model transformations have been specified in the deliverable D3.2 to provide the semantics of the Mancoosi DSL

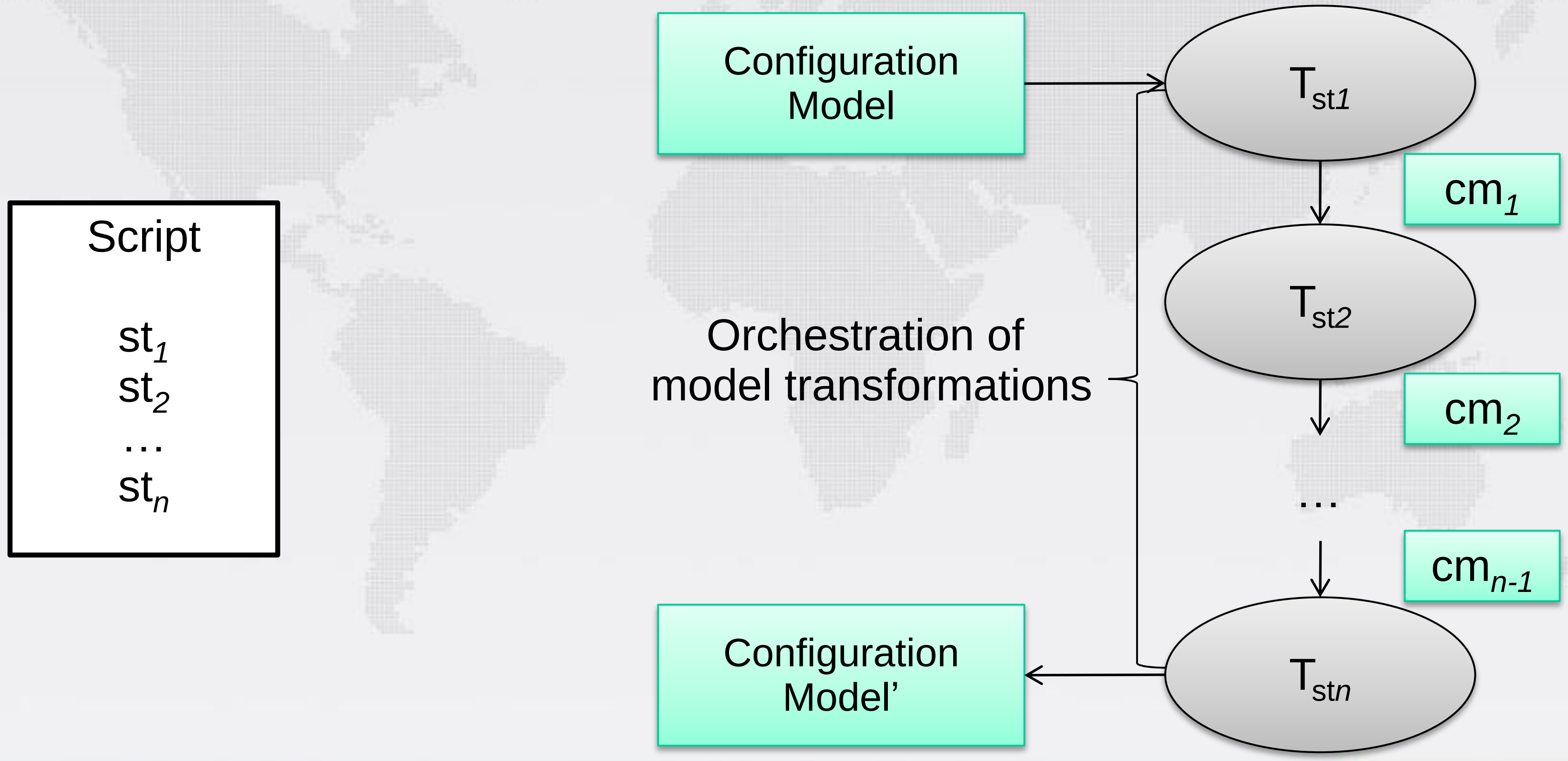
Upgrade Simulator: Script simulator

- The scripts are specified by means of the Mancoosi DSL
- For each DSL statement (*st*) a corresponding model transformation (T_{st}) is defined in order to specify how the execution of st affects a source configuration model (*cm*)

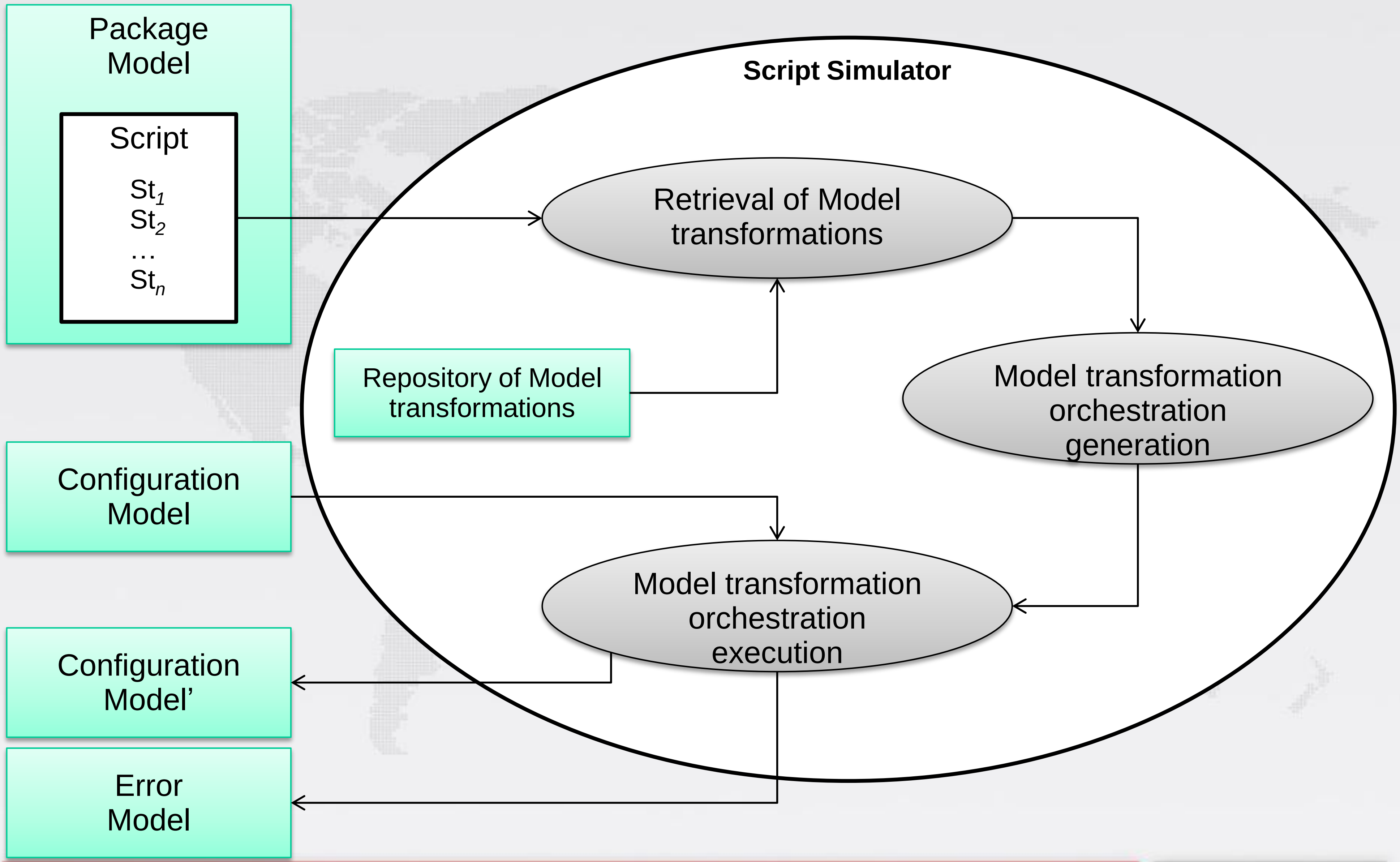


Upgrade Simulator: Script simulator

- The scripts are specified by means of the Mancoosi DSL
- For each DSL statement (*st*) a corresponding model transformation (T_{st}) is defined in order to specify how the execution of st affects a source configuration model (*cm*)

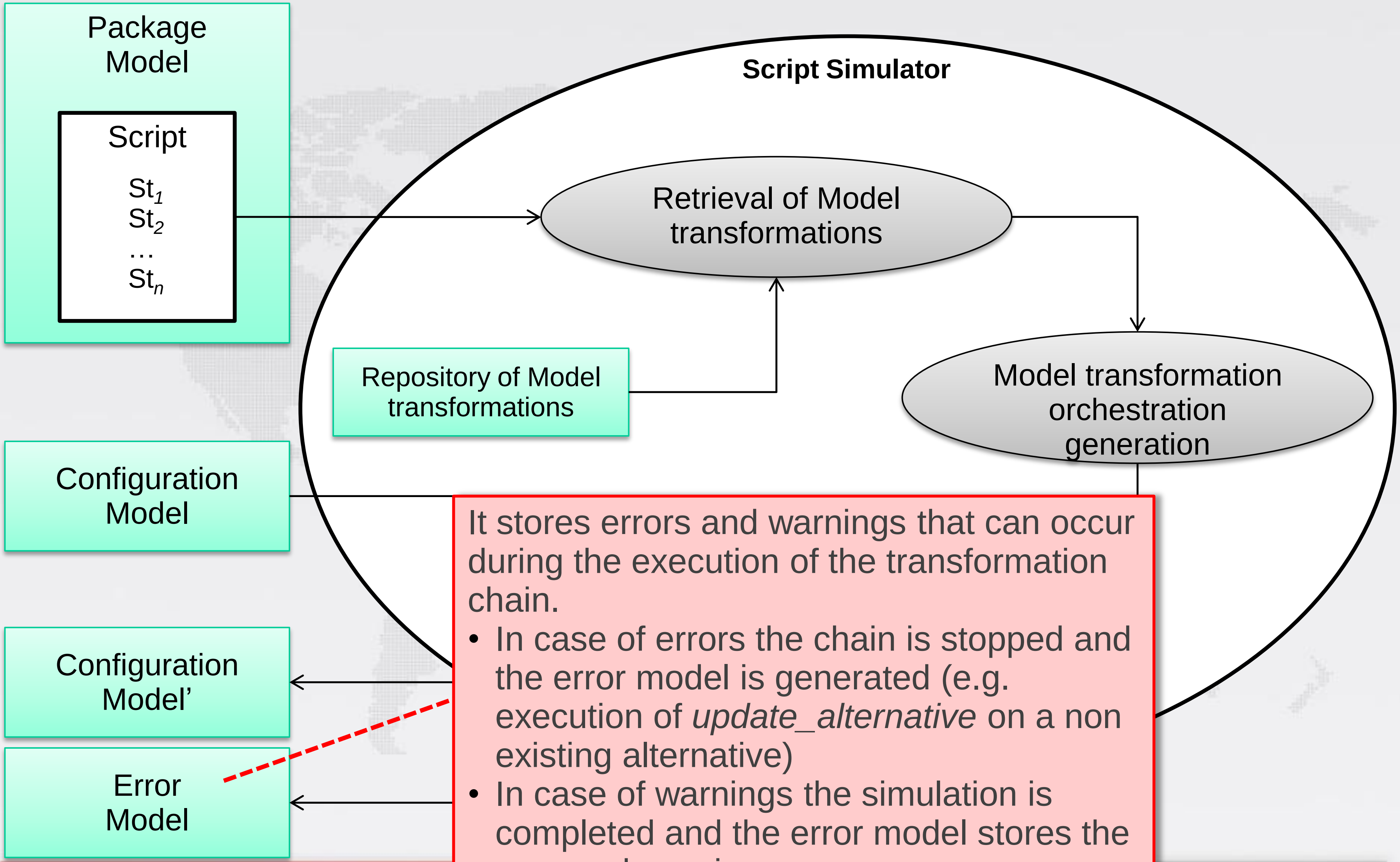


Upgrade Simulator: Script simulator


Package Model
Script
St_1
St_2
…
St_n
Script Simulator
Retrieval of Model transformations
Repository of Model transformations
Model transformation orchestration generation
Configuration Model
Model transformation orchestration execution
Configuration Model'
Error Model

mancoosi

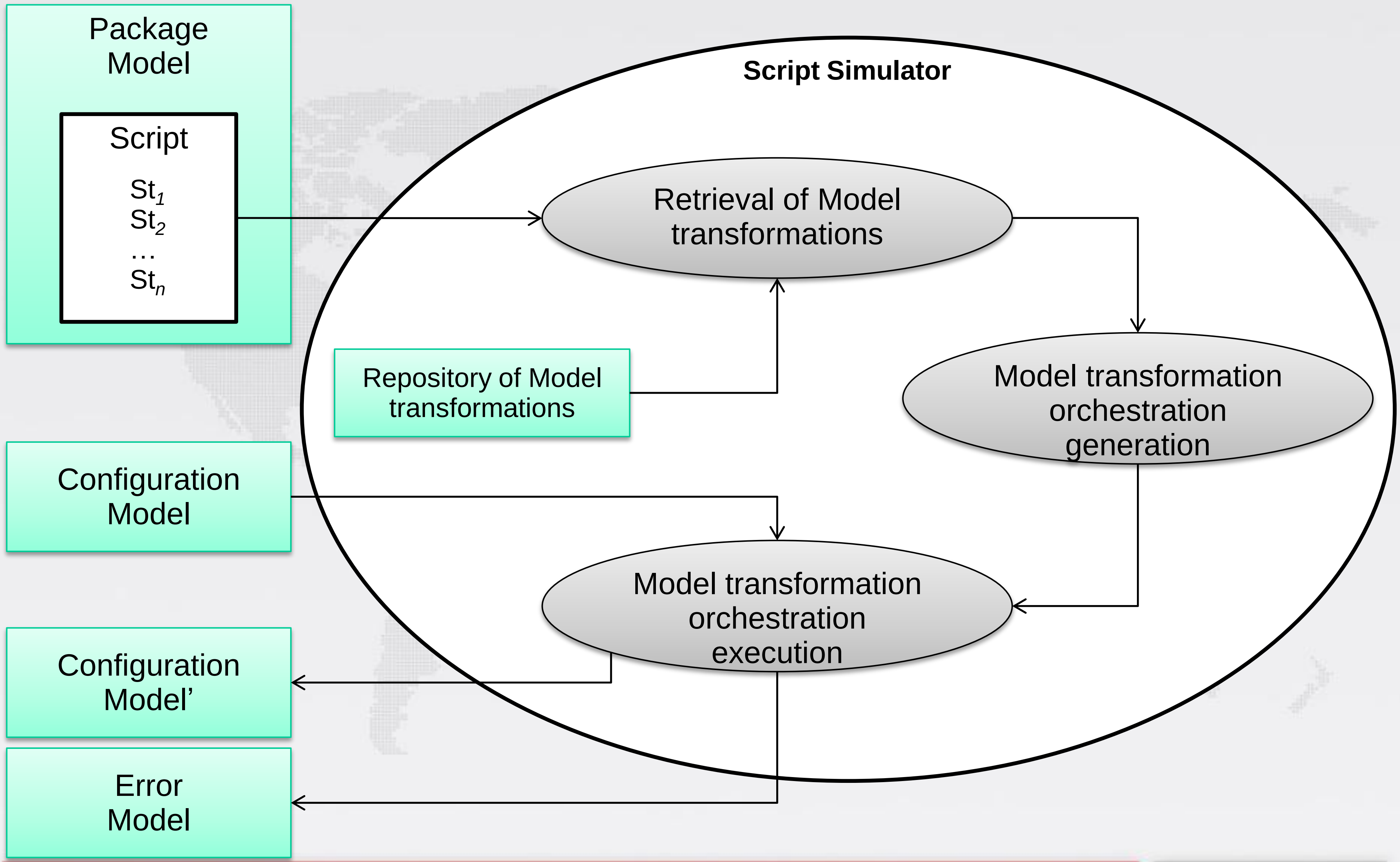
Upgrade Simulator: Script simulator


Package Model
Script
St_1
St_2
…
St_n
Script Simulator
Retrieval of Model transformations
Repository of Model transformations
Model transformation orchestration generation
Configuration Model
Configuration Model’
Error Model

It stores errors and warnings that can occur during the execution of the transformation chain.

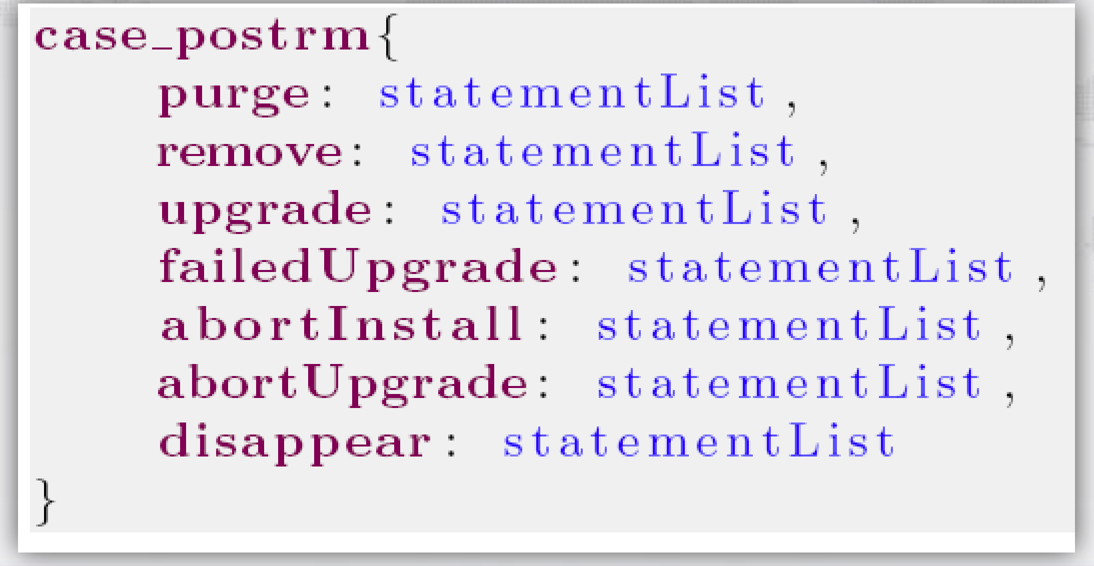
- In case of errors the chain is stopped and the error model is generated (e.g. execution of *update_alternative* on a non existing alternative)
- In case of warnings the simulation is completed and the error model stores the occurred warnings.

Upgrade Simulator: Script simulator


Package Model
Script
St_1
St_2
…
St_n
Script Simulator
Retrieval of Model transformations
Repository of Model transformations
Model transformation orchestration generation
Configuration Model
Model transformation orchestration execution
Configuration Model'
Error Model

Upgrade Simulator: Model Transformation Orchestration

- Model Transformation Orchestration (MTO) aims at supporting the construction of complex model transformations from other transformations already defined
- Model transformations are defined in ATL that does not provide a native support to compose different transformations
- A proper support to MTO has to be able to deal with situations like the following:



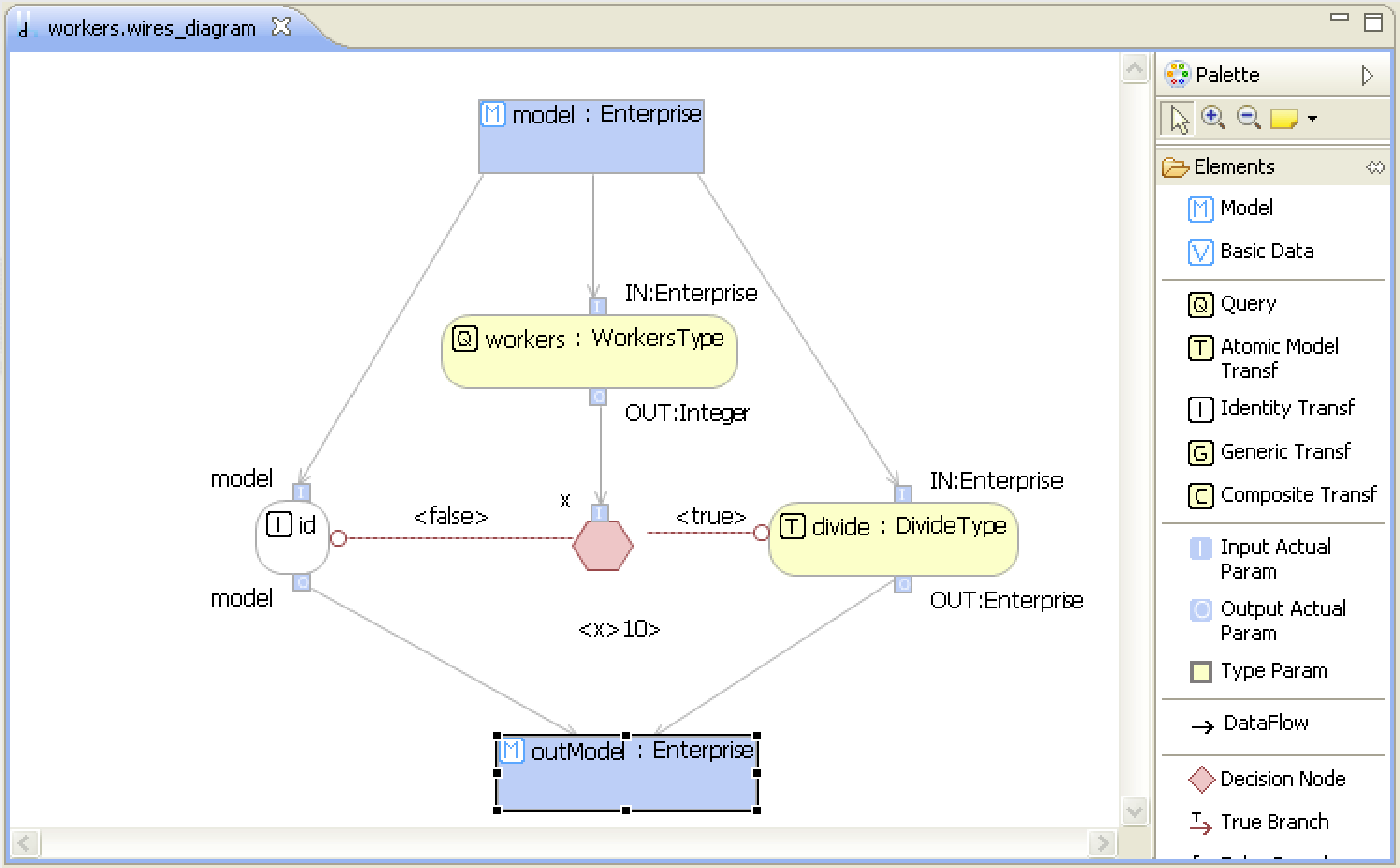
```
case_postrm{
      purge: statementList ,
      remove: statementList ,
      upgrade: statementList ,
      failedUpgrade: statementList ,
      abortInstall: statementList ,
      abortUpgrade: statementList ,
      disappear: statementList
}
```

Upgrade Simulator: Model Transformation Orchestration

- » Tipically model transformations are composed by means of Ant scripts that are difficult to manage
- » Wires* is a Domain Specific Language that enables the high-level orchestration of model transformations [1]
  - It provides a visual notation for defining *chains of model transformation in a modular and compositional manner*
  - It is supported by a graphical framework and an *execution engine* that loads the appropriate models and execute the transformations along the predefined path

[1] J.E. Rivera, D. Ruiz-Gonzalez, F. Lopez-Romero, J. Bautista, and A. Vallecillo ***Orchestrating ATL Model Transformations***. In Proc. of MtATL 2009. Nantes, France, 8-9 July 2009

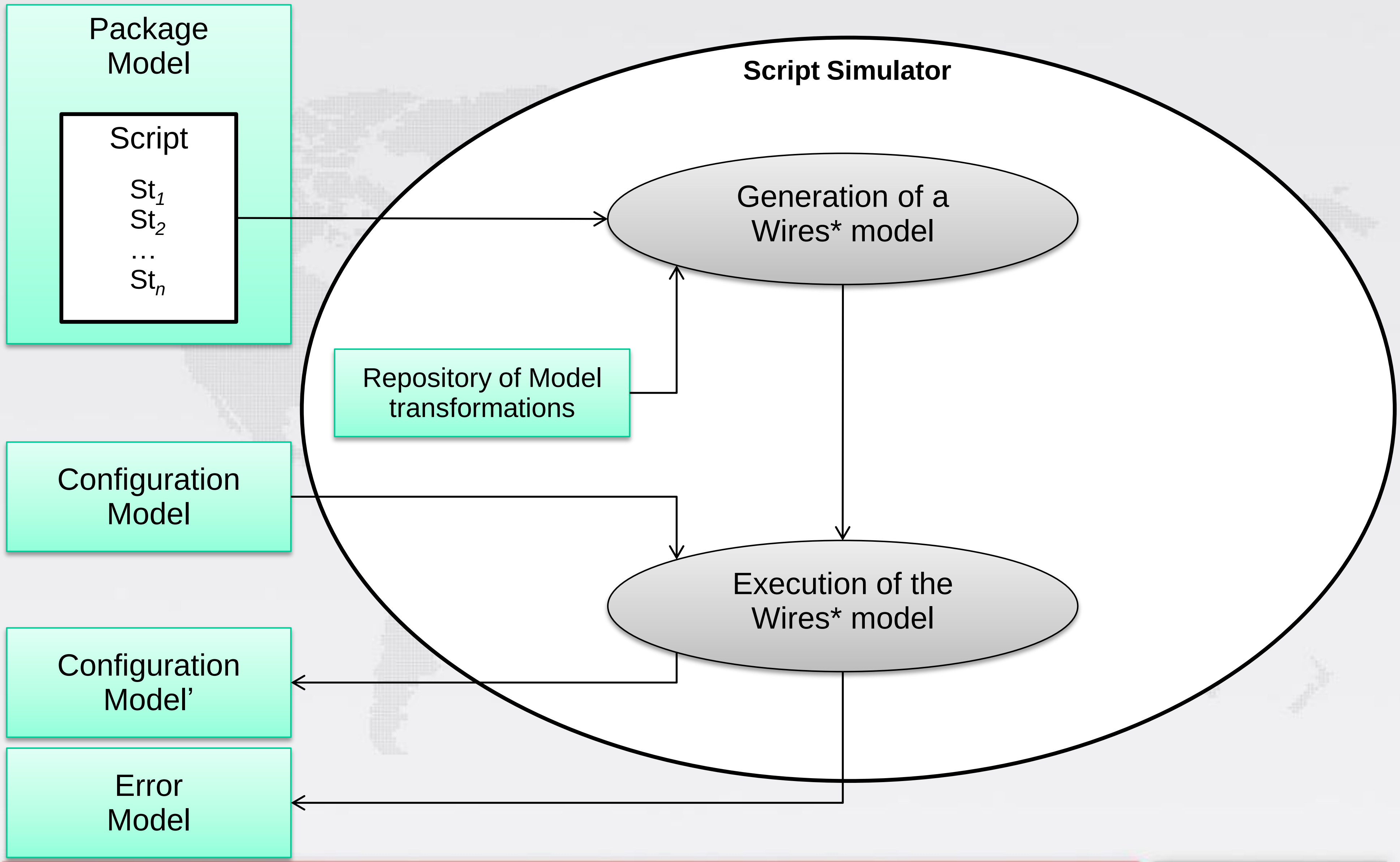
Upgrade Simulator: Model Transformation Orchestration



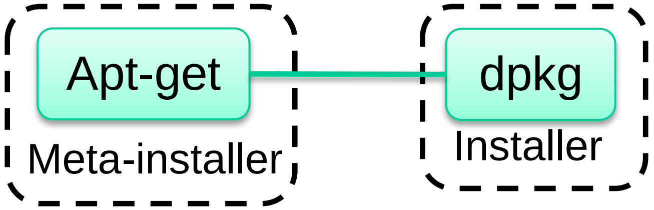
» Sample Wires* model which orchestrates the *workers* query and the *divide* transformation

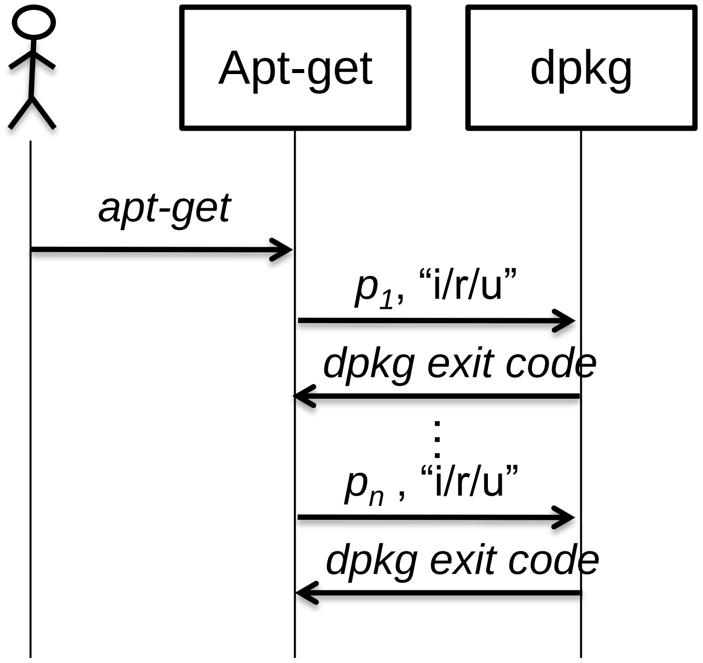
mancoosi

Upgrade Simulator: Using Wires* in the Script simulator

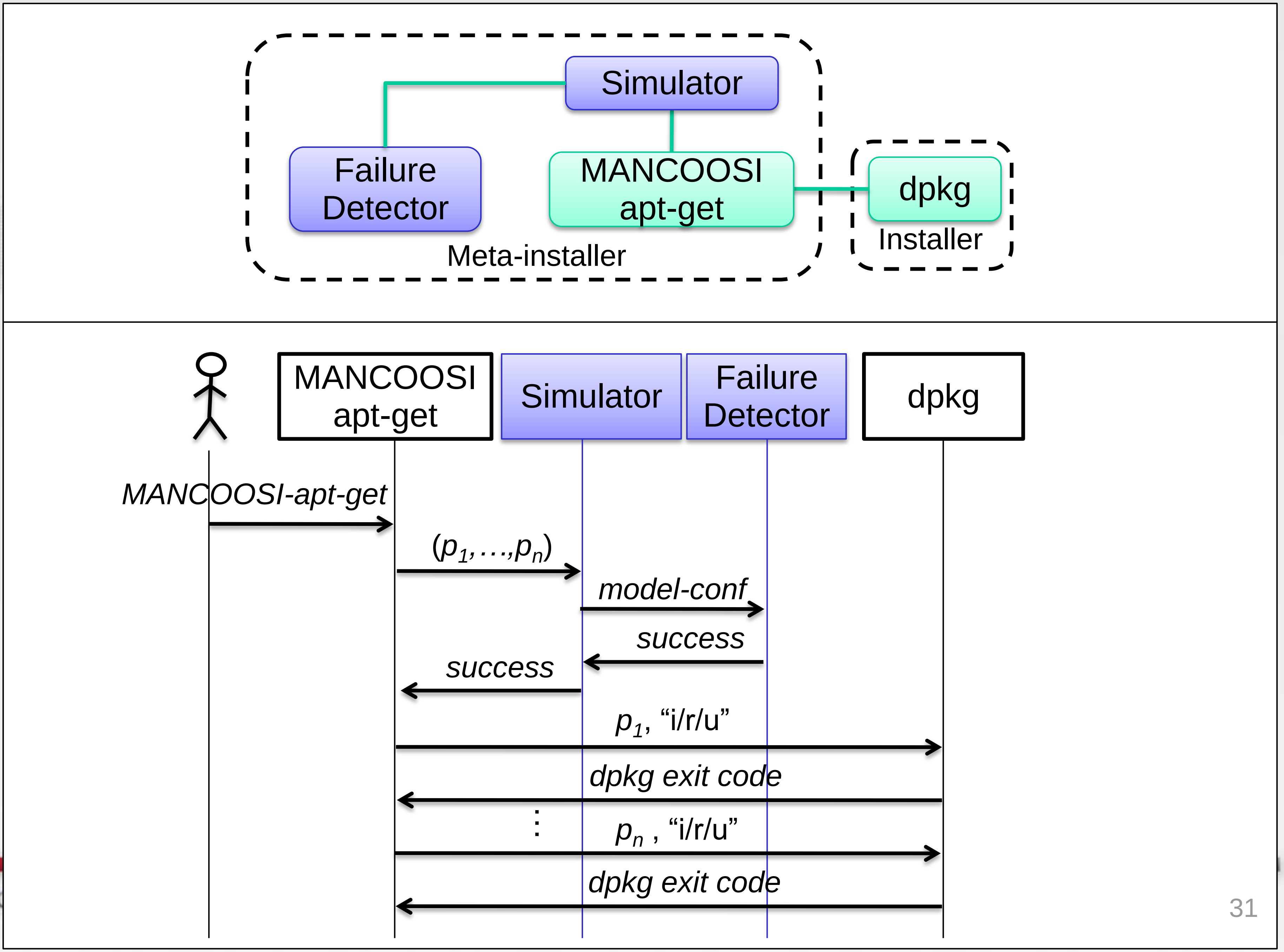


The apt-get meta-installer: current solution

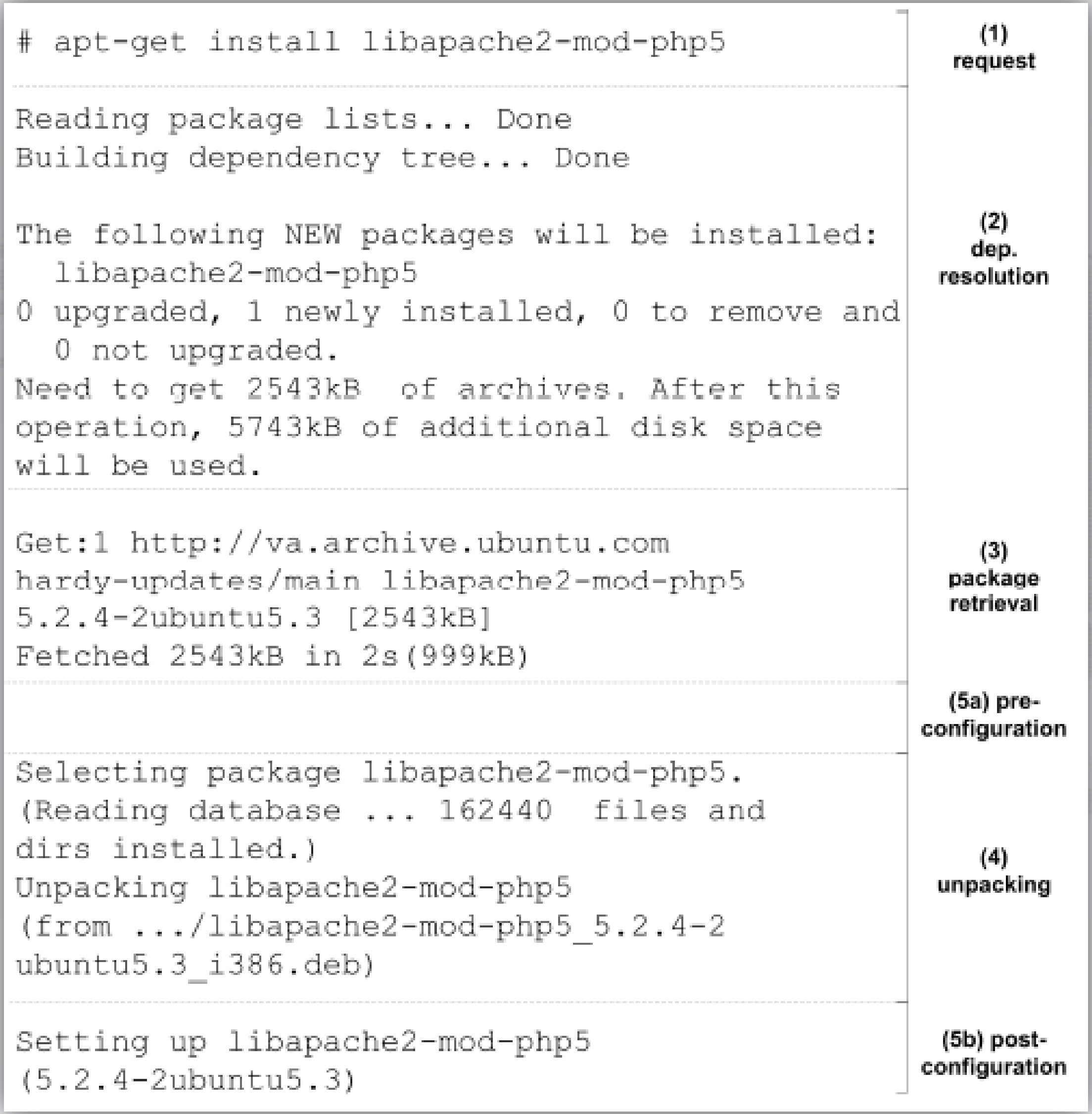




The apt-get meta-installer: proposed solution



The apt-get meta-installer: current solution



```
# apt-get install libapache2-mod-php5
```

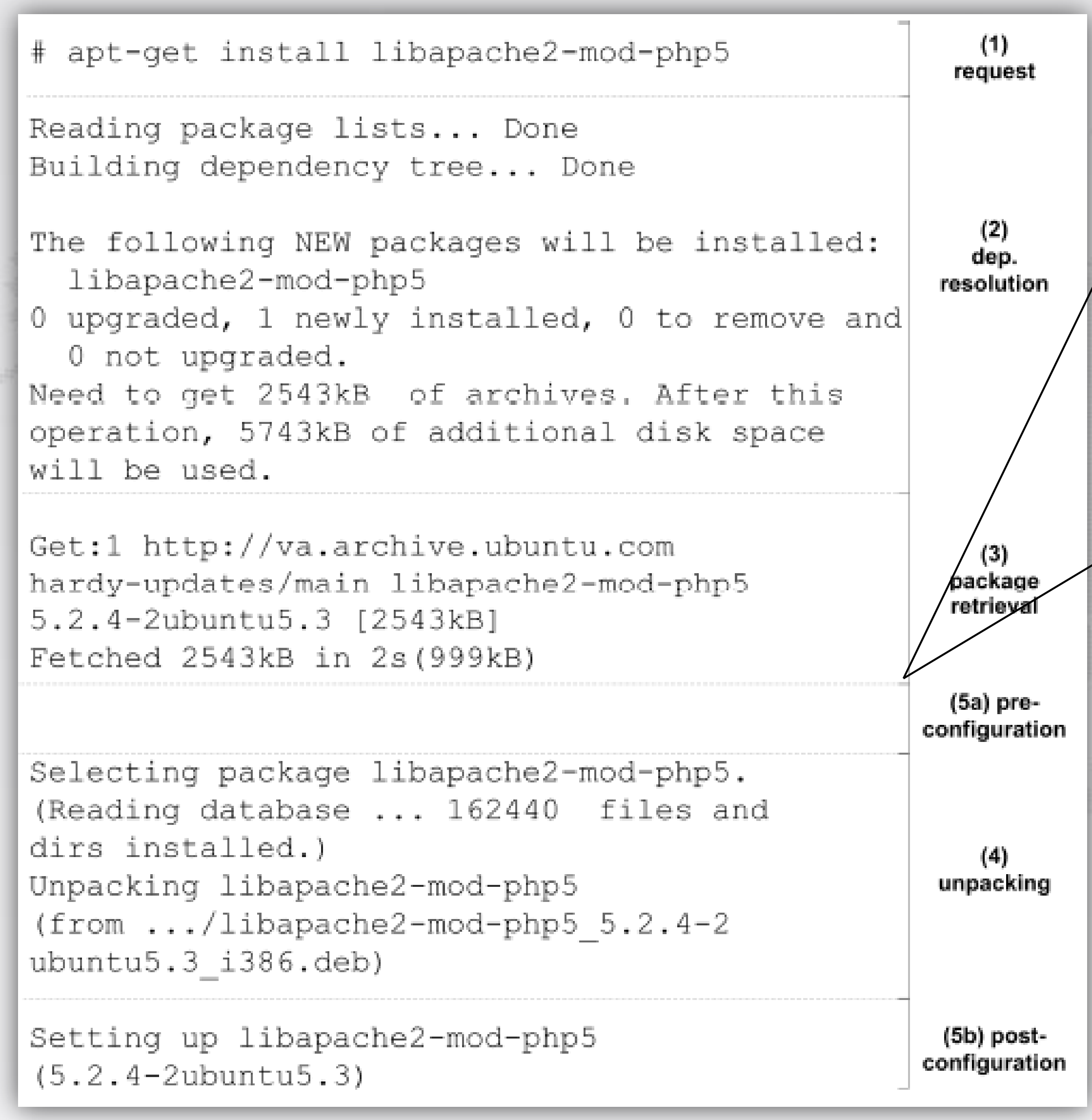
```
Reading package lists... Done
Building dependency tree... Done

The following NEW packages will be installed:
  libapache2-mod-php5
0 upgraded, 1 newly installed, 0 to remove and
  0 not upgraded.
Need to get 2543kB  of archives. After this
operation, 5743kB of additional disk space
will be used.
```

```
Get:1 http://va.archive.ubuntu.com
hardy-updates/main libapache2-mod-php5
5.2.4-2ubuntu5.3 [2543kB]
Fetched 2543kB in 2s(999kB)
```

```
Selecting package libapache2-mod-php5.
(Reading database ... 162440  files and
dirs installed.)
Unpacking libapache2-mod-php5
(from .../libapache2-mod-php5_5.2.4-2
ubuntu5.3_i386.deb)
```

```
Setting up libapache2-mod-php5
(5.2.4-2ubuntu5.3)
```

The apt-get meta-installer: proposed solution



```
# apt-get install libapache2-mod-php5
```

```
Reading package lists... Done
Building dependency tree... Done

The following NEW packages will be installed:
  libapache2-mod-php5
0 upgraded, 1 newly installed, 0 to remove and
  0 not upgraded.
Need to get 2543kB  of archives. After this
operation, 5743kB of additional disk space
will be used.
```

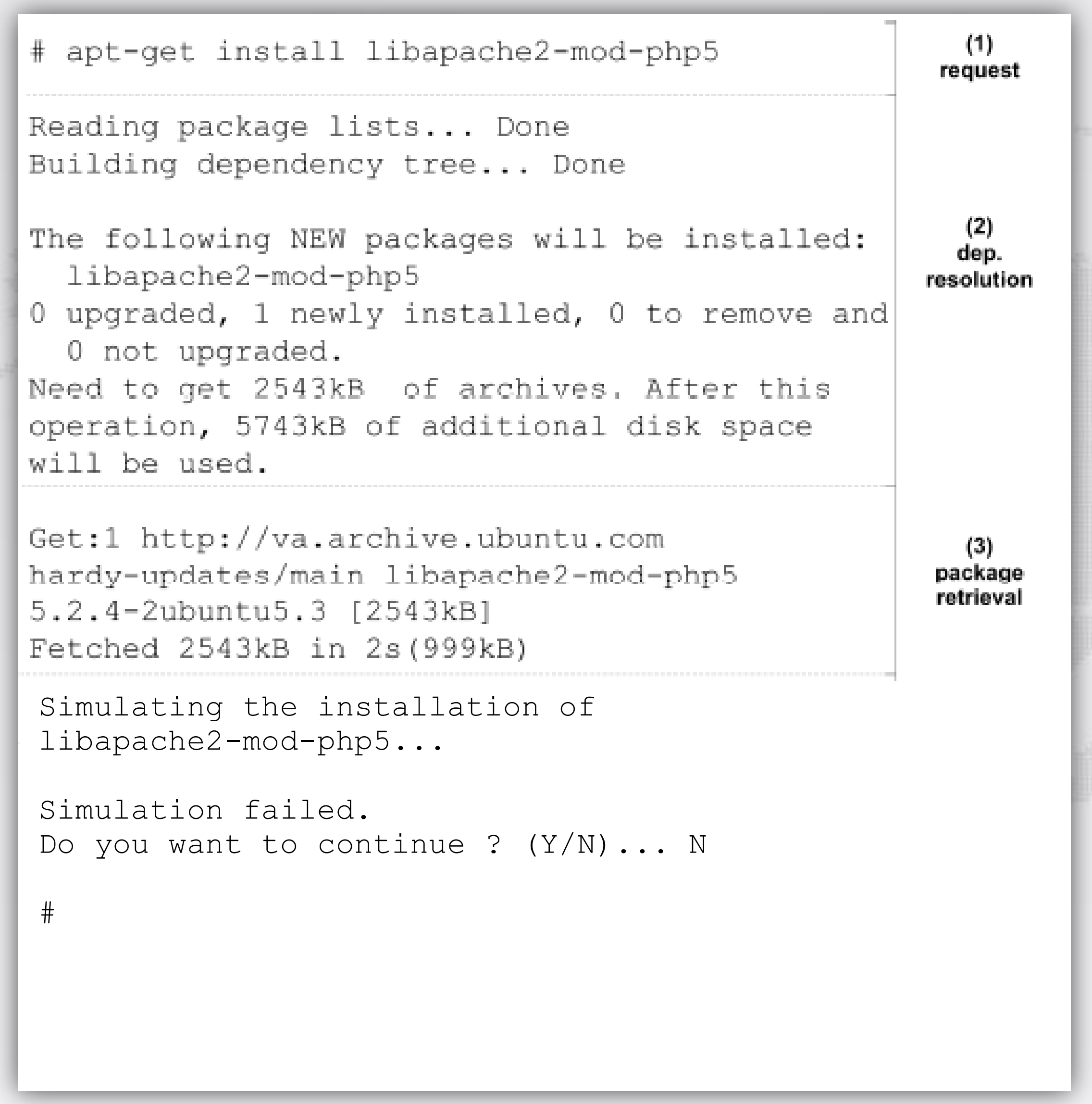
```
Get:1 http://va.archive.ubuntu.com
hardy-updates/main libapache2-mod-php5
5.2.4-2ubuntu5.3 [2543kB]
Fetched 2543kB in 2s(999kB)
```

```
Selecting package libapache2-mod-php5.
(Reading database ... 162440  files and
dirs installed.)
Unpacking libapache2-mod-php5
(from .../libapache2-mod-php5_5.2.4-2
ubuntu5.3_i386.deb)
```

```
Setting up libapache2-mod-php5
(5.2.4-2ubuntu5.3)
```

```
Simulating the installation of
libapache2-mod-php5...

Simulation succeeded.
```

The apt-get meta-installer: proposed solution



```
# apt-get install libapache2-mod-php5
```

```
Reading package lists... Done
Building dependency tree... Done

The following NEW packages will be installed:
  libapache2-mod-php5
0 upgraded, 1 newly installed, 0 to remove and
  0 not upgraded.
Need to get 2543kB  of archives. After this
operation, 5743kB of additional disk space
will be used.
```

```
Get:1 http://va.archive.ubuntu.com
hardy-updates/main libapache2-mod-php5
5.2.4-2ubuntu5.3 [2543kB]
Fetched 2543kB in 2s(999kB)
```

```
Simulating the installation of
libapache2-mod-php5...

Simulation failed.
Do you want to continue ? (Y/N)... N

#
```

Overview of the upgrade simulator implementation

» **Input**:

- Upgrade plan ($(p_1,u_1), (p_2,u_2),\ldots,(p_n,u_n)$)
- System configuration model

» **For each couple (p_i,u_i)**

1. the model corresponding to p_i is created by means of the package injection or retrieved if already existing
2. the *pre* upgrade script is simulated
   - A Wires* model is created to chain the model transformations corresponding to the semantics of the statements to be executed
   - The created Wires* model is executed
3. the *unpacking* operation is performed
   - The system configuration model is updated to include the representation of those files contained in the considered package
4. the *post* upgrade script is simulated (as in 2)
5. the *finalize* operation is performed
   - The system configuration model is updated to add or remove the considered package according to u_i

» **Output:**

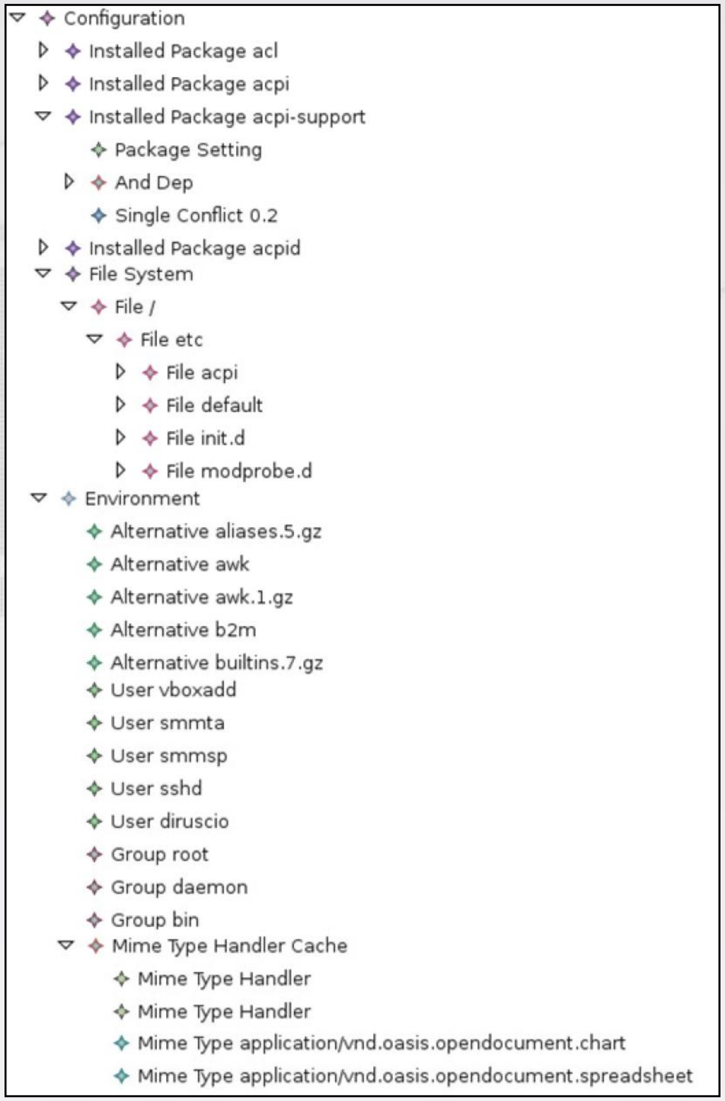
- Updated system configuration model or an error model which reports the error raised during the simulation

Overview of the upgrade simulator implementation

» Input

```
<?xml version="1.0"?>
<sequence>
       <package name="swi-prolog">
         <action>install</action>
       </package>
       <package name="swi-prolog">
         <action>remove</action>
       </package>
</sequence>
```

Upgrade plan

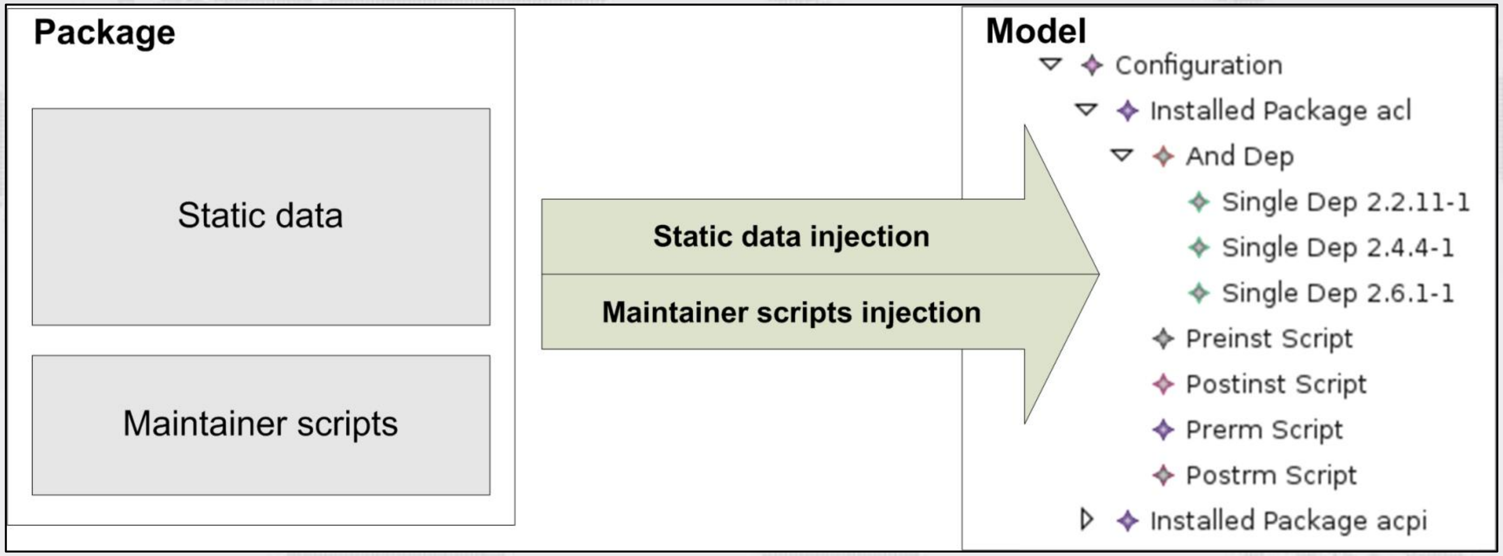


```
▽ Configuration
  ▷ Installed Package acl
  ▷ Installed Package acpi
  ▽ Installed Package acpi-support
      Package Setting
    ▷ And Dep
      Single Conflict 0.2
  ▷ Installed Package acpid
  ▽ File System
    ▽ File /
      ▽ File etc
        ▷ File acpi
        ▷ File default
        ▷ File init.d
        ▷ File modprobe.d
  ▽ Environment
      Alternative aliases.5.gz
      Alternative awk
      Alternative awk.1.gz
      Alternative b2m
      Alternative builtins.7.gz
      User vboxadd
      User smmta
      User smmsp
      User sshd
      User diruscio
      Group root
      Group daemon
      Group bin
    ▽ Mime Type Handler Cache
        Mime Type Handler
        Mime Type Handler
        Mime Type application/vnd.oasis.opendocument.chart
        Mime Type application/vnd.oasis.opendocument.spreadsheet
```

System configuration model

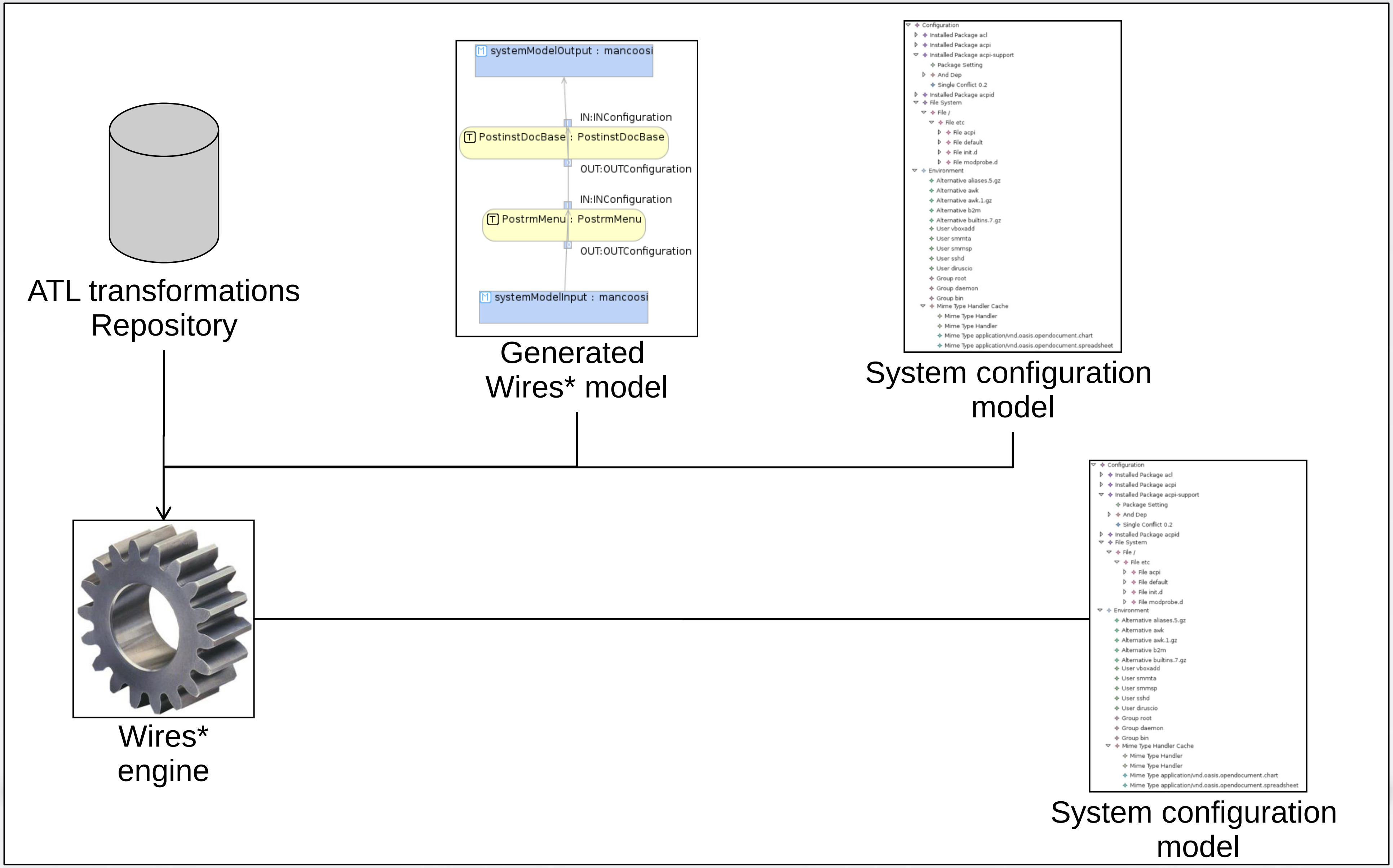
Overview of the upgrade simulator implementation

» Step 1: package injection



Overview of the upgrade simulator implementation

» Step 2: simulation of the *pre* upgrade script

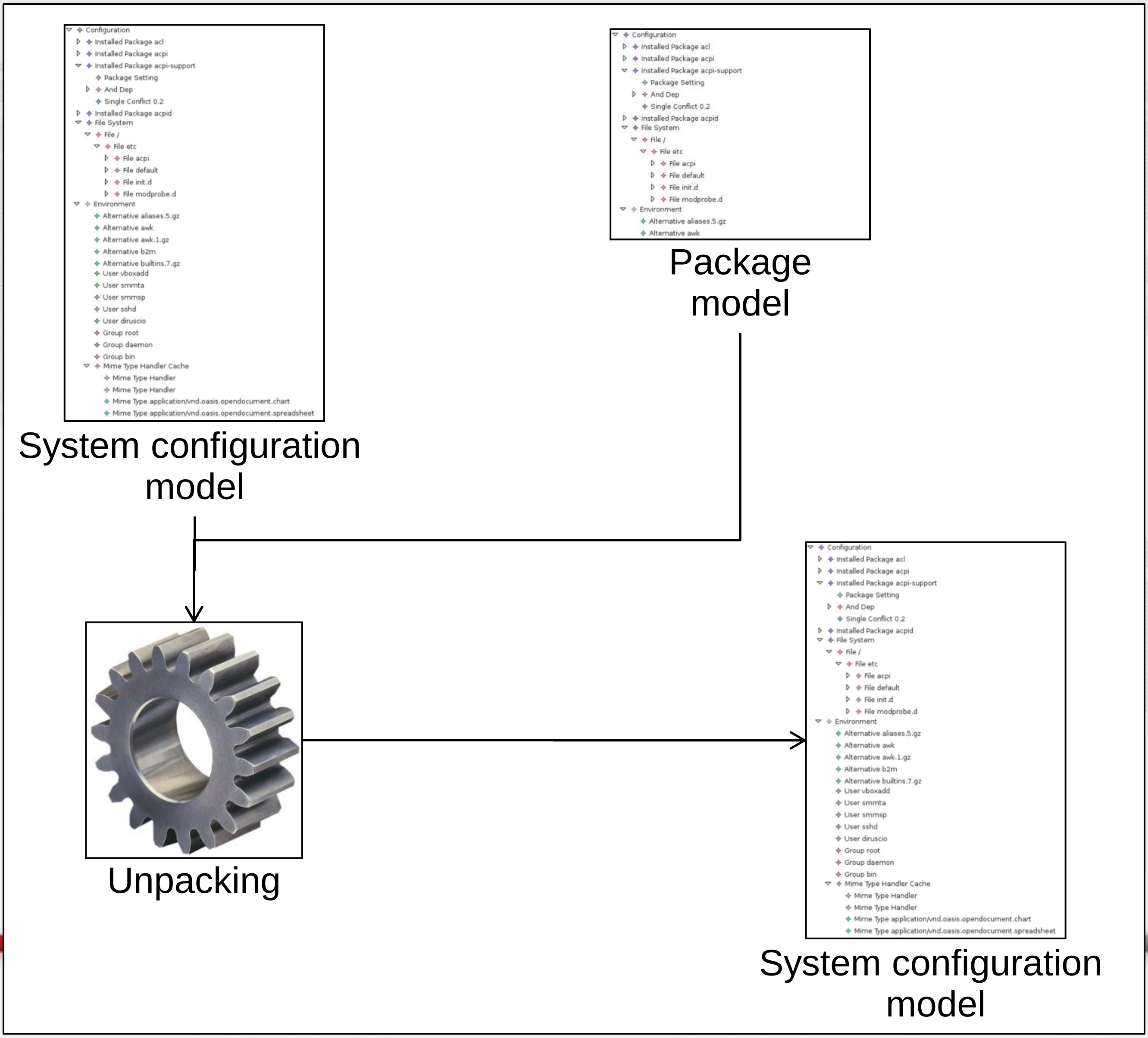

ATL transformations Repository
systemModelOutput : mancoosi
IN:INConfiguration
PostinstDocBase : PostinstDocBase
OUT:OUTConfiguration
IN:INConfiguration
PostrmMenu : PostrmMenu
OUT:OUTConfiguration
systemModelInput : mancoosi
Generated Wires* model
System configuration model
Wires* engine
System configuration model

```
▽ Configuration
  ▷ Installed Package acl
  ▷ Installed Package acpi
  ▽ Installed Package acpi-support
      Package Setting
    ▷ And Dep
      Single Conflict 0.2
  ▷ Installed Package acpid
  ▽ File System
    ▽ File /
      ▽ File etc
        ▷ File acpi
        ▷ File default
        ▷ File init.d
        ▷ File modprobe.d
  ▽ Environment
      Alternative aliases.5.gz
      Alternative awk
      Alternative awk.1.gz
      Alternative b2m
      Alternative builtins.7.gz
      User vboxadd
      User smmta
      User smmsp
      User sshd
      User diruscio
      Group root
      Group daemon
      Group bin
    ▽ Mime Type Handler Cache
        Mime Type Handler
        Mime Type Handler
        Mime Type application/vnd.oasis.opendocument.chart
        Mime Type application/vnd.oasis.opendocument.spreadsheet
```

```
▽ Configuration
  ▷ Installed Package acl
  ▷ Installed Package acpi
  ▽ Installed Package acpi-support
      Package Setting
    ▷ And Dep
      Single Conflict 0.2
  ▷ Installed Package acpid
  ▽ File System
    ▽ File /
      ▽ File etc
        ▷ File acpi
        ▷ File default
        ▷ File init.d
        ▷ File modprobe.d
  ▽ Environment
      Alternative aliases.5.gz
      Alternative awk
      Alternative awk.1.gz
      Alternative b2m
      Alternative builtins.7.gz
      User vboxadd
      User smmta
      User smmsp
      User sshd
      User diruscio
      Group root
      Group daemon
      Group bin
    ▽ Mime Type Handler Cache
        Mime Type Handler
        Mime Type Handler
        Mime Type application/vnd.oasis.opendocument.chart
        Mime Type application/vnd.oasis.opendocument.spreadsheet
```

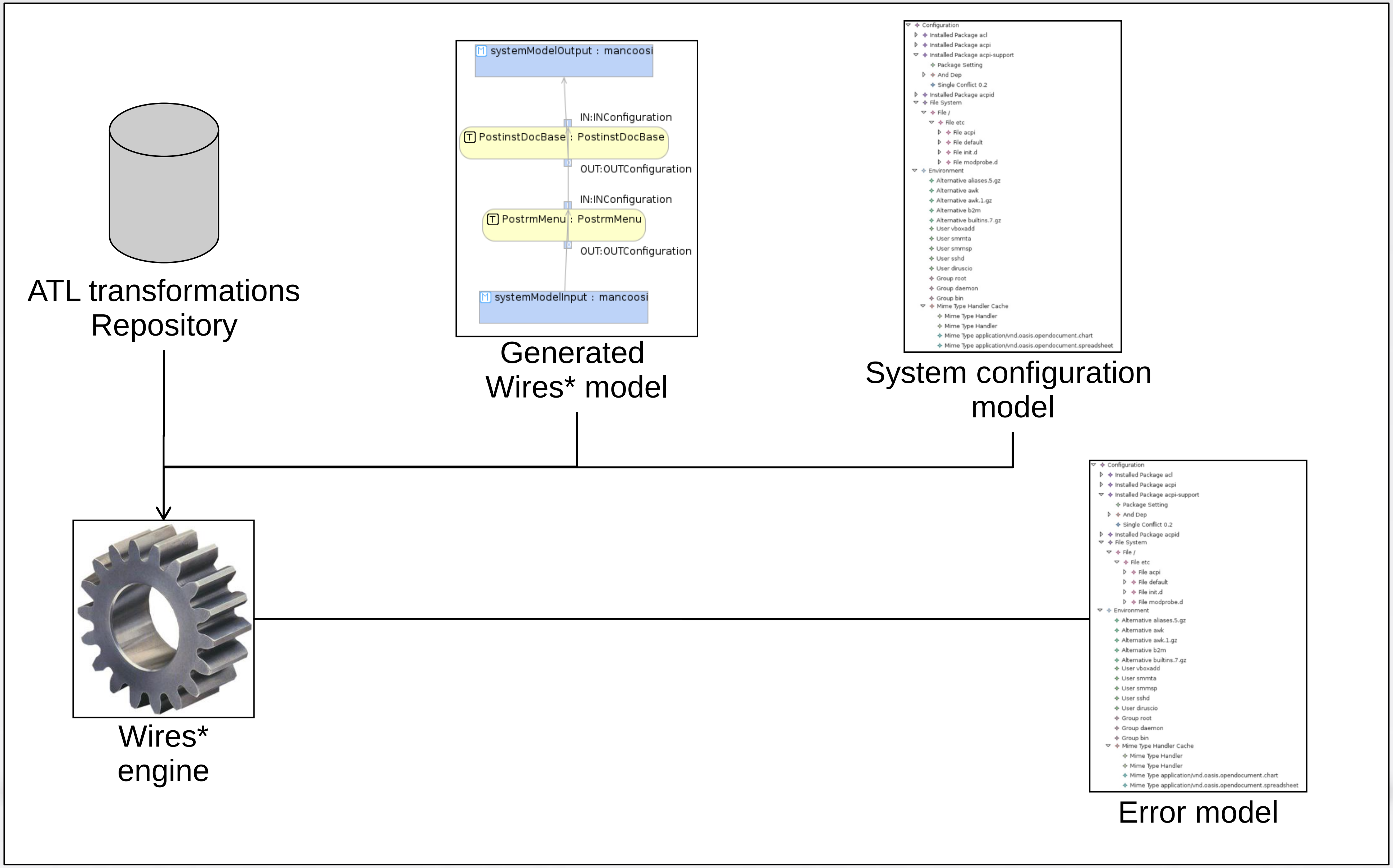
Overview of the upgrade simulator implementation

» Step 3: simulation of the *unpacking*



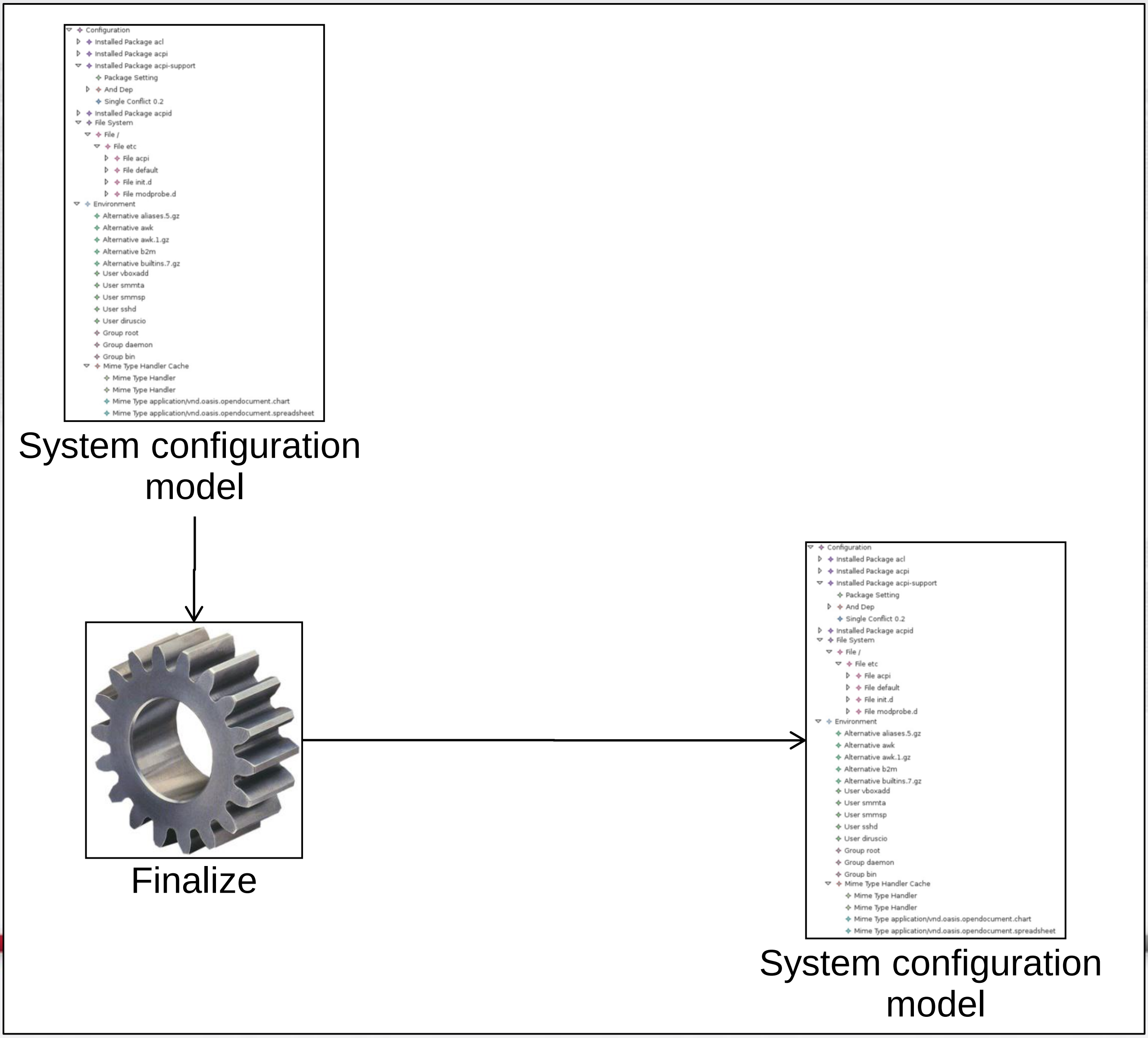
Overview of the upgrade simulator implementation

» Step 4: simulation of the *post* upgrade script



Overview of the upgrade simulator implementation

» Step 5: execution of the *finalize* operation



```
Configuration
  Installed Package acl
  Installed Package acpi
  Installed Package acpi-support
    Package Setting
    And Dep
    Single Conflict 0.2
  Installed Package acpid
  File System
    File /
      File etc
        File acpi
        File default
        File init.d
        File modprobe.d
  Environment
    Alternative aliases.5.gz
    Alternative awk
    Alternative awk.1.gz
    Alternative b2m
    Alternative builtins.7.gz
    User vboxadd
    User smmta
    User smmsp
    User sshd
    User diruscio
    Group root
    Group daemon
    Group bin
    Mime Type Handler Cache
      Mime Type Handler
      Mime Type Handler
      Mime Type application/vnd.oasis.opendocument.chart
      Mime Type application/vnd.oasis.opendocument.spreadsheet
```

```
Configuration
  Installed Package acl
  Installed Package acpi
  Installed Package acpi-support
    Package Setting
    And Dep
    Single Conflict 0.2
  Installed Package acpid
  File System
    File /
      File etc
        File acpi
        File default
        File init.d
        File modprobe.d
  Environment
    Alternative aliases.5.gz
    Alternative awk
    Alternative awk.1.gz
    Alternative b2m
    Alternative builtins.7.gz
    User vboxadd
    User smmta
    User smmsp
    User sshd
    User diruscio
    Group root
    Group daemon
    Group bin
    Mime Type Handler Cache
      Mime Type Handler
      Mime Type Handler
      Mime Type application/vnd.oasis.opendocument.chart
      Mime Type application/vnd.oasis.opendocument.spreadsheet
```

Upgrade simulator: status and limitations

Status

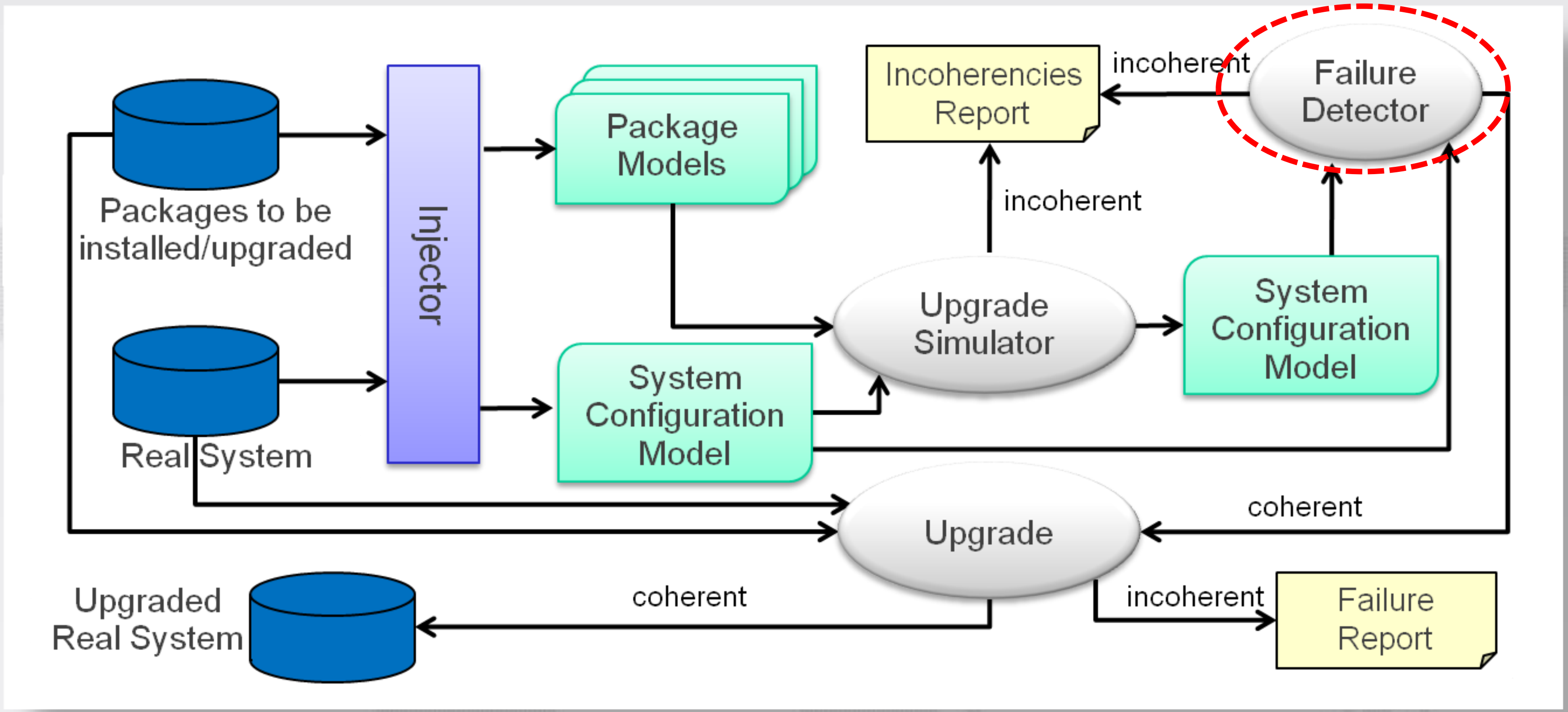
» The simulator is a standard Java application which can be executed at command line without the need of the overall Eclipse platform

Current limitations

» The generation of Wires* models related to complex condition expressions is not implemented yet

» The tagging support has to be finalized

Using models to enhance package upgrades



» A model-based approach is introduced to support the package upgrades and enhance the failure detection possibilities:

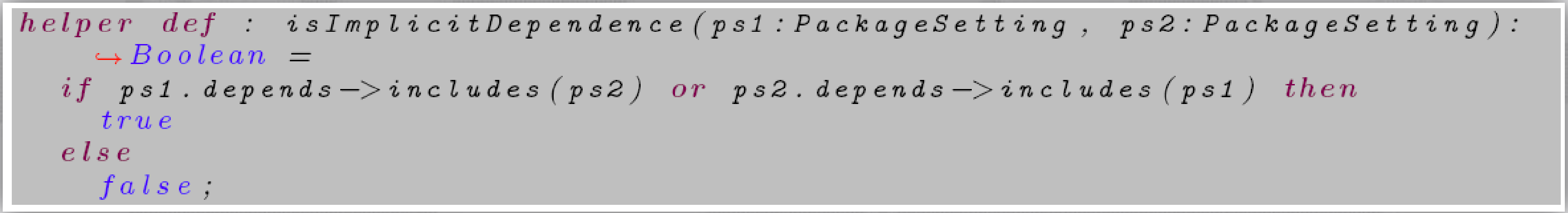
- A simulator is used to predice the effect of maintainer script executions (*deploy-time failures*)
- A failure detector is used to deal with *undetected failures*

Failure detector

- *Discovering implicit dependencies among packages*: we are able to discover dependencies that are not declared into the package's meta-information
- *Discovering missing configuration files*: according to the system configuration model, some configuration files are required but they are not available in the system
- *Discovering Mime-type dangling handlers*: according to the available information, the considered system should be able to manage a mime type, but the corresponding handler is missing in the system
- *Discovering missing services:* the *init.d* file contains services that should start at the system start-up; by querying the configuration model, it is possible to detect missing services
- …

Failure detector implementations

» The failure detector relies on OCL queries each corresponding to a possible failure



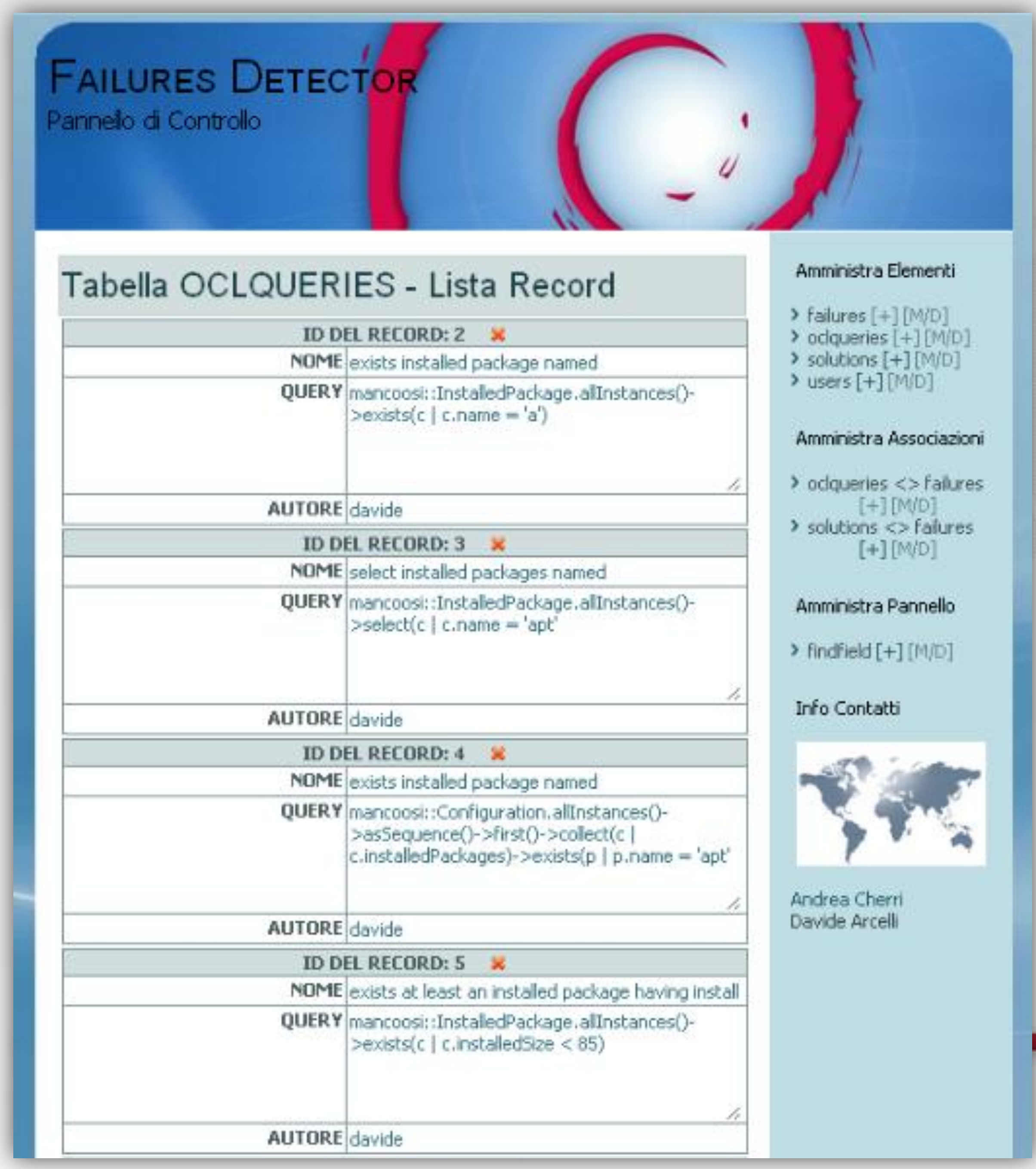
```
helper def : isImplicitDependence(ps1:PackageSetting, ps2:PackageSetting):
    ↪Boolean =
  if ps1.depends->includes(ps2) or ps2.depends->includes(ps1) then
    true
  else
    false;
```

» **Desktop** a local implementation of the failure detector is available to execute a set of queries on the system configuration model generated by the simulator

» **Client/Server** the failure detector is available in a server which is able to query system configuration models uploaded by clients

mancoosi

Failure detector implementations: Client/Server



FAILURES DETECTOR
Pannello di Controllo

Tabella OCLQUERIES - Lista Record

ID DEL RECORD: 2	
NOME	exists installed package named
QUERY	mancoosi::InstalledPackage.allInstances()->exists(c | c.name = 'a')
AUTORE	davide
ID DEL RECORD: 3	
NOME	select installed packages named
QUERY	mancoosi::InstalledPackage.allInstances()->select(c | c.name = 'apt'
AUTORE	davide
ID DEL RECORD: 4	
NOME	exists installed package named
QUERY	mancoosi::Configuration.allInstances()->asSequence()->first()->collect(c | c.installedPackages)->exists(p | p.name = 'apt'
AUTORE	davide
ID DEL RECORD: 5	
NOME	exists at least an installed package having install
QUERY	mancoosi::InstalledPackage.allInstances()->exists(c | c.installedSize < 85)
AUTORE	davide

Amministra Elementi

- failures [+] [M/D]
- oclqueries [+] [M/D]
- solutions [+] [M/D]
- users [+] [M/D]

Amministra Associazioni

- oclqueries <> failures [+] [M/D]
- solutions <> failures [+] [M/D]

Amministra Pannello

- findfield [+] [M/D]

Info Contatti

Andrea Cherri
Davide Arcelli

Failure detector implementations: Client/Server

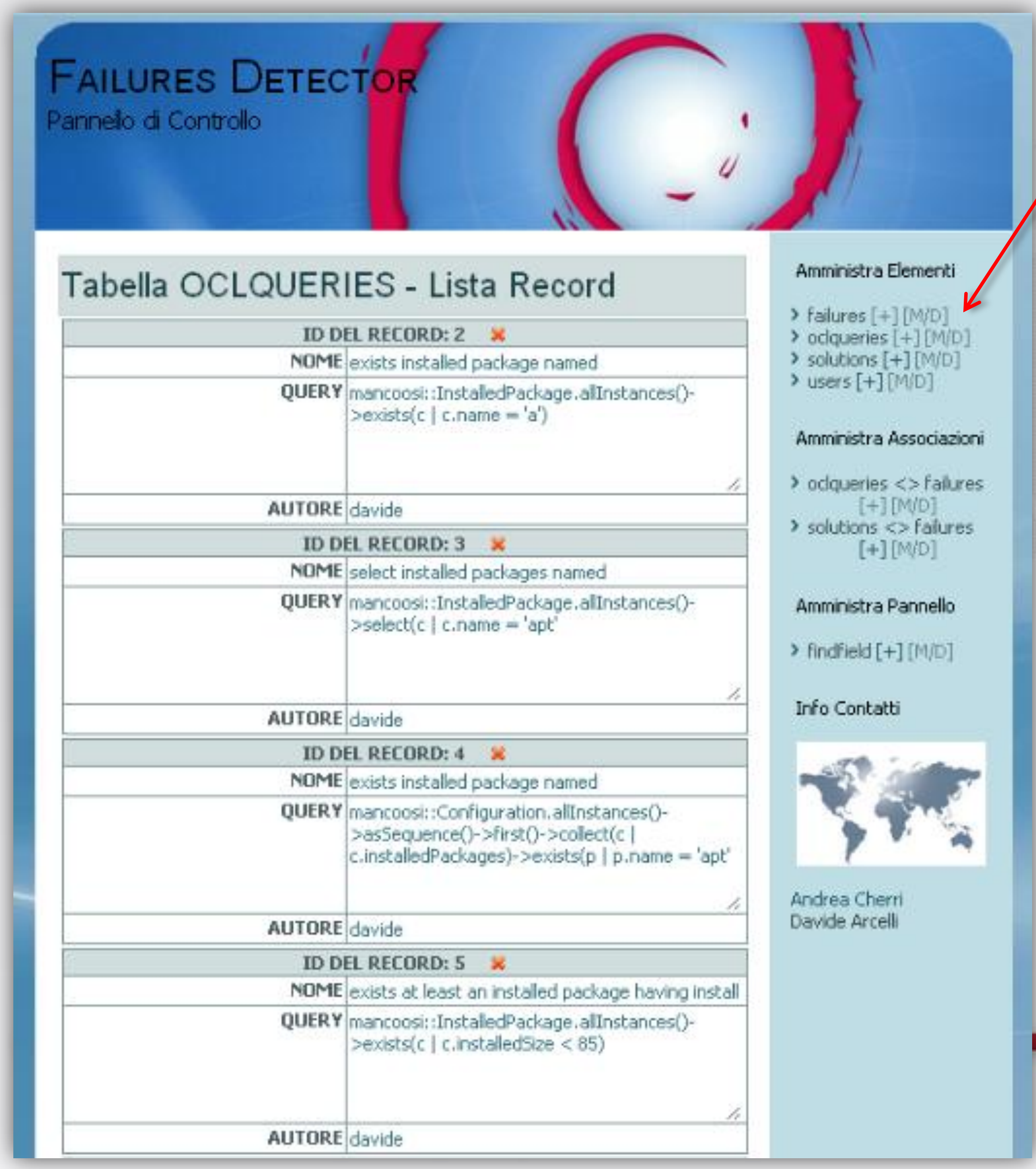


Tabella OCLQUERIES - Lista Record

ID DEL RECORD: 2	
NOME	exists installed package named
QUERY	mancoosi::InstalledPackage.allInstances()->exists(c | c.name = 'a')
AUTORE	davide
ID DEL RECORD: 3	
NOME	select installed packages named
QUERY	mancoosi::InstalledPackage.allInstances()->select(c | c.name = 'apt'
AUTORE	davide
ID DEL RECORD: 4	
NOME	exists installed package named
QUERY	mancoosi::Configuration.allInstances()->asSequence()->first()->collect(c | c.installedPackages)->exists(p | p.name = 'apt'
AUTORE	davide
ID DEL RECORD: 5	
NOME	exists at least an installed package having install
QUERY	mancoosi::InstalledPackage.allInstances()->exists(c | c.installedSize < 85)
AUTORE	davide

Addition of failure specifications

Failure detector implementations: Client/Server

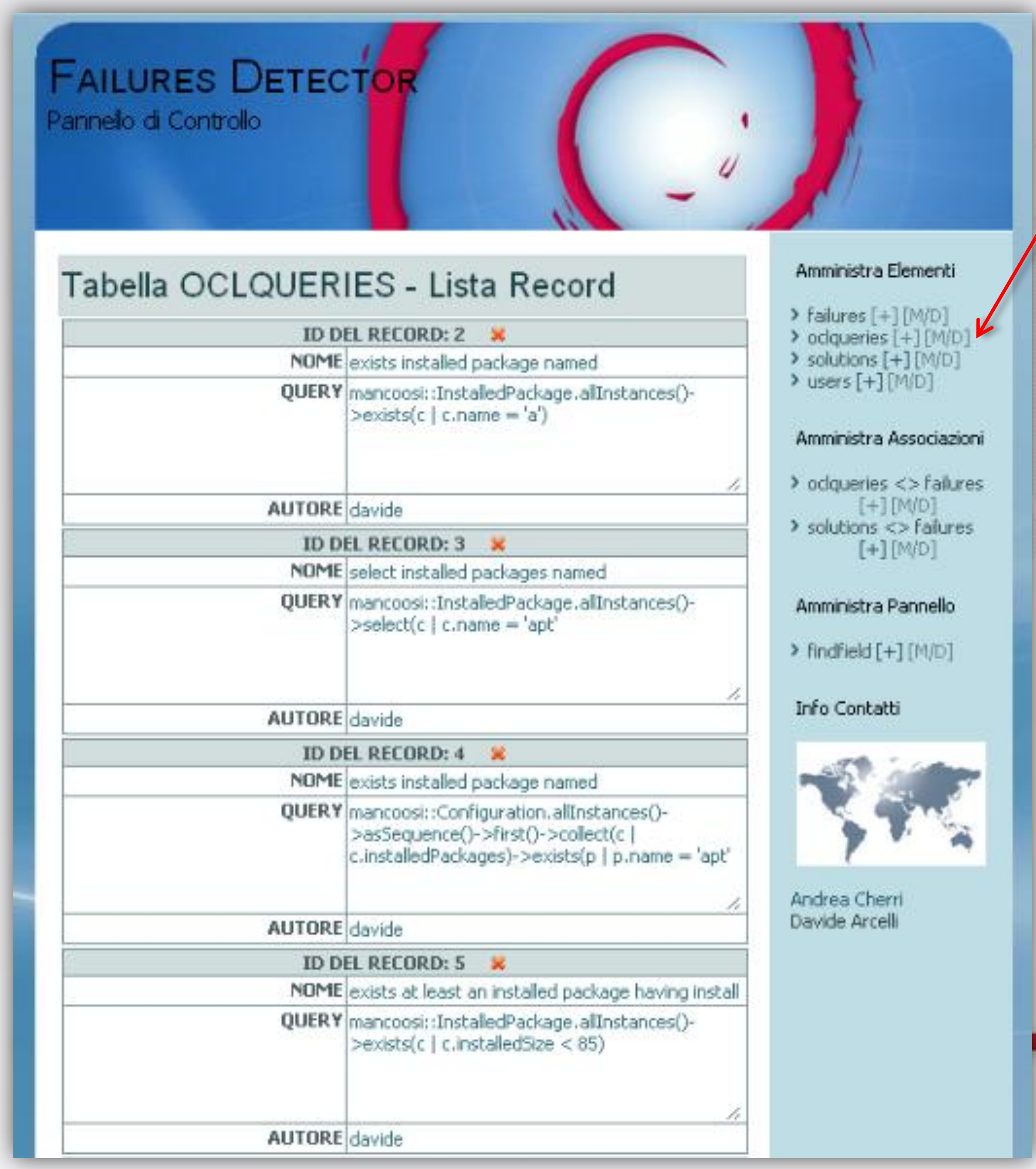


Tabella OCLQUERIES - Lista Record

ID DEL RECORD: 2	
NOME	exists installed package named
QUERY	mancoosi::InstalledPackage.allInstances()->exists(c | c.name = 'a')
AUTORE	davide
ID DEL RECORD: 3	
NOME	select installed packages named
QUERY	mancoosi::InstalledPackage.allInstances()->select(c | c.name = 'apt'
AUTORE	davide
ID DEL RECORD: 4	
NOME	exists installed package named
QUERY	mancoosi::Configuration.allInstances()->asSequence()->first()->collect(c | c.installedPackages)->exists(p | p.name = 'apt'
AUTORE	davide
ID DEL RECORD: 5	
NOME	exists at least an installed package having install
QUERY	mancoosi::InstalledPackage.allInstances()->exists(c | c.installedSize < 85)
AUTORE	davide

Addition of new OCL queries to increase the number of failures which can be detect

Failure detector implementations: Client/Server

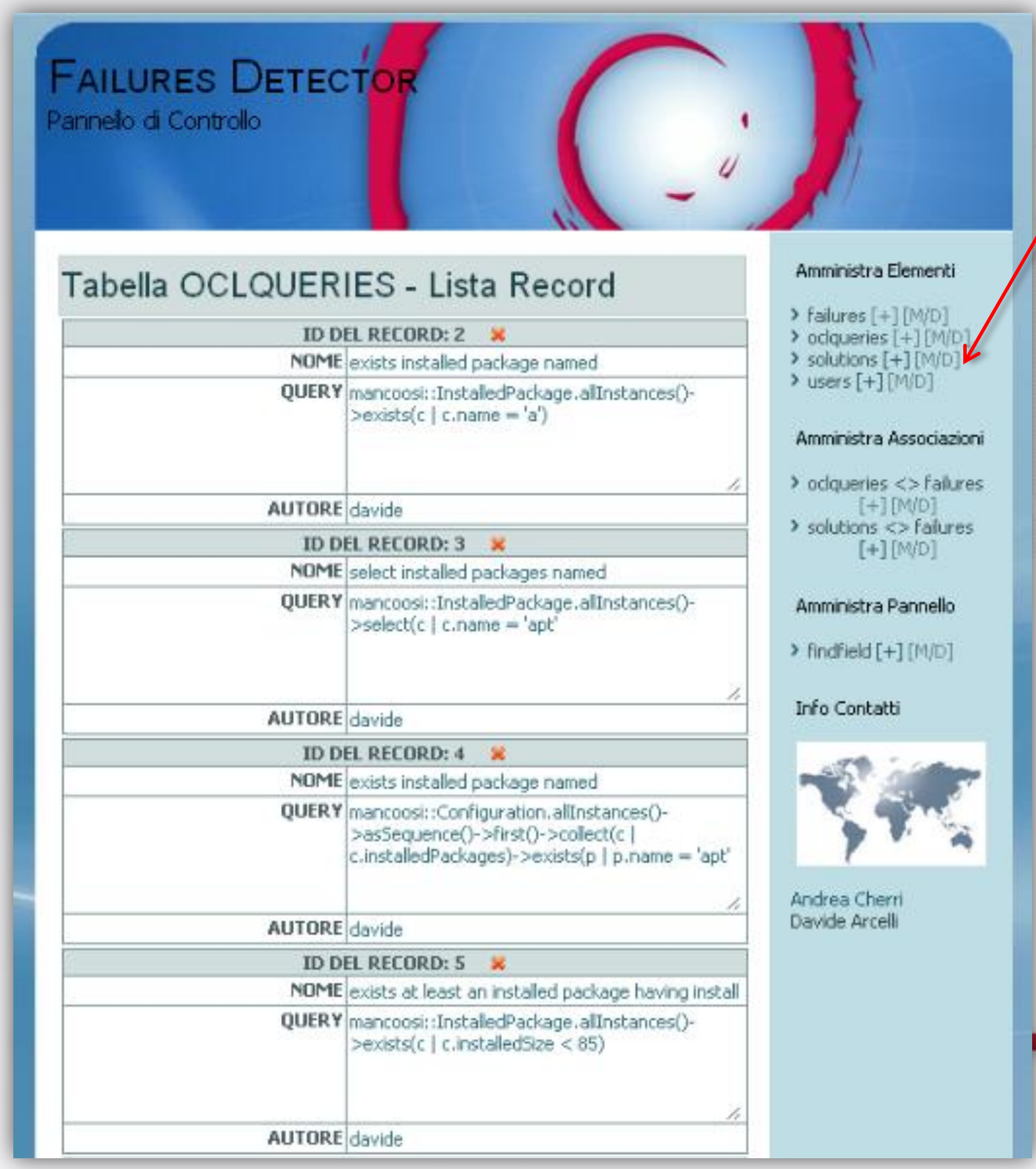


Tabella OCLQUERIES - Lista Record

ID DEL RECORD: 2	
NOME	exists installed package named
QUERY	mancoosi::InstalledPackage.allInstances()->exists(c | c.name = 'a')
AUTORE	davide
ID DEL RECORD: 3	
NOME	select installed packages named
QUERY	mancoosi::InstalledPackage.allInstances()->select(c | c.name = 'apt'
AUTORE	davide
ID DEL RECORD: 4	
NOME	exists installed package named
QUERY	mancoosi::Configuration.allInstances()->asSequence()->first()->collect(c | c.installedPackages)->exists(p | p.name = 'apt'
AUTORE	davide
ID DEL RECORD: 5	
NOME	exists at least an installed package having install
QUERY	mancoosi::InstalledPackage.allInstances()->exists(c | c.installedSize < 85)
AUTORE	davide

It is possible to associate possible solutions to solve detected failures

Integration points: WP2/WP4

» The simulator does not calculate the packages which have to be upgraded to satisfy user requests. It consider the ordered list of packages provided by WP4

» The upgrade plans calculated by the tools of WP4 have to be represented in terms of XML documents

```
<?xml version="1.0"?>
<sequence>
       <package name="swi-prolog">
         <action>install</action>
       </package>
       <package name="swi-prolog">
         <action>remove</action>
       </package>
</sequence>
```

mancoosi

Discussion

- We identified the following use cases related to a user's interaction with the packaging system:
  - **Install**: Starting with a package which is not installed on the system, the user asks the packaging system to install a package
  - **Remove**: Starting with a package that is currently installed on the system, the user asks the packaging system to remove it
  - **Purge**: Starting with the state when only configuration files remain on the system, the user asks the package system to remove even the configuration files
  - **Reinstall**: Starting with just the configuration files remaining on the system, the user asks the packaging system to install the package again (potentially newer version of the package)
  - **Upgrade**: Starting with a version of the package installed on the system, the user asks the packaging system to install a newer version of the package

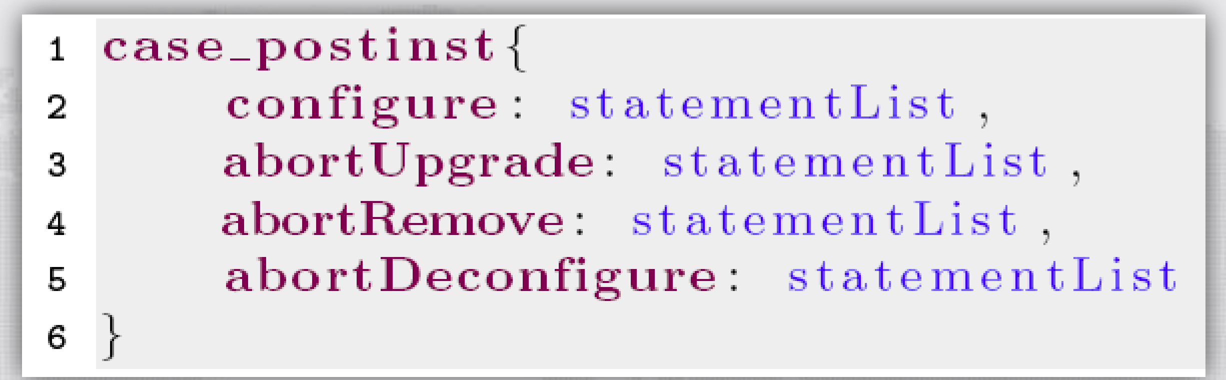
Discussion

- Each use case involves the execution of an activity diagram
- Each activity diagram is characterized by standard actions which identify the parameters to be passed to the maintainer scripts that are executed

```
#!/bin/sh
set -e
case "$1" in
    configure)
        # register gedit as a gnome-text-editor in the alternatives system
        update-alternatives \
            --install \
                /usr/bin/gnome-text-editor \
                gnome-text-editor \
                /usr/bin/gedit \
                50 \
            --slave \
                /usr/share/man/man1/gnome-text-editor.1.gz \
                gnome-text-editor.1.gz \
                /usr/share/man/man1/gedit.1.gz
    ;;
    abort-upgrade|abort-remove|abort-deconfigure)
    ;;
    *)
        echo "postinst called with unknown argument \`$1'" >&2
        exit 0
    ;;
esac
# Automatically added by dh_installmenu
if [ "$1" = "configure" ] && [ -x "`which update-menus 2>/dev/null`" ]; then
     update-menus
fi
# End automatically added section
```

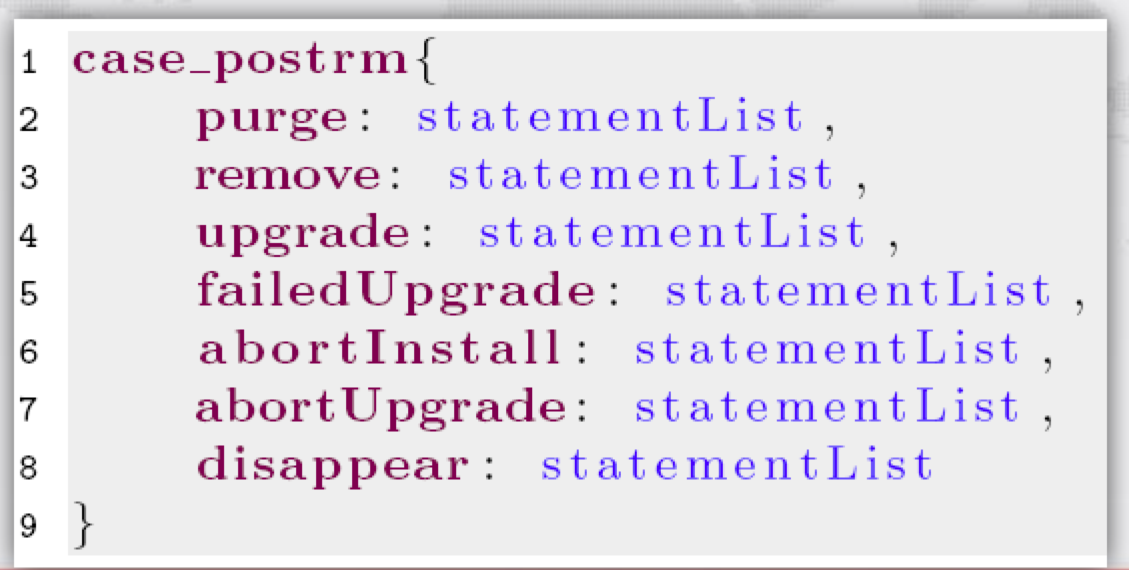
Discussion > DSL statements

» case_postinst



```
case_postinst{
      configure: statementList ,
      abortUpgrade: statementList ,
      abortRemove: statementList ,
      abortDeconfigure: statementList
}
```

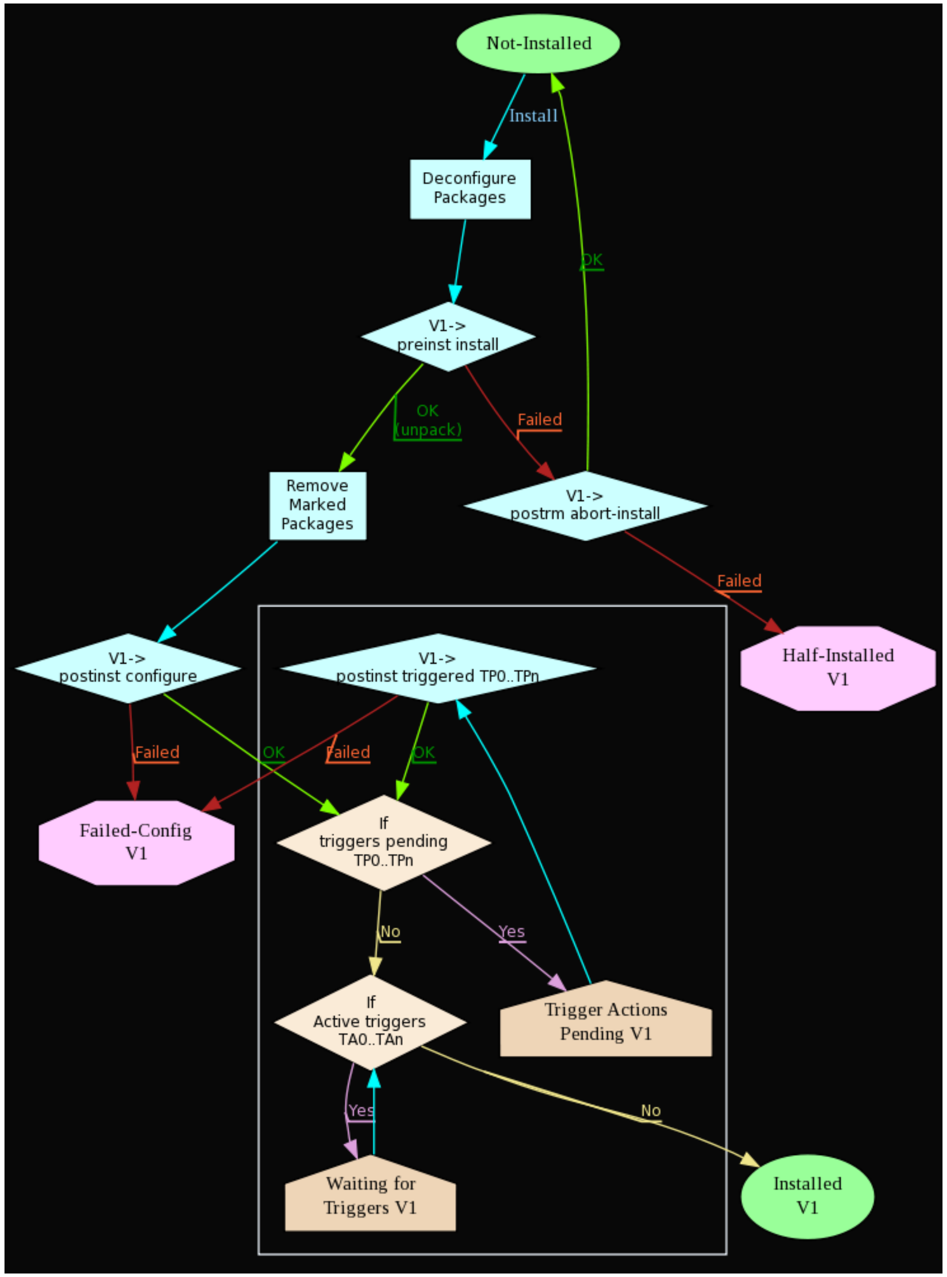
» case_portrm



```
case_postrm{
      purge: statementList ,
      remove: statementList ,
      upgrade: statementList ,
      failedUpgrade: statementList ,
      abortInstall: statementList ,
      abortUpgrade: statementList ,
      disappear: statementList
}
```

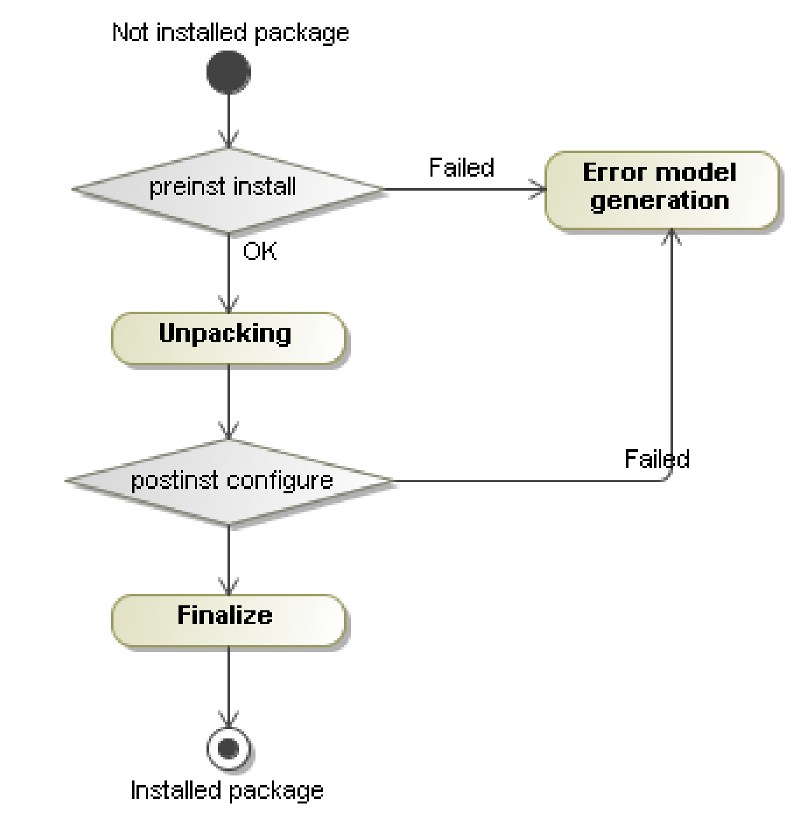
mancoosi

Package Installation use case



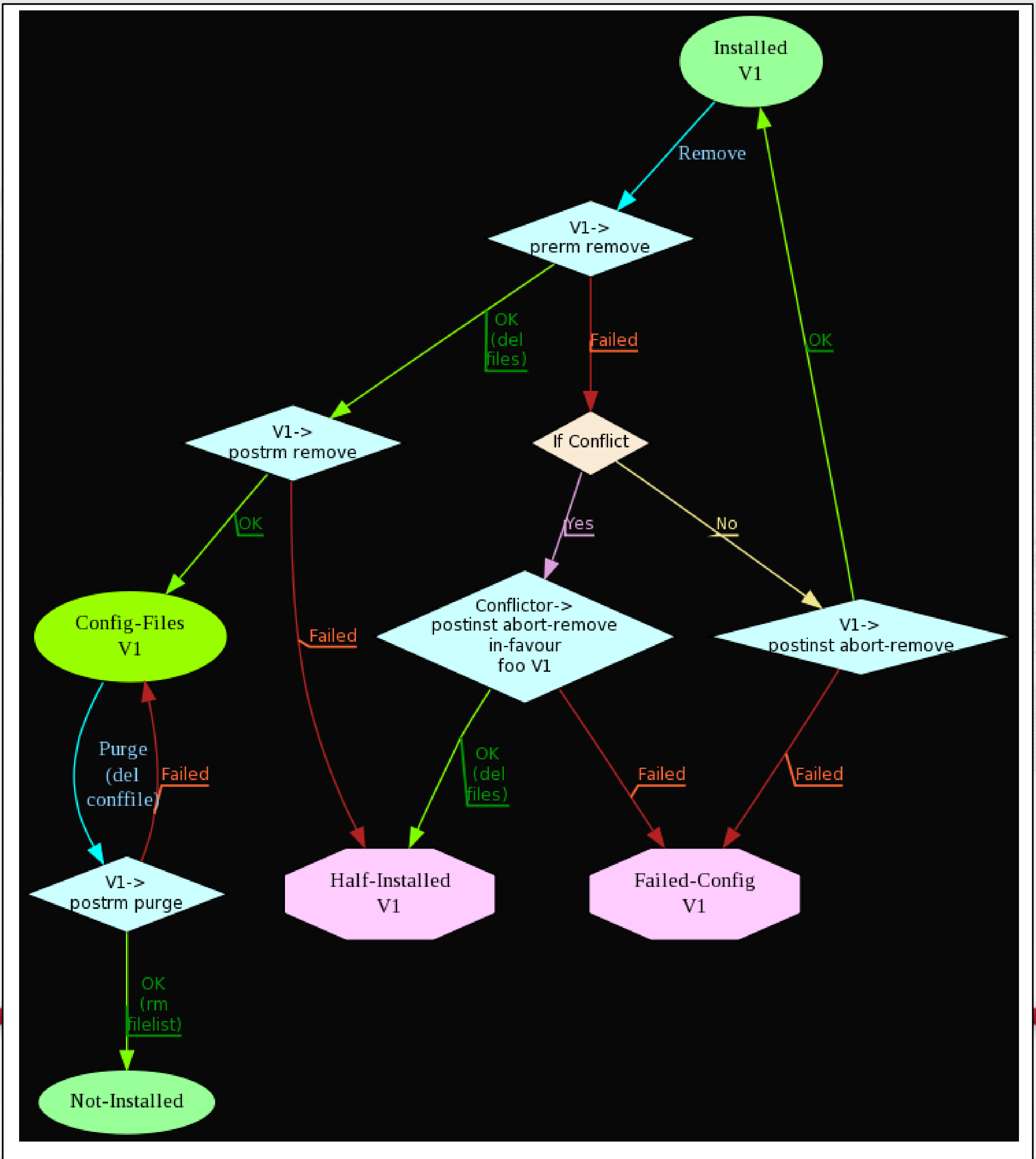
Real

[http://people.debian.org/~srivasta/MaintainerScripts.html]

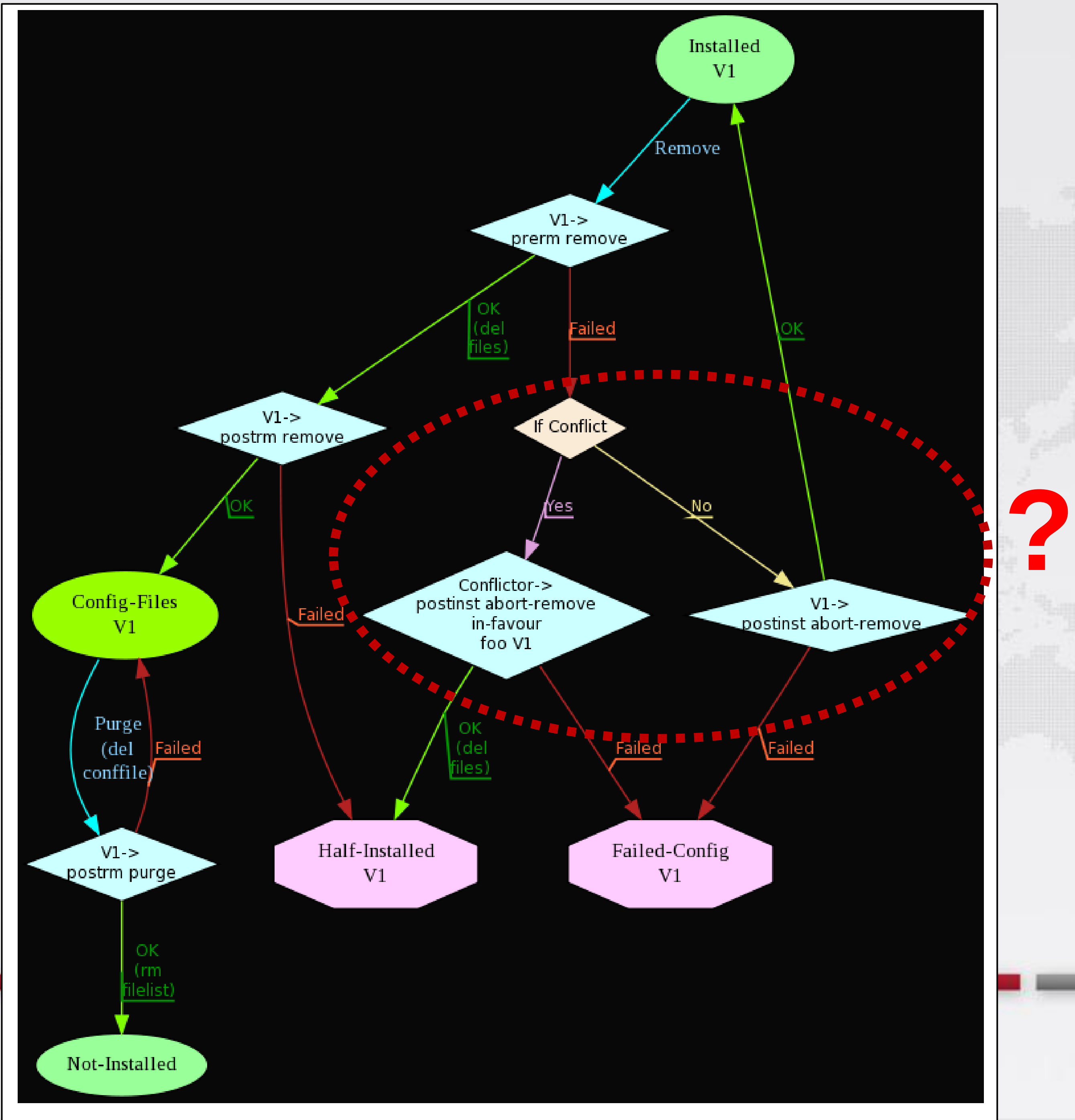


Simulated

Package Remove use case

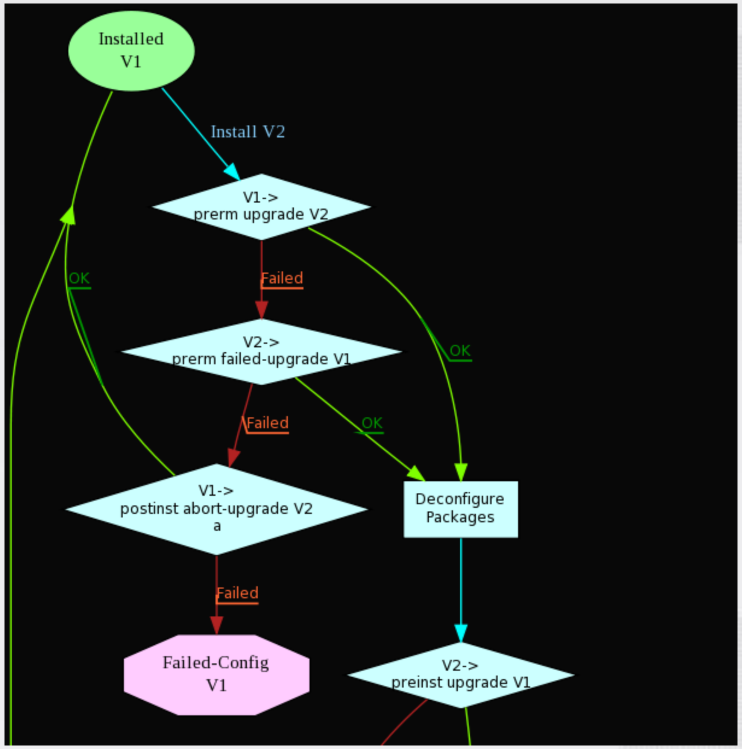


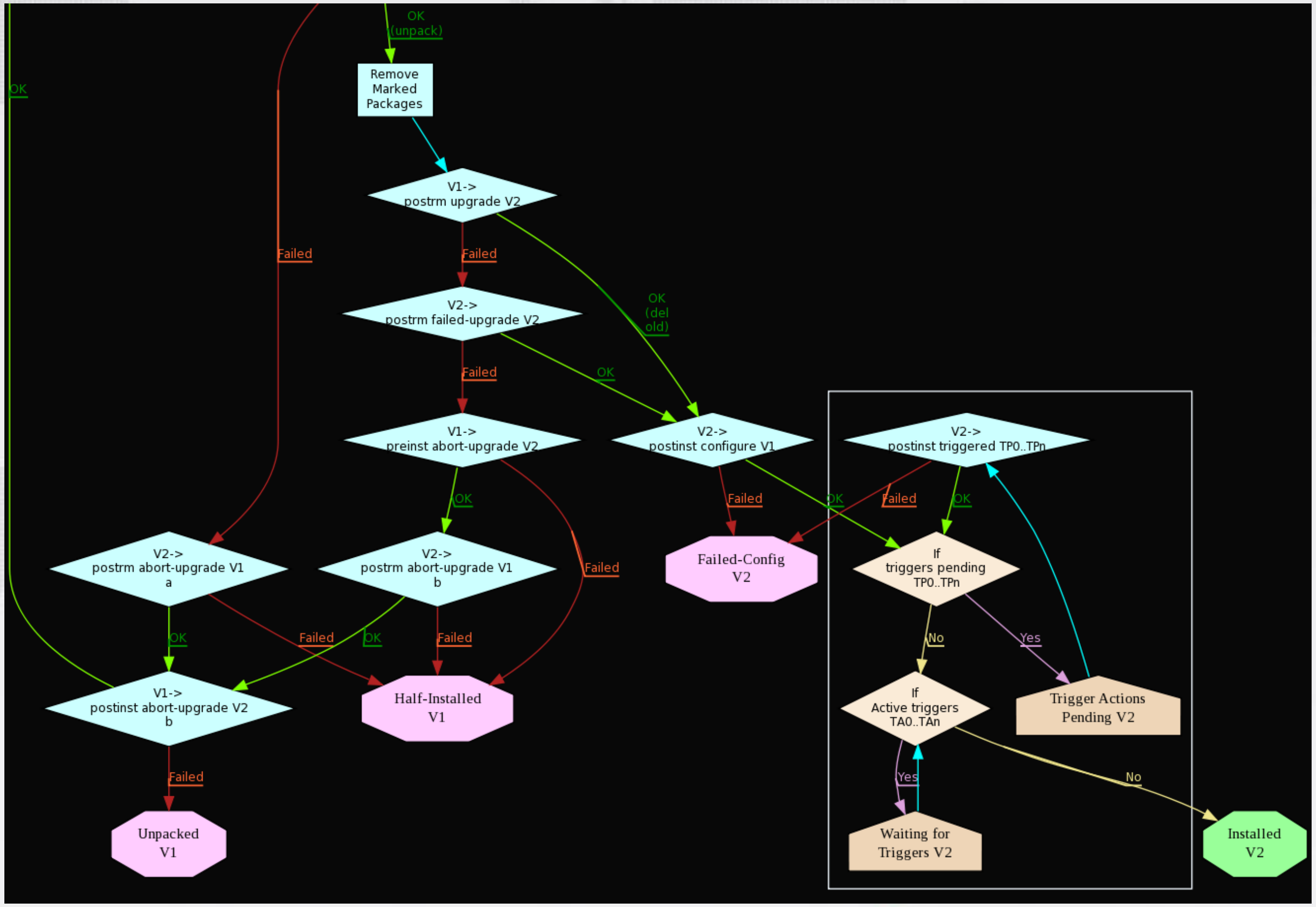
Package Remove use case



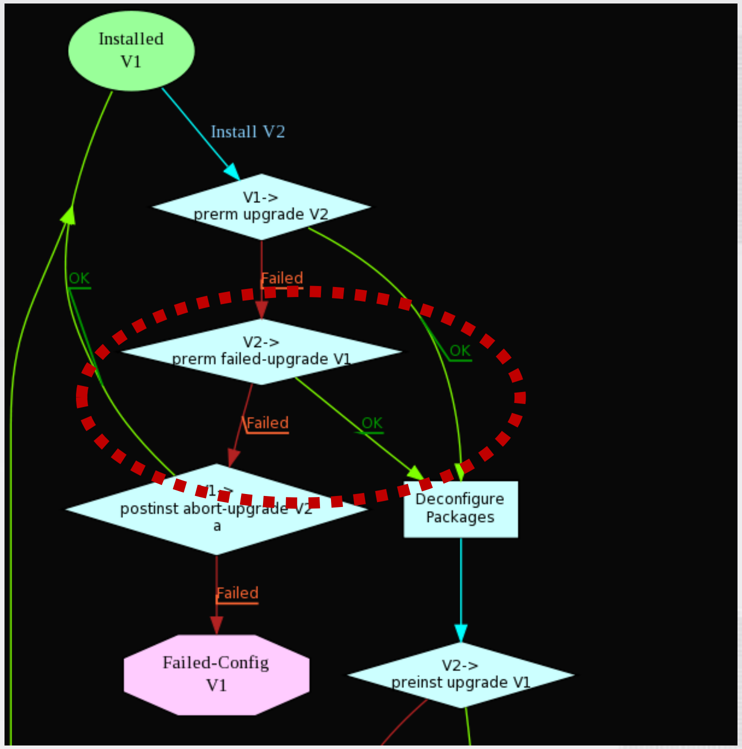
mancoosi

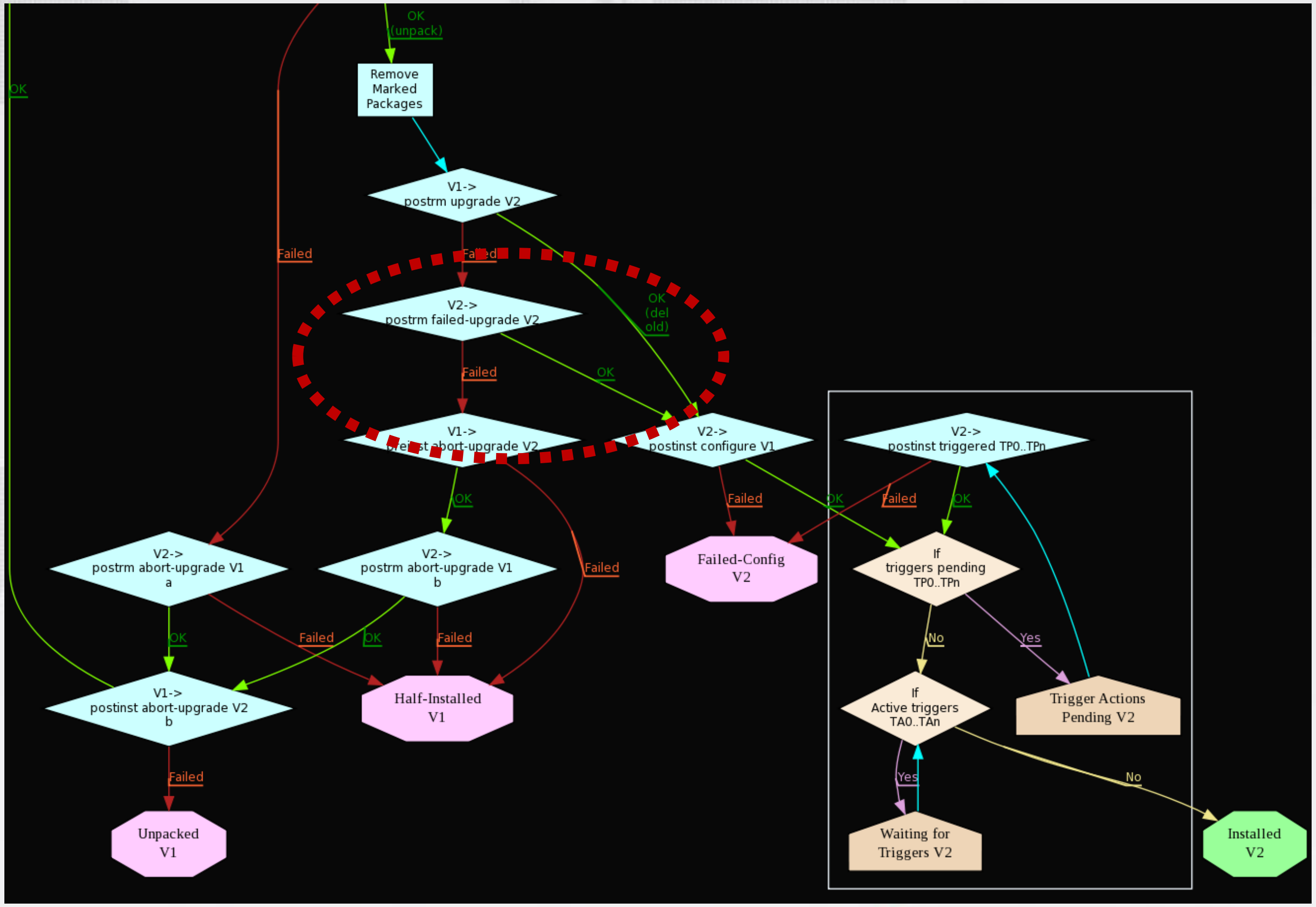
Package Upgrade use case




OK (unpack)
OK
Remove Marked Packages
V1-> postrm upgrade V2
Failed
Failed
V2-> postrm failed-upgrade V2
OK (del old)
Failed
OK
V1-> preinst abort-upgrade V2
V2-> postinst configure V1
V2-> postinst triggered TP0..TPn
OK
Failed
OK
Failed
Failed
OK
V2-> postrm abort-upgrade V1 a
V2-> postrm abort-upgrade V1 b
Failed-Config V2
If triggers pending TP0..TPn
OK
Failed
OK
Failed
No
Yes
V1-> postinst abort-upgrade V2 b
Half-Installed V1
If Active triggers TA0..TAn
Trigger Actions Pending V2
Failed
Yes
No
Unpacked V1
Waiting for Triggers V2
Installed V2

Package Upgrade use case




OK (unpack)
OK
Remove Marked Packages
V1-> postrm upgrade V2
Failed
Failed
OK (del old)
V2-> postrm failed-upgrade V2
Failed
OK
V1-> preinst abort-upgrade V2
V2-> postinst configure V1
V2-> postinst triggered TP0..TPn
OK
OK
Failed
Failed
OK
Failed
V2-> postrm abort-upgrade V1 a
V2-> postrm abort-upgrade V1 b
Failed-Config V2
If triggers pending TP0..TPn
OK
Failed
OK
Failed
No
Yes
V1-> postinst abort-upgrade V2 b
Half-Installed V1
If Active triggers TA0..TAn
Trigger Actions Pending V2
Failed
Yes
No
Unpacked V1
Waiting for Triggers V2
Installed V2

mancoosi

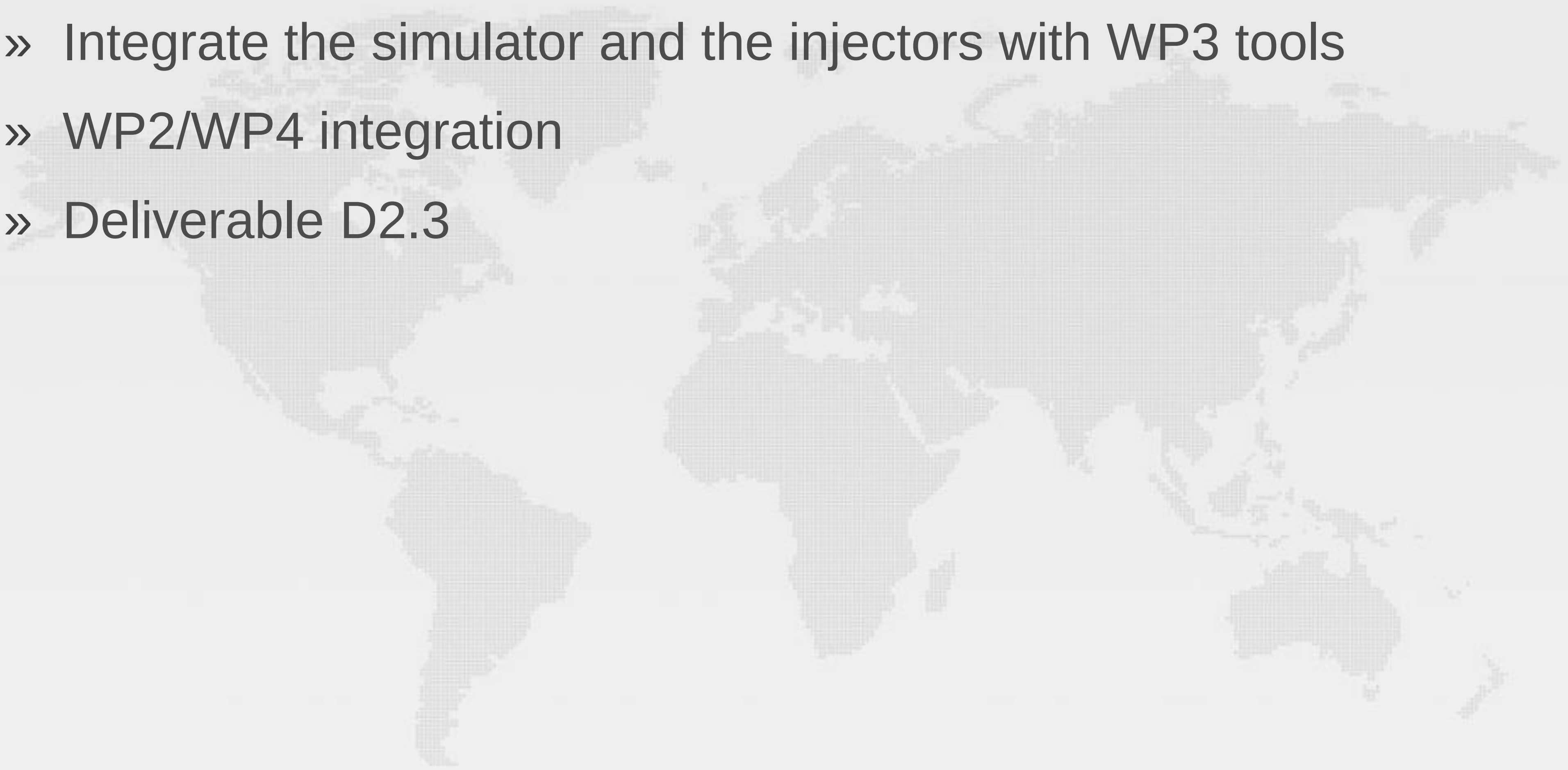
Open questions

» Currently the simulator manages two package states: *Not-installed*, and *Fully-installed*

- Are they enough for the simulation purposes ?
- What about the following states ?
  - ✓ *Config-Files state* : Only the configuration files of the package exist on the system
  - ✓ *Half-Installed state* : The installation of the package has been started, but not completed for some reason
  - ✓ *Unpacked state* : The package is unpacked, but not configured
  - ✓ *Half-Configured state* : The package is unpacked and configuration has been started, but not yet completed for some reason
  - ✓ *Triggers-Awaited state* : The package awaits trigger processing by another package
  - ✓ *Triggers-Pending state* : Another package has activated a trigger that this package had earlier expressed an interest in, and now some work has to be done

[http://people.debian.org/~srivasta/MaintainerScripts.html]

Next steps



- » Finalize the implementation of the simulator
- » Integrate the simulator and the injectors with WP3 tools
- » WP2/WP4 integration
- » Deliverable D2.3

References

R. Di Cosmo, D. Di Ruscio, P. Pelliccione, A. Pierantonio, S. Zacchiroli **Supporting Software Evolution in Component-Based FOSS Systems,** submitted for publication to Elsevier Science of Computer Programming