WP2: Modeling system configurations and packages

Davide Di Ruscio
Dipartimento di Informatica
Università degli Studi dell'Aquila

diruscio@di.univaq.it

Outline

- Objectives
- Scripts analysis
- MANCOOSI Metamodel(s)
  - Configuration
  - Package
  - Logs
- Example
- Conclusion and future plan

Objectives

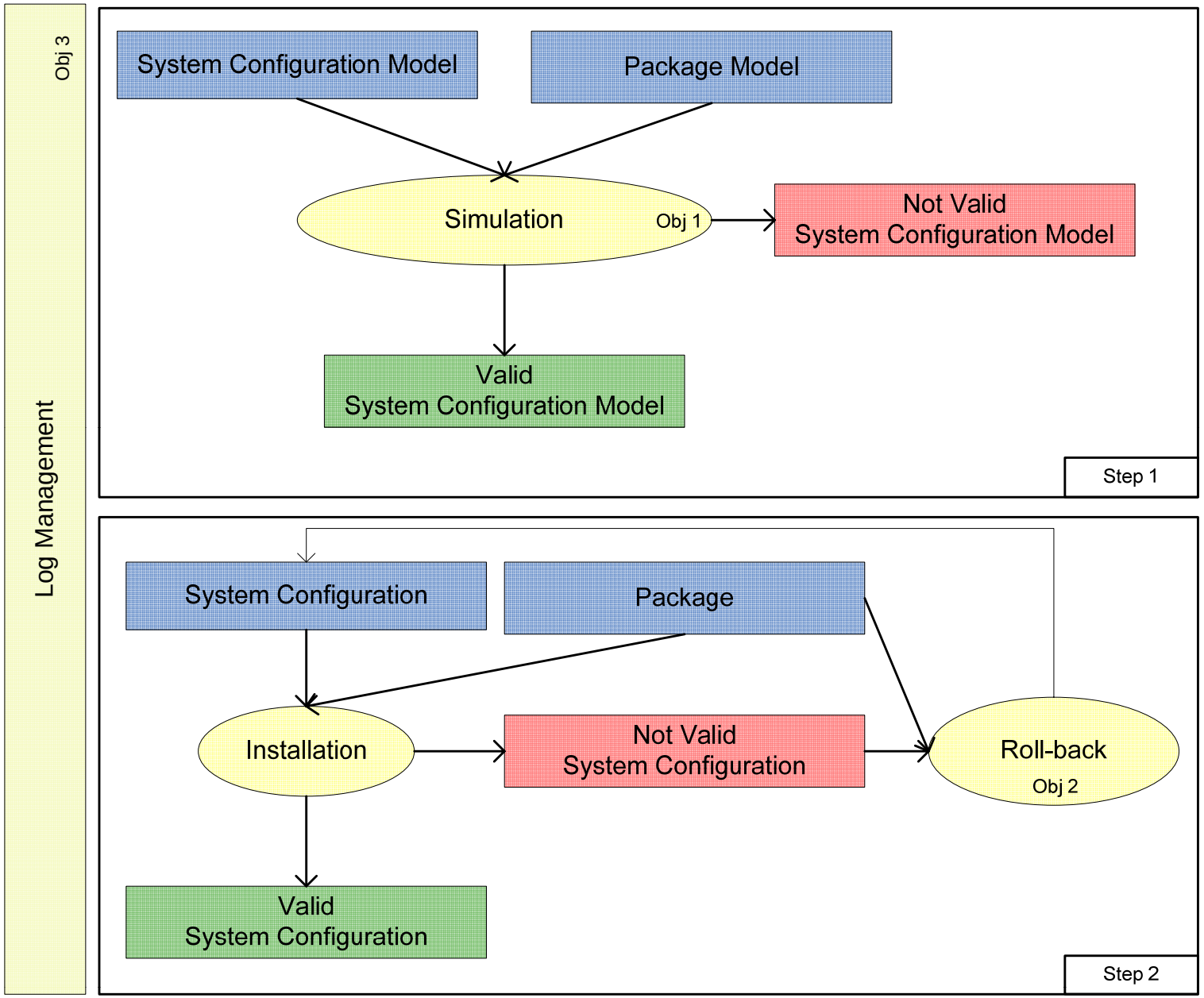
- Definition of metamodels to describe system configurations in order to support:
  - Simulation of package installations (obj 1)
  - Fault and roll-back management (obj 2)
  - Log management (obj 3)
- Metamodels must be defined to capture the relevant aspects of a system according to the DoW:

  "Since it is not possible to specify in detail every single part of systems, trade-offs between model completeness and usefulness must also be evaluated in order to have a good model that fits well with them"

Which are the concepts in the metamodel ?

- The definition of the metamodels is an intrinsically difficult task (and only the last stage of the analysis and engineering of the domain under observation)
- The metamodels should describe
  - Packages
    - Name, version, installed size, maintainer, architecture, package size, checksum, description, dependencies, conflicts, etc
  - Configurations
    - Installed packages, running services, hardware devices, etc
  - Logs
    - Information to keep the system and the corresponding model synchronized

Objectives

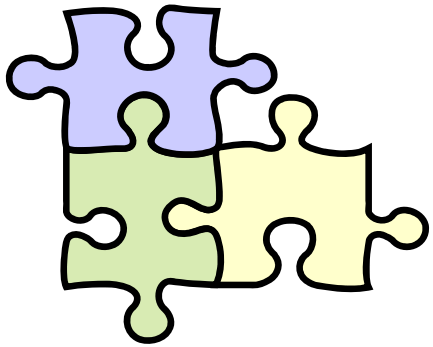

Obj 3
Log Management
System Configuration Model
Package Model
Simulation
Obj 1
Not Valid
System Configuration Model
Valid
System Configuration Model
Step 1
System Configuration
Package
Installation
Not Valid
System Configuration
Roll-back
Obj 2
Valid
System Configuration
Step 2

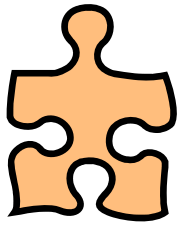
Problem : Script behavior

- A useful simulation has to consider the effects of package scripts, how to describe them ?
- Scripts should be given in terms of models which *(a)* on one hand abstract from the real system and *(b)* on the other hand must be enough expressive to predict the effects
- Therefore, the problem is to define:
  - Which kind of information must be represented ?
  - How this information can be (conveniently) elicited ?

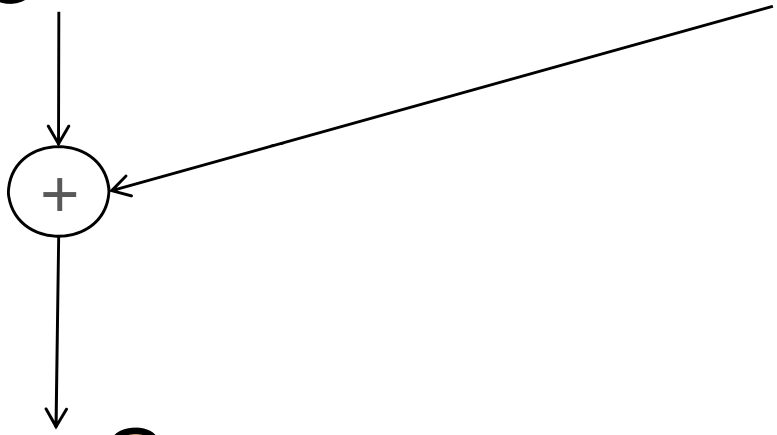
Sample Scenario

Configuration 1

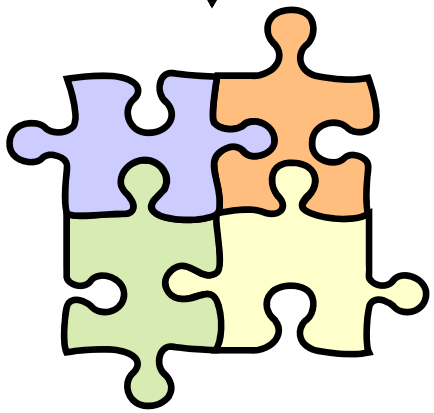




package
libapache-mod-ssl

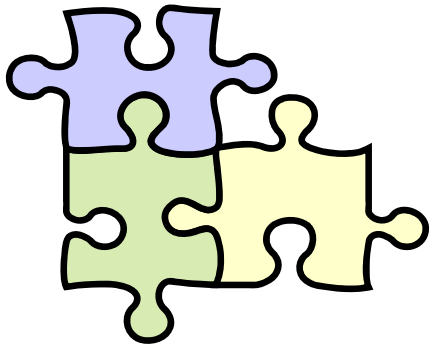


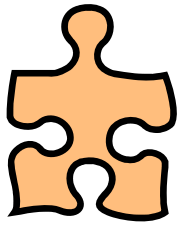
Configuration 2



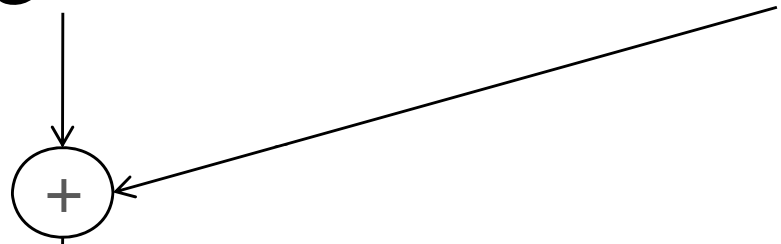
Sample Scenario

Configuration 1



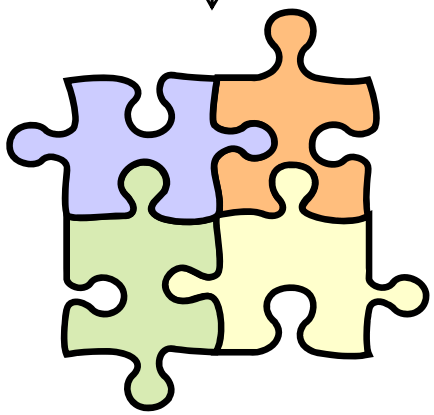


package
libapache-mod-ssl



```
apt-get install libapache-mod-ssl
```

Configuration 2



Sample Scenario


```
Package: libapache-mod-ssl
Version: 2.8.22-1sarge1
Section: web
Priority: optional
Architecture: amd64
Depends: libc6 (>= 2.3.2.ds1-21), libdb4.2, libexpat1 (>= 1.95.8), libssl0.9.7, openssl, apache-
common (>= 1.3.33-1), apache-common (<< 1.3.34)
Suggests: ca-certificates, libapache-mod-ssl-doc, apache | apache-perl
Installed-Size: 724
Maintainer: Domenico Andreoli <cavok@debian.org>
Description: Strong cryptography (HTTPS support) for Apache
 This Apache module provides strong cryptography for the Apache 1.3 webserver
 via the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS
 v1) protocols.
 .
   o  Open-Source software (BSD-style license)
   o  Usable for both commercial and non-commercial use
   o  Available for both Unix and Win32 platforms
   o  128-bit strong cryptography world-wide
   o  Support for SSLv2, SSLv3 and TLSv1 protocols
   o  Clean reviewable ANSI C source code
   o  Clean Apache module architecture
   o  Integrates seamlessly into Apache through an Extended API (EAPI)
   o  Full Dynamic Shared Object (DSO) support
   o  Support for the OpenSSL+RSAref US-situation
   o  Advanced pass-phrase handling for private keys
   o  X.509 certificate based authentication for both client and server
   o  Support for per-URL renegotiation of SSL handshake parameters
   o  Support for explicit seeding of the PRNG from external sources
```

Sample Scenario


```
Package: libapache-mod-ssl
Version: 2.8.22-1sarge1
Section: web
Priority: optional
Architecture: amd64
Depends: libc6 (>= 2.3.2.ds1-21), libdb4.2, libexpat1 (>= 1.95.8), libssl0.9.7, openssl, apache-
common (>= 1.3.33-1), apache-common (<< 1.3.34)
Suggests: ca-certificates, libapache-mod-ssl-doc, apache | apache-perl
Installed-Size: 724
Maintainer: Domenico Andreoli <cavok@debian.org>
Description: Strong cryptography (HTTPS support) for Apache
 This Apache module provides strong cryptography for the Apache 1.3 webserver
 via the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS
 v1) protocols.
 .
   o  Open-Source software (BSD-style license)
   o  Usable for both commercial and non-commercial use
   o  Available for both Unix and Win32 platforms
   o  128-bit strong cryptography world-wide
   o  Support for SSLv2, SSLv3 and TLSv1 protocols
   o  Clean reviewable ANSI C source code
   o  Clean Apache module architecture
   o  Integrates seamlessly into Apache through an Extended API (EAPI)
   o  Full Dynamic Shared Object (DSO) support
   o  Support for the OpenSSL+RSAref US-situation
   o  Advanced pass-phrase handling for private keys
   o  X.509 certificate based authentication for both client and server
   o  Support for per-URL renegotiation of SSL handshake parameters
   o  Support for explicit seeding of the PRNG from external sources
```

Sample Scenario

libapache-mod-ssl_2...1_amd64.deb.postinst

```
sslcrtdir=/etc/apache/ssl.crt
supportdir=/usr/share/apache/mod_ssl

umask 022

# we do not want to bother the user with useless questions at install time
if [ "$1" = "reconfigure" -o "${DEBCONF_RECONFIGURE}" = "1" ]; then
        mod-ssl-makecert
fi

if [ "$1" = "configure" -o "$1" = "reconfigure" -o "${DEBCONF_RECONFIGURE}" = "1" ]; then
        if [ -d ${sslcrtdir} ]; then
                cd ${sslcrtdir}
                . ${supportdir}/functions.sh
                modssl_update_hashes
        fi
fi

if [ "$1" = "configure" ]; then
        # apache-ssl does not need us as DSO
        for i in apache apache-perl; do
                if [ -e "/etc/$i/httpd.conf" ]; then
                        apache-modconf "$i" enable mod_ssl
                fi
        done
fi
```

Ln 4, Col 37 INS

Sample Scenario

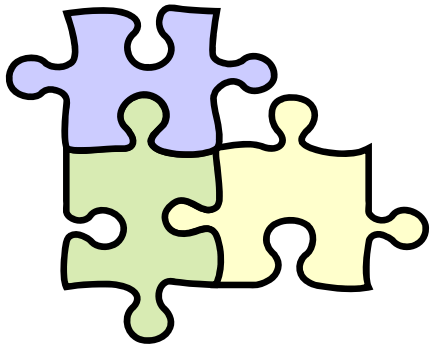
libapache-mod-ssl_2...1_amd64.deb.postinst

```
sslcrtdir=/etc/apache/ssl.crt
supportdir=/usr/share/apache/mod_ssl

umask 022

# we do not want to bother the user with useless questions at install time
if [ "$1" = "reconfigure" -o "${DEBCONF_RECONFIGURE}" = "1" ]; then
        mod-ssl-makecert
fi

if [ "$1" = "configure" -o "$1" = "reconfigure" -o "${DEBCONF_RECONFIGURE}" = "1" ]; then
        if [ -d ${sslcrtdir} ]; then
                cd ${sslcrtdir}
                . ${supportdir}/functions.sh
                modssl_update_hashes
        fi
fi

if [ "$1" = "configure" ]; then
        # apache-ssl does not need us as DSO
        for i in apache apache-perl; do
                if [ -e "/etc/$i/httpd.conf" ]; then
                        apache-modconf "$i" enable mod_ssl
                fi
        done
fi
```

Ln 4, Col 37 INS

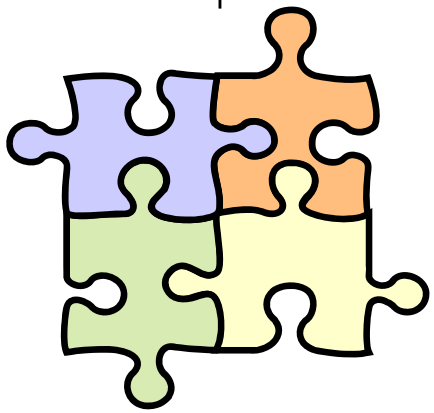
Sample Scenario

Configuration 1




```
apt-get remove libapache-mod-ssl
```

Configuration 2



Sample Scenario

libapache-mod-ssl_2...rge1_amd64.deb.prerm

```
#! /bin/sh

sslcrtdir=/etc/apache/ssl.crt
supportdir=/usr/share/apache/mod_ssl

umask 022

if [ "$1" = "upgrade" -o "$1" = "remove" ]; then
        if [ -d ${sslcrtdir} ]; then
                cd ${sslcrtdir}
                . ${supportdir}/functions.sh
                modssl_clean_hashes
        fi
fi

if [ "$1" = "remove" ]; then
        # there is no need to remove the entries for apache-ssl
        for i in apache apache-perl; do
                if [ -e "/etc/$i/httpd.conf" ]; then
                        apache-modconf "$i" disable mod_ssl quiet
                fi
        done
fi
```

Ln 21, Col 19 INS

Sample Scenario

libapache-mod-ssl_2...rge1_amd64.deb.prerm

```
#! /bin/sh

sslcrtdir=/etc/apache/ssl.crt
supportdir=/usr/share/apache/mod_ssl

umask 022

if [ "$1" = "upgrade" -o "$1" = "remove" ]; then
        if [ -d ${sslcrtdir} ]; then
                cd ${sslcrtdir}
                . ${supportdir}/functions.sh
                modssl_clean_hashes
        fi
fi

if [ "$1" = "remove" ]; then
        # there is no need to remove the entries for apache-ssl
        for i in apache apache-perl; do
                if [ -e "/etc/$i/httpd.conf" ]; then
                        apache-modconf "$i" disable mod_ssl quiet
                fi
        done
fi
```

Ln 21, Col 19 INS

What happen if the maintainer does not write this statement ?

Sample Scenario

libapache-mod-ssl_2...rge1_amd64.deb.prerm

```
#! /bin/sh

sslcrtdir=/etc/apache/ssl.crt
supportdir=/usr/share/apache/mod_ssl

umask 022

if [ "$1" = "upgrade" -o "$1" = "remove" ]; then
        if [ -d ${sslcrtdir} ]; then
                cd ${sslcrtdir}
                . ${supportdir}/functions.sh
                modssl_clean_hashes
        fi
fi

if [ "$1" = "remove" ]; then
        # there is no need to remove the entries for apache-ssl
        for i in apache apache-perl; do
                if [ -e "/etc/$i/httpd.conf" ]; then
                        apache-modconf "$i" disable mod_ssl quiet
                fi
        done
fi
```

Ln 21, Col 19 INS

An “inconsistent” configuration is reached and it can be noticed only at run-time since the package dependencies are not enough

A new model-based scripting language

- The dependencies among packages are not enough to discover unsatisfactory situations which currently can be noticed only at run-time
  - Inconsistency management can be considered as a strategy to detect fallacious situations at run-time since just observing the behavior of the single package management could be not enough
- Currently, it is not clear from the user point of view which services will be available after having installed or removed a given package
- Roll-back operations require more information than package dependencies

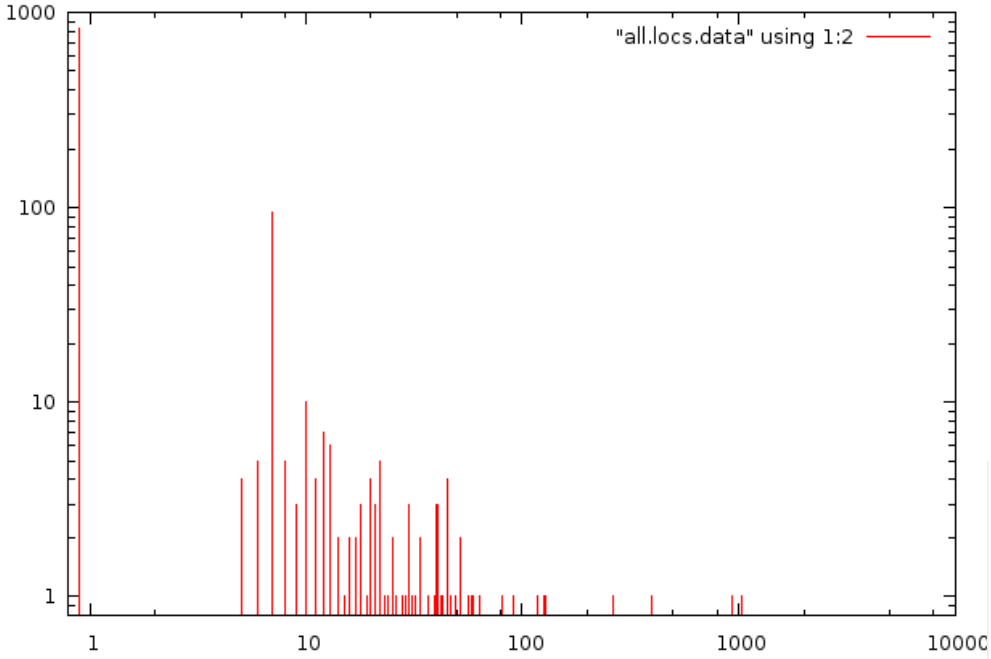
A new model-based scripting language

- A brand new model-based language for specifying the system configurations and packages (maintainer scripts included)
- Transitions are model transformations which let a configuration C1 evolve into a configuration C2
- Having specified the behavior of the scripts enables a more realistic simulation

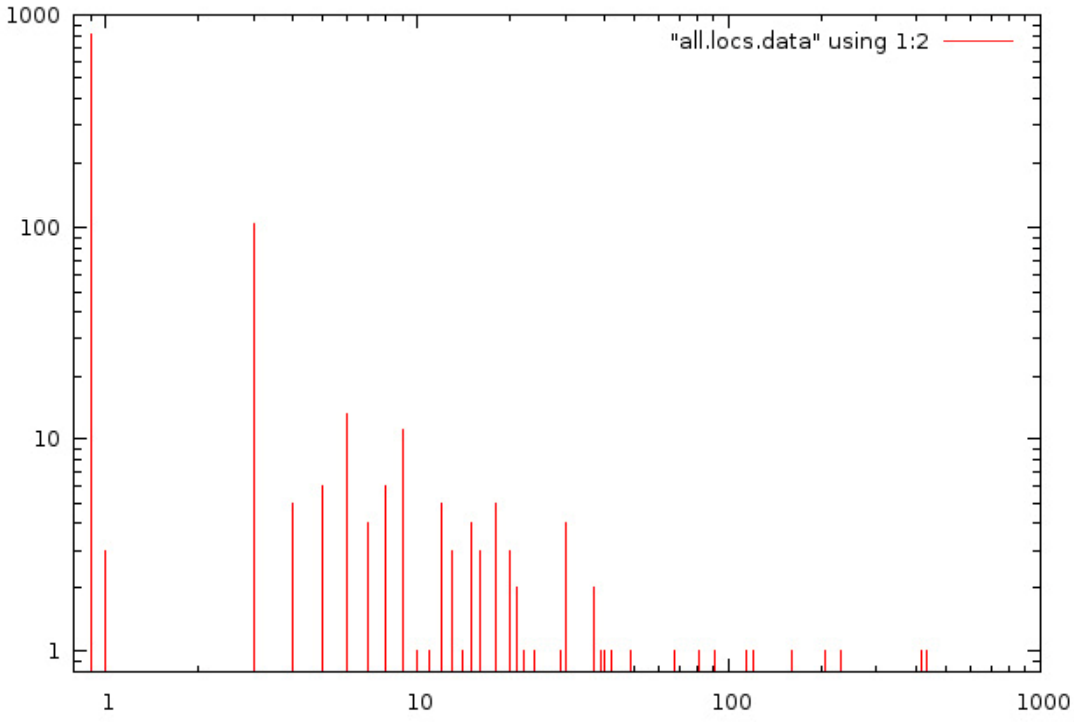
Domain analysis

- We considered all the packages of the *debian distribution* for the *amd64 architecture* (2 dvds, ~ 6 GB)
- The lenght of the maintainer scripts have been calculated and graphically reported, in particular we compared the “complexity” of scripts before and after a pruning operation
  - Due to space limitation we present the results of *all*, *preinst*, and *postrm* maintainer scripts

Domain analysis : all maintainer scripts

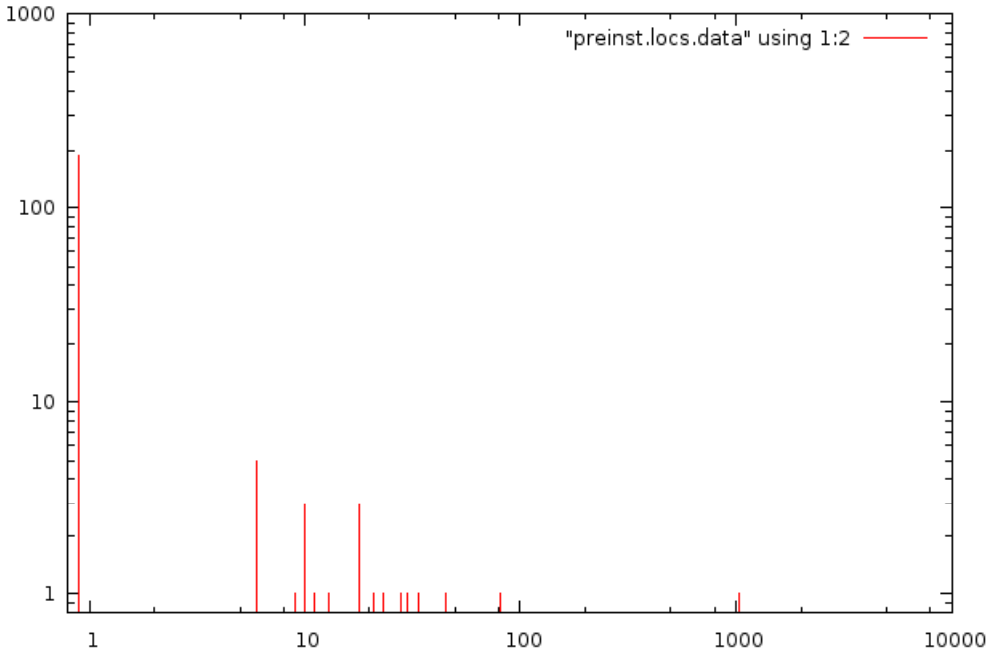


Original scripts

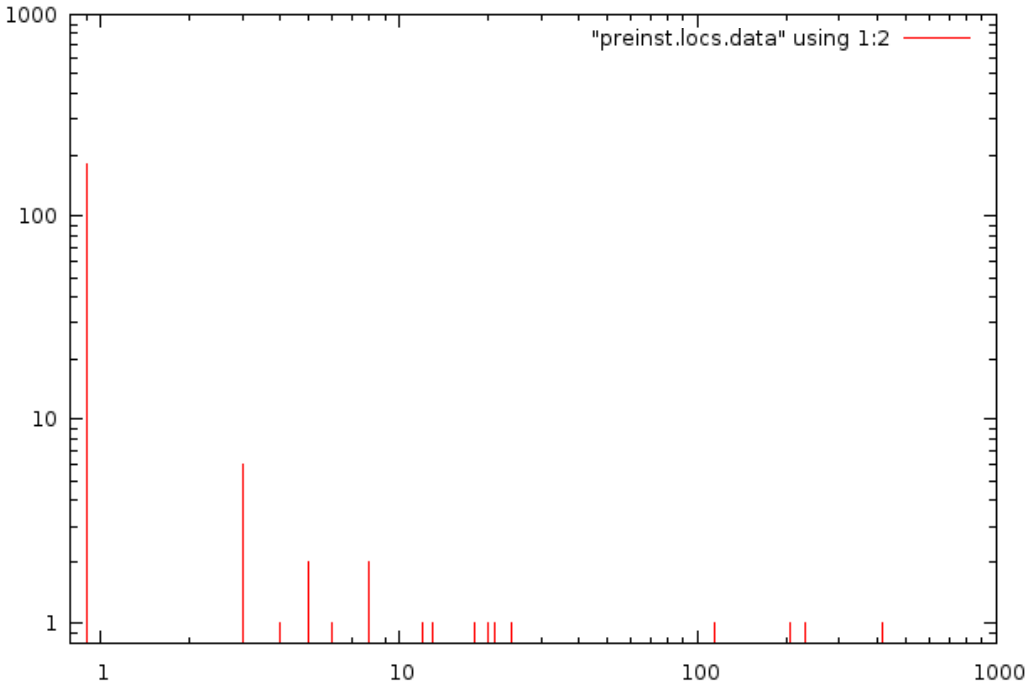


Pruned scripts

Domain analysis : preinst scripts

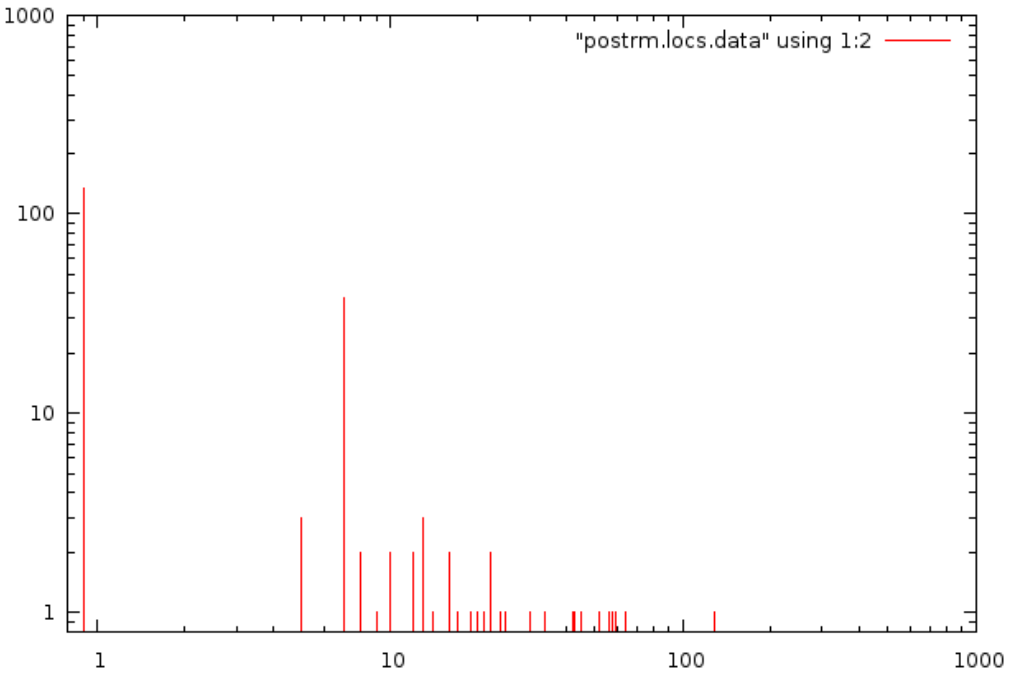


Original scripts

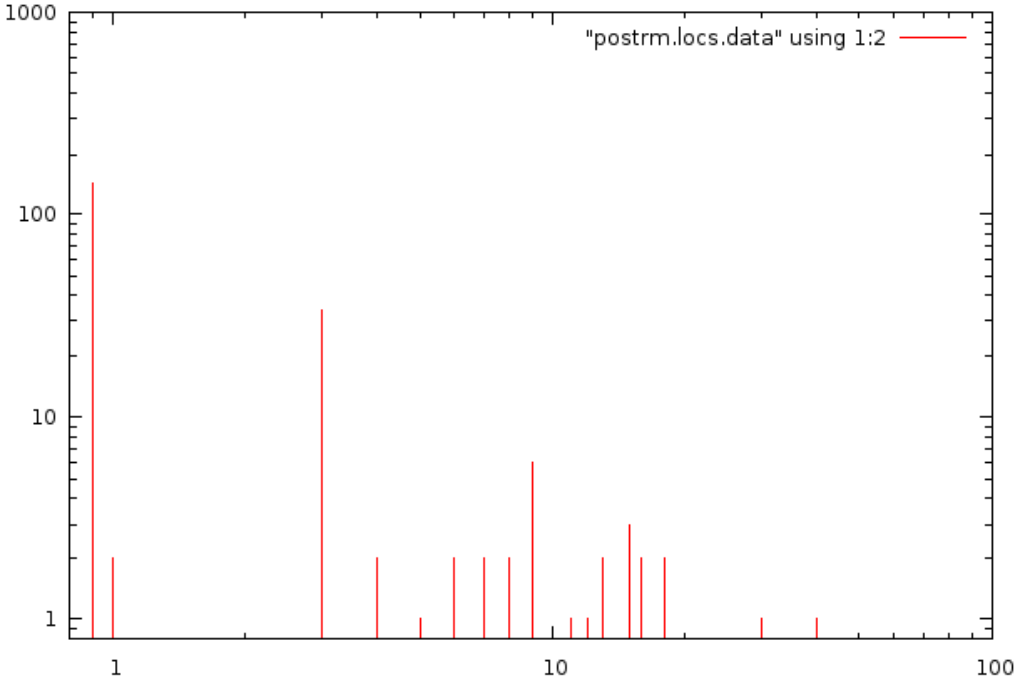


Pruned scripts

Domain analysis : postrm scripts



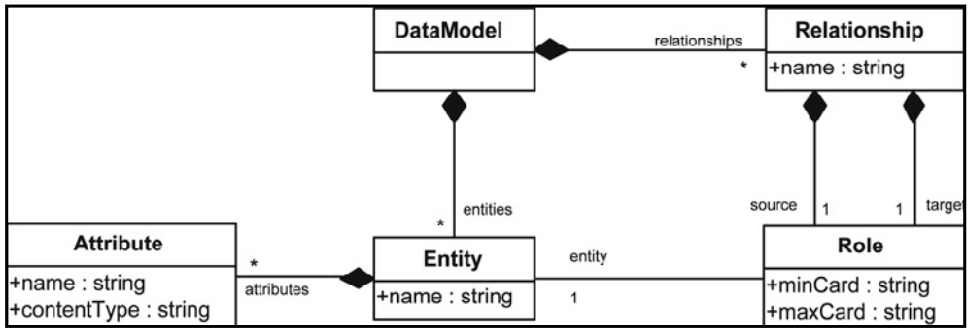
Original scripts



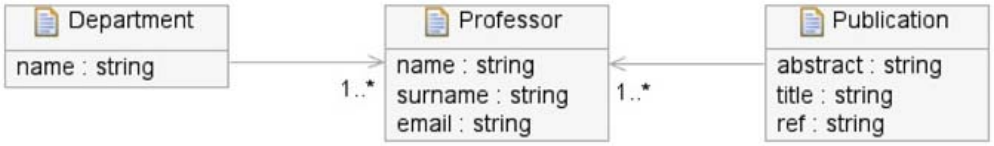
Pruned scripts

Metamodeling

- After analyzing the domain we identified the main concerns (packages, configurations, logs) which have been iteratively formalized with corresponding metamodels
- Metamodeling is a useful technique to formalize and engineer a specific domain by providing the syntactical means to express its concepts



E/R Metamodel



Sample E/R Model

Metamodeling

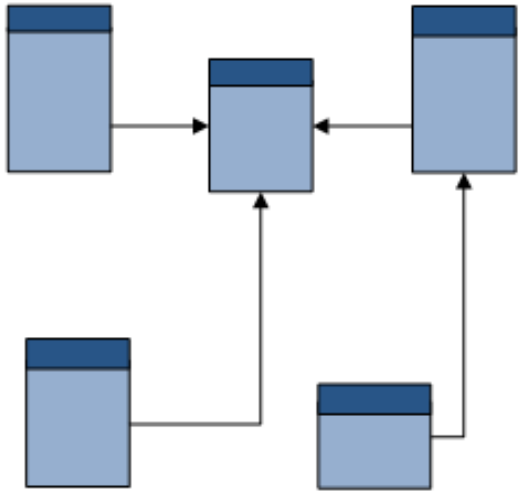
» Metamodels are models that make statements about modeling, more precisely a metamodel describes

- the possible structure of models
- the constructs of a modeling language and their relationships
- constraints and modeling rules

» A metamodel describes the **abstract syntax** and the static semantics of a modeling language, a suitable **concrete syntax** is the interface to the modeler depending on the purpose

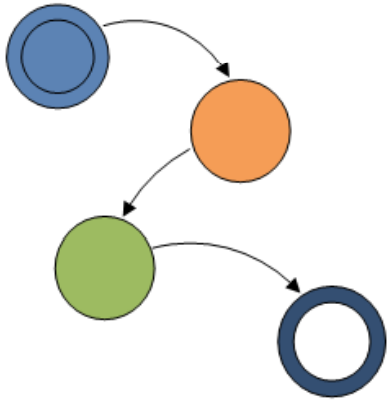
» The metamodel is the basis for the automated, tool-supported processing of models (which represents its implicit **semantics**)

Metamodeling



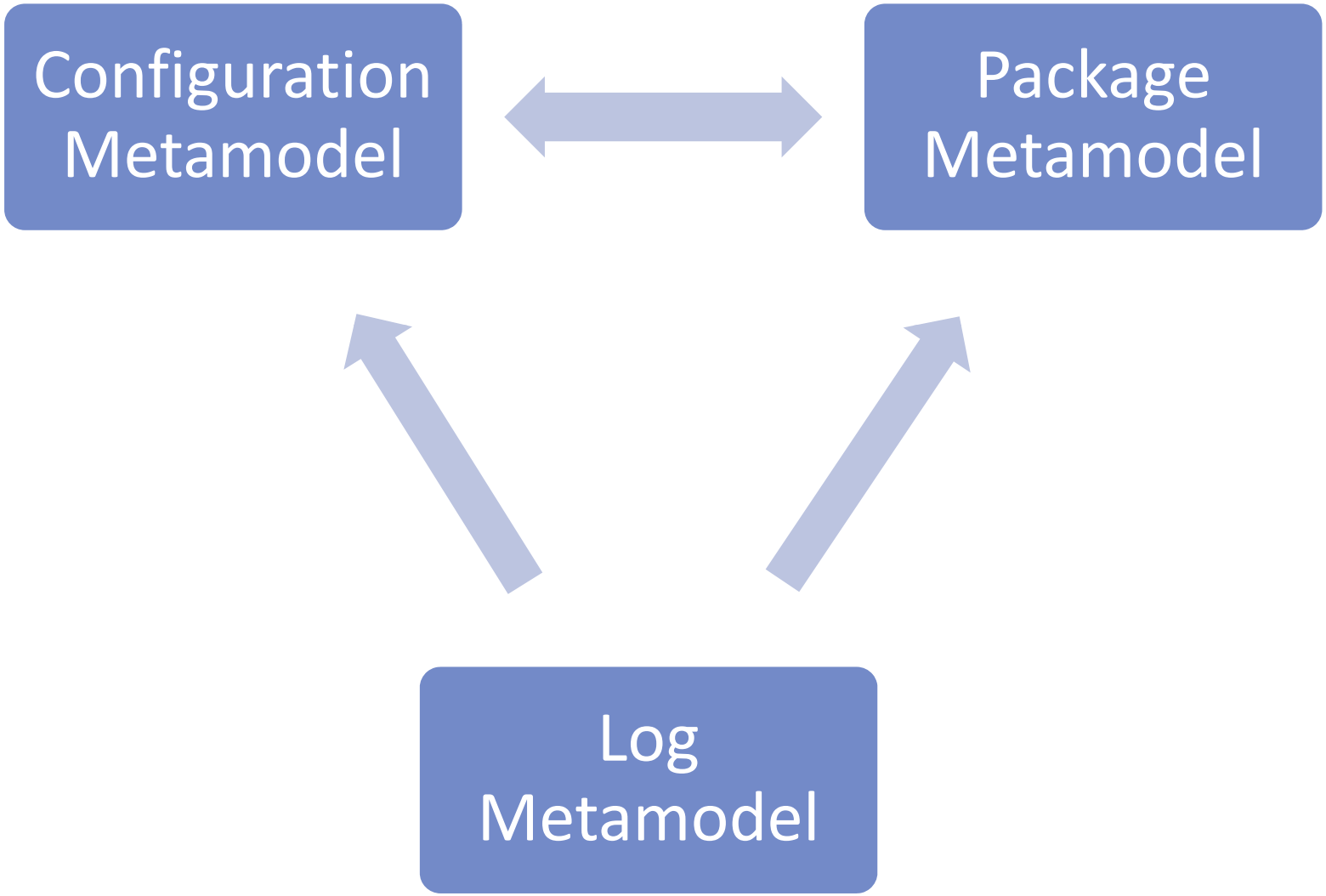
Sample Metamodel

These three may be three concrete syntaxes associated with the same Metamodel



```
process 1 {
        job x {}
        job y {}
        ...
}
```

MANCOOSI metamodels



MANCOOSI metamodels

» Configuration metamodel

- Installed packages
- Package settings
- Environment (e.g. loaded modules, shared libraries)
- File systems
- Hardware devices

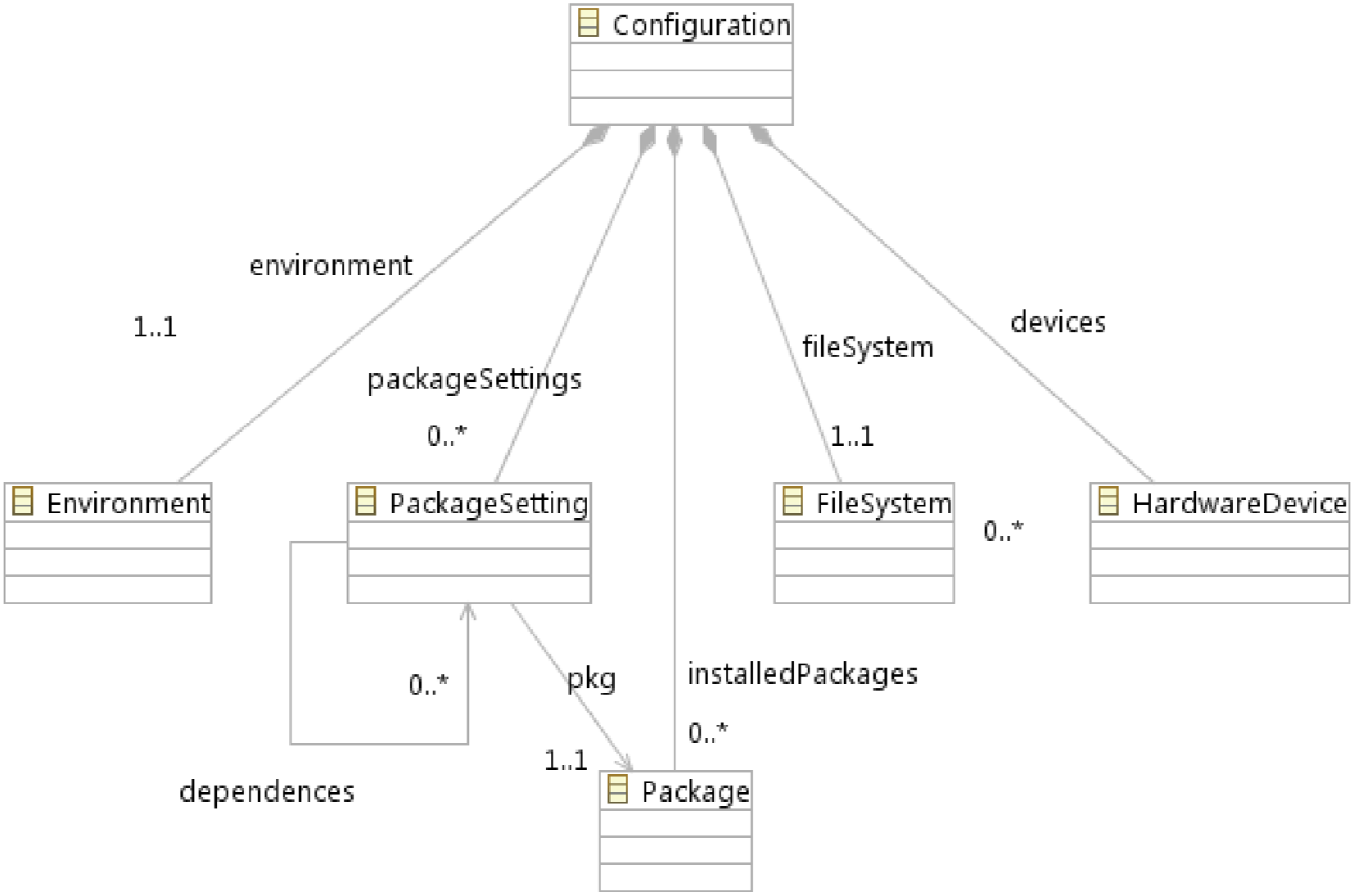
» Package metamodel

- Dependences, conflicts, etc.
- Behavior of Scripts

» Log metamodel

- Transactions (set of statements which change the configurations)

Configuration metamodel



Sample configuration

Configuration 1

-**Environment**

- running services: sendmail, apached, cron…

-**Filesystem**

-Files: /usr/share/…

-**Installed packages**

- apache2, sendmail, cron, …

-**Package settings**

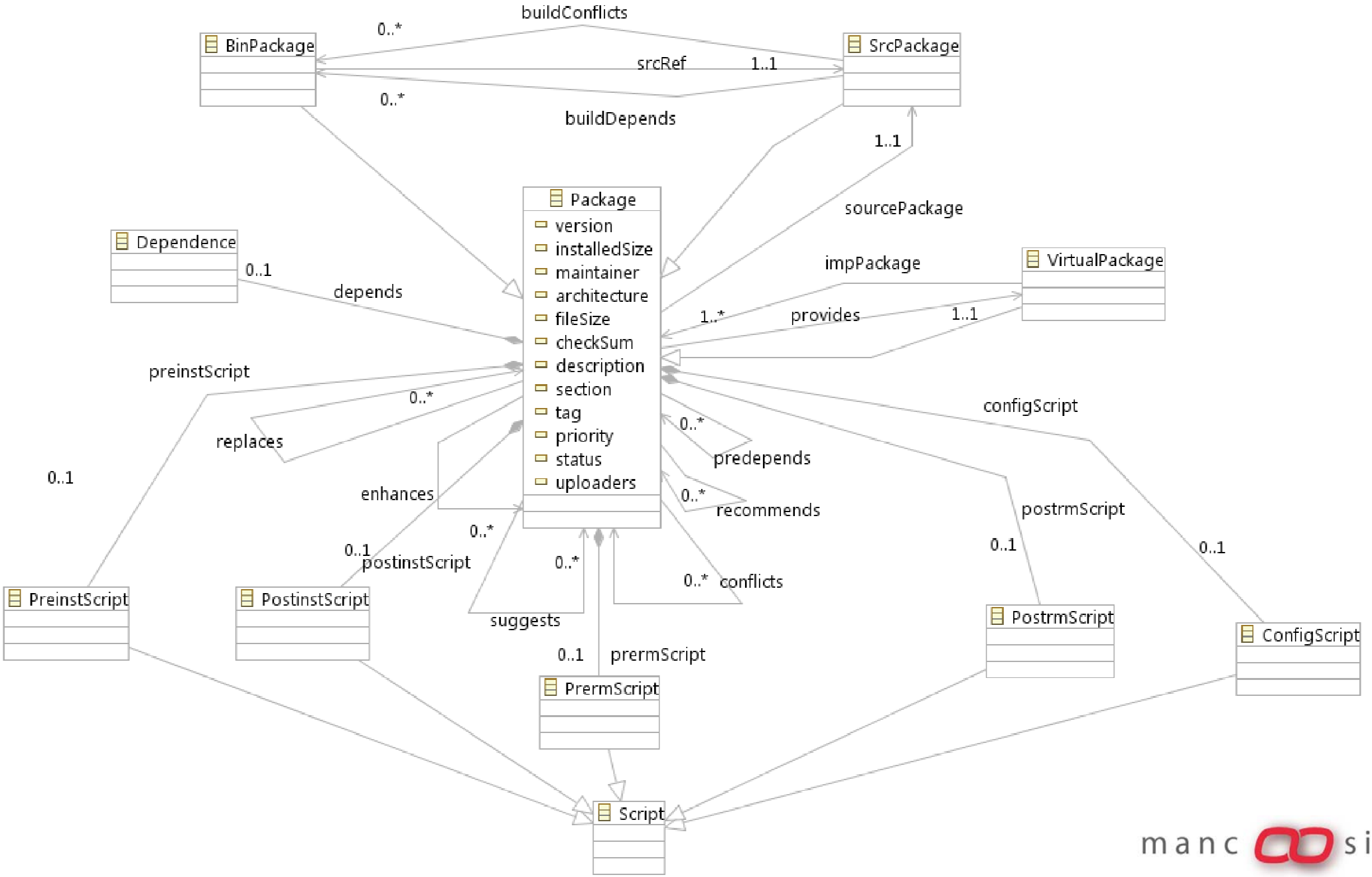
-apache2: the configuration files in /etc/apache2

-sendmail: the configuration files in /etc/mail

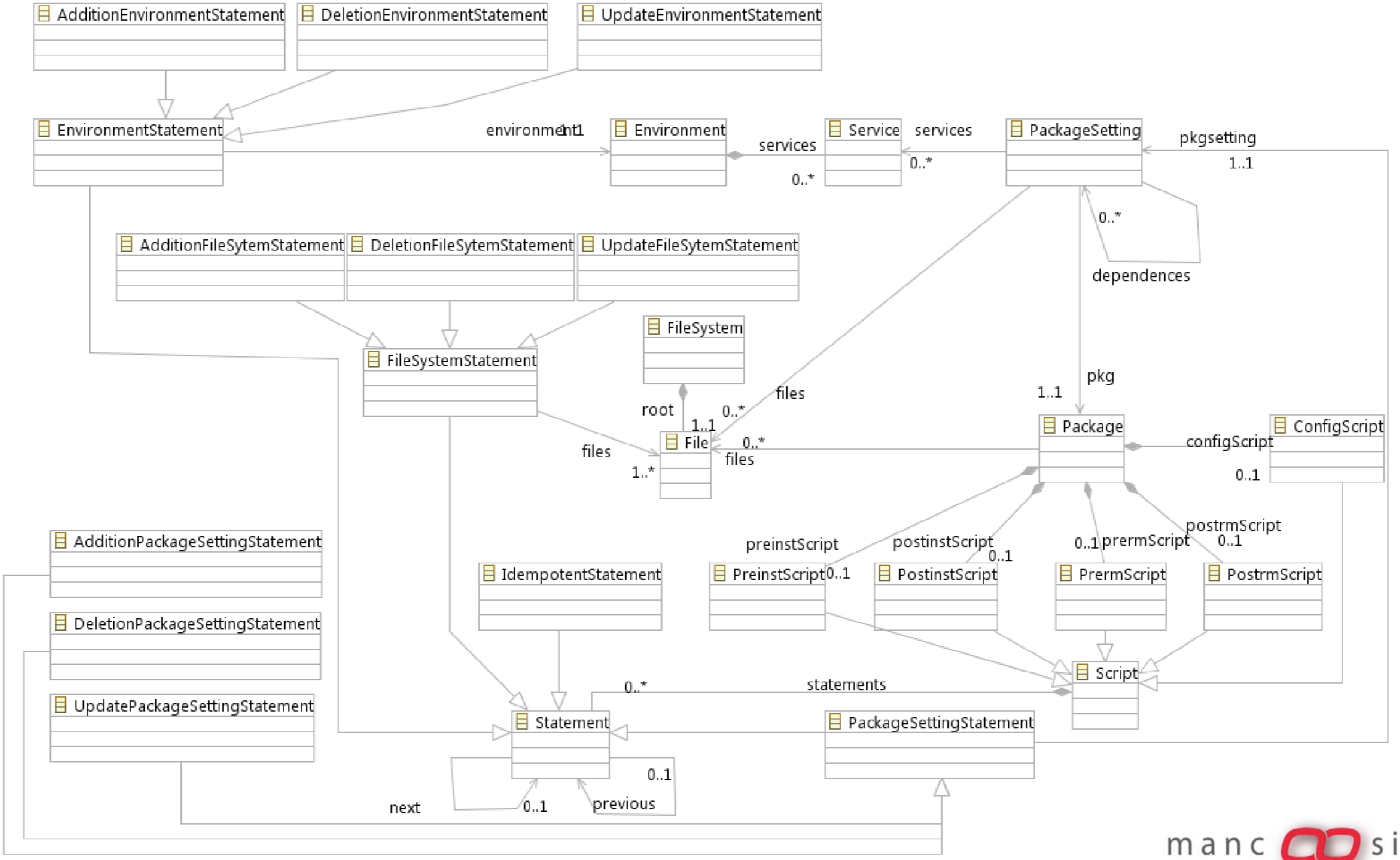
(it has a dependence with the package setting of cron)

-cron: the configurtion files in /etc/cron.*

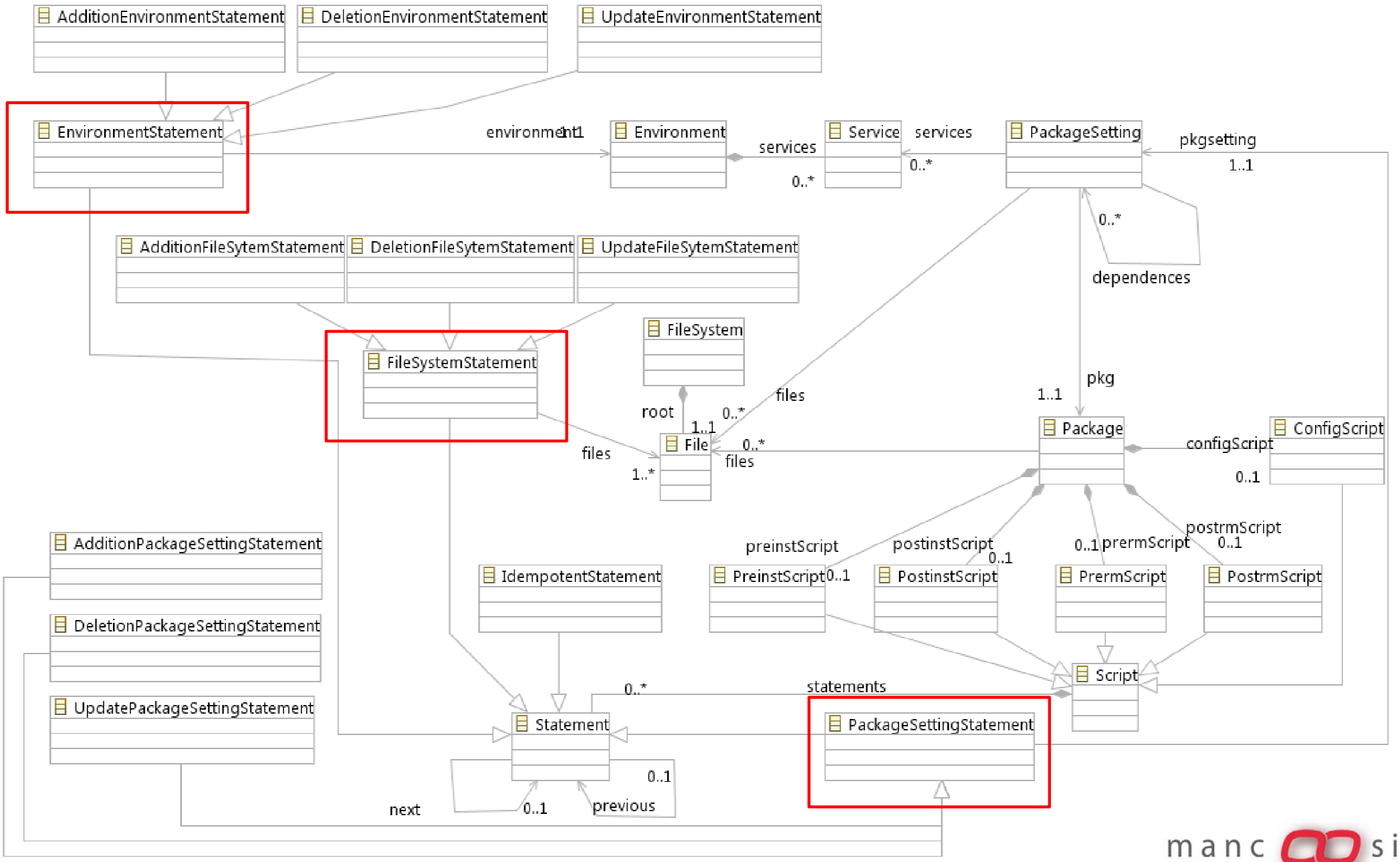
Package metamodel



Package metamodel : Script



Package metamodel : Script



Sample postinst script

libapache-mod-ssl_2...1_amd64.deb.postinst

```
if [ "$1" = "configure" ]; then
        # apache-ssl does not need us as DSO
        for i in apache apache-perl; do
                if [ -e "/etc/$i/httpd.conf" ]; then
                        apache-modconf "$i" enable mod_ssl
                fi
        done
fi
```

Ln 4, Col 37 INS

Sample postinst script

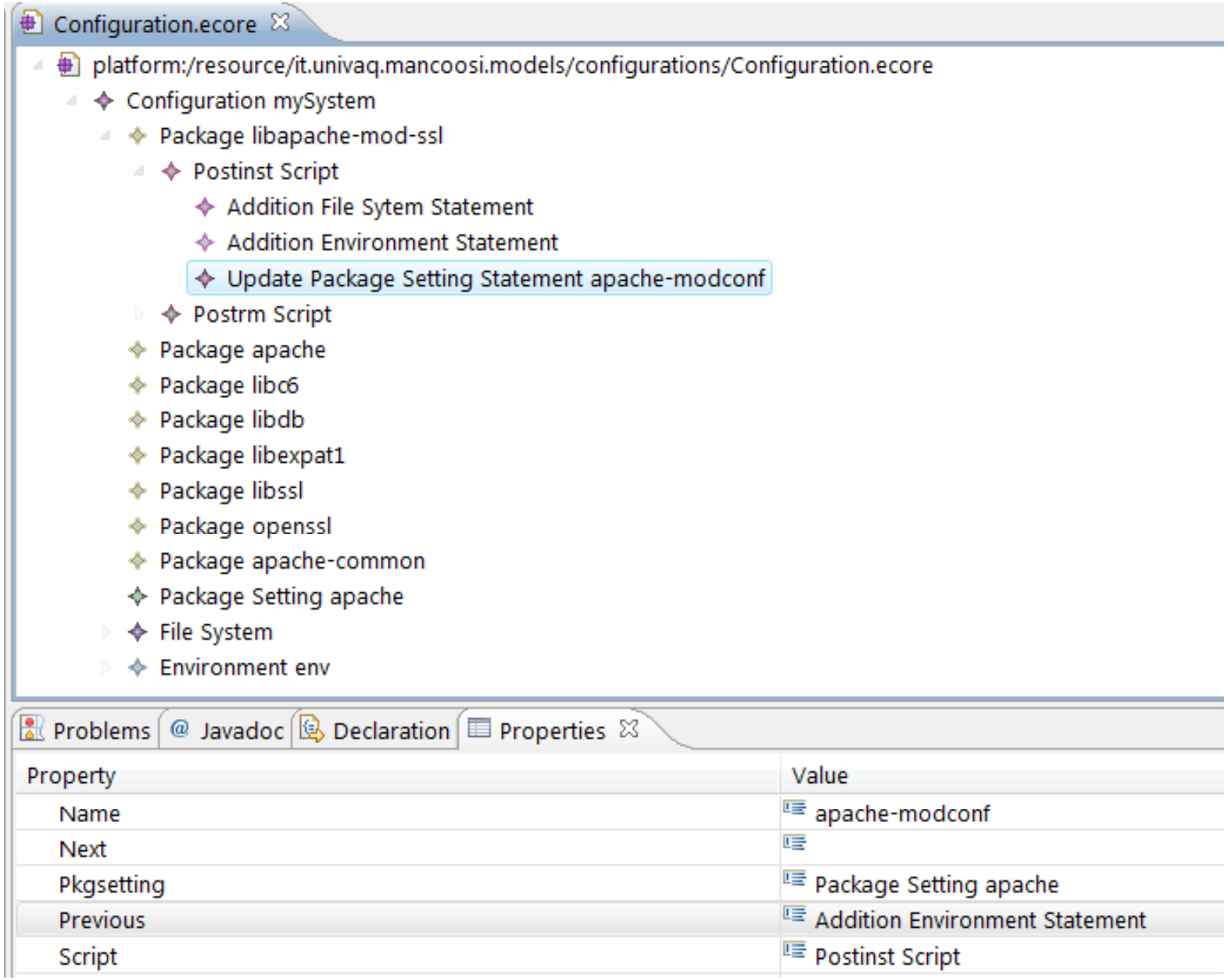
libapache-mod-ssl_2...1_amd64.deb.postinst

```
if [ "$1" = "configure" ]; then
        # apache-ssl does not need us as DSO
        for i in apache apache-perl; do
                if [ -e "/etc/$i/httpd.conf" ]; then
                        apache-modconf "$i" enable mod_ssl
                fi
        done
fi
```

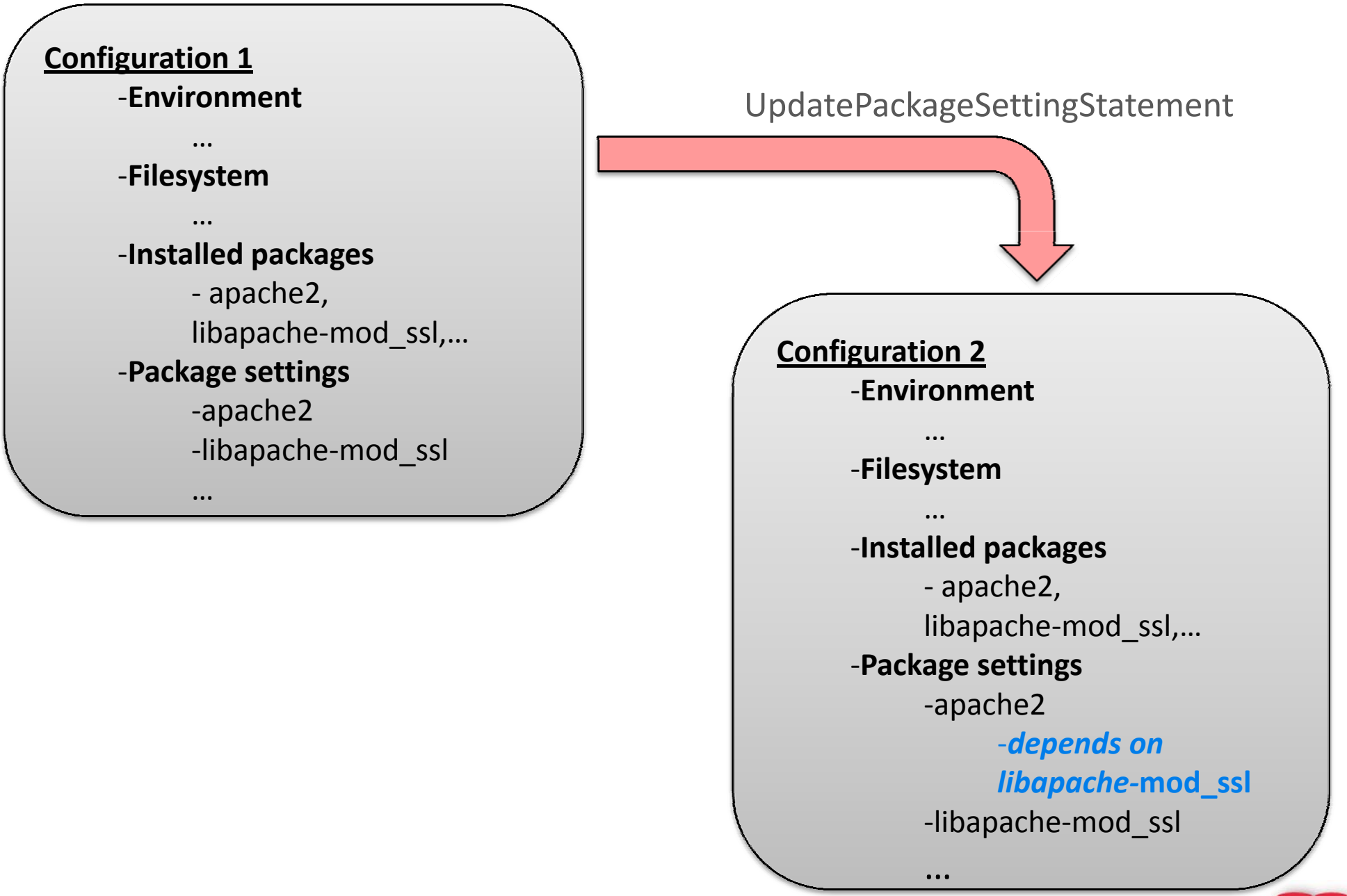
INS

UpdatePackageSettingStatement
- *name*: apache-modconf
- *pkgsetting:* apache

Sample postinst script model



Application of the postinst script



Application of the prerm script

libapache-mod-ssl_2...rge1_amd64.deb.prerm

```
#! /bin/sh

sslcrtdir=/etc/apache/ssl.crt
supportdir=/usr/share/apache/mod_ssl

umask 022

if [ "$1" = "upgrade" -o "$1" = "remove" ]; then
        if [ -d ${sslcrtdir} ]; then
                cd ${sslcrtdir}
                . ${supportdir}/functions.sh
                modssl_clean_hashes
        fi
fi

if [ "$1" = "remove" ]; then
        # there is no need to remove the entries for apache-ssl
        for i in apache apache-perl; do
                if [ -e "/etc/$i/httpd.conf" ]; then
                        apache-modconf "$i" disable mod_ssl quiet
                fi
        done
fi
```

Ln 21, Col 19 INS

Without this *UpdatePackageSettingStatement*

Application of the prerm script

Configuration 3

-**Environment**

…

-**Filesystem**

…

-**Installed packages**

- apache2, …

-**Package settings**

-apache2

-depends on libapache-mod_ssl

-libapache-mod_ssl

…

Inconsistency

Configuration 2

-**Environment**

…

-**Filesystem**

…

-**Installed packages**

- apache2, mod_ssl,…

-**Package settings**

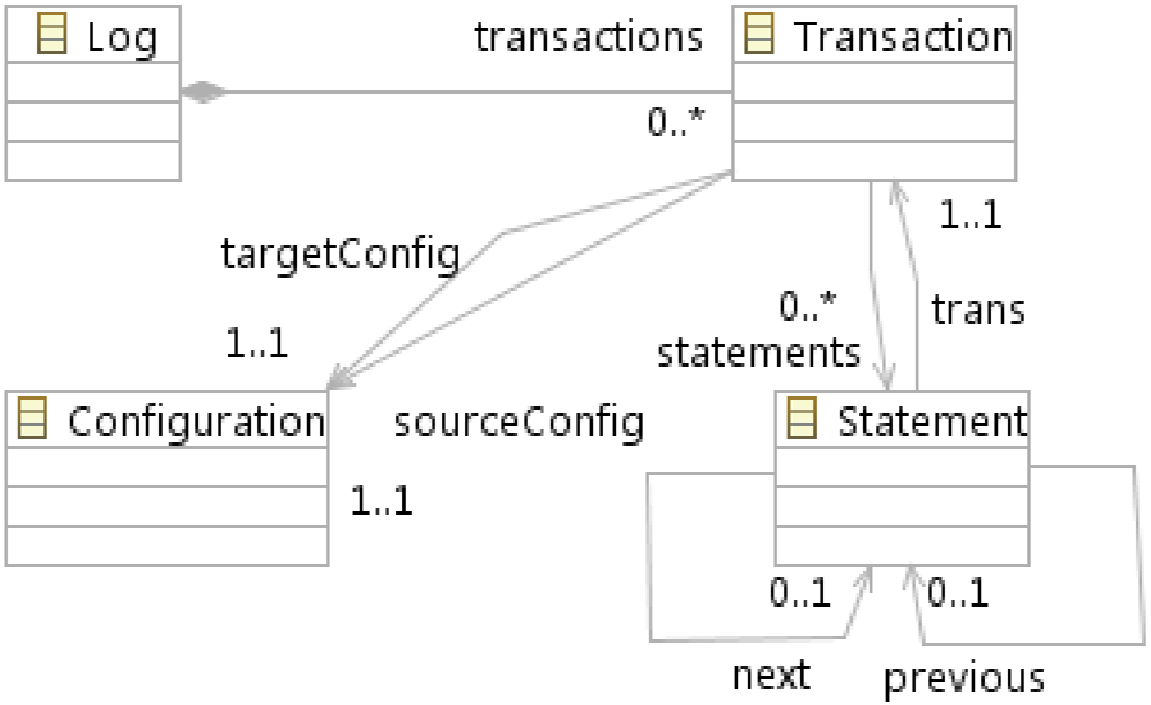
-apache2

-depends on libapache-mod_ssl

-libapache-mod_ssl

…

Log metamodel



Conclusions

- The proposed metamodels represents a step toward the following objectives
  - Simulation of package installations (obj 1)
  - Fault and roll-back management (obj 2)
  - Log management (obj 3)

Conclusions

» The proposed metamodels represents a step toward the following objectives

- Simulation of package installations (obj 1)
- Fault and roll-back management (obj 2)
- Log management (obj 3)

Configuration and Package metamodels

Benefits

-Installation and removal simulation takes into account both package dependencies and the behaviour of the maintainer scripts

- Complex model inconsistency checks

Open Issues

-Because of legacy problems, we could implement such metamodels as a wrap of existing scripting languages

-Expressiveness, do we really need the expressive power of the existing scripting languages ?

Conclusions

» The proposed metamodels represents a step toward the following objectives

- Simulation of package installations (obj 1)
- Fault and roll-back management (obj 2)
- Log management (obj 3)

Log metamodel

Benefits

-Roll-back operations can exploit the information contained in the log model which stores the transactions between different configurations

-Log management can rely on model differences and versioning

Open Issues

-Integration points between WP2 and WP3 have to be established

Future plan

» Short term
  – Refinement of the MANCOOSI metamodels
  – Deliverable D2.1

» Medium term
  – Specification of the Debian distribution by means of the MANCOOSI metamodels
    • Sample configurations
    • Specification of Packages
    • Complexity problems have to be considered
  – Deliverable D2.2

» Long term
  – Implementation of the overall simulation approach
    • Scripting language
    • Model difference management (some results in [MoDELS 2008])
    • Deliverable D2.3

Future plan

» Short term
  – Refinement of the MANCOOSI metamodels
  – Deliverable D2.1

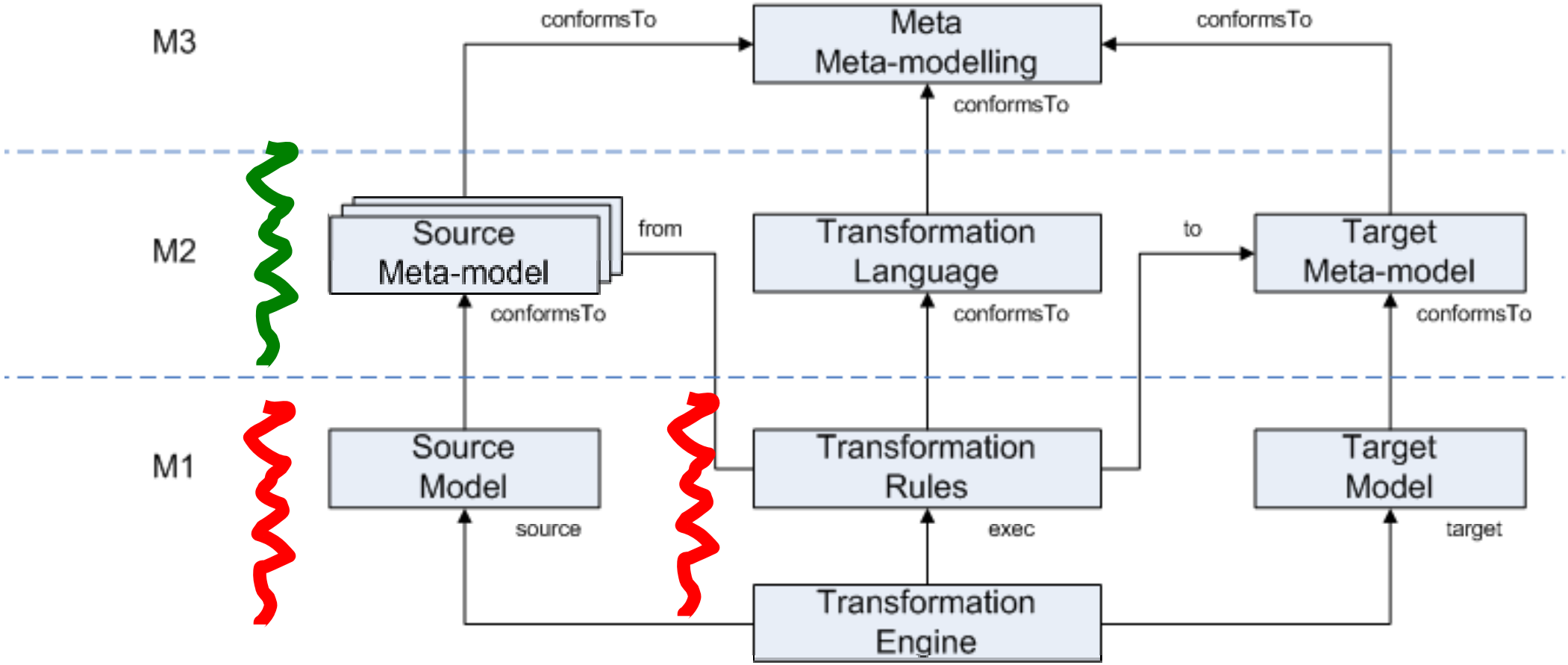
» Medium term
  – Specification of the Debian distribution by means of the MANCOOSI metamodels
    • Sample configurations
    • Specification of Packages
    • Complexity problems have to be considered
  – Deliverable D2.2

» Long term
  – Implementation of the overall simulation approach
    • Scripting language
    • Model difference management (some results in [MoDELS 2008])
    • Deliverable D2.3

Metamodel evolution

» A metamodel is defined with an iterative process, which requires to assess metamodel-level decisions against the application domain (abstraction lift)

Metamodel evolution



» Metamodels are expected to evolve during their life-cycle

» As a consequence, corresponding models have to be updated for preserving their conformance (**co-evolution**)

» This is a complex problem and we proposed an approach supporting it [EDOC 2008]

WP2: Modeling system configurations and packages

Davide Di Ruscio
Dipartimento di Informatica
Università degli Studi dell'Aquila

diruscio@di.univaq.it