Applying MDE for the Validation of Correct Eclipse Plugin Bundles

G. Doux, M. Didonet Del Fabro, F. Jouault, P. Albert, J. Bézivin, F. Madiot, S. Lee

19 mai 2010

Guillaume Doux
Guillaume.doux@inria.fr
Plan

- Introduction
- Eclipse Configuration Discovery
- Eclipse Configuration Validation
- Global Plugin Management
- Questions
Introduction - Context of study

- Complex systems build by assembling components from repositories
  - Large set of interconnected components
    - Different versions of them
    - Strong dependencies between some
    - Incompatibilities between some others
  - Examples:
    - Eclipse platform (OSGI system)
    - Linux systems
    - ...
Introduction - Context of study

- Need tooling for easier management of components and dependencies
  - Illustrating example:
    - Eclipse core runtime plug-in
      → 11 dependencies
      → Specific versions for required plug-ins...
Introduction - Context of study

Idea

- Using MDE and Configuration techniques for:
  - Components and links representation
  - Building consistent set of components
  - Validating existing distributions
Introduction – Use Case

Eclipse Platform use case

- Implementing OSGI framework
- High extension and integration abilities
- Huge number of plug-ins
  - Wide community of tool provider
  - Rapid adoption by users
  - Acceleration in the plug-in development
  - Increasing number of possible platform configuration
Introduction – State of the Art

Eclipse Platform use case

- Existing plugin management solutions (1)
  - P2:
    - Current Eclipse plug-in management tool
    - Create a constraint problem from plug-ins metadata
    - Use SAT4J for the solution creation
    - Failure explanation presentation if problem
Introduction – State of the Art

Eclipse Platform use case

- Existing plugin management solutions (2)
  - b3:
    - Focus on the build management
    - Represents build informations into models
      - From dependencies to build scripts
  - Goal:
    - Making build processes simpler, repeatable, reproducible and adaptable
  - Currently no implementation, only proposal
Plan

○ Introduction

○ **Eclipse Configuration Discovery**

○ Eclipse Configuration Validation

○ Global Plugin Management

○ Questions
Eclipse Configuration Discovery

- **Idea:**
  - Using platform information to build a complete platform model
Eclipse Configuration Discovery

- Eclipse plug-ins *manifest*
  - Simple declarative format
    - Describes dependencies with versions

```java
Manifest-Version: 1.0
Bundle-SymbolicName: org.eclipse.core.runtime; singleton=true
Bundle-Activator: org.eclipse.core.internal.runtime.PlatformActivator
Bundle-ManifestVersion: 2
Bundle-Version: 3.4.0.v20080512
Bundle-Vendor: %providerName
DynamicImport-Package: org.eclipse.core.internal.runtime.auth
Require-Bundle: org.eclipse.osgi; bundle-version="[3.2.0,4.0.0]"; visibility="reexport,org.eclipse.equinox.common
[3.2.0,4.0.0]"; visibility="reexport,org.eclipse.core.jobs; bundle-version="[3.2.0,4.0.0]"; visibility="reexport,org.eclipse.equinox.registry;bundle-version="[3.4.0,4.0.0]"; visibility="reexport,org.eclipse.equinox.preferences; bundle-version="[3.2.0,4.0.0]"
```
Eclipse Configuration Discovery

- Eclipse configuration metamodel
Plan

- Introduction
- Eclipse Configuration Discovery
- **Eclipse Configuration Validation**
- Global Plugin Management
- Questions
Eclipse Configuration Validation

○ Need for validation of the configurations
  ● Valid configuration:
    ○ All dependencies satisfied
    ○ Each bundle activable
    ○ Not just a running Configuration

○ Approach:
  ● Using configuration techniques for the solution build.
Eclipse Configuration Validation

- **Approach Overview**

![Diagram]

- Eclipse Plug-ins Dependencies
- Constraints on Models
- Adapted OCL Representation
- CP Representation Of Constraints
- Configuration Validation
- CP Engine
- Valid Eclipse Products Models
- All Valid Eclipse Products Models

19 mai 2010
Eclipse Configuration Validation

- Constraints on models expressed in OCL++

  ```
  package SPL {
    context Class inv :
      name == "Watch" ;
    context m1 : Method , m2 : Method inv :
      m1 != m2 implies m1.name != m2.name ;
    context Class inv :
      methods.exists ( m | m.name == "start" ) ;
    context Class inv:
      methods.exists ( m | m.name == "displayTime" ) ;
  }
  ```

- Enable easy expression of constraints on models
Eclipse Configuration Validation

- Zoom on the configuration validation part
Plan

- Introduction
- Eclipse Configuration Discovery
- Eclipse Configuration Validation
- **Global Plugin Management**
- Questions
Global Plug-in Management

- Idea: managing plug-in in the same way as we manage models with AM3

- AM3 offers several generic facilities:
  - Navigation
  - Query support
  - Metrics Extraction
  - Visualization

These facilities are available for models or any other kind of supported entities
Global Plug-in Management

- Eclipse plug-in management extension for AM3
Global Plug-in Management

- Dependencies Matrix constructed from the megamodel

- Cycles detection
Global Plug-in Management

- Generated visualization of a Eclipse configuration excerpt
  - Required bundle and exported packages from the org.eclipse.ui.net
Global Plug-in Management

- Birdview of an Eclipse installation
Plan

- Introduction
- Eclipse Configuration Discovery
- Eclipse Configuration Validation
- Global Plugin Management
- Questions
Questions

- If you have any questions...

?  

- Links: