

p2 enablement

Pascal Rapicault
IBM Rational™, p2 lead

p2-dev@eclipse.org

What is p2?

- A replacement for update manager
 - ◆ New UI, simplified workflows
 - ◆ Complete management of Eclipse™ and surroundings
 - exe, ini, plug-ins, windows keys,...
 - ◆ Sharing of bundles across eclipse-based products
 - ◆ An installer
- A provisioning solution for OSGi™ systems (in progress)
 - ◆ Manage non-running instance
 - ◆ Start level, framework extension
 - ◆ Deal with dependencies
- A provisioning framework
 - ◆ Pluggable and modular infrastructure
 - ◆ To provision other than eclipse based systems
 - ◆ No UI required

Agenda

- Concepts and architecture
- p2 repository generation
- Aggregating existing pieces
- p2-enabled RCP applications

Concepts and architecture

A few facts about p2

- Describe a complete Eclipse-based application
 - ◆ Exe, ini, plug-ins, features
 - ◆ Installer [1]
- Guarantee same result independently of the delivery mechanism
 - ◆ SDK.zip == Installer

p2 offers a **continuum**
from the installation to the update
and the uninstallation

[1] – Installer – <http://download.eclipse.org/eclipse/equinox/drops/R-3.4-200806172000/index.php> equinox.p2.installer file

Installable Unit, one entity to manage them all

An abstraction to decouple p2 from what is being installed

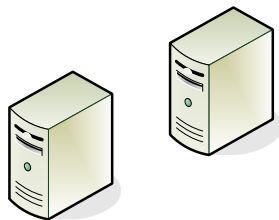
- Id and version
- Dependencies
- Describe the provisioning behavior
 - ◆ Configuration information usually stored in other IUs
- Reference the associated artifacts

Everything is an IU
Everything is installable

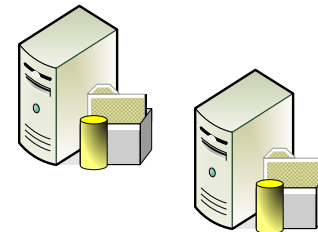
Separation of concerns

- Installable Unit
 - ◆ Exist independently of the repository
- Metadata Repository
 - ◆ Store only Installable Units
- Artifact Repository
 - ◆ Store only Artifacts

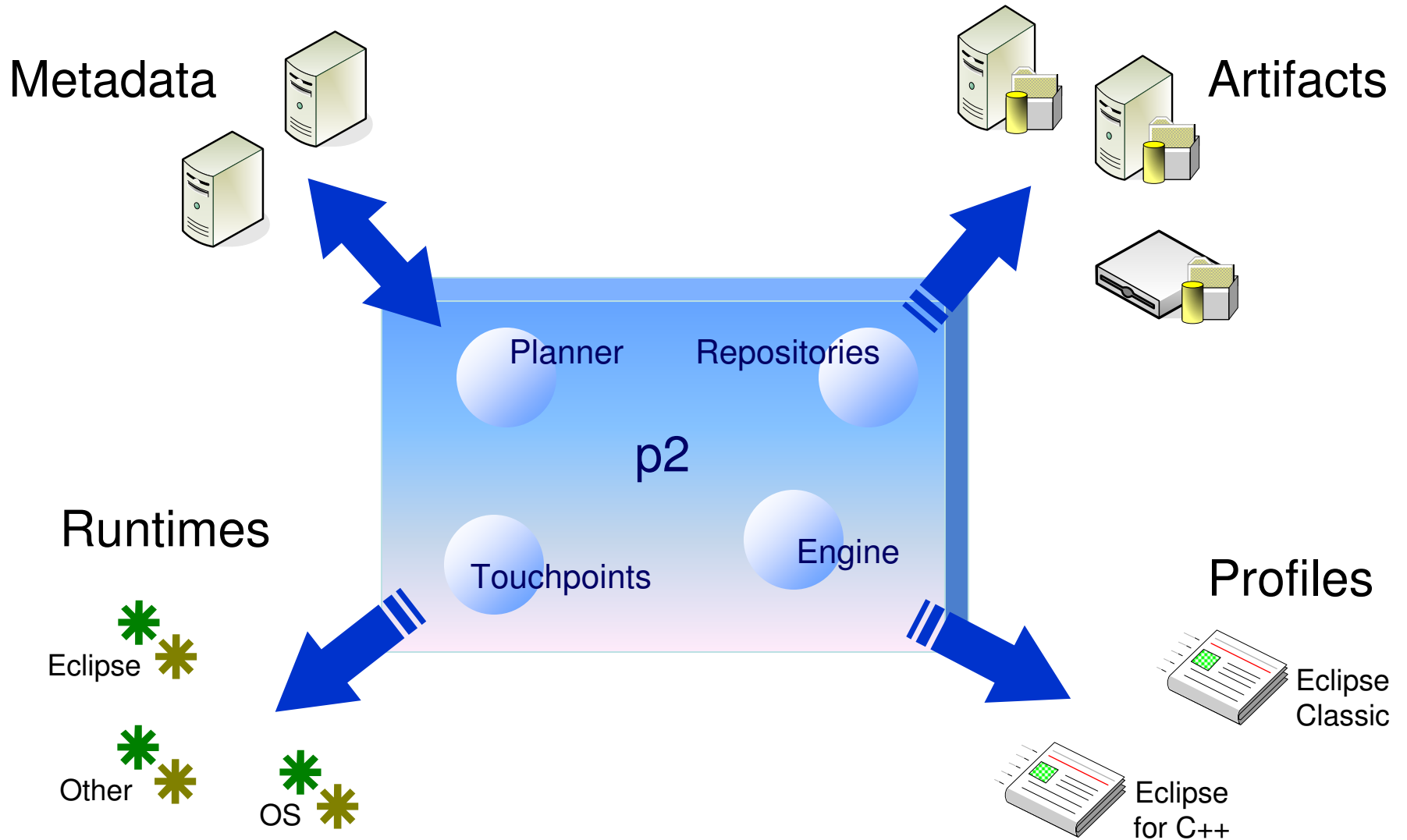
Metadata



Artifacts



5000 foot view



Recap on Terminology

- **p2 / Agent**
 - ◆ The provisioning infrastructure on client machines
- **Installable Unit (IU)**
 - ◆ **Metadata** that describes things that can be installed/configured
- **Artifact**
 - ◆ The actual content being installed/configured(e.g., bundle JARs)
- **Repository**
 - ◆ A store of metadata or artifacts
- **Profile**
 - ◆ The target of install/management operations
- **Planner**
 - ◆ The decision-making entity in the provisioning system
- **Engine**
 - ◆ The mechanism for executing provisioning requests
- **Touchpoint**
 - ◆ The part of the engine responsible for integrating the provisioning system to a particular runtime or management system

Generating p2 repositories (formerly known as update sites)

What is a p2 repository and why should I make one?

- What is it?
 - ◆ A “p2 repository” is the p2 “native” format for a repository
 - ◆ Metadata repository: content.jar (or .xml)
 - ◆ Artifact repository: artifacts.jar (or .xml)
- Why do I care?
 - ◆ Improved the overall user-experience
 - Faster site browsing
 - Precise dependency resolution

It installs it runs

p2 repository from an existing update site

- Prerequisite
 - ◆ An existing update site
- How
 - ◆ Run the metadata generator on the update site [1]

```
eclipse -console -application  
org.eclipse.equinox.p2.metadata.generator.EclipseGenerator  
-updateSite <folder> -site file:<folder>/site.xml  
-metadataRepository file:<folder>/ -metadataRepositoryName  
<Repository name> -artifactRepository file:<folder>/  
-artifactRepositoryName <Repository name> -compress  
-reusePack200Files -noDefaultIUS -vmargs -Xmx256m
```

- ◆ Put the artifacts.jar and content.jar next to the site.xml
- Note
 - ◆ If the plug-ins are only to be installed from p2, you can delete the site.xml after the generation

[1] – Generator doc – http://help.eclipse.org/ganymede/index.jsp?topic=/org.eclipse.platform.doc.isv/guide/p2_metadata_generator.html

p2 repository when exporting from the UI

- Prerequisite
 - ◆ An update site project in the IDE
- How
 - ◆ Simply use context menu action “PDE Tools > Build site”
- Note
 - ◆ This generates a content.xml and artifacts.xml file. They can be zipped to content.jar and artifacts.jar [1]

[1] – Bug on repo format – https://bugs.eclipse.org/bugs/show_bug.cgi?id=222962

p2 repository using PDE Build

- Prerequisite
 - ◆ A PDE Build based build [1]
- How
 - ◆ Set additional properties in the main build.properties [2]

<code>generate.p2.metadata</code>	enable p2 repository generation
<code>p2.metadata.repo</code>	metadata repository location
<code>p2.artifact.repo</code>	artifact repository location
<code>p2.metadata.repo.name</code>	user readable name of metadata repo
<code>p2.artifact.repo.name</code>	user readable name of artifact repo

[1] – Introduction to PDE Build – <http://www.eclipsecon.org/2008/?page=sub/&id=317>

[2] – PDE Build / p2 doc – http://help.eclipse.org/ganymede/topic/org.eclipse.pde.doc.user/tasks/pde_p2_integration.htm

Lighter weight repositories

- A feature will suffice
 - ◆ Full blown update sites are not necessary. No site.xml.
 - ◆ Even just a plug-in is enough
 - They are not shown in the UI for clarity
- A repository could be as simple as a content.jar
 - ◆ Artifact repositories referenced from content.jar

Aggregating existing pieces

Creating the aggregate

- Problem statement
 - ◆ You have several different plug-ins and features that you want to group to make them installable all at once
- How (part 1)
 - ◆ Create the grouping feature
 - Add reference to the feature. You don't have to have them installed.
 - Be careful to use qualifier in your feature

Referring to artifacts and IUs

- How – part 2

- ◆ Getting the artifacts and metadata you depend on
 - Referencing other sites
 - ✦ Use an “associated site” file in your site.xml [3]

```
eclipsec -nosplash -application org.eclipse.equinox.p2.metadata.generator.EclipseGenerator  
-updateSite D:\dev\webinar\UpdateSite4 -site file:D:\dev\webinar\UpdateSite4\site.xml  
-metadataRepository file:D:\dev\webinar\UpdateSite4\ -metadataRepositoryName "My Team Site"  
-artifactRepository file:D:\dev\webinar\UpdateSite4 -artifactRepositoryName "My Team site"  
-compress -reusePack200Files -noDefaultIUs -vmargs -Xmx256m
```

- Mirroring content locally in your repository
 - ✦ Use the p2 mirroring applications [1,2].

[1] – Doc on mirror app – http://help.eclipse.org/ganymede/index.jsp?topic=/org.eclipse.platform.doc.isv/guide/p2_mirror.html

[2] – Bug on selective mirroring – https://bugs.eclipse.org/bugs/show_bug.cgi?id=239492

[3] – Bug on site.xml being ignored – https://bugs.eclipse.org/bugs/show_bug.cgi?id=222961

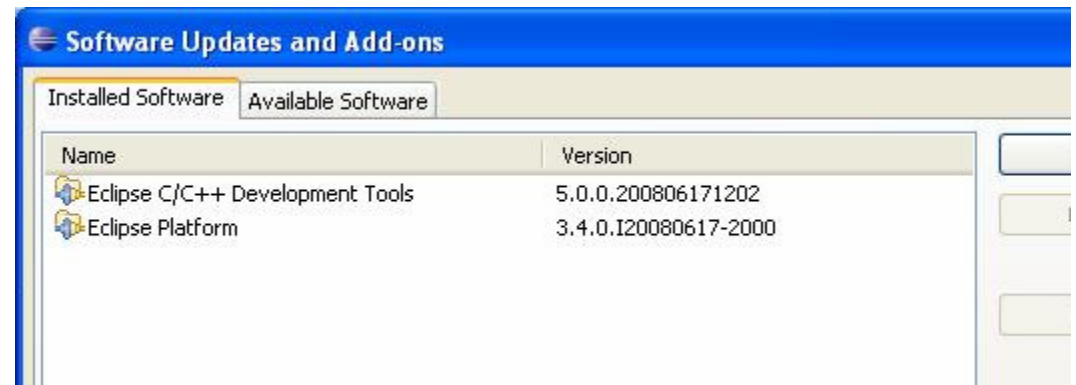
p2-enabled products

What is a p2-enabled product and why should I make one?

- What is it?
 - ◆ A p2-enabled product is an installation of an eclipse-based product resulting from a p2 install (e.g. eclipse/p2/...)
 - The profile for the product is complete
 - The product is completely described with metadata
- When does it matter?
 - ◆ When a product is delivered as an archive (e.g. Eclipse SDK)
- Why do I care?
 - ◆ Properly configure all the eclipse bits and pieces (e.g. start level)
 - ◆ Improved startup time
 - All the plug-ins are known before the first start
 - ◆ The product can be fully introspected
 - ◆ Cleaner when looking at the “Installed software” page

Extending an existing p2-enabled product

- Prerequisite
 - ◆ An existing product
 - ◆ A set of things to add



- How
 - ◆ Run the director application [1]

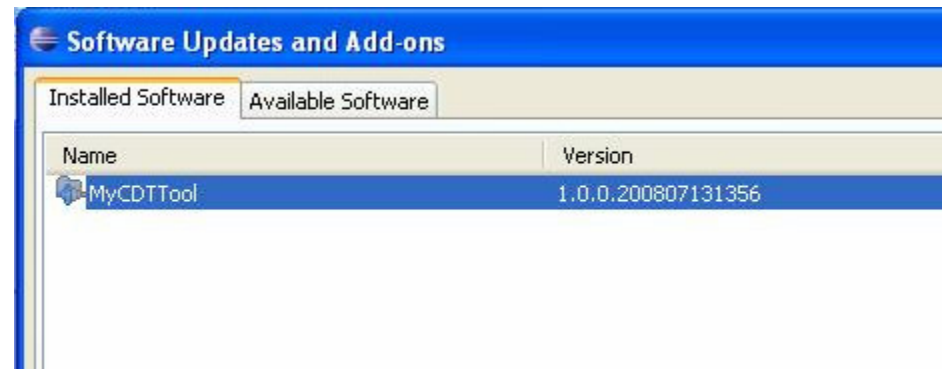
```
eclipsec -nosplash -application org.eclipse.equinox.p2.director.app.application  
-metadataRepository http://download.eclipse.org/releases/ganymede  
-artifactRepository http://download.eclipse.org/releases/ganymede  
-installIU org.eclipse.cdt.feature.group  
-destination d:/tmp/platform/eclipse  
-profile PlatformProfile  
-vmArgs -Declipse.p2.data.area=d:/tmp/platform/eclipse/p2
```

[1] – Director app doc – http://help.eclipse.org/ganymede/index.jsp?topic=/org.eclipse.platform.doc.isv/guide/p2_director.html

Appropriating/wrapping an existing p2-enabled product

- Prerequisite

- ◆ An existing product
- ◆ A set of things to add



- How

- ◆ Create an IU referring to the product and your additions
- ◆ Run the director application [1]

```
eclipse -nosplash -application org.eclipse.equinox.p2.director.app.application
-metadataRepository <list Of Metadata Repo>
-artifactRepository <list Of Artifact Repo>
-installIU <IU of the product>
-destination <installFolder>/ -profile <profileID>
-profileProperties org.eclipse.update.install.features=true
-bundlepool <installFolder>/ -p2.os linux -p2.ws gtk -p2.arch x86 -roaming
-vmargs -Declipse.p2.data.area=<installFolder>/p2/
```

[1] – Director app doc – http://help.eclipse.org/ganymede/index.jsp?topic=/org.eclipse.platform.doc.isv/guide/p2_director.html

Extending vs. Wrapping

- Extension

- ◆ The base and your extension can be updated at different times.

- Wrapping

- ◆ You are in complete control of the pieces you use. You will be the logical provider of the underlying pieces (e.g. RCP, platform, etc.).
- ◆ Hide the base you are building on

p2-enabled RCP applications

Creating a p2-repository for a product

- Prerequisite
 - ◆ An RCP application and a product file
- How
 - ◆ Build the application creating the metadata repository
 - Use the “Export product” action from the UI
 - ◆ Run the director to create the p2-enabled application
 - See slide 22

[1] – Kai Toedter’s blog on RCP and p2 – <http://toedter.com/blog/?p=27>

[2] – Andrew Niefer’s blog on RCP, p2 and PDE Build – <http://aniefer.blogspot.com/2008/06/example-headless-build-for-rcp-product.html>

Tool summary

- Generator application

Generate repositories from eclipse

- ◆ Input: plug-ins, features, config.ini
- ◆ Output: Installable units in a repository, artifact repository

- Mirroring application

Copy a repository into another one

- ◆ Input: Artifact or metadata repository
- ◆ Output: Artifact or metadata repository

- Director application

Install / uninstall into an existing application or create it

- ◆ Input: Repositories + Installable Unit
- ◆ Output: p2-enabled application

Future work

- Improvements to the p2 runtime [1]
- Repository tooling
- IU editor [2]

[1] – 3.5 plan – http://wiki.eclipse.org/Equinox_p2_3.5_contributions

[2] – IU editor – http://wiki.eclipse.org/Equinox_p2_Metadata_Authoring

Thank you

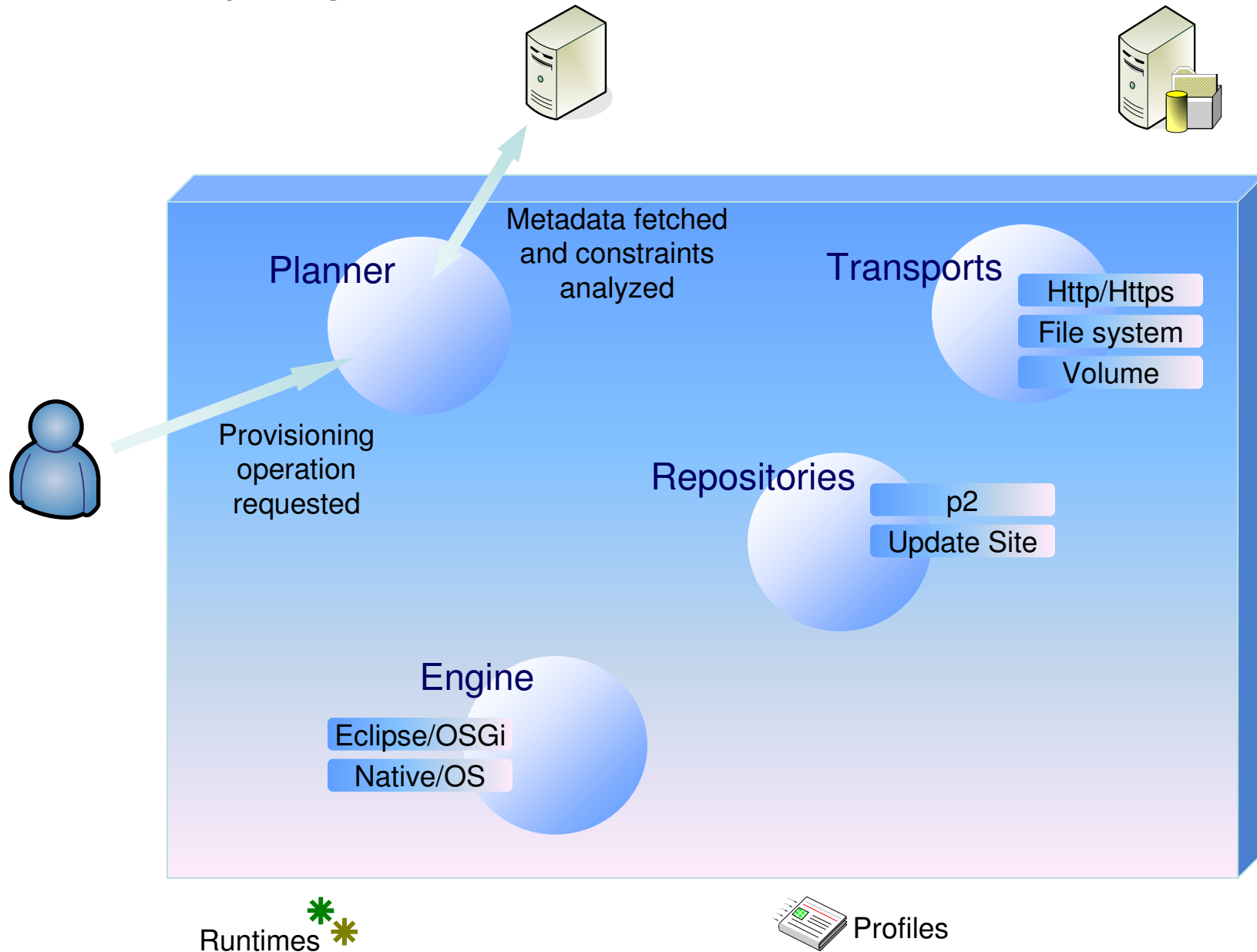
p2-dev@[eclipse.org](mailto:p2-dev@eclipse.org)
http://wiki.eclipse.org/Equinox_p2

Legal Notices

- ◆ Eclipse and the Eclipse logo are trademarks of Eclipse Foundation, Inc.
- ◆ IBM and the IBM logo are trademarks or registered trademarks of IBM Corporation, in the United States, other countries or both.
- ◆ Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both
- ◆ OSGi is a registered trademark of the OSGi Alliance in the United States, other countries or both.
- ◆ Other company, product, or service names may be trademarks or service marks of others

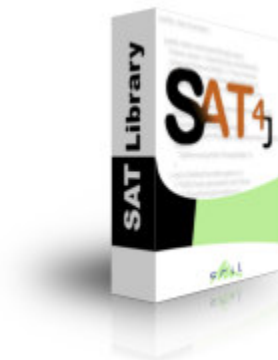
THE INFORMATION DISCUSSED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, AND IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, SUCH INFORMATION. ANY INFORMATION CONCERNING IBM'S PRODUCT PLANS OR STRATEGY IS SUBJECT TO CHANGE BY IBM WITHOUT NOTICE.

Inside the p2 agent



SAT-based resolution

- The installation problem is NP-complete
 - ◆ Map dependencies to boolean formula
 - ◆ Use SAT4J
 - If there is no solution it will tell us
 - If there is a solution we will get an optimal one



Special thanks to
Daniel LeBerre
and Anne Parrain

Inside the p2 agent

