




Provisioning Eclipse in the Enterprise

Jeff McAffer | EclipseSource

Henrik Lindberg | Cloudsmith

Pascal Rapicault | IBM

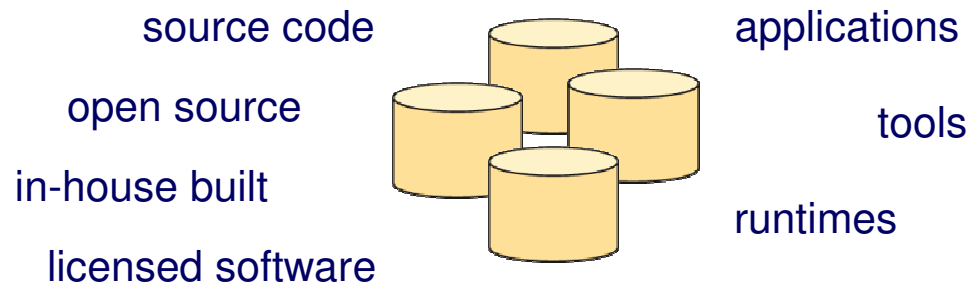
Provision what – where




Engineers



Users



Automated Builds
Test



Test



Wide range of requirements

- packaged applications – source code
- strict policy control – full flexibility
- approved repositories – any repository
- fixed configuration – dynamic configuration
- fully automatic – user driven – (headless)
- update to latest – update to policy controlled (downgrade)

Eclipse and Provisioning



p2



Unzipping is not installing!





p2 is installing!

- A replacement for the old Update Manager
 - New UI, simplified workflow
 - Manage Eclipse, RCP and more
 - exe, ini, bundles, registry keys, native code, ...)
 - Shared bundles across Eclipse-based products
 - An installer
- A provisioning solution for OSGi™ systems
 - Managing non-running instance
 - Start level, framework extension
 - Fine-grained dependency management



Trends in Eclipse

- Eclipse proves the power of componentization
- Componentization naturally spreads
- More components → more management
- Management is hard

- Its all about the Contract
 - Defining
 - Instantiating
 - Executing
 - Maintaining



How does p2 help?

- Manages the contract
 - Dependencies
 - Code
 - Settings
 - Integrations
 - Non-Eclipse parts
- Extensible
- GUI and Headless
- One consistent model



New user interactions

- Simple update workflow
 - Replace multi-steps wizards
- New metaphors
 - Drag n Drop for install, adding repos, ...
- More flexible repositories
 - Connect to p2 repos, Update Sites, OBR, Maven, ...
- Managed folders
 - Explicit “watched” locations
 - Drop content to have it installed
 - No need to unzip and -clean



From Install to Update

- Installing and Updating are the same operations
- Installed shape same as zip shape
 - Flexibility in delivery
- Programmatic API for all operations

p2 offers a **continuum**
from installation to updating



Demo

- Install the SDK using the installer
- Add a repository
 - Discover and install some new function
- Install a second SDK
 - Observe how blazingly fast it goes



Concepts and Architecture



Installable Units

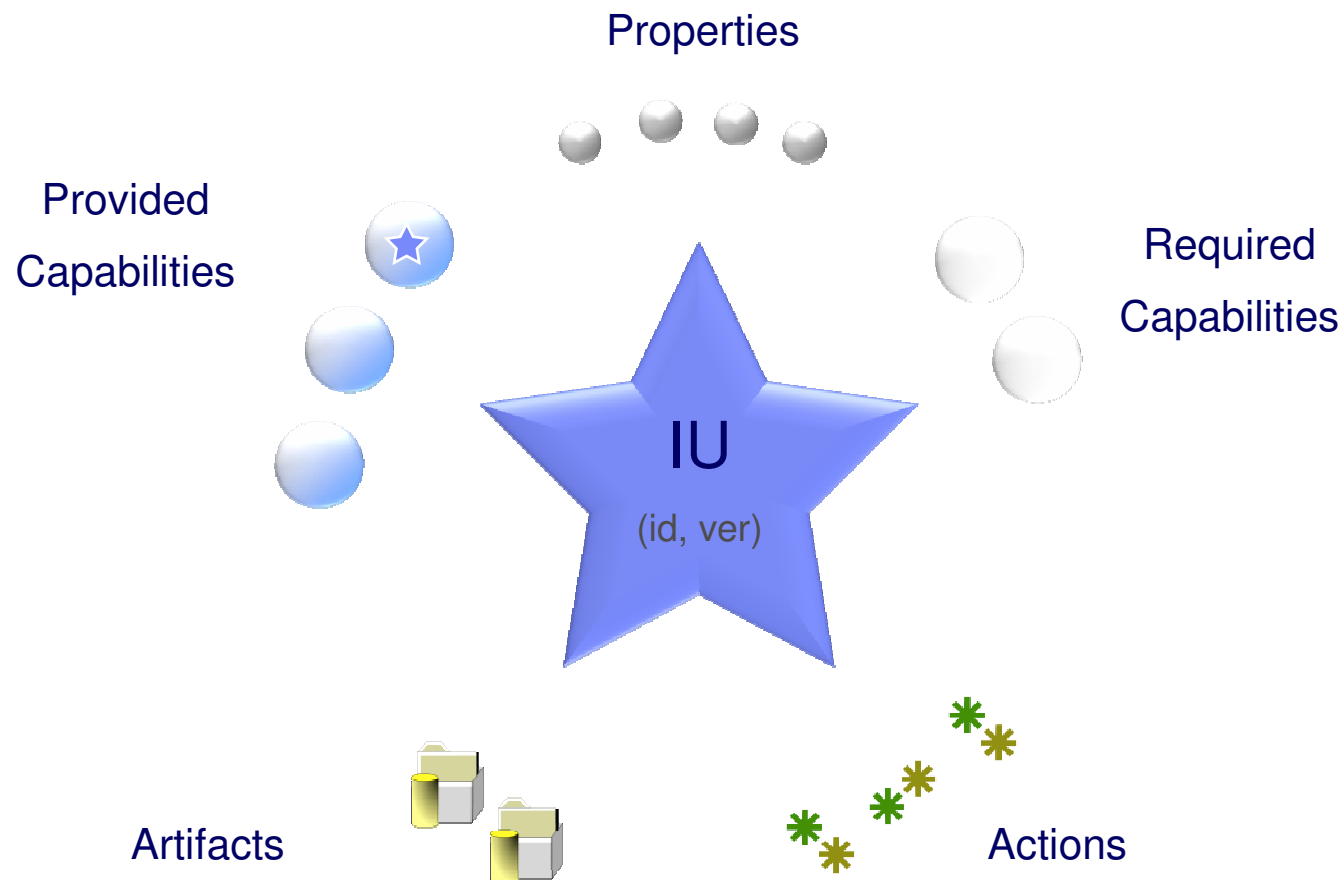
Decouple decision making from the actual content



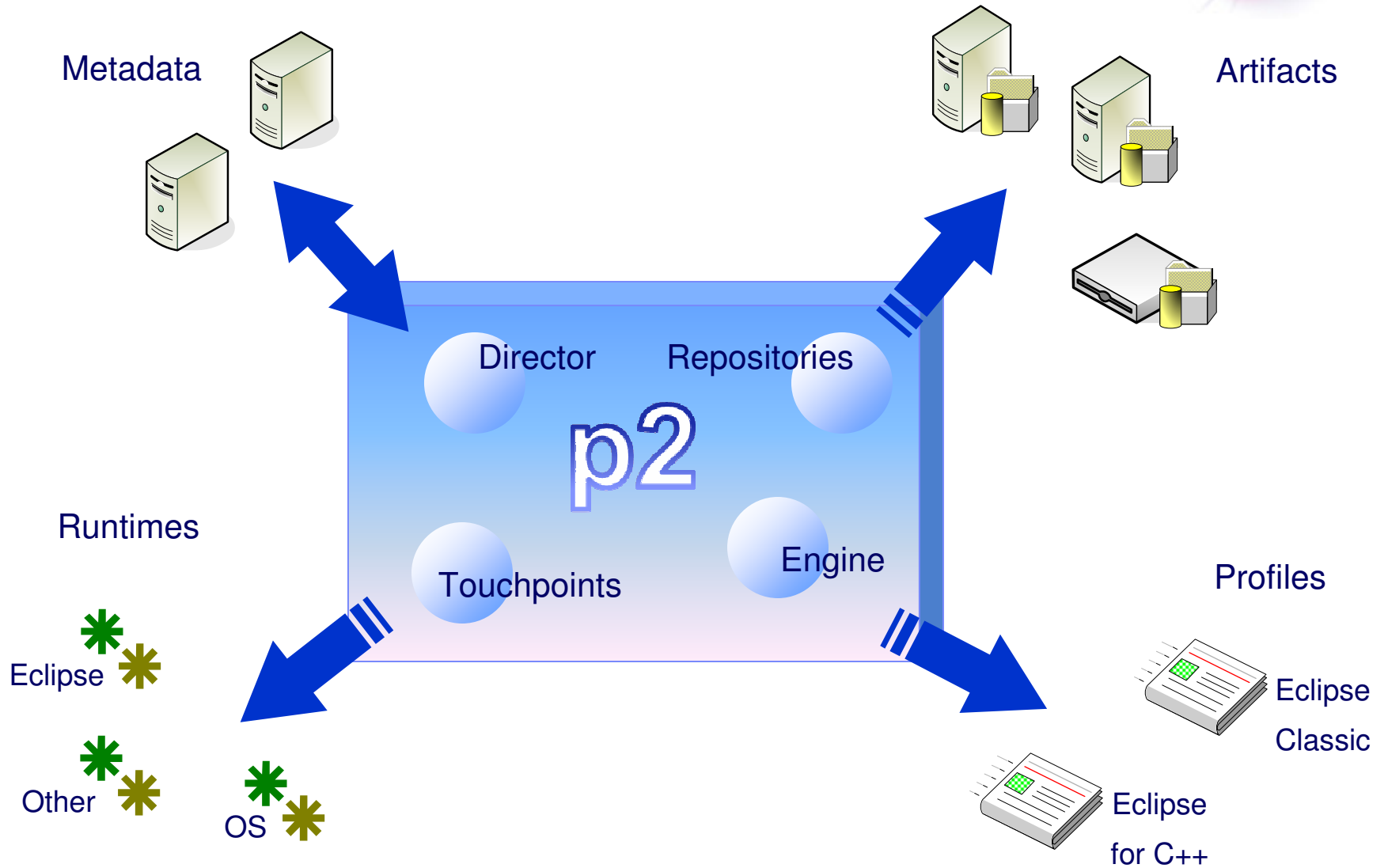
Everything is an IU
Everything is installable



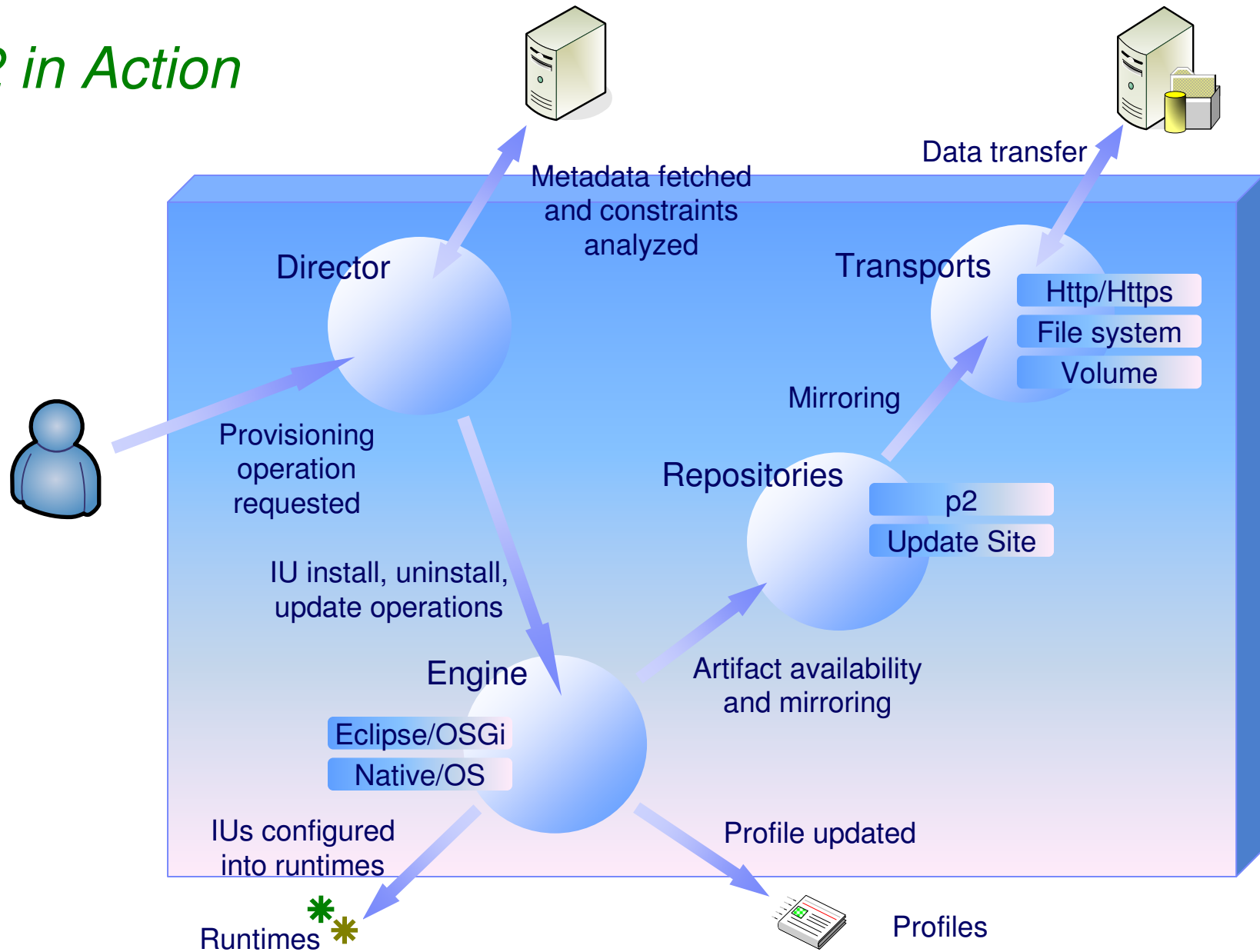
Anatomy of an IU



p2 Architecture



p2 in Action





SAT-based resolution

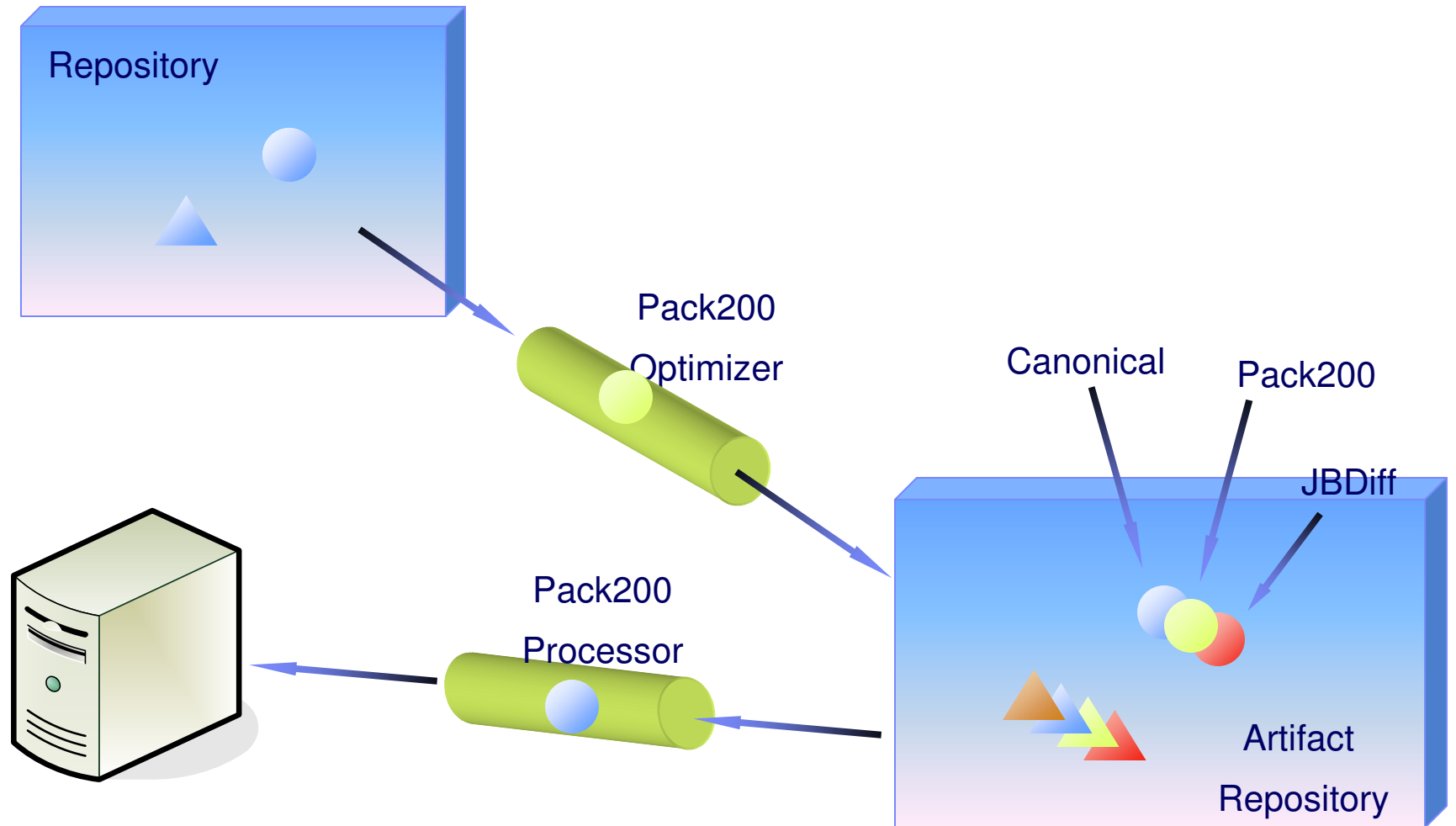
- The installation problem is NP-complete
 - Map dependencies to boolean formula
 - Use SAT4J
 - If there is no solution it will tell us
 - If there is a solution we will get an optimal one



Special thanks to
Daniel LeBerre
and Anne Parrain



Optimizing and Processing Artifacts





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
- **Director**
 - The decision-making entity in the provisioning system
- **Engine**
 - The mechanism for executing provisioning requests
- **Touchpoints**
 - Integration with particular runtime or management systems



Another Demo Toast Deployment



p2 in Ganymede



1.0 Goals

- Replace Update Manager in Ganymede (June 2008)
 - Improved functionality
 - Minimal disruption
- Starting point for new provisioning platform
 - Initial design
 - Provisional API (try and give us feedback)
- “Client-side” only



Update Manager Compatibility

- Read existing update sites directly
- Install features and manage platform.xml
- “Optimize” update sites to have p2 metadata
 - In-place artifact management
- Ganymede supports 3 modes:
 - UM only
 - p2 only (but still read existing update sites)
 - UM / p2 co-existence (Default mode for SDK)
- UM API supported by UM code, not p2



Impact on you?

- User
 - UI improvements
 - Faster install (Pack200, incremental, multi-threaded download, ...)
- Update site owner
 - Good practice to “convert” but not required
- Product producer
 - Change requirements depend on provisioning mode
 - No code changes
- You do not write IUs
 - All the information is already available



p2 in Galileo



Stability & Completeness

User Interaction

Download Technology

Core Facilities

Tooling



User Interaction

- Improve overall usability (1)
- Closer integration with the VM (1)
- Promote the usage of an installer (2)
- Tighter desktop integration (2)
- Better shared install (2)
- Ease installation of extension and bridges plug-ins (2)
- Ease the management of complementary items (language, docs, source) (2)
- Installation duplication (2)
- Recovery application (3)
- Profile interchange (3)



Download Technology

- Download integrity through MD5/SHA1 and signature verification (1)
- Robustness / responsiveness / user friendliness (1)
- Improve Adaptive downloads and mirror selection (2)
- Restart from partial downloads (2)
- Download time estimation (2)
- Media support (2)



Core Facilities

- API (1)
- Review touchpoint contribution model (1)
- Reaction to configuration changes (1)
- Improve robustness of installation (1)
- Improve the eclipse touchpoint (1)
- Garbage collection (1-2)
- Improve overall traceability of the install (1)
- Improve test suites (1)
- Repository enhancements (1)
- Multiple processes modifying the same profile registry / profile (1)



Core Facilities cont...

- Making the agent fully dynamic (2)
- Dynamic provisioning of missing touchpoints (2)
- UI building blocks and programmatic configurability (2)
- Dependency model improvement(2)
- Linkability in metadata (2)
- Sequenced provisioning (2)
- Profile initialization (2)



Tooling - Plugin Developer

- Repository browsers and editors (2)
- Repository tooling (copy, clean, verify) (1-2)
- Metadata authoring (2)



Other projects are...

- EPP Wizard
 - Configure your own custom Eclipse on the web and install using p2
- Buckminster - will be p2 based
 - Installing into workspace
 - using properties to deliver links and feeds in OPML format
 - Installing to multiple profiles (multiple products/tools/runtime/source)
 - Handle other version types than OSGi
- IAM – Maven integration on the plan



Summary

- p2 is a powerful provisioning platform
 - Highly extensible
 - Enable the creation of comprehensive solutions
 - The basis of Eclipse provisioning going forward
-
- <mailto:p2-dev@eclipse.org>
 - <http://wiki.eclipse.org/Equinox/p2>