What are the aims of Kepler
Defining a collaboration model
Project facet extensions
Version facet extensions
How is the model defined?
Model adaptors
Next steps
what are the aims of Eclipse Kepler
what are the aims of Kepler

- To build the concept of a *community* or *collaboration* model
- To service a loosely coupled collaboration approach
- To integrate common collaboration technologies’
  - artifact repositories
  - issue tracking
  - build servers
  - instant message solutions (IRC/IM)
  - mailing lists
what are the aims of Kepler

- To develop in an open, transparent fashion at the Eclipse Software Foundation
- To leverage other Eclipse projects
  - Building/Assembly through Buckminster
  - Issue tracking through Mylyn
  - Source Control Management integration Eclipse SCM
  - On-line collaboration through Corona
  - ECF for communication protocols
- To bring new technologies into Eclipse (Maven)
what are the aims of Kepler

Kepler should:

- Gather information for collaboration
- Adapt to existing sources of meta-data
- Provide links out to existing tooling to work with collaboration technologies (Issue tracking etc)
- Provide a format for searching and finding projects
- Provide collaboration tooling integration where is it missing
  - Mailing lists, Forums etc
Defining a collaboration model
We need very basic concepts to start collaboration

- What is a project?
- What is a version?
- Does the project have dependencies?
- What artifact(s) does a project produce that can be used?

Beyond this core of information, projects can vary wildly in their content.
defining a collaboration model

How does this information help?

- It provides the basis to identify a project
- It understands the concept of slow moving dimensions on collaboration (time, releases, versions)
- It accepts that there is a product that can be consumed from the project
- It can be created from a number of sources
- It is not specific to any build tool or language
defining a collaboration model
Extending the collaboration model

- The Core model provides the basis for extension.
- Extensions are provided at three places:
  - ProjectFacet - 0 or more project facets
  - VersionFacet - 0 or more version facets
  - DependencyFacet - 0 or more dependency facets
- The model defines all facets as abstract types.
- Also the dependency and artifact types are abstract.
  - Dependencies/Artifacts are by their nature not concrete
  - This provides freedom in the core model for supporting the concept of dependency without constraining it.
defining a collaboration model
public interface ProjectType extends EObject {

    String getId();

    void setId(String value);

    String getDescription();

    void setDescription(String value);

    EList<Version> getVersion();

    EList<ProjectFacet> getFacet();

}
public interface Version extends EObject {
    void setId(String value);
    EList<VersionFacet> getFacet();
    EList<Dependency> getDependency();
    EList<Artifact> getArtifact();
}
defining a collaboration model

Kepler will work to provide a set of common implementations
- ProjectFacets
- VersionFacets
- Dependency types
- Artifact types

Other facets will be able to register themselves

A few examples of the common implementations are .....
project facet extensions

- licenseUrl
- name
- comments
- distributionMechanism

http://org/eclipse/emf/ecore/util/ExtendedMetaData
project facet extensions
version facet extensions
version facets

Licensing
- licenseUrl
- name
- comments
- distributionMechanism

http://org/eclipse/emf/ecore/util/ExtendedMetaData
how is the model defined?
The collaboration model is defined as XSD’s. It is handled and leveraged through the Eclipse EMF tooling. An EMF model is generated from the XSD’s that define the core schema:

- We also include the common extensions along with the schema.
- We generate standard EMF code to represent the model.
Kepler will provide Model Adaptors

- These will provide a way to source meta-data from projects into the collaboration model
- They would be bi-directional
- They would provide a list of the facet types that they support
- This would also allow more than one source to be found in a project
Consider a PDE model adaptor

- Would support the base of the core model (project id, version if defined in MANIFEST.MF)
- Would support a source locations extensions (.classpath)
- Would support an understanding of dependencies (.classpath)
- Would support an understanding of OSGi imports/exports

Therefore it would support those facet types
Consider a Maven model adaptor

- Would support the base of the core model (pom.xml)
- Would support a source locations extensions (pom.xml)
- Would support an understanding of dependencies (pom.xml)
- Would support an understanding of licensing (if in the pom.xml)
- Would support other facets (based on pom.xml)

Therefore it would support those facet types
<?xml version="1.0" encoding="UTF-8"?>
<core:projectType xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:community="http://www.eclipse.org/kepler/schema/project/facet/community"
    xmlns:core="http://www.eclipse.org/kepler/schema/project/core"
    xmlns:licensing="http://www.eclipse.org/kepler/schema/project/version/facet/licensing"
    xmlns:organization="http://www.eclipse.org/kepler/schema/project/facet/organization"
    xmlns:participants="http://www.eclipse.org/kepler/schema/project/facet/participants"
    xmlns:project="http://www.eclipse.org/kepler/schema/dependency/project">
  <core:id>velocity.velocity</core:id>
  <core:description>Velocity is a Java-based template engine. It permits anyone to use the simple yet powerful template language to reference objects defined in Java code.</core:description>
  <core:version>
    <core:id>1.4</core:id>
    <core:facet xsi:type="licensing:licensing">
      <licensing:name>The Apache Software License, Version 2.0</licensing:name>
      <licensing:distributionMechanism>repo</licensing:distributionMechanism>
    </core:facet>
    <core:dependency xsi:type="project:runtimeDependency">
      <project:projectId>velocity.velocity-dep</project:projectId>
      <project:versionId>1.4</project:versionId>
    </core:dependency>
    <core:dependency>
    </core:dependency>
  </core:version>
  <core:facet xsi:type="community:community">
    <community:mailingList>
      <community:name>Maven User List</community:name>
      <community:unsubscribeEmailAddress>velocity-user-unsubscribe@jakarta.apache.org</community:unsubscribeEmailAddress>
      <community:unsubscribesAddress>
    </community:mailingList>
  </core:facet>
</core:projectType>