Unleash the power of Eclipse technologies

The benefits of modernizing your project

Dirk Fauth, Harald Mackamul (Robert Bosch GmbH)
Unleash the power of Eclipse technologies

Speakers

Dirk Fauth
Research Engineer
Eclipse Committer
Robert Bosch GmbH
Motorstraße 28
70499 Stuttgart
dirk.fauth@de.bosch.com
www.bosch.com
blog.vogella.com/author/fipro/
Twitter: fipro78

Harald Mackamul
Senior Expert
Eclipse Committer
Robert Bosch GmbH
Renningen
70465 Stuttgart
harald.mackamul@de.bosch.com
www.bosch.com
Unleash the power of Eclipse technologies

Agenda

- Motivation
- General project improvements (Eclipse infrastructure)
- Issues and possible solutions (Eclipse technologies)
- Benefits – Eclipse APP4MC examples
- Conclusion
Unleash the power of Eclipse technologies

Motivation

► Situation
  ► large codebase with contributions from many participants
  ► diversity is a barrier for new contributors
  ► monolithic build
  ► multiple dependencies between central model and tools

► Goals
  ► reduce the maintenance effort on builds
  ► increase maintainability of the code base
  ► simplify contributions to the project
  ► make it easier for adopters to seamlessly extend the project
General Project Improvements

Dependencies, Libraries, ...
Unleash the power of Eclipse technologies

Dependencies to (external) libraries

Limit dependencies to really required libraries

"Know and use the libraries"

By using standard libraries:
- you take advantage of the knowledge of expert authors
- you don’t waste your time writing ad hoc solutions
- Standard libraries performance tends to improve over time
- Standard libraries tend to gain functionality over time

"Dependency hell"

larger runtime
conflicting versions
difficult to reuse in other contexts

https://pixabay.com/illustrations/arm-wrestling-competition-strength-1020222/
Unleash the power of Eclipse technologies

Dependencies to (external) libraries

Example: Model files and folders

Java

```
java.io
```

- Object
- File

Eclipse

```
... core.resources
```

- IResource
  - IContainer
    - IFolder
    - IProject
  - IFile

Eclipse EMF

```
... emf.ecore.resource
```

- Notifier
  - IResource
    - Resource

Eclipse APP4MC

```
... app4mc.amalthea.model.emf
```

- AmaltheaResource
  - AmaltheaResourceSet

Increasing number of dependencies
Unleash the power of Eclipse technologies

Dependencies to (external) libraries

Is Eclipse workspace required?

Example: Service that generates SystemC code based on model content

- Smaller deployment artifact (~ 45% less)
- Faster startup time
  No directories to create, no filesystem listener to install,
  simply start the OSGi framework and trigger the function/service

OS dependent archive | Portable executable JAR

- aml2systemc-app.jar 20.962 KB
General Project Improvements

Eclipse Infrastructure
Unleash the power of Eclipse technologies
Modularization – Builds and resulting artifacts

Products and components
Unleash the power of Eclipse technologies
Modularization – Results in Eclipse download area

download.eclipse.org/app4mc/components
|-- addon
   `-- migration
      |-- releases
      |   |-- 1.1.0
      |   |   |-- p2repo
      |   |   `-- org.eclipse.app4mc.converters.p2repo-1.1.0.zip
      |-- 1.2.0
      |   |-- p2repo
      |   `-- headless
      |   `-- org.eclipse.app4mc.converters.p2repo-1.1.0.zip
      `-- latest
         |-- p2repo
         `-- headless
            |-- org.eclipse.app4mc.converters.p2repo-1.1.0.zip
     `-- snapshot
         |-- p2repo
         `-- headless
            `-- org.eclipse.app4mc.converters.p2repo-2.0.0-SNAPSHOT.zip

headless
- README.TXT
- amalthea_migration.jar
Unleash the power of Eclipse technologies
Modularization – Results in Maven Central

- General useful bundles published
  - Amalthea Model
  - Model Migration
  - Model Validation
  - Utils
  - No APP4MC UI plug-ins
- Deployment via `mvn deploy` using pom-less Maven Tycho
  - Maven dependencies need to be maintained twice

https://wiki.eclipse.org/Tycho:How_to_deploy_to_a_Maven_repository
Unleash the power of Eclipse technologies
Sonar scans as part of dedicated builds

- Code quality improvements with Sonar scans
  - https://sonarcloud.io/organizations/eclipse/projects
- Coverage of PDE structured projects
- Examples
Issues and possible solutions

Eclipse Technologies
Unleash the power of Eclipse technologies

Eclipse Technologies

- Modular product updates
- Startup Performance – Plugin Activators
- Builds with pomless Tycho
- Extension Points → Declarative Services
- Eclipse 3 → Eclipse 4
Unleash the power of Eclipse technologies

Modular product updates

- Feature based product
  - Easier product definition
  - Modular updates
Unleash the power of Eclipse technologies
Modular product updates

Configure product independent features as root features to enable modular product updates
Unleash the power of Eclipse technologies

Modular product updates

- Feature based product
  - Easier product definition
  - Modular updates
Unleash the power of Eclipse technologies

Startup Performance – Remove Activator

- Created by Eclipse PDE tooling
- Executed in the UI thread when bundle is started by the framework
- Not used as intended but as single point of feature access

<table>
<thead>
<tr>
<th>Activator Usage</th>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugin / Bundle ID</td>
<td>FrameworkUtil.getBundle(getClass()).getSymbolicName()</td>
</tr>
<tr>
<td>Logging functionality</td>
<td>Various approaches available, for simple 1-to-1 change use Platform.getLog()</td>
</tr>
<tr>
<td>Access to preferences</td>
<td>use preference Singletons provided by the Eclipse runtime, e.g.</td>
</tr>
<tr>
<td></td>
<td>InstanceScope.INSTANCE.getNode(&quot;&quot;)</td>
</tr>
<tr>
<td></td>
<td><a href="https://www.vogella.com/tutorials/EclipsePreferences/article.html">https://www.vogella.com/tutorials/EclipsePreferences/article.html</a></td>
</tr>
<tr>
<td>Access to resources like images</td>
<td>ResourceLocator#imageDescriptorFromBundle(String, String)</td>
</tr>
<tr>
<td></td>
<td>JFace ResourceManager</td>
</tr>
<tr>
<td></td>
<td><a href="https://www.vogella.com/tutorials/EclipseJFace/article.html#jface_localresoucemanager">https://www.vogella.com/tutorials/EclipseJFace/article.html#jface_localresoucemanager</a></td>
</tr>
<tr>
<td>Access to dialog settings</td>
<td>PlatformUI.getDialogSettingsProvider(getBundle()).getDialogSettings()</td>
</tr>
<tr>
<td>(since 4.18 / 2020-12)</td>
<td><a href="http://eclip.se/549929">http://eclip.se/549929</a></td>
</tr>
</tbody>
</table>
Unleash the power of Eclipse technologies

Startup Performance – Remove Activator

- Deal with required actions on startup
  - Use OSGi Declarative Services *Immediate Components*
  - Use an EventHandler for **APP_STARTUP_COMPLETE**

```java
@Component(property = {
    EventConstants.EVENT_TOPIC + "=" + UIEvents.UILifeCycle.APP_STARTUP_COMPLETE })
public class ViewpointRegistration implements EventHandler {

    @Override
    public void handleEvent(Event event) {
        Display.getDefault().asyncExec(() -> {
            // TODO Auto-generated method stub
        });
    }
}
```
Unleash the power of Eclipse technologies
pomless Tycho

- no pom.xml for plugins, features and tests
- Since 1.5
  - no pom.xml for target definitions, products and update sites
  - no connector pom.xml files needed for “structured environments”

→ Number of pom.xml files reduced to a minimum
→ Maintenance effort for build topics (e.g. version update) reduced

http://blog.vogella.com/2015/12/15/pom-less-tycho-builds-for-structured-environments/
Unleash the power of Eclipse technologies

Extension Points → Declarative Services

- **Extension Points**
  - Definition via XML & XML Schema
  - one-to-many relation design
  - Consumed programmatically using ExtensionRegistry via ID → No type safety!
  - No dynamics at runtime
  - Equinox only

- **Declarative Services**
  - Definition via XML → Generated by tooling (DS annotations)
  - many-to-many relation design
  - Consumed via injection → type safe!
  - Dynamics supported at runtime
  - OSGi standard
Unleash the power of Eclipse technologies
Eclipse 3 → Eclipse 4

- Migrate UI contributions to Eclipse 4
  - Contribute via model fragments (fragment.e4xmi)
  - Commands, Handler, Menu Contributions
  - ViewPart → Part

- Migrate **Core Expressions** to **Imperative Expressions** (@Evaluate)

- Add context values via **Model Addons**

- Usage of the **Event Bus** in several places
  
  - Less code bloat
  - Participating in E4 programming model
  - Usage of injection even for OSGi services
  - Loose coupling between modules
Unleash the power of Eclipse technologies

Examples: Extension Point vs. Declarative Service

- Service Provider: Extension Point → Declarative Service
  - Add `@Component` to implementation
  - Delete plugin.xml
  - Create component references via `@Reference`
Unleash the power of Eclipse technologies

Examples: E3 vs. E4

Service Consumer: **Extension Point / E3**

```java
public class AmaltheaModelMigrationHandler extends AbstractModelConverterHandler {

    @Override
    public Object execute(final ExecutionEvent event) throws ExecutionException {
        // extract Shell and selection from event
        ...
    }

    IExtensionRegistry registry = Platform.getExtensionRegistry();
    IExtensionPoint extensionPoint = registry.getExtensionPoint("org.eclipse.app4mc.amalthea.model.converters.cachebuilders");
    IConfigurationElement[] extensions = extensionPoint.getConfigurationElements();

    for (IConfigurationElement iConfigurationElement : extensions) {
        String definedInputModelVersion = iConfigurationElement.getAttribute("input-model-version");
        if (definedInputModelVersion.equals(inputModelVersion)) {
            Object cacheBuilder = iConfigurationElement.createExecutableExtension("class");
            caches.add((ICache) cacheBuilder);
        }
    }
```
Unleash the power of Eclipse technologies

Examples: E3 vs. E4

Service Consumer: **Declarative Service / E4**

```java
public class AmaltheaModelMigrationHandler {

  @Execute
  public void execute(
    Shell shell,
    @Service MigrationProcessor migrationProcessor,
    @Named(IServiceConstants.ACTIVE_SELECTION) ISelection selection,
    @Optional @Named("executioncontext") String executionContext,
    @Optional @Named("APP4MC_MODEL_VERSION") String modelEditorVersion) {

    // implement the desired functionality
    ...
  }
```
Unleash the power of Eclipse technologies

Examples: Events

Loose coupling: Trigger model migration from model editor

```java
@Inject
@Optional
private void handleConvertEvent(
    @UIEventTopic("org/eclipse/app4mc/amalthea/converter/CONVERT") Map<String, String> conversionArgs,
    @Service MigrationProcessor migrationProcessor,
    @Named(IServiceConstants.ACTIVE_SELECTION) ISelection selection) {

    String type = conversionArgs.get("type");
    String version = conversionArgs.get("version");

    AmaltheaModelMigrationHandler handler = new AmaltheaModelMigrationHandler();
    handler.execute(Display.getDefault().getActiveShell(), migrationProcessor, selection, type, version);
}
```

```
IEventBroker broker = getSite().getService(IEventBroker.class);
if (broker != null) {
    String executionContext = (result == 0) ? "simplemigration" : "dialogmigration";
    HashMap<String, String> args = new HashMap<>();
    args.put("type", executionContext);
    args.put("version", versionFromMetaModel);
    broker.send("org/eclipse/app4mc/amalthea/converter/CONVERT", args);
}
```
Unleash the power of Eclipse technologies

Examples: Events

- Loose coupling: Trigger selection in model editor tree from anywhere
- Any contributor can trigger the selection in the tree viewer via event without any tight dependency on app4mc

```java
public void addSelectionSubscriber() {
    IEventBroker broker = getSite().getService(IEventBroker.class);
    if (broker != null) {
        broker.subscribe("org/eclipse/app4mc/amalthea/editor/SELECT", this.selectionListener);
    }
}
```

```java
@Execute
public void execute(@Active MPart activePart, IEventBroker broker) {
    if (activePart.getObject() instanceof VisualizationPart) {
        VisualizationPart part = (VisualizationPart) activePart.getObject();
        if (part.hasActiveModelElement()) {
            HashMap<String, Object> data = new HashMap<>();
            data.put("modelElements", part.getActiveModelElements());
            broker.send("org/eclipse/app4mc/amalthea/editor/SELECT", data);
        }
    }
}
```
Benefits

Eclipse APP4MC Examples
Unleash the power of Eclipse technologies

Eclipse APP4MC Examples

- Migration Framework
  - Small command line application

- Cloud Services

- Model Validation Framework

- Model Visualization Framework
Unleash the power of Eclipse technologies
Command line application

▶ Create small Equinox “headless” application using Tycho
  ▶ About 5MB
  ▶ Folder structure
  ▶ Native executable

▶ Create a small executable jar using bnd Maven plugins
  ▶ About 4,5 MB
  ▶ Single executable JAR

http://blog.vogella.com/2020/01/20/building-a-headless-rcp-application-with-tycho/
Unleash the power of Eclipse technologies

**APP4MC Cloud Services**

- Create cloud services using OSGi specifications
  - HTTP Whiteboard
  - JAX-RS Whiteboard
- Extract APP4MC functions/services as services in a cloud infrastructure
- Chain APP4MC cloud services without an installed platform

Migration  ➔  Validation  ➔  Transformation  ➔  Simulation  ➔  Visualization

http://blog.vogella.com/2017/04/20/access-osgi-services-via-web-interface/
http://blog.vogella.com/2021/01/08/build-rest-services-with-osgi-jax-rs-whiteboard/
Unleash the power of Eclipse technologies
APP4MC – Validation Framework

- Flexible framework based on standard EMF validations

- Main concepts
  - **Validation**
    implements validation(s) for a specific model class
  - **Profile**
    allows grouping and (hierarchical) structuring of validations

- Details are specified with Java annotations
- Top level profiles are OSGi components
Unleash the power of Eclipse technologies
APP4MC – Validation Framework Example

```java
@Component
public class AmaltheaProfile implements IProfileConfiguration {
}

@Profile(name = "Hardware Validation")
@ValidationGroup(
    severity = Severity.ERROR,
    validations = {
        AmaltheaStructure.class,
        AmaltheaPort.class,
        AmaltheaConnection.class,
        AmaltheaAccessPath.class,
        AmaltheaDefinition.class,
    }
)

@Validation(id = "AM-HW-Connection",
    checks = {
        "HWConnections must refer to two HwPorts",
        "HWConnections must be linked to HwPorts of the same Interface"
    })
public class AmaltheaConnection extends AmaltheaValidation {
    @Override
    public EClassifier getClassifier() {
        return ePackage.getHWConnection();
    }
    @Override
    public void validate(final EObject object, final List<ValidationDiagnostic> results) {
        ...
    }

    Hardware Validations
    AM-HW-AccessPath
    * HwAccessPath ranges and memory size must be consistent
    * HwAccessPath elements must be consistent
    AM-HW-Port-Definition
    * PortType must be set
    * PortInterface must be set
    AM-HW-Connection
    * HWConnections must refer to two HwPorts
    * HWConnections must be linked to HwPorts of the same Interface
    AM-HW-Port
    * A HwPort can only have one (non internal) HwConnection
    AM-HW-Structure
    * Connections must only refer to contained HwPorts
    * Inner connections always need one Initiator and one Responder HwPort
    * Delegated connections always connect HwPorts of the same type
    AM-HW-Definition
    * Only one feature of a category can be referred
    AM-HW-Module-Definition
    * Memory definition must be set
    * ProcessingUnit definition must be set
    * ConnectionHandler definition must be set
    * Cache definition must be set
```
Unleash the power of Eclipse technologies

APP4MC – Visualization Framework Example

- Generic view part integrated in APP4MC
  - Consumes OSGi services of type `Visualization`
    \( \rightarrow \) Make it easy to contribute
  - Executes methods annotated with `@PostConstruct` via Eclipse 4 injection
    \( \rightarrow \) Make it easy to implement

```java
@Component(property= {
   "name=Runnable Visualization Demo",
   "description= Shows the runnable name (demo)"
})
public class RunnableDemo implements Visualization {

    @PostConstruct
    public void createVisualization(Runnable runnable, Composite parent) {
        Label label = new Label(parent, SWT.NONE);
        label.setText(runnable.getQualifiedName());
    }
}
```
Unleash the power of Eclipse technologies
APP4MC – Visualization Framework Example

Complete code of a demo visualization component

```java
@Component(property={
    "name=Runnable Visualization Demo",
    "description=Shows the runnable name (demo)"
})
public class RunnableDemo implements Visualization {
    @PostConstruct
    public void createVisualization(Runnable runnable, Composite parent) {
        Label label = new Label(parent, SWT.NONE);
        label.setText(runnable.getQualifiedName());
    }
}
```
Unleash the power of Eclipse technologies

APP4MC – Editor Contributions

```java
@Component
public class AddModificationTimestamp implements ProcessingService {
    private final SimpleDateFormat timestampFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

    @PostConstruct
    public void addTimestamp(Innotatable object) {
        CustomPropertyUtil.customPut(object, "modified", timestampFormat.format(new Date()));
    }
}
```

Complete code of a simple editor action
Conclusion
Unleash the power of Eclipse technologies

Conclusion

- Main achievements
  - smaller code base
  - reduced maintenance effort
  - simplified contribution and adoption
  - easier to consume (plain Java, ...)

- Advantages (of the proposed actions)
  - independent steps
  - can be applied gradually
Thank you!

Join the conversation:

@EclipseCon | #EclipseCon
Evaluate the Sessions

Vote in Swapcard

Rate the session
Share your feedback with the organizers.

⭐⭐⭐⭐⭐

Your feedback will be only visible to the organizer.

Not bad :-(

SEND

ECLIPSE 2021 CON