logi.CAD3 → logi.CLOUD

An experience report on migrating an industrial-grade IDE to the Cloud using the Eclipse ecosystem
Agenda of this Talk

• Introduction (7 Minutes)
  • Goal
  • Company focus (products, technical)
  • High-level goal
• Experience report (20 Minutes)
  • Architectural decisions
  • Challenges
• Outlook (3 Minutes)
  • Future steps
• Q&A (5 Minutes)
Goals of this Talk

- Usage scenario of Eclipse technologies
  - Industrial Automation
- Sharing experiences: Make Fat Desktop App usable in Cloud Environment
  - Overview of frameworks and projects
  - Challenges, Pitfalls
  - How we got started
About logi.cals

1987 | 35 | 2

FOUNDED | EMPLOYEES | OFFICES
Overview: Our (Original) Goal
IDE for Programmable Logic Controllers

- IEC 61131-3
  - Structure Text (ST)
  - Function Block Diagram (FBD)
  - Sequential Function Chart (SFC)
Overview: Our (Original) Goal
IDE for Programmable Logic Controllers
Solutions Overview

**AUTOMOTIVE**
80% of all buses in Europe

**BUILDING**
Thousands of controllers for security and comfort in modern ships

**ENERGY**
30% of all hydro-electric power plants

**MOBILE AUTOMATION**
Controllers for off-highway and agricultural machines

**SYSTEM**
Projects with 6 million I/O and communication data points

Logi.cals

---

Solutions Overview

**AUTOMOTIVE**
80% of all buses in Europe

**BUILDING**
Thousands of controllers for security and comfort in modern ships

**ENERGY**
30% of all hydro-electric power plants

**MOBILE AUTOMATION**
Controllers for off-highway and agricultural machines

**SYSTEM**
Projects with 6 million I/O and communication data points

Logi.cals

---
Overview: Current State
High-Level Architecture

- logi.CAD3
  - ST-/FBD-editor
  - Binary Build Infrastructure
  - Target Communication

- logi.RTS
  - Target-Binary (Shared Object)
  - Scheduler
  - Target-Binary (Shared Object)

TCF
Structure of logi.CAD3
Architecture and Components

- FBD-Editor
- ST-Editor
- Xcore Model
- Xtext
- EMF Model Elements (EObjects)
- Formatter
- Synchronizer/Transformer
- Project Index
- Project Database
- Validator
- Resolver
- C-Code Generator
- Binary Code Generation
- .c/.h files
- OEM SDK
- Variable Values View
- Instance View
- Target Proxy
- Gateway
- BuiltInPlc
- Platform Toolkits

WE EMPOWER AUTOMATION
Structure of logi.CAD3
What will be re-used for the Cloud?

- FBD-Editor
- ST-Editor
- Xcore Model
- Xtext
- EMF Model Elements (EObjects)
- Formatter
- Synchronizer/Transformer
- Project Index
- Project Database
- Validator
- Resolver
- C-Code Generator
- Binary Code Generation
- .c/.h files
- OEM SDK
- Variable Values View
- Instance View
- COMM
  - Target Proxy
  - Gateway
  - BuiltInPlc
- Platform Toolkits
- WE EMPower Automation
Structure of logi.CAD3
What will be re-used for the Cloud?

- FBD-Editor
- ST-Editor
- Xcore Model
- Xtext
- EMF Model Elements (EObjects)
- Formatter
- Synchronizer/Transformer
- Project Index
- Project Database
- Validator
- Resolver
- C-Code Generator
- .c/.h files
- OEM SDK
- Variable Values View
- Instance View
- Target Proxy
- Gateway
- BuiltInPlc
- Binary Code Generation
- Platform Toolkits
- OEM SDK
logi.CAD3 offers public API

- logi.CAD3 is a tool integrated by OEMs
- API is pretty comprehensive

Makes it quite simple to

- Extend logi.CAD3 itself
- Create stand-alone applications (with extensions)
Architecture of logi.Cloud
Architecture and Components

Server
- LC Server
- IEC support
  - Xtext via LSP4J
- IEC FBD via GLSP
- Project Index
- Read
- Write into DB
- LC3 API Server
- LC Target Com Server
- LC Build Server

Theia Backend
- ... (Theia core stuff)
- IEC Language Backend
- IEC FBD Backend
- LC API Service Backend
- LC Target Com Backend
- LC Build Service Backend

HTTP GET
Web Socket
LSP over Web Socket
GLSP over Web Socket

Browser
- Theia Frontend
- ... (Theia core stuff)
- IEC Language Frontend
- IEC FBD Frontend
- ... Instance View
- LC3 Service
- Target Com Service
- Build Service

WE EMPOWER AUTOMATION
Key Enablers
Making logi.CAD3 ready for the Cloud

- **Eclipse projects**
  - Xtext: language mechanisms
  - LSP4J: JSON RPC over WebSockets (incl. support for LSP)
  - Eclipse Theia: Frontend/Backend framework
  - Eclipse GLSP: Graphical Language Server Protocol
  - Eclipse Che: future execution context of logi.CLOUD
Challenges: Dependency Categorization
Making logi.CAD3 ready for the Cloud

- **UI dependencies**
  - Rely on Eclipse UI to work
  - **Example**: custom editors

- **Eclipse dependencies**
  - Rely on common Eclipse functionality (but no UI)
  - **Example**: Eclipse builders, Workspace

- **OSGi**
  - Low-level Eclipse functionality
  - **Example**: Service access

- **OK**
  - Functionality that works in any Java environment
  - **Example**: (Generated) Xtext parsers
Challenges: Dependencies
Making logi.CAD3 ready for the Cloud

• Dependency Analysis of Java Code using CDA
  • Class Dependency Analyzer
    http://www.dependency-analyzer.org/
  • Handles class, package, and plug-in level dependencies
  • Can visualize dependency hierarchies

• Dependencies exported to CSV format

• Categorization and aggregation in Spreadsheet Software
  • Identification of (potentially) problematic dependencies
## Challenges: Dependency Categorization

Making logi.CAD3 ready for the Cloud

<table>
<thead>
<tr>
<th>Plugin</th>
<th>Included in heads</th>
<th>Direct dependency types (plugin)</th>
<th>Direct dependency types (class)</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.logicals.buildservice_2.6.0.201909060757.jar</td>
<td>Yes</td>
<td>Eclipse 2 UI 1 OSGi 7 ?</td>
<td>Eclipse 12 UI 2 OSGi 7 ?</td>
</tr>
<tr>
<td>com.logicals.common_2.6.0.201909060757.jar</td>
<td>Yes</td>
<td>2 2 4 2 w/p</td>
<td>13 3 21 22</td>
</tr>
<tr>
<td>com.logicals.debug_common_2.6.0.201909060757.jar</td>
<td>Yes</td>
<td>4 4 4 2</td>
<td>16 8 14 26</td>
</tr>
<tr>
<td>com.logicals.iec.fbd_2.6.0.201909060757.jar</td>
<td>Yes</td>
<td>1 0 0 1</td>
<td>1 0 7 5</td>
</tr>
<tr>
<td>com.logicals.iec.model_2.6.0.201909060757.jar</td>
<td>Yes</td>
<td>2 1 3 1</td>
<td>9 4 10 1</td>
</tr>
<tr>
<td>com.logicals.iec.st_2.6.0.201909060757.jar</td>
<td>Yes</td>
<td>2 0 3 1</td>
<td>7 0 8 2</td>
</tr>
<tr>
<td>com.logicals.iec.st.ide_2.6.0.201909060757.jar</td>
<td>Yes</td>
<td>2 2 2 0</td>
<td>26 9 12 0</td>
</tr>
<tr>
<td>com.logicals.lc3_api_2.6.0.201909060757.jar</td>
<td>Yes</td>
<td>2 2 4 0 w/p</td>
<td>5 3 17 0</td>
</tr>
<tr>
<td>com.logicals.lc3.api.module.st.impl_2.6.0.201909060757.jar</td>
<td>Yes</td>
<td>0 0 2 1 w/p</td>
<td>0 0 2 1</td>
</tr>
<tr>
<td>com.logicals.lc3.api.mrc_2.6.0.201909060757.jar</td>
<td>Yes</td>
<td>1 0 2 1</td>
<td>4 0 4 1</td>
</tr>
<tr>
<td>com.logicals.lc3.help_2.6.0.201909060757.jar</td>
<td>Yes</td>
<td>1 2 2 3</td>
<td>1 7 5 9</td>
</tr>
<tr>
<td>com.logicals.library_2.6.0.201909060757.jar</td>
<td>Yes</td>
<td>1 0 0 1</td>
<td>9 0 13 4</td>
</tr>
<tr>
<td>com.logicals.mrc_2.6.0.201909060757.jar</td>
<td>Yes</td>
<td>2 4 4 1 w/p</td>
<td>9 18 10 1</td>
</tr>
<tr>
<td>com.logicals.projectindex_2.6.0.201909060757.jar</td>
<td>Yes</td>
<td>2 1 3 1</td>
<td>11 2 18 7</td>
</tr>
<tr>
<td>com.logicals.targetproxy_2.6.0.201909060757.jar</td>
<td>Yes</td>
<td>1 2 3 1</td>
<td>1 3 8 10</td>
</tr>
<tr>
<td>com.logicals.uploader.lc3uploader_2.6.0.201909060757.jar</td>
<td>Yes</td>
<td>1 1 2 0</td>
<td>4 2 6 0</td>
</tr>
</tbody>
</table>
**Challenges: Workspace handling**

Making logi.CAD3 ready for the Cloud

**Challenge:**
- Eclipse Workspace depending functionality (Project Index; H2-based)
- Theia IDE does not know Eclipse Workspaces

**Solution:**
- Custom “Cloud-aware” Workspace implementation
- Will behave differently to Eclipse file system
- Limits possibility of shared code with logi.CAD3
Challenges: Workspace handling
Making logi.CAD3 ready for the Cloud

- **Workspace types being used**
  - Theia Backend: FS/TS-based workspace
  - logi.CAD3 Server: Cloud-aware workspace
  - logi.CAD3 Build Server: Eclipse workspace (generated; snapshot)
  - logi.CAD3 Target Comm Server: none needed
Outlook
The (near) Future

• Full support for C/C++ editing
  • Based on clangd

• Diagnostic Features
  • View variable values

• Continue FBD support
  • Variable table
  • Context menus
  • Full editing, e.g. routing for connections
  • Copy-paste
  • ...

WE EMPOWER AUTOMATION
Outlook
The (far) Future

- **Integration of new platforms**
  - Licensing issues/challenges
- **Workspace management**
  - Based on Eclipse Che
- **Decision on the execution platform for logi.CLOUD**
  - RedHat OpenShift, MS Azure, …
Outlook
The (far) Future

- **Offering logi.CLOUD as an integrable solution**
  - logi.cals products typically sold as integrable framework
  - OEMs use our product and add branding
  - OEMs expect a plugin infrastructure
Summary

Findings, Recommendations

• **Build Proof-of-Concept Prototypes**
  • Based on well-defined use-cases
  • Getting to know new technologies and frameworks
  • Helps to estimate the time frame for developing the actual product
Bibliography

Further reading

- **Overall Architecture**
  - (Initial) Eclipse Theia Design Document:
    https://docs.google.com/document/d/1aodR1LJEZu7xBis2MjpHRyv7JKJzW7EWI9XRYCt48/
  - EclipseSource Documents on Eclipse Theia and Eclipse Che:
    https://eclipsesource.com/blogs

- **Getting Started (Motivational example)**
  - https://theia-ide.org/docs/
Thank you for your attention!
Challenges: Additional concerns
Making logi.CAD3 ready for the Cloud

LC3/logi.CLOUD: General interface of LC3 projects:

```java
public interface Lc3Project extends Comparable<Lc3Project> {
    [...] File getFile(String name);
}
```
Challenges: Additional concerns
Making logi.CAD3 ready for the Cloud

LC3: Using Eclipse IProject to retrieve files

```java
@Override
public File getFile(final String name) {
    final IFile eclipseFile = this.eclipseProject.getFile(name);
    return new File(eclipseFile.getLocation().toOSString());
}
```

logi.CLOUD: Using Language Server Project Manager to retrieve files (LC3 Server; JAVA)

```java
@Override
public File getFile(final String name) {
    final URI baseDir = this.projectManager.getBaseDir();
    return new File(baseDir.toFileString() + name);
}
```
Challenges: Additional concerns
Making logi.CAD3 ready for the Cloud

LC3: Using Eclipse Workspace and Projects to initialize Project Index

```java
private void initializeProjects() {
    final IProject[] iProjects = ResourcesPlugin.get Workspace().getRoot().getProjects();
    for (final IProject iProject : iProjects) {
        if (iProject.isOpen() && IECProjectNature.isIECProject(iProject)) {
            final Lc3Project project = new Lc3ProjectImpl(iProject);
            projects.put(project.getName(), project);
        }
    }
}
```
Challenges: Additional concerns
Making logi.CAD3 ready for the Cloud

logi.CLOUD: Use Language Server Workspace Manager and Project Manager
to initialize Project Index (LC3 Server; JAVA)

```java
private void initializeProjects() {
    for (final ProjectManager projectManager : workspaceManager.getProjectManagers()) {
        if (projectManager.getBaseDir() != null) {
            final Lc3Project project = new LC3CloudProject(projectManager);
            projects.put(project.getName(), project);
        }
    }
}
```
Challenges: Additional concerns
Making logi.CAD3 ready for the Cloud

LC3: Build and Upload Program triggered by client

```java
protected void buildAndLoadApplication(final ActionEvent event) {
    TargetSession targetSession = TargetProxy.getTargetSessionForUI(resource);
    new BuildProgramJob(targetSession.getResource(), new BuildOptions()).schedule();
    new UploadProgramJob(targetSession).schedule();
}

public static NormalizedFile getTargetOutputDirectory(IProject project) {
    return new NormalizedFile(project.getLocation().toFile()).append("target");
}
```
**Challenges: Additional concerns**
Making logi.CAD3 ready for the Cloud

**logi.CLOUD: Build and Upload Program triggered by client (Theia Backend; TypeScript)**
```javascript
async buildAndLoadApplication(connectionInfo: Lc3ResourceConnectionInfo): Promise<void> {
  const jobId = await this.buildService.buildWithProgress(connectionInfo.resource.instancePath);
  await this.buildService.onJobFinished(jobId);
  const buildLocation = await this.buildService.getBuildLocation();
  return this.targetService.upload(connectionInfo, buildLocation);
}
```

**logi.CLOUD: Determining the actual build location (LC3 Build Server; JAVA)**
```java
public NormalizedFile getTargetOutputDirectory(String buildLocation, String projectName) {
  return new NormalizedFile(buildLocation).append(projectName).append("target");
}
```
Overview: PLC-Programming
Programming Languages: ST (IEC61131-3)