Veronika Domova, Ettore Ferranti, Thijmen de Gooijer, Aneta Vulgarakis

4DIAC integration into the FASA project

A success story of increased maintainability and modularity

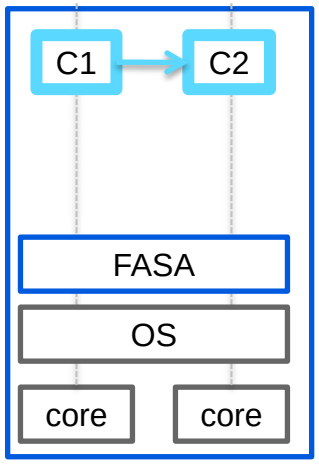
Agenda

- Background and motivation
- Project idea
- Development approach
- Tools used for implementation
- Implementation logic
- End-user workflow
- Summary and conclusions

Background(1)
The Future Automation System Architecture (FASA)

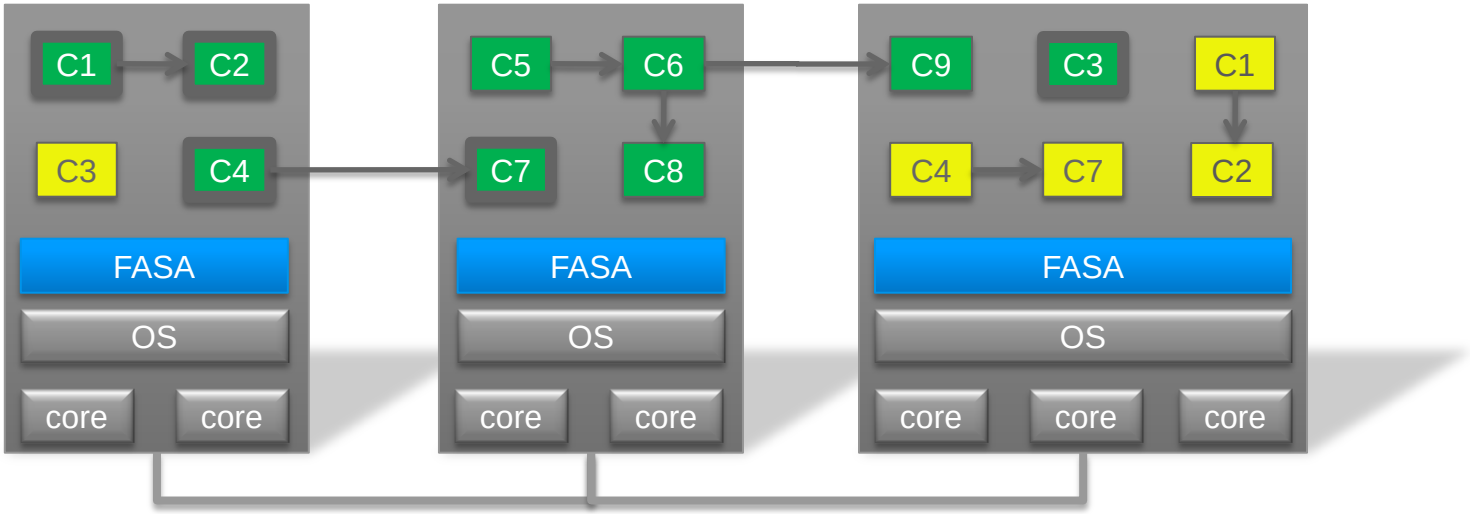
Experimental distributed control system framework

- Real-time middleware
- Loosely based on IEC61499 standard
- Based on components
- Implemented in C++
- Linear execution



Background(2)
The Future Automation System Architecture (FASA)

- **Execution transparency** for control applications across multiple controllers and CPU cores
- **Dynamic changes** to the control system configuration without any disruption



Motivation

- FASA **lacks an IDE** to create, deploy and edit its applications
- Large amount of C++ code and configuration files have to be created and maintained **manually**
- Development and testing processes are noticeably **slowed down**

Project Idea

What?

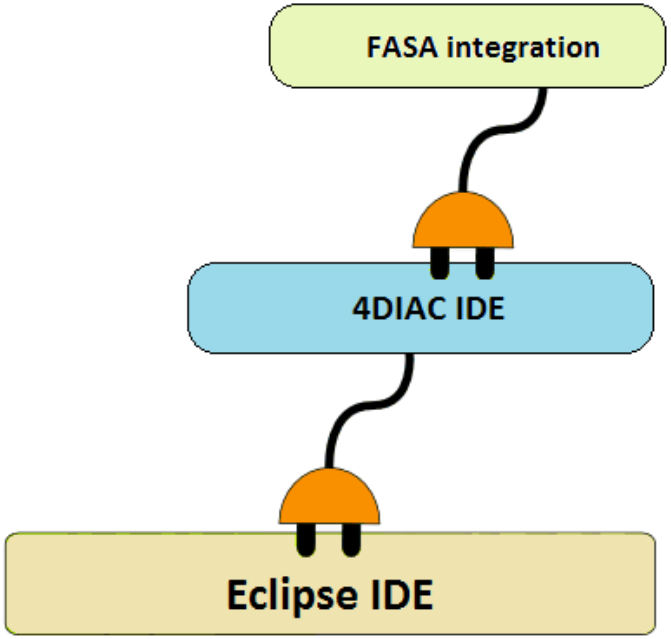
- Create applications in 4DIAC IDE
- Automatically generate FASA code

How?

- Implement 4DIAC IDE – FASA integration
- Make an extension to the 4DIAC IDE platform
- Implement the extension as an Eclipse plugin

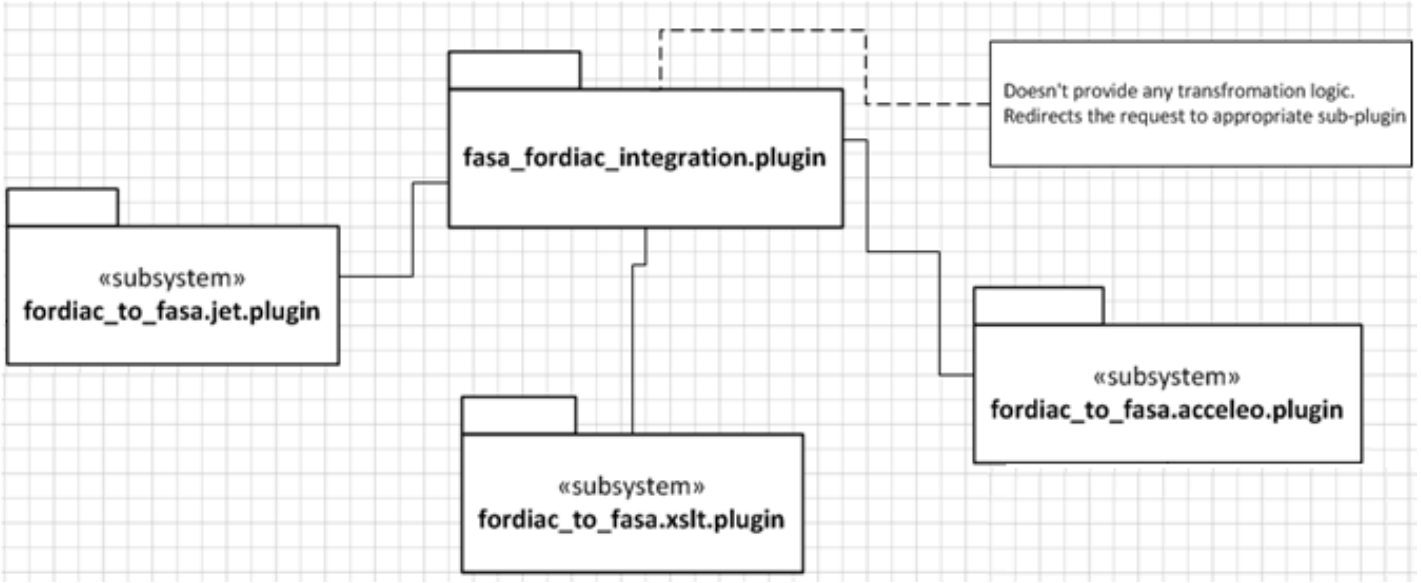
Approach

- Implement an Eclipse plugin
- Rely on 4DIAC IDE model
- Generate FASA code using M2T transformation



Tools used for implementation

- **Eclipse IDE, Java**
- **Model-to-text transformation tools**:
  - XSLT
  - Acceleo
  - Jet



Implementation logic

- Retrieving necessary objects using 4DIAC IDE model and API
- Applying JET transformation templates
- Dynamically mapping data in the templates
- Generating output folders structure
- Generating output files into appropriate folders

End-user Workflow

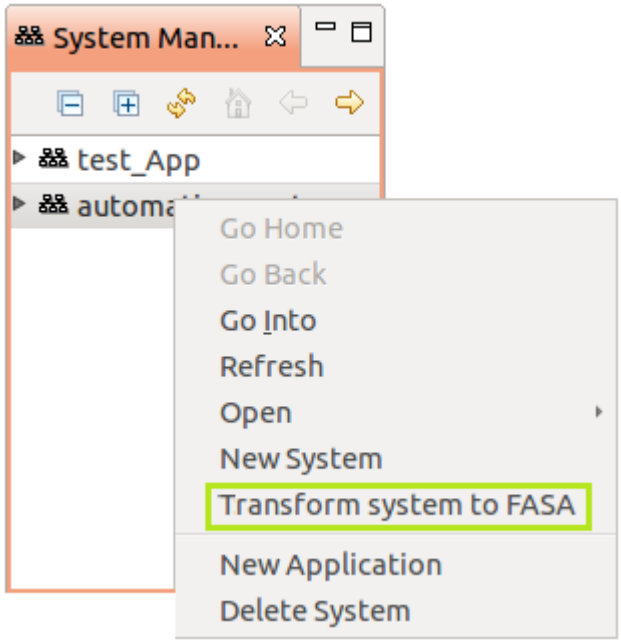
Step 1: Create a system in 4DIAC IDE

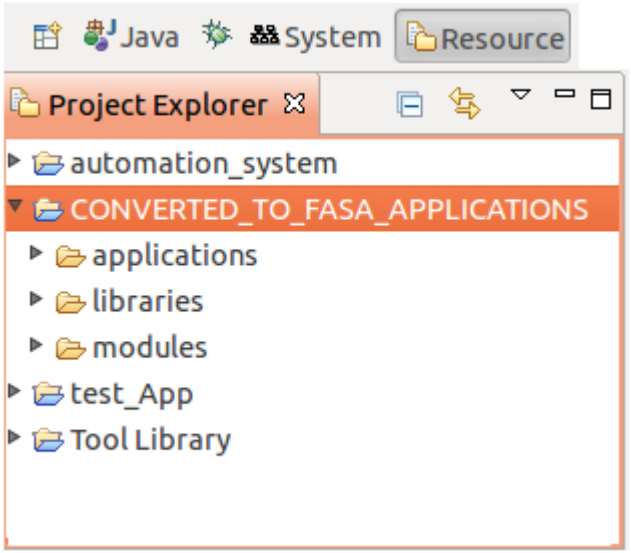
Step 2: Right-click on the system in the browser

Step 3: In appeared menu select ”Transform system to FASA”

Step 4: Wait a moment until all the transformation is done

Step 5: Observe the generated code in the workspace





Summary and Conclusions

Contribution to FASA

- Graphical IDE for FASA applications
- Increased maintainability and modularity

Future work

- Improving integration logic
- Making the integration process entirely automatic

Contribution to 4DIAC

- Application in a new automation domain
- Extendibility and integration with other systems

Power and productivity
for a better world™