Papyrus UML
the first stage of a journey to the cloud

Remi Schnekenburger
rschnekenburger@eclipsesource.com
Papyrus

- Open Source UML modeling platform
  - Based on the Eclipse Modeling Framework
  - Based on modeling standards: UML, SysML, fUML, Alf, OCL, …
  - Supported by an active open-source community
  - Enables to build domain-specific tools based on UML, SysML, etc.
Is Papyrus only a diagram editor?

- Rendering definition
- Editing commands & Element Types
- Palette definition
- Serialization
- Model management
- UML Support
- UML Profile definition & usage
- Validation
- UI Customization
- Rendering technology
- UI technology
Is Papyrus only a diagram editor?

Rendering definition
Editing commands & Element Types
Palette definition

Serialization
Model management
UI Customization
Rendering technology
UI technology

UML Support
UML Profile definition & usage
Validation
Papyrus and underlying technologies for diagraming

- Papyrus diagrams depend on
  - Eclipse GMF (Graphical Modeling Framework),
  - GEF3 (Graphical Editing Framework)
  - SWT (Standard Widget Toolkit)
  - CSS integration

- Lots of functionality
  - Hardly possible and desirable to re-implement

- How can we keep a lot of the functionality?
  - But evolve the UI & rendering technology
Past experiments

GEF3/GMF to GEF5/JavaFX
⇒ keeping the same notation
⇒ reuse some of the GMF.tooling
Emergence of web technologies

- Theia
- EMF.cloud
- GLSP (and LSP!)
- Sprotty
Focus on Eclipse Theia

- IDE platform for both desktop or web based IDEs sharing the same code base
- Flexible Architecture, with dependency injection and modularization
- Eclipse project
- Already adopted by the industry:
  - Ericsson
  - Arm
  - RedHat
Focus on Eclipse GLSP and Sprotty

Development of web-based diagram editors

- Sprotty / SVG based rendering
- Generic GLSP diagram client with extensibility
- Domain specific language server encapsulation
Focus on EMF.cloud

Umbrella for several projects, among:

- Model Server
- Theia Tree editor
- Editors Examples
  - Ecore GLSP
  - Coffee editor
More details on these technologies!

- **Model validation, diffing and more with EMF.cloud**
  Wed 27.10., 16:10 CET

- **Building a model application with Theia and EMF.cloud: the DISCO experience**
  Thu, 28.10., 16:50 CET

- **Diagram editors with Eclipse GLSP - 1.0**
  Tue, 26.10., 14:30 CET

- **Migrating Eclipse-based Tools/Plugins to Eclipse Theia or VSCode**
  Tue, 26.10., 16:50 CET

- **Spotlight session**
  Tue, 26.10., 17:30 CET

- **Birds of a Feather (BoF): Building (web-based) tools with Eclipse**
  Wed, 27.10., 18:15 CET
More experiments

● In collaboration with TU Wien
  ○ Business Informatics Group, Dominik Bork
● 5 groups of 2 students
● Implementation of 5 UML diagram editors
  ○ Activities
  ○ Use Case
  ○ State Machines
  ○ Deployment
  ○ Class
Experiment configuration

Agenda

- 4 hour training on technology stack
- Experiment spread over 3 months, 1-2 days / week

Initial technology stack

- GLSP
- GLSP Theia Integration
- Model Server with UML extension (metamodels and custom commands)
  - UML Metamodel, EMF/UML editing domain, resource set & command stack APIs
  - Some command implementation reused from Papyrus
- Generic EMF-based notation model
Experiment goals

● Diagram editors
  ○ Front-end diagram rendering
  ○ Operation handlers for CRUD of nodes and edges (partly only for a subset of language)
  ○ Implementation of UML commands on the model server

● Further customizations in the direction of their choice
  ○ Integration of validation based on Eclipse UML
  ○ Custom edge rendering
  ○ Auto-completion and label validation for property editing
Demos
Feedback from students

All groups could produce a very decent initial state of a diagram editor

Steep learning curve:

- EMF APIs / UML Metamodel / Model Server / GLSP + Sprotty with mixed languages like Typescript and java

But provides a clean architecture, with clear separation of the concerns
Feedback from students

Missing features reported

- Better default compartment support
- Multi-line support in comments
- For complex edge and node creation/reorient, client side selection may not be sufficient
- Anchoring to lines (not supported yet)

Most of them addressed since the experiment
Feedback for Papyrus project

- Diagram part only study
- Form based and table based editors to be also investigated (partially done for Property views)

An incremental transition

- Many customization frameworks may be reused initially on the server side, like profile extensions, element type framework, architecture
- Improved architecture, leveraging services & dependency-injection
Evaluate the Sessions

- Please help by leaving feedback on the sessions you attend!
- To rate a session, you must be registered for it in Swapcard BEFORE the talk starts.
- Swapcard will prompt you to leave feedback after the end of each session.
- You may also rate a talk by locating the session from the “Agenda” or “My Event” buttons on the Event Home page. Click on the session and look for the “Give your feedback” box.