The evolution of Papyrus
An open UML and SysML tool

Camille Letavernier and Philip Langer
{cletavernier|planger}@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.
Domain-specific Modeling

- Model-based engineering is most successful if it is *domain-specific*
  - Highly customized modeling environments
  - Directly reflecting specific needs of a domain and its users
  - User roles, their backgrounds, methodologies, and tool chains
Domain-specific Modeling
Domain-specific Modeling vs UML, SysML, etc.

- Standardized Modeling Languages (UML, SysML, …)
  - Reuse well-known and -proven language concepts
  - Reuse existing tools and components
  - Interoperability and connectability with other models
  - Conformance to industry standards

- Domain-specific Modeling vs. UML/SysML?
  - Contradiction? NO!
  - Papyrus can be used as a platform
    - Graphical syntax, palette, property views, editing behavior, etc.
    - Based on EMF and Eclipse RCP
Domain-specific Modeling with UML, SysML, etc.

- Thanks to the great Open Source Eclipse Modeling Ecosystem...
Papyrus as a Platform

- Adding “domain-specificness” to UML
  - “Plain” UML is a general purpose modeling language
  - UML Profiles allow to extend UML with domain-specific concepts
Papyrus as a Platform

- Building a domain-specific modeling tool
  - UML profile defines only structure of the model (metamodel)
  - Off-the-shelf UML tools provide *generic* graphical syntax
  - A domain-specific modeling environment
    - Domain-specific graphical syntax and tooling for editing models
    - Rich client platform to support domain-specific workflows, ...
    - Powerful API to process models (e.g., for codegen, validation, etc.)
Example and Demo

- **User Workflow Modeling Language**
  - A workflow contains tasks
  - Manual task and automatic tasks
  - Flow between tasks and decisions
  - Duration, responsibilities, and probabilities
Custom Tooling

- Abstract and concrete syntax is just the beginning of a custom tool
  - More customizations are required for a domain-specific modeling environment
  - Papyrus supports full customizability of palette, property views, menus, ...
  - Papyrus builds on Eclipse → rich tool development platform
    - Integration with other tools, code generators, etc.
    - Wizards, menus, toolbars, views, ...

![Diagram of Papyrus tooling features]

© 2019 EclipseSource | http://eclipsesource.com | Camille Letavernier and Philip Langer | The evolution of Papyrus, an open SysML and UML tool
What’s new around Papyrus?

- **Papyrus UMLLight**
  - Optimization of Papyrus for the most common UML features
  - Simplified menus, wizards, palettes, property views, fresh diagram style, etc.
  - Source code and customization guide publicly available *
  - Funded by the Papyrus Industry Consortium
  - Implemented by EclipseSource

- **SysML 1.6**
  - Implementation of the profile
  - Element and diagram types
  - Migration tool from SysML 1.4 to 1.6
  - Implemented by CEA

* github.com/eclipsesource/papyrus-umllight
What’s new around Papyrus?

- Papyrus Compare
  - Allows diffing and merging Papyrus models
  - Enables managing Papyrus models in Git repository
  - Enhancements of the last year
    - Huge performance and usability improvements
    - Changing merge and conflict resolution decisions
    - Property compare

 wiki.eclipse.org/Papyrus_Compare
What’s next with Papyrus?

- Today and tomorrow

- Rendering definition
- Editing commands
- Palette definition
- Element types
- Customizability
- Rendering technology
- UI technology

- Serialization
- Model management
- Language types
- Profile definition

© 2019 EclipseSource | http://eclipsesource.com | Camille Letavernier and Philip Langer | The evolution of Papyrus, an open SysML and UML tool
What’s next with Papyrus?

- Aging UI & Rendering Technology
  - Papyrus depends on GMF
  - GMF depends on GEF3
  - Rich-client: SWT → JavaFX
  - Browser-based: SWT → HTML5/CSS/SVG

- 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
Papyrus GEFx: Papyrus with GEF5.x

- Goals
  - Migration to modern diagram components & UI
  - More modular architecture, improving extension & customization
  - Compatibility has to be preserved: incremental migration

- How
  - Remove the GEF 3.x (“Draw2D”) parts of Papyrus Diagram UI
  - Re-implement Papyrus Diagram UI with GEF 5.x (“JavaFX”)

- Result
  - Papyrus Diagrams based on GEF 5.x
  - Compatible with the existing Papyrus Editor & Models
Papyrus GEFx

- **GEFx Connector**
  - Some generic services & diagram parts on top of GEF (5.x, JavaFX)

- **GMF Connector**
  - Implement GEFx services (Notation, EMF Transactions)
  - Base interactions (Move, Resize, Create, etc.)

- **Papyrus Integration**
  - Semantic models
  - Customization models & CSS (Edition logic, Palettes, Diagram Structure, ...)
  - Retain editor integration with Model Explorer & Properties
Papyrus GEFx

● Integration effort
  ○ Custom figures & labels
  ○ Custom user interaction
  ○ Custom ... (depends on the diagram implementation)

● Benefits
  ○ Modern UI toolkit (JavaFX)
  ○ Compatibility with Papyrus, including customizations
  ○ Reuse of GMF Runtime (or EMF/UML)
  ○ Integrated with Eclipse, but not based on it
  ○ Improved architecture, leveraging services & dependency-injection
Papyrus GEFx with GLSP: Web-based Papyrus

- **GLSP (Graphical Language Server Protocol)**
  - Enables browser-based clients
  - Protocol for abstracting the diagram logic from the rendering technology

- **GLSP connector**
  - Server side: based on Papyrus-GEFx
  - Client side: generic web client based on GLSP client (Theia, Sprotty)

- **Any diagram supported by Papyrus-GEFx can be opened with GLSP-GEFx**
  - Structure & edition is supported by the server
  - Server can control styling (via CSS classes and diagram structure)
  - But implementation is required on the client for rendering
Top Left: Papyrus (GMF + Draw2D)
Bottom Left: Papyrus-GEFx (JavaFX)
Top Right: GEFx-GLSP (Web-based, Sprotty/Theia)
Thanks a lot!

- Contact us for more details
  - planger@eclipsesource.com
  - cletavernier@eclipsesource.com

- Links
  - https://www.eclipse.org/papyrus/download.html
  - https://github.com/eclipsesource/papyrus-uml-light
  - https://wiki.eclipse.org/Papyrus_Compare
  - https://github.com/eclipsesource/papyrus-gefx
  - https://github.com/eclipsesource/papyrus-gefx-glsp