EclipseSource

Diagrams in web and space with Eclipse GLSP

(Graphical Language Server Platform)

Philip Langer

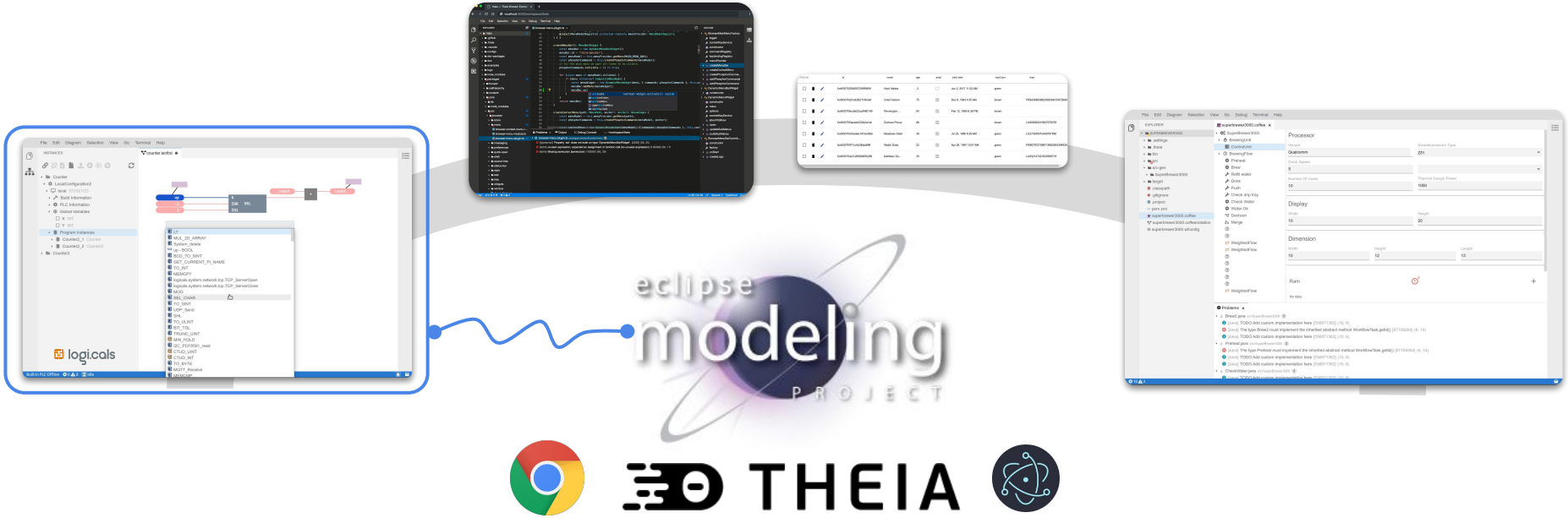
planger@eclipsesource.com

Martin Fleck

mfleck@eclipsesource.com

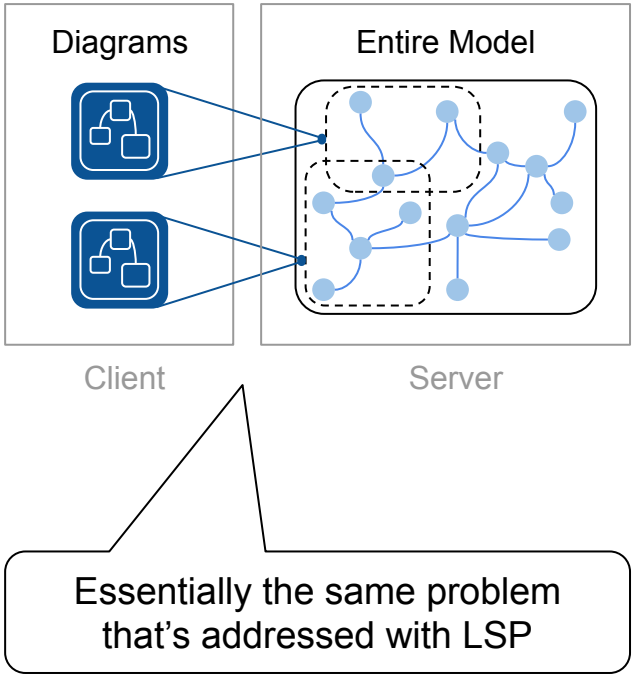
ECLIPSE INCUBATION

Building domain-specific cloud-based (modeling) tools



Client-server architecture for diagram editors

- Re-use Eclipse Modeling Technologies
  → Modeling language “smarts” runs on a Java-based server
- Migration of existing diagram editors
  → Only rendering & edit UI depends on client
- One diagram != one model
  → Whole-model understanding required for editing

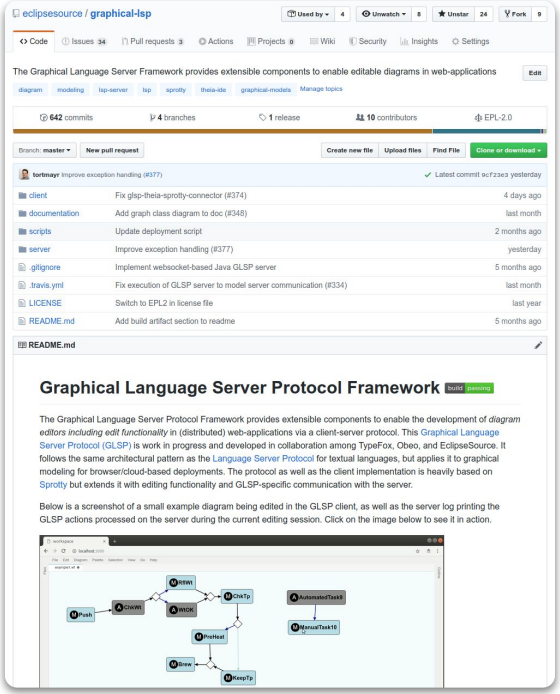


Eclipse Graphical Language Server Platform (GLSP)

Applying the architectural pattern of LSP to graphical modeling

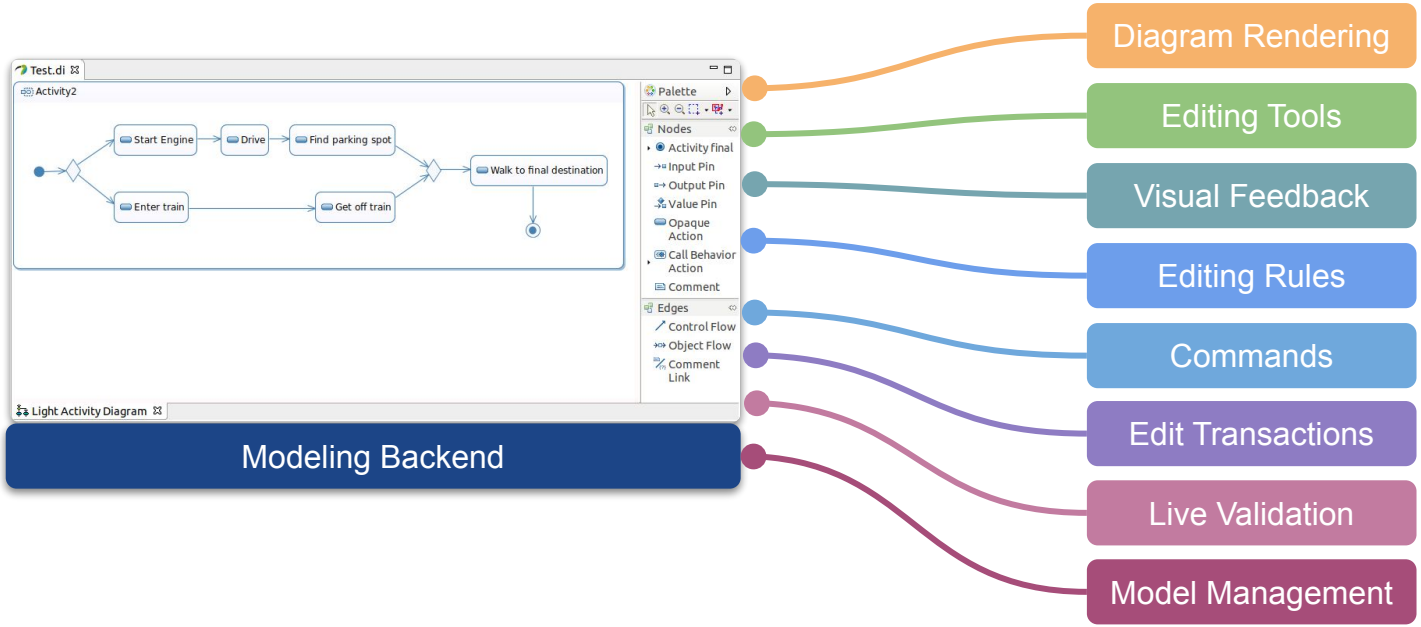
- Enable development of web-based diagram clients
  - Or clients in any technology
  - Decouple client implementation from modeling language implementation
- Encapsulate language know-how on the server
  - Reuse of existing frameworks & diagram implementation
  - Management of large models
- Front-end focused on rendering & user interaction
  - Everything else is obtained from the server
  - With the minimum amount of rountrips

ECLIPSE INCUBATION

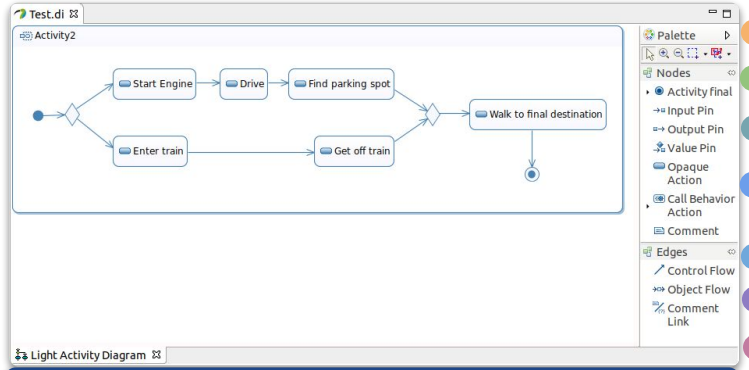


github.com/eclipsesource/graphical-lsp

Separation of Concerns with GLSP



Separation of Concerns with GLSP



Modeling Backend

Diagram Rendering

Editing Tools

Visual Feedback

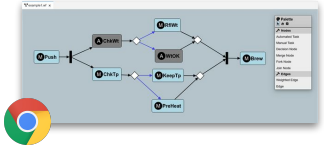
Editing Rules

Commands

Edit Transactions

Live Validation

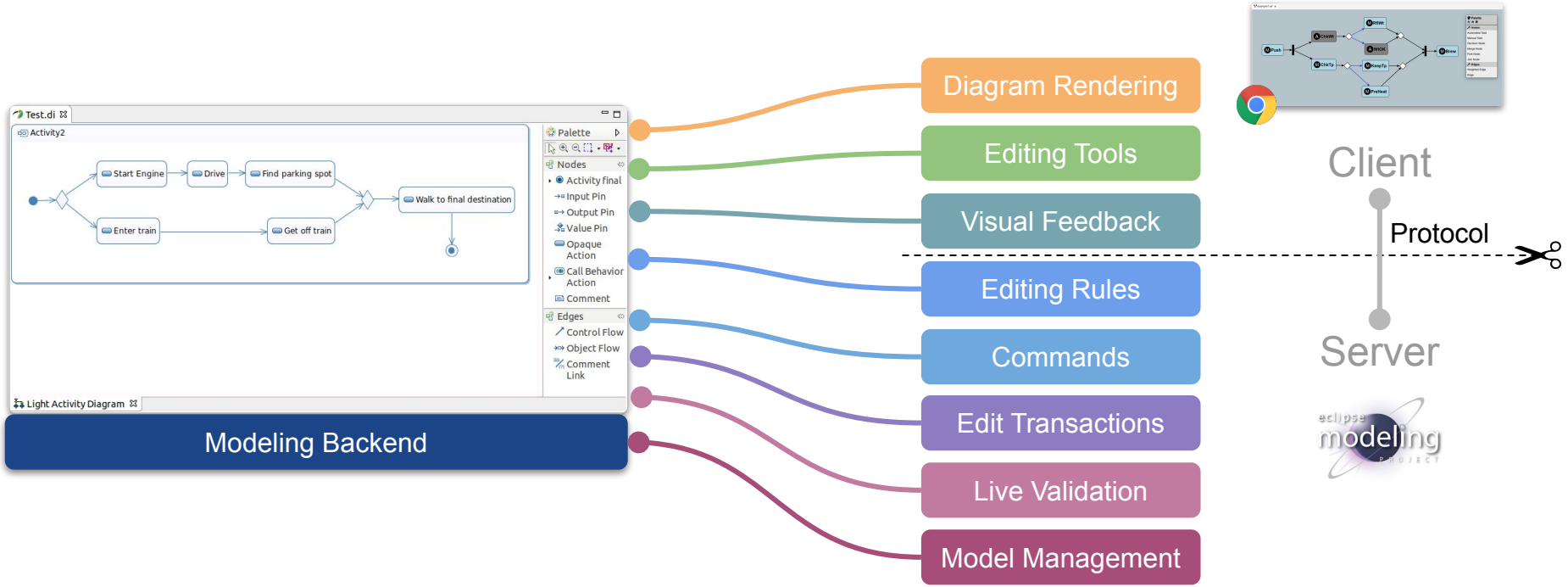
Model Management



Client

Server

Separation of Concerns with GLSP

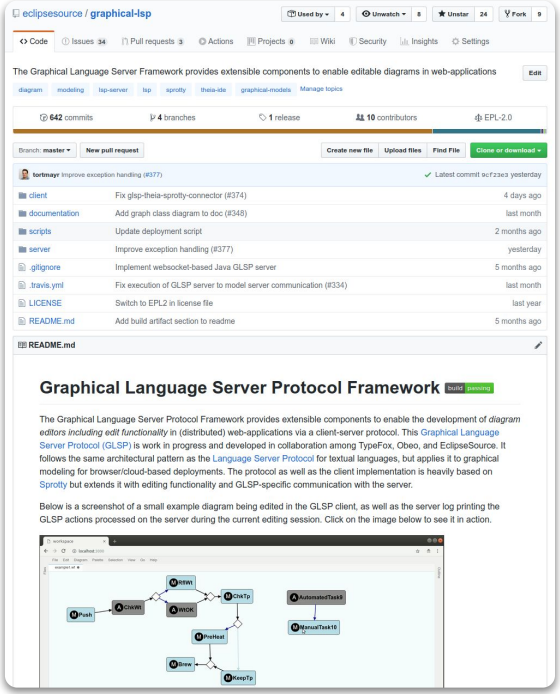


Eclipse Graphical Language Server Platform (GLSP)

1. Java-based server framework
   - Standalone server implementation
2. Graphical Language Server Protocol
   - Language config, executing operations, ...
3. Web-based Client framework
   - Connection and interaction to a graphical language server
   - Integrated with Theia (usable stand-alone too)
   - UI for editing support, hooking up the protocol for editing

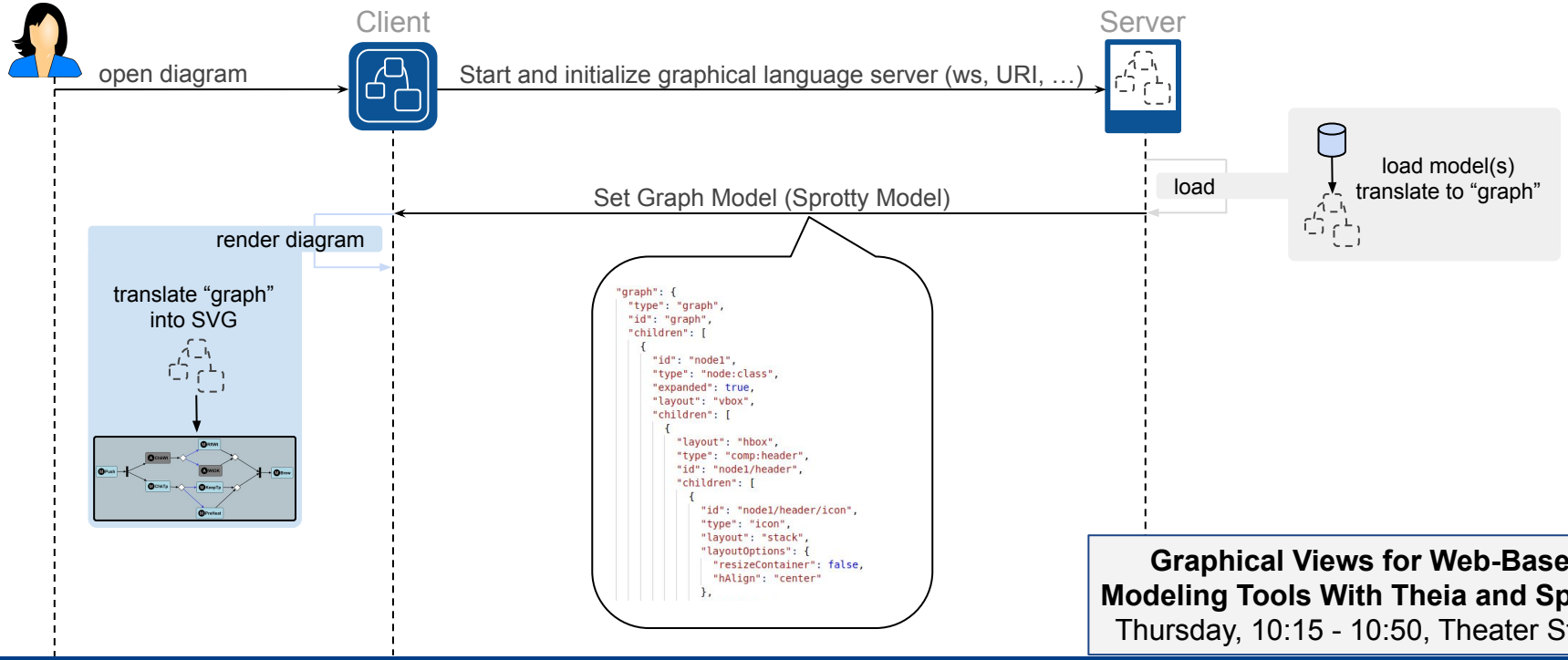
- Based on ...
  - Eclipse LSP4J
  - Eclipse Sprotty
    - Client implementation for diagram rendering
    - Sprotty's client-server protocol as a base for model transfer

ECLIPSE INCUBATION



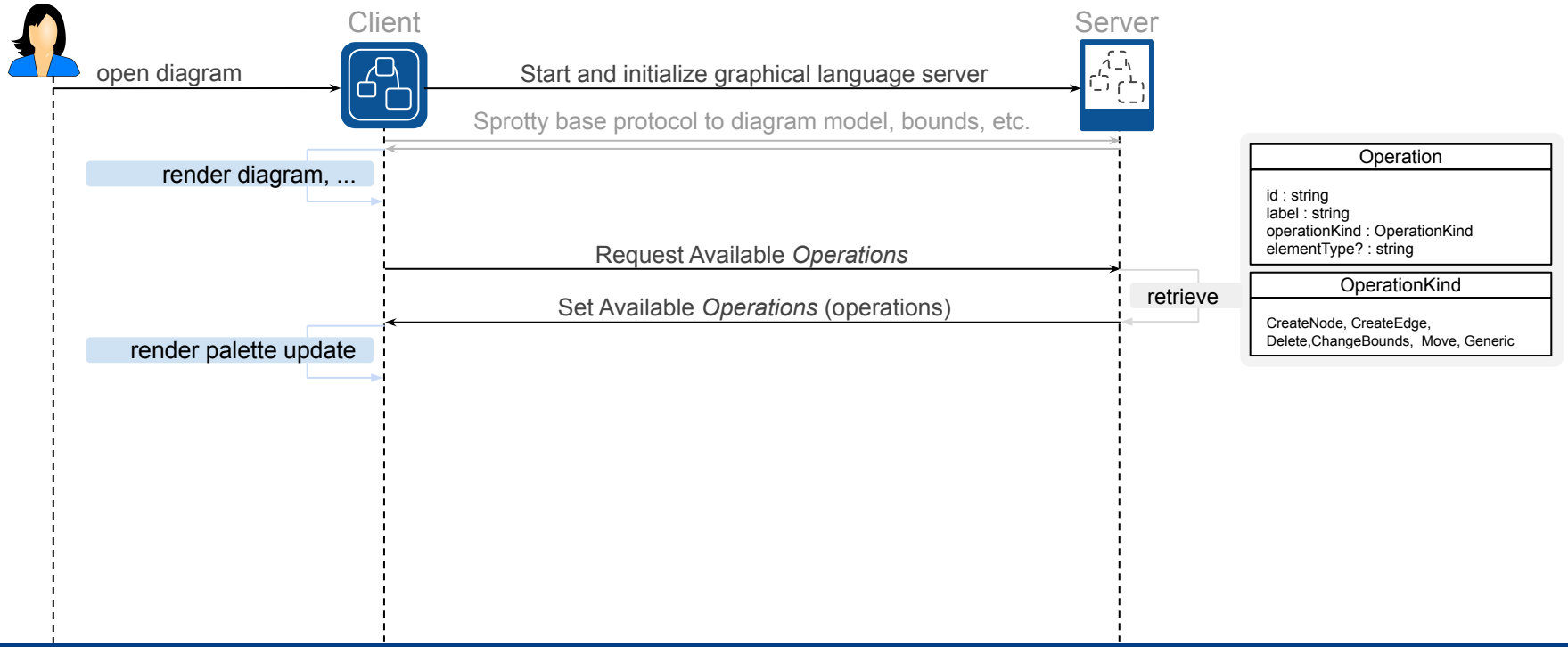
github.com/eclipsesource/graphical-lsp

Initialization and Rendering with Eclipse Sprotty

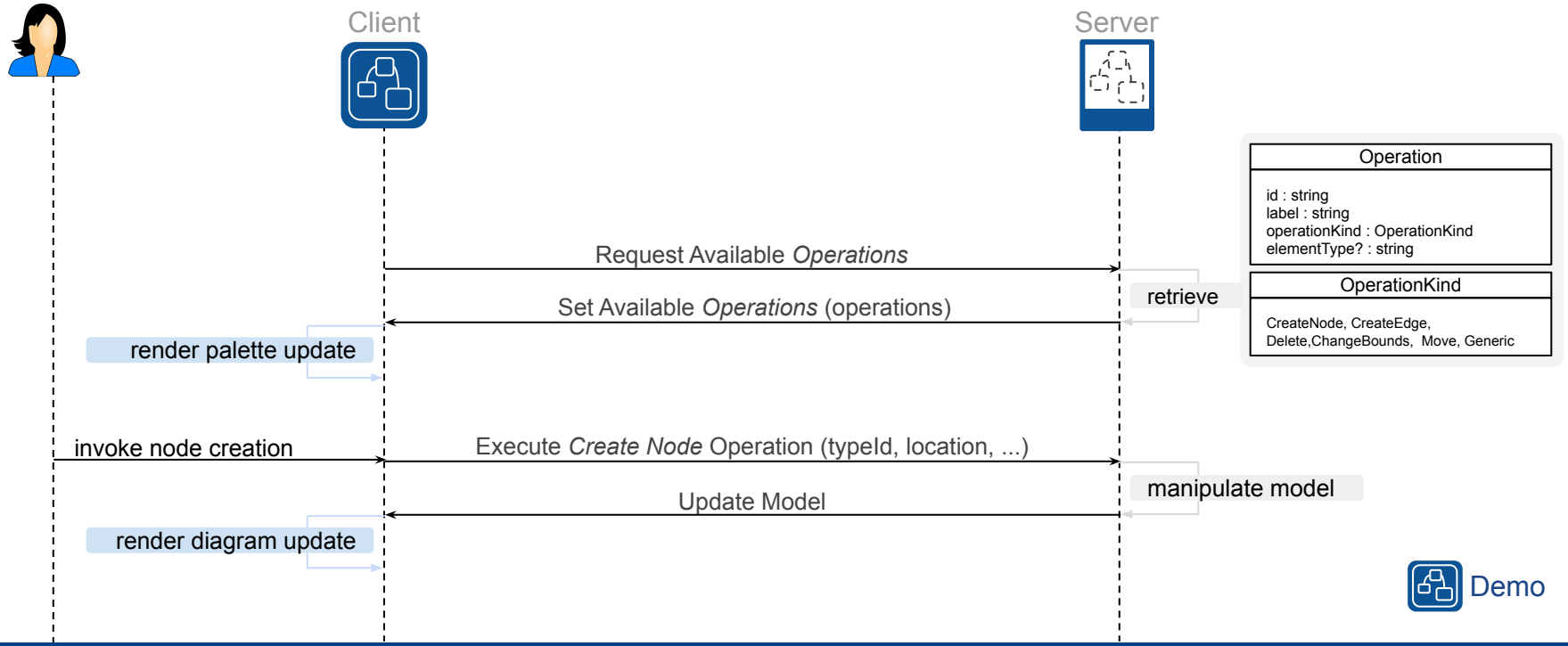


Graphical Views for Web-Based Modeling Tools With Theia and Sprotty
Thursday, 10:15 - 10:50, Theater Stage

Client-Server Interaction: Initialization and Editing Tools



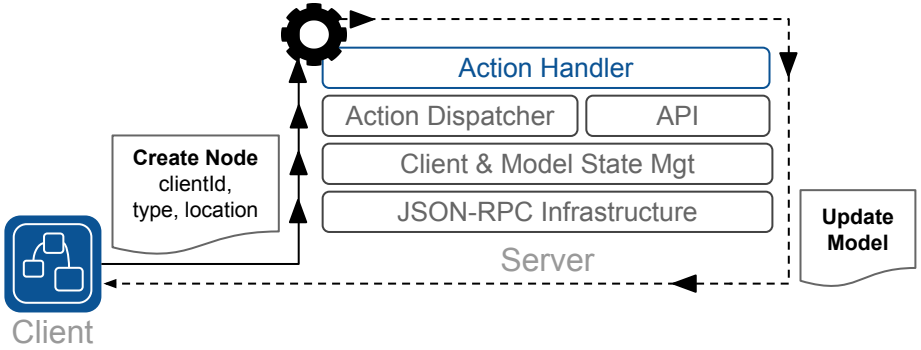
Client-Server Interaction: Operation Execution



Demo

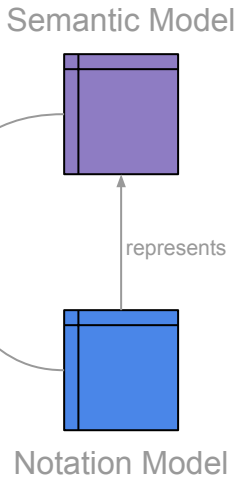
Server Framework

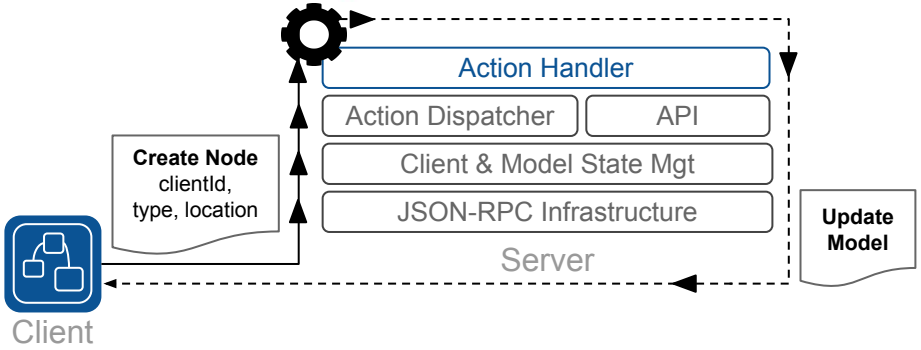
```
public class CreateActivityNodeHandler extends CreateNodeOperationHandler {

    @Override
    protected GNode createNode(Optional<GPoint> point, GraphicalModelState modelState) {
        String nodeType = ModelTypes.toNodeType(ModelTypes.MERGE_NODE);
        return new ActivityNodeBuilder(ModelTypes.MERGE_NODE, nodeType) //
                .position(point.orElse(null)) //
                .build();
    }

}
```



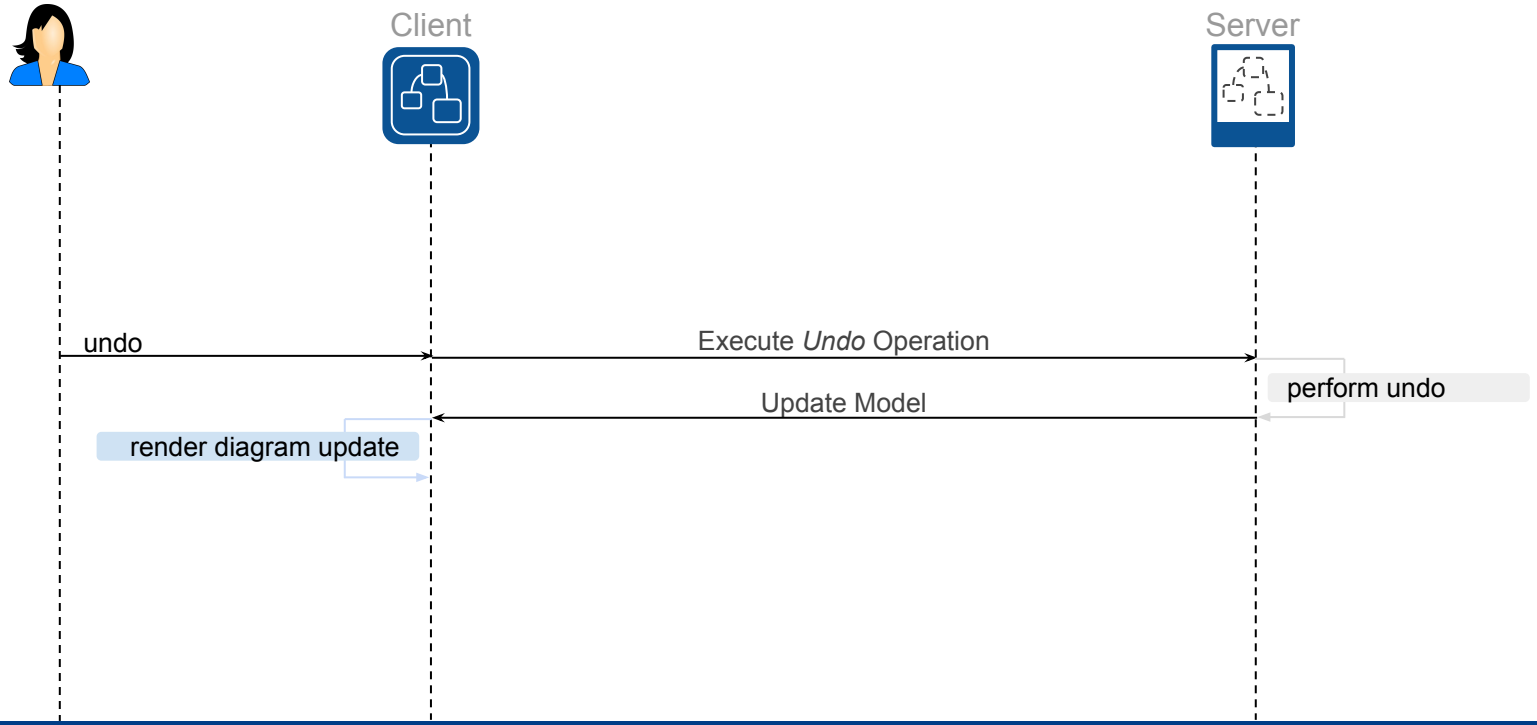
Server Framework

```java
@Override
public void doExecute(AbstractOperationAction action, GraphicalModelState modelState,
        WorkflowModelServerAccess modelAccess) throws Exception {
    CreateNodeOperationAction createNodeOperationAction = (CreateNodeOperationAction) action;
    WorkflowFacade workflowFacade = modelAccess.getWorkflowFacade();
    Workflow workflow = workflowFacade.getCurrentWorkflow();

    Merge node = CoffeeFactory.eINSTANCE.createMerge();

    Command addCommand = AddCommand.create(modelAccess.getEditingDomain(), workflow,
            CoffeePackage.Literals.WORKFLOW__NODES, node);

    workflowFacade.findDiagram(workflow).ifPresent(diagram -> {
        Shape shape = WfnotationFactory.eINSTANCE.createShape();
        shape.setSemanticElement(workflowFacade.createProxy(node));
        shape.setPosition(ShapeUtil.point(createNodeOperationAction.getLocation()));
        diagram.getElements().add(shape);
    });

    if (!modelAccess.edit(addCommand).thenApply(res -> res.body()).get()) {
        throw new IllegalAccessError("Could not execute command: " + addCommand);
    }
}
```



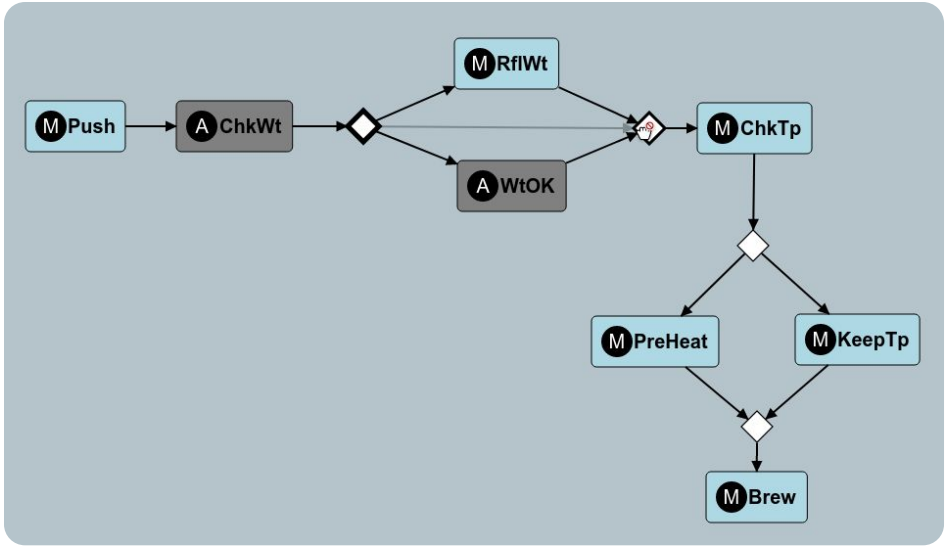


Client-Server Interaction: Undo and redo, delete, ...



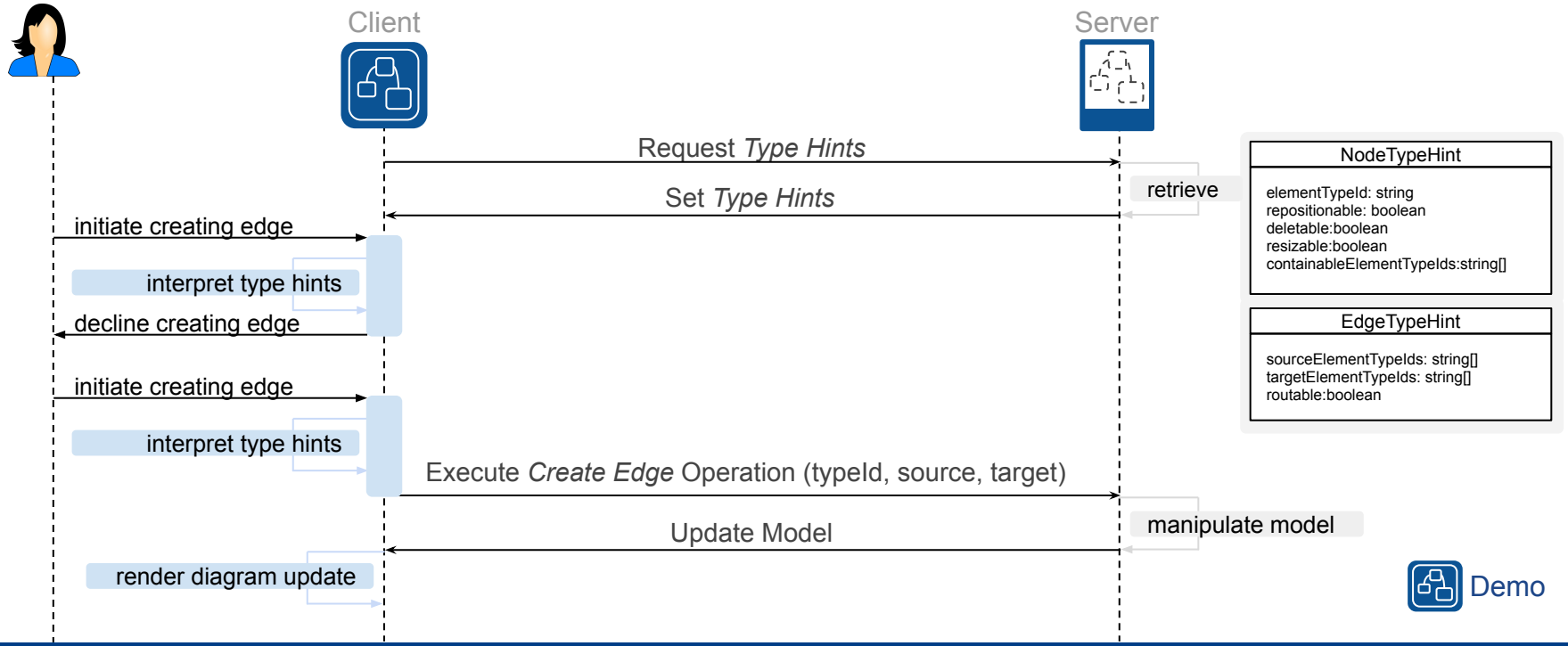
Demo

Client-Server Interaction: Type Hints

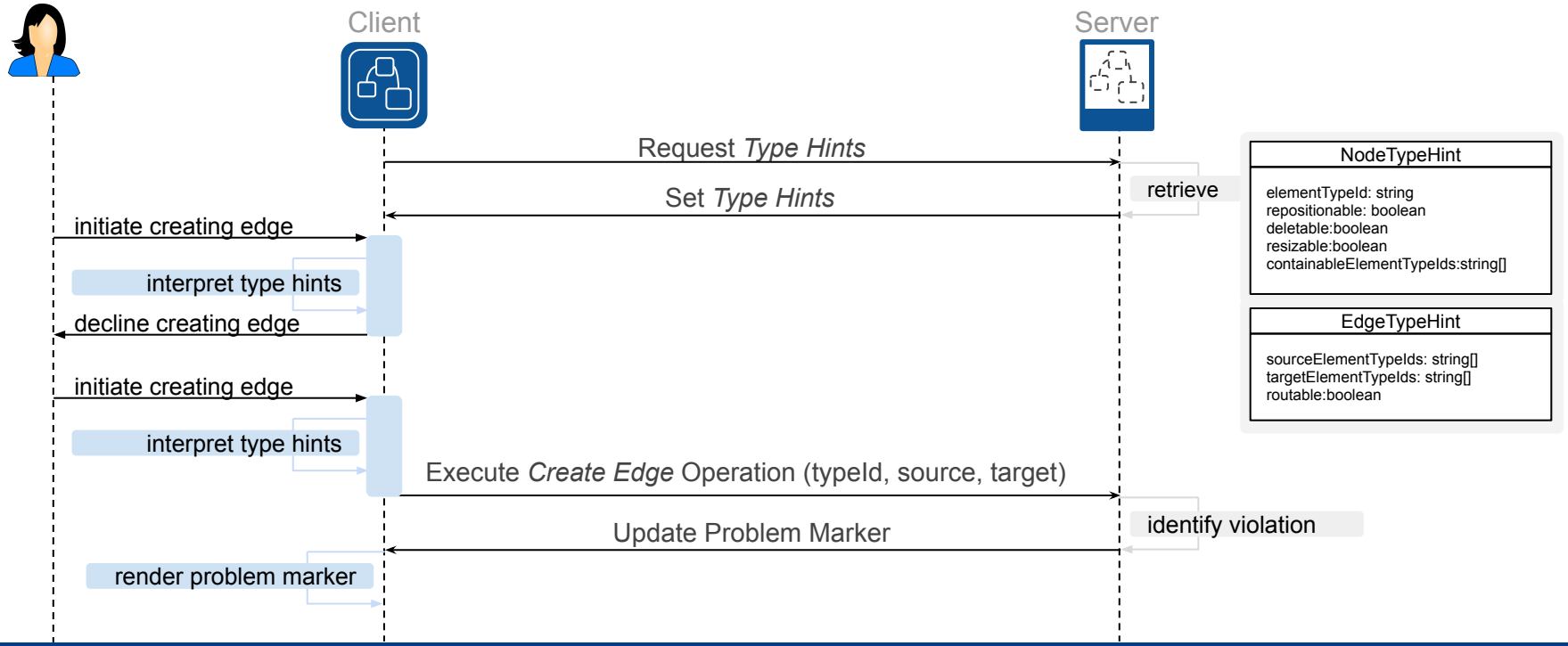


Avoiding server-roundtrip on direct user interaction!

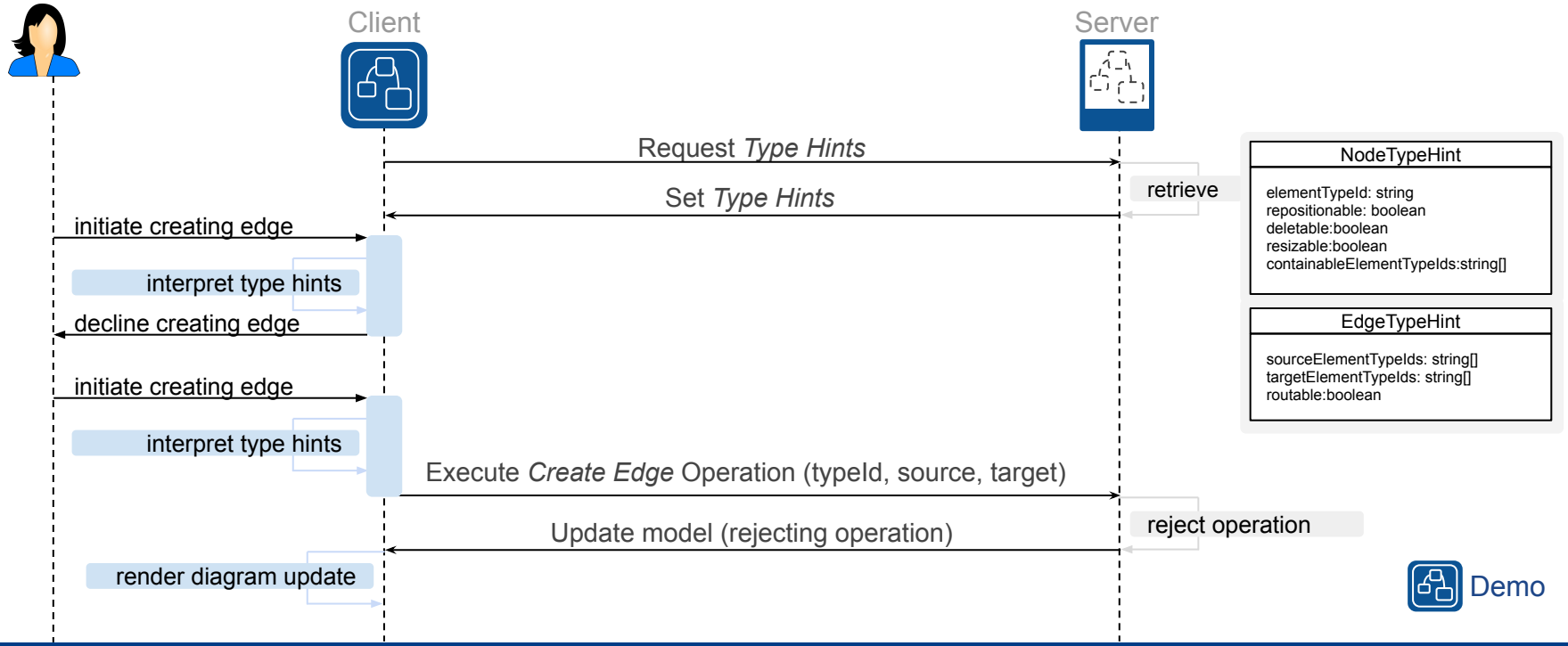
Client-Server Interaction: Type Hints



Client-Server Interaction: Type Hints



Client-Server Interaction: Type Hints



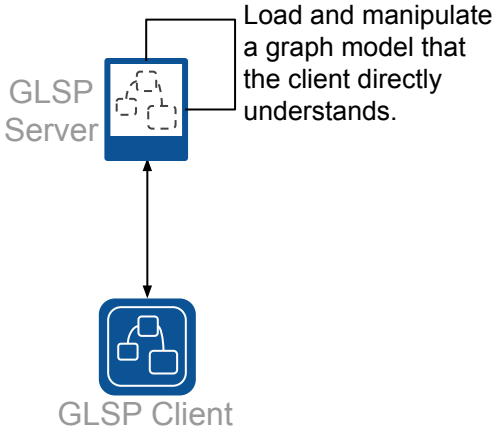
Client Framework

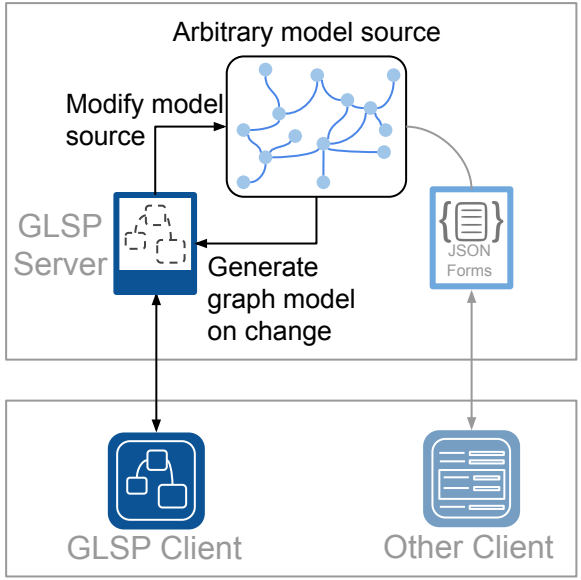
- Rendering of graph models
  - Pure Sprotty → all of Sprotty's rendering flexibility
- Client-side GLSP Server communication
  - Connects to a stand-alone GLSP server instance
  - Handles dispatching events locally or to the server
- Theia integration of the GLSP editor (also Eclipse RCP integration available)
- Sprotty extensions to enable editing capabilities
  - Editing tools, tools management, and palette
  - Visual feedback for tools
  - Dirty state handling
  - Command palette for keyboard-oriented actions
  - Direct editing of labels, ...

Demo

What else can we do with GLSP? Reuse and Flexibility!

- Integrate with existing model implementations
  - Just let server load and manipulate them
  - The client doesn't need to know

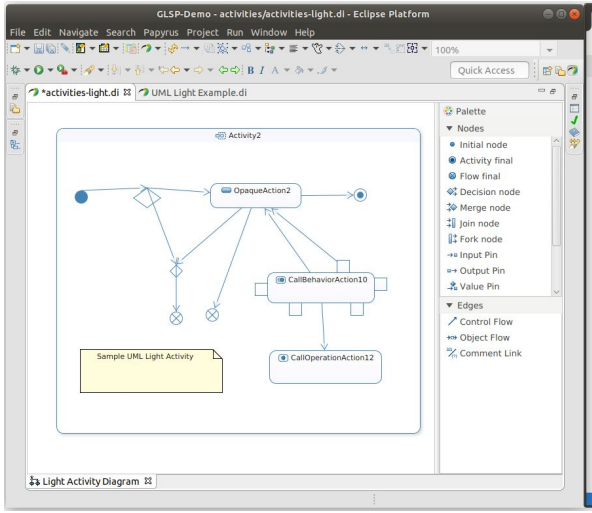


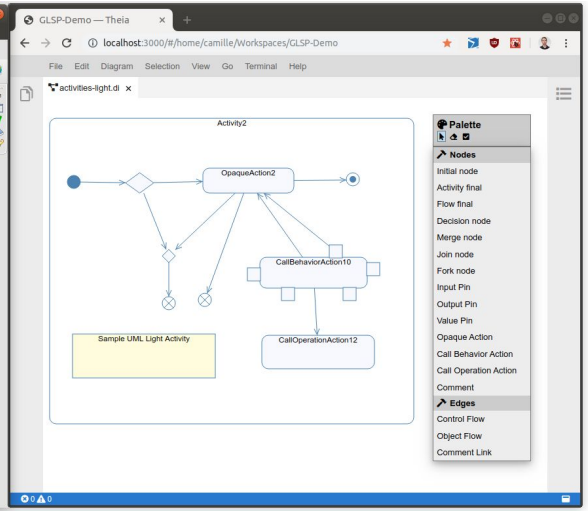


Demo

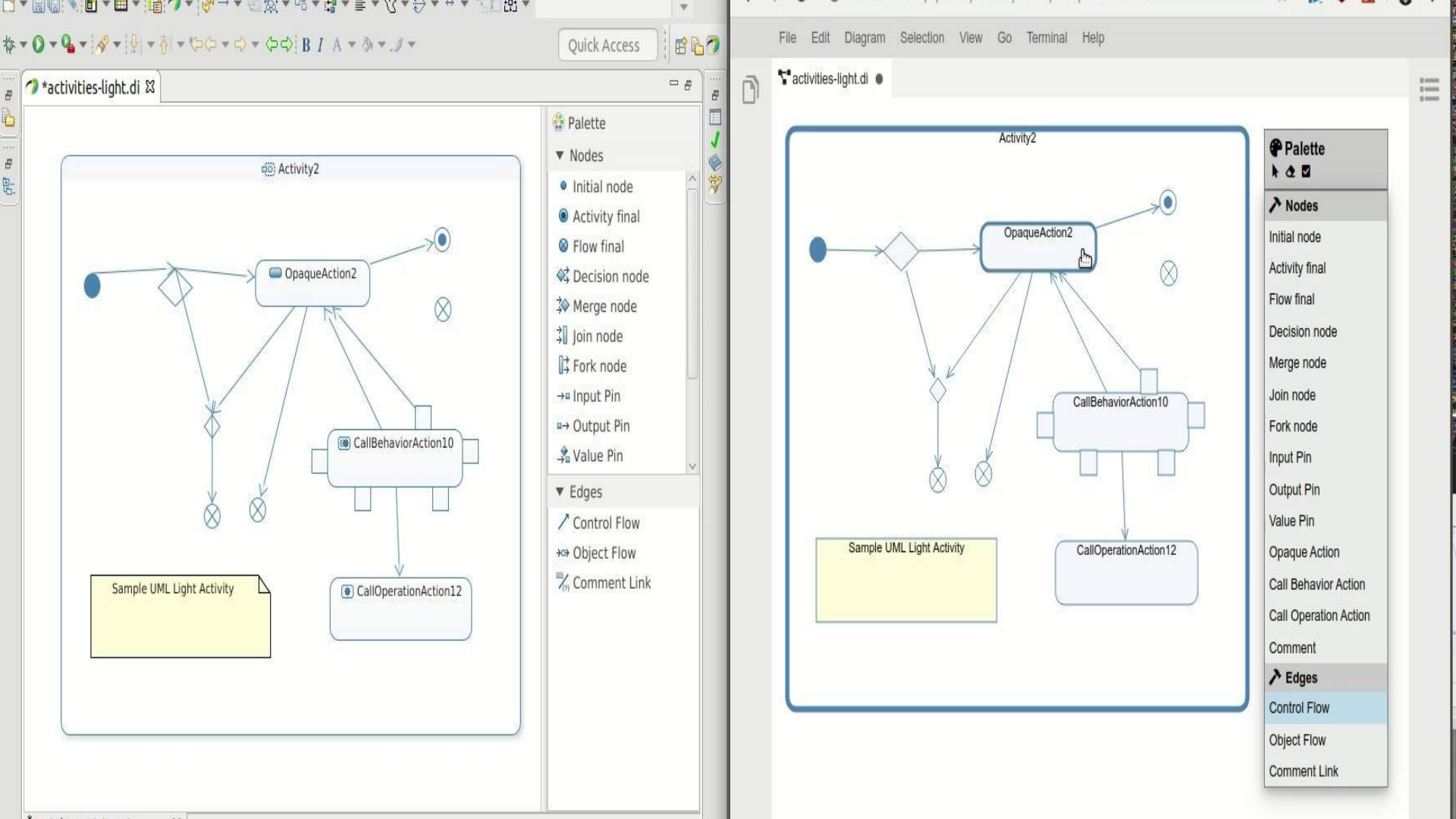
What else can we do with GLSP? Reuse and Flexibility!

- Integrate with existing editor implementations
  - Reuse a great deal of your work (e.g. model management, editing behavior, ...)



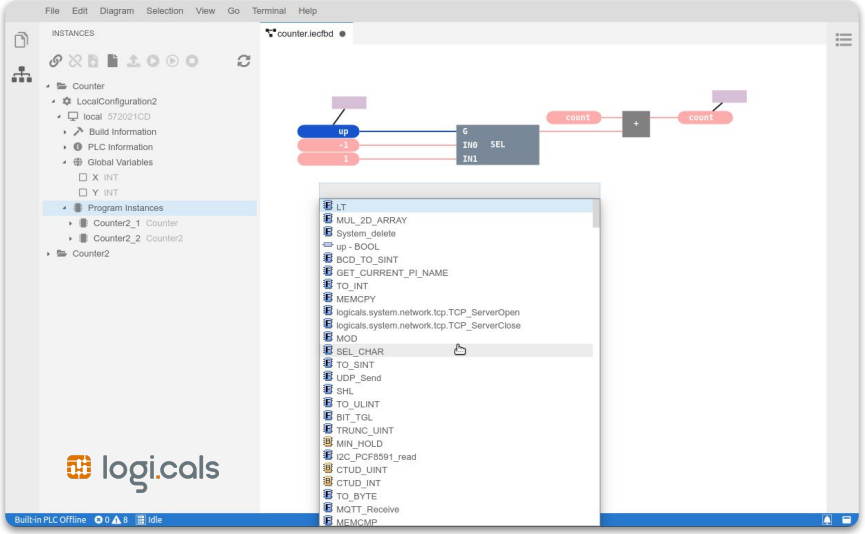


The evolution of Papyrus,
an open SysML and UML tool
Tuesday, 14:30 - 15:05, Seminarraum 5


Quick Access
*activities-light.di
Activity2
OpaqueAction2
CallBehaviorAction10
CallOperationAction12
Sample UML Light Activity
Palette
Nodes
Initial node
Activity final
Flow final
Decision node
Merge node
Join node
Fork node
Input Pin
Output Pin
Value Pin
Edges
Control Flow
Object Flow
Comment Link
File
Edit
Diagram
Selection
View
Go
Terminal
Help
activities-light.di
Activity2
OpaqueAction2
CallBehaviorAction10
CallOperationAction12
Sample UML Light Activity
Palette
Nodes
Initial node
Activity final
Flow final
Decision node
Merge node
Join node
Fork node
Input Pin
Output Pin
Value Pin
Opaque Action
Call Behavior Action
Call Operation Action
Comment
Edges
Control Flow
Object Flow
Comment Link

What else can we do with GLSP? Reuse and Flexibility!

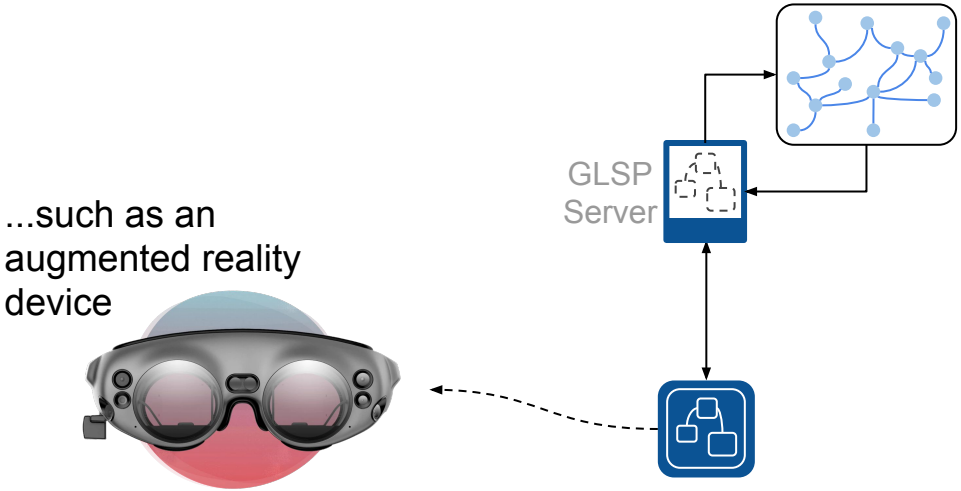
- Integrate with existing editor implementations
  - Reuse a great deal of your work (e.g. model management, editing behavior, ...)



An experience report on migrating an industrial-grade IDE to the Cloud
Wednesday, 10:15 - 10:50, Bürgersaal 2

What else can we do with GLSP? Reuse and Flexibility!

- Explore completely different clients
  - Use an existing server and have a client implementation on any device!



Demo

Property editors in space (AR)
Thursday, 13:00 - 13:35, Theater Stage

Conclusion and Outlook

- Conclusion
  - Sprotty-based client allows for developing editors with great flexibility
  - GLSP allows for very high reuse when migrating existing diagram editors
  - Separation of concerns → more future proof and flexible
  - Young technology, but ready for being adopted for several use cases
- Outlook
  - Property view support based on JsonForms
  - Further editing capabilities on the client
  - Identify best balance of powerfulness / simplicity of edit hints
- Get in contact with us to tell us about your use cases!
  - planger@eclipsesource.com
  - mfleck@eclipsesource.com

eclipse.org/glsp

github.com/eclipsesource/graphical-lsp