Building Web-based Diagram Editors

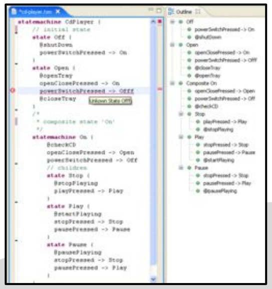
Towards a Graphical Language Server Protocol for Diagrams?

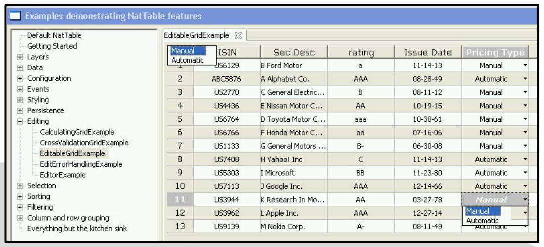
Philip Langer
EclipseSource

Tobias Ortmayr
EclipseSource

Building domain-specific (modeling) tools for various domains

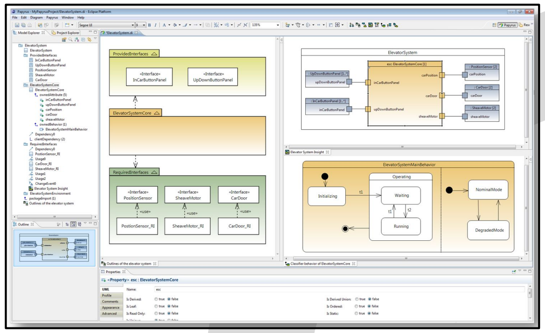
Software and systems engineering
Hardware producers,
Mobile networks,
Firmware, ...

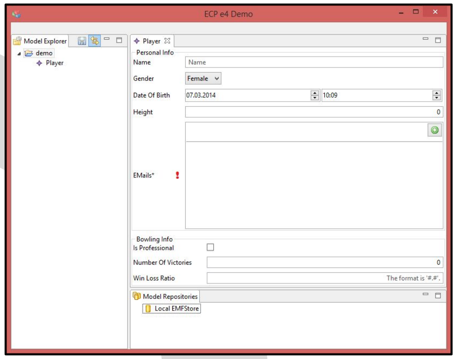

```
statemachine CdPlayer {
  // initial state
  state Off {
    @shutDown
    powerSwitchPressed -> On
  }
  state Open {
    @openTray
    openClosePressed -> On
    powerSwitchPressed -> Offf
    @closeTray
  }
  /*
   * composite state 'On'
   */
  statemachine On {
     @checkCD
     openClosePressed -> Open
     powerSwitchPressed -> Off
     // children
     state Stop {
       @stopPlaying
       playPressed -> Play
     }
     state Play {
       @startPlaying
       stopPressed -> Stop
       pausePressed -> Pause
     }
     state Pause {
       @pausePlaying
       stopPressed -> Stop
       pausePressed -> Play
     }
```



	ISIN	Sec Desc	rating	Issue Date	Pricing Type
1	US6129	B Ford Motor	a	11-14-13	Manual
2	ABC5876	A Alphabet Co.	AAA	08-28-49	Automatic
3	US2770	C General Electric...	B	08-11-12	Manual
4	US4436	E Nissan Motor C...	AA	10-19-15	Manual
5	US6764	D Toyota Motor C...	aaa	10-30-61	Manual
6	US6766	F Honda Motor C...	aa	07-16-06	Manual
7	US1133	G General Motors ...	B-	06-30-08	Manual
8	US7408	H Yahoo! Inc	C	11-14-13	Automatic
9	US5303	I Microsoft	BB	11-23-80	Automatic
10	US7113	J Google Inc.	AAA	12-14-66	Automatic
11	US3944	K Research In Mo...	AA	03-27-78	Manual
12	US3962	L Apple Inc.	AAA	12-27-14	Manual / Automatic
13	US9139	M Nokia Corp.	A-	08-11-49	Automatic

Information Systems
Insurances, Accounting, information management,
...





Name	Name
Gender	Female
Date Of Birth	07.03.2014 10:09
Height	0
EMails*	
Is Professional	☐
Number Of Victories	0
Win Loss Ratio	The format is '#.#'.

Modeling tools development

Business application development

Web-based diagram editors

1 Deployment and integration

- Browser is all that is needed
- Installing tool vs opening a link

+ Integration of diagram editors anywhere

+ Avoid the smell of an IDE

+ More people in the modeling process

2 Modern UI technology

- SWT vs HTML5
- GEF 3 vs SVG
- CSS3 Styling

+ Modern-looking diagrams

+ Flexibility in diagram style

+ Visual feedback and animations

- Frameworks?

JointJS

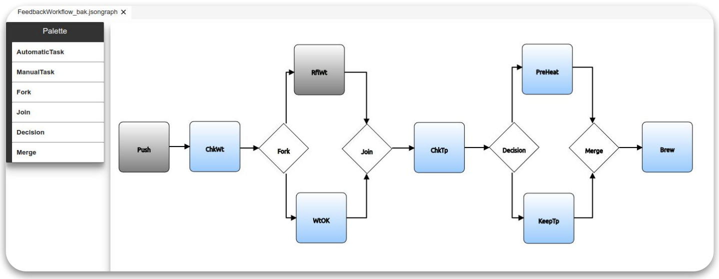
SVG-based Diagram Framework in JavaScript

+ Pros

- Decently implemented MVC architecture
- Nice editing support
- Feature-rich: many shapes, edge routing strategies, 2D function library

- Cons

- Not really community-driven open-source framework
- Everything is on the client

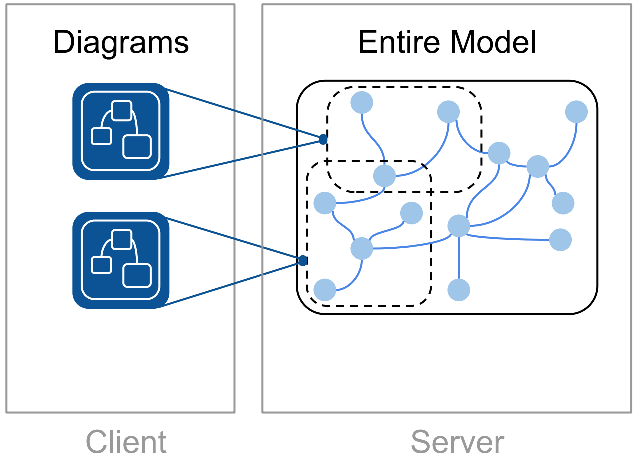


Why a client-server architecture is important to us

- One model != one diagram
  - Large models
  - One diagram only shows a part of it
  - Other parts are edited in other diagrams, forms, etc.
  - Prevent loading entire model into browser

- Whole-model understanding required for editing
  - User feedback before/after editing operations
  - Update on outside-of-diagram model changes
  - Live-validation may access outside-of-diagram parts

- Modeling language “smarts” is Java-based
  - Avoid having to re-implement those in JS
  - But re-use them in browser-based implementation

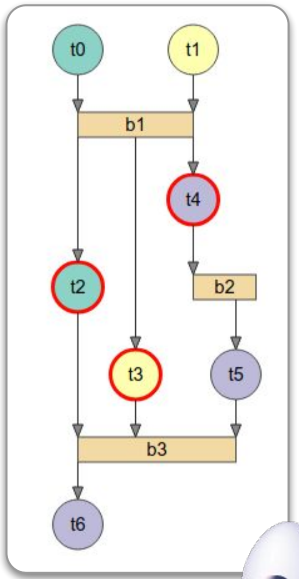


→ Essentially the same problem that’s addressed with LSP

Eclipse Sprotty

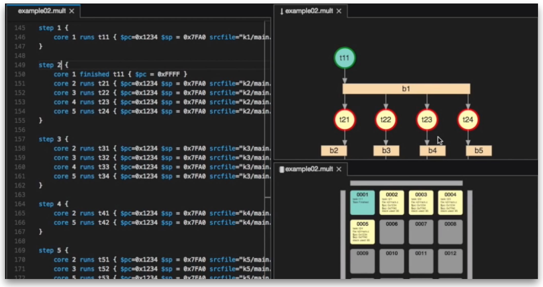
SVG-based Diagram Framework in TypeScript

+ Pros
  - TypeScript
    - Not plain JavaScript
  - Great and extensible architecture
    - Slim abstractions
    - DI-based configuration
  - Truly open-source software
    - Open development
    - Open to contributions
    - Recently became an Eclipse project
  - Integration with Xtext / Language Server Protocol



Sprotty & LSP = Client-server Protocol

- Sprotty's Language Server Extension
  - Visualize models owned by the Language Server
- Sprotty Client-server Protocol
  - Sprotty messages are tunneled through LSP
  - C → S: Request Model
  - S → C: Set/Update Model
  - Bounds
  - Collapse state
  - Selection
  - Pop-ups
- Server manages whole model
- Client doesn't need to know entire model
- Protocol is front-end oriented (just as LSP)



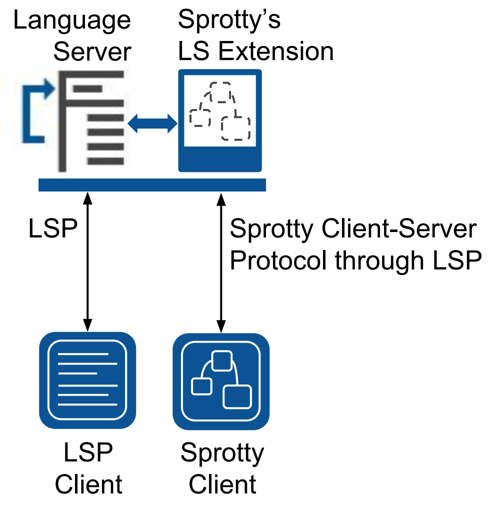
```
step 1 {
    core 1 runs t11 { $pc=0x1234 $sp = 0x7FA0 srcfile="k1/main.
}

step 2 {
    core 1 finished t11 { $pc = 0xFFFF }
    core 2 runs t21 { $pc=0x1234 $sp = 0x7FA0 srcfile="k2/main.
    core 3 runs t22 { $pc=0x1234 $sp = 0x7FA0 srcfile="k2/main.
    core 4 runs t23 { $pc=0x1234 $sp = 0x7FA0 srcfile="k2/main.
    core 5 runs t24 { $pc=0x1234 $sp = 0x7FA0 srcfile="k2/main.
}

step 3 {
    core 2 runs t31 { $pc=0x1234 $sp = 0x7FA0 srcfile="k3/main.
    core 3 runs t32 { $pc=0x1234 $sp = 0x7FA0 srcfile="k3/main.
    core 4 runs t33 { $pc=0x1234 $sp = 0x7FA0 srcfile="k3/main.
    core 5 runs t34 { $pc=0x1234 $sp = 0x7FA0 srcfile="k3/main.
}

step 4 {
    core 2 runs t41 { $pc=0x1234 $sp = 0x7FA0 srcfile="k4/main.
    core 5 runs t42 { $pc=0x1234 $sp = 0x7FA0 srcfile="k4/main.
}

step 5 {
    core 2 runs t51 { $pc=0x1234 $sp = 0x7FA0 srcfile="k5/main.
    core 3 runs t52 { $pc=0x1234 $sp = 0x7FA0 srcfile="k5/main.
```



So what’s missing?

- Sprotty server implementation
  - Depends on (textual) language server
  - Textual model is master
  - Graphical model is slave

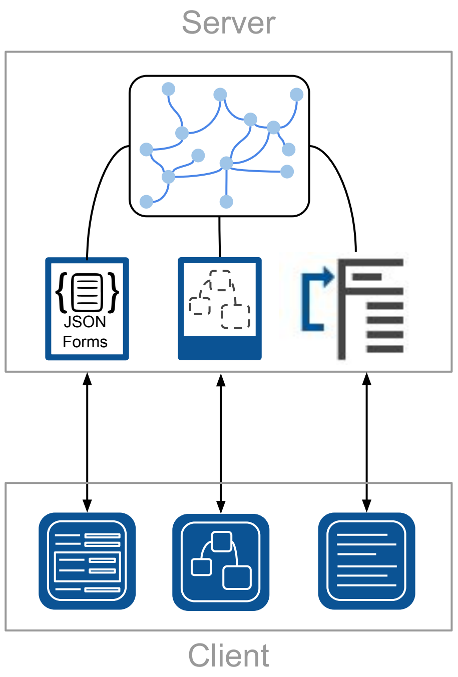
  → Independent diagram server API and impl

- Client-server protocol
  - Viewing and navigation capabilities only

  → Editing capabilities

- Client
  - UI for visualization purposes

  → Support and UI for editing



Graphical LSP Framework

github.com/eclipsesource/graphical-lsp

Applying the architectural pattern of LSP to graphical modeling

- Based on Sprotty
  - Client implementation
  - Sprotty's client-server base protocol

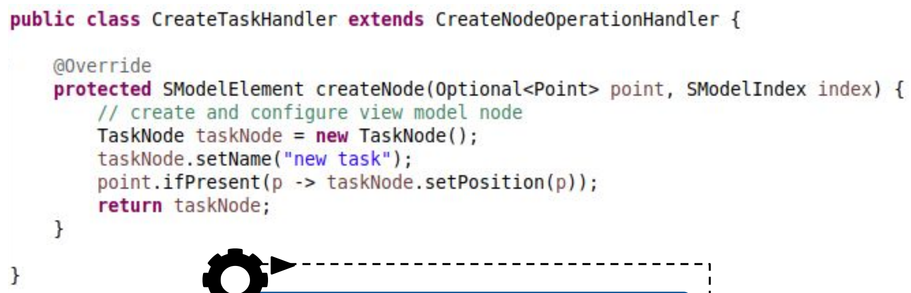
1. Java-based server framework
   - Standalone server implementation
   - Independent from any text language server
2. Extension of Sprotty's protocol for editing
3. Client framework
   - Server connector ("model source") decoupled from LSP
   - UI for editing support hooking up the protocol for editing

1 Server Framework

Framework for building specific diagram servers

- Server infrastructure
  - JSON-RPC communication: Sending and receiving action messages
  - Client and model state management
  - Model manipulation infrastructure

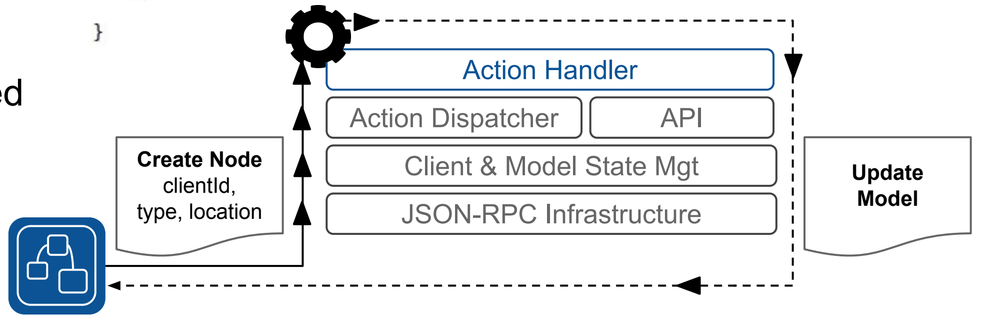
- Extensible DI module
  - Action registry
    - Predefined actions
    - Optionally custom actions
  - Action handler registry
    - Generic action handlers available
    - Custom handlers can be configured



```
public class CreateTaskHandler extends CreateNodeOperationHandler {

    @Override
    protected SModelElement createNode(Optional<Point> point, SModelIndex index) {
        // create and configure view model node
        TaskNode taskNode = new TaskNode();
        taskNode.setName("new task");
        point.ifPresent(p -> taskNode.setPosition(p));
        return taskNode;
    }

}
```

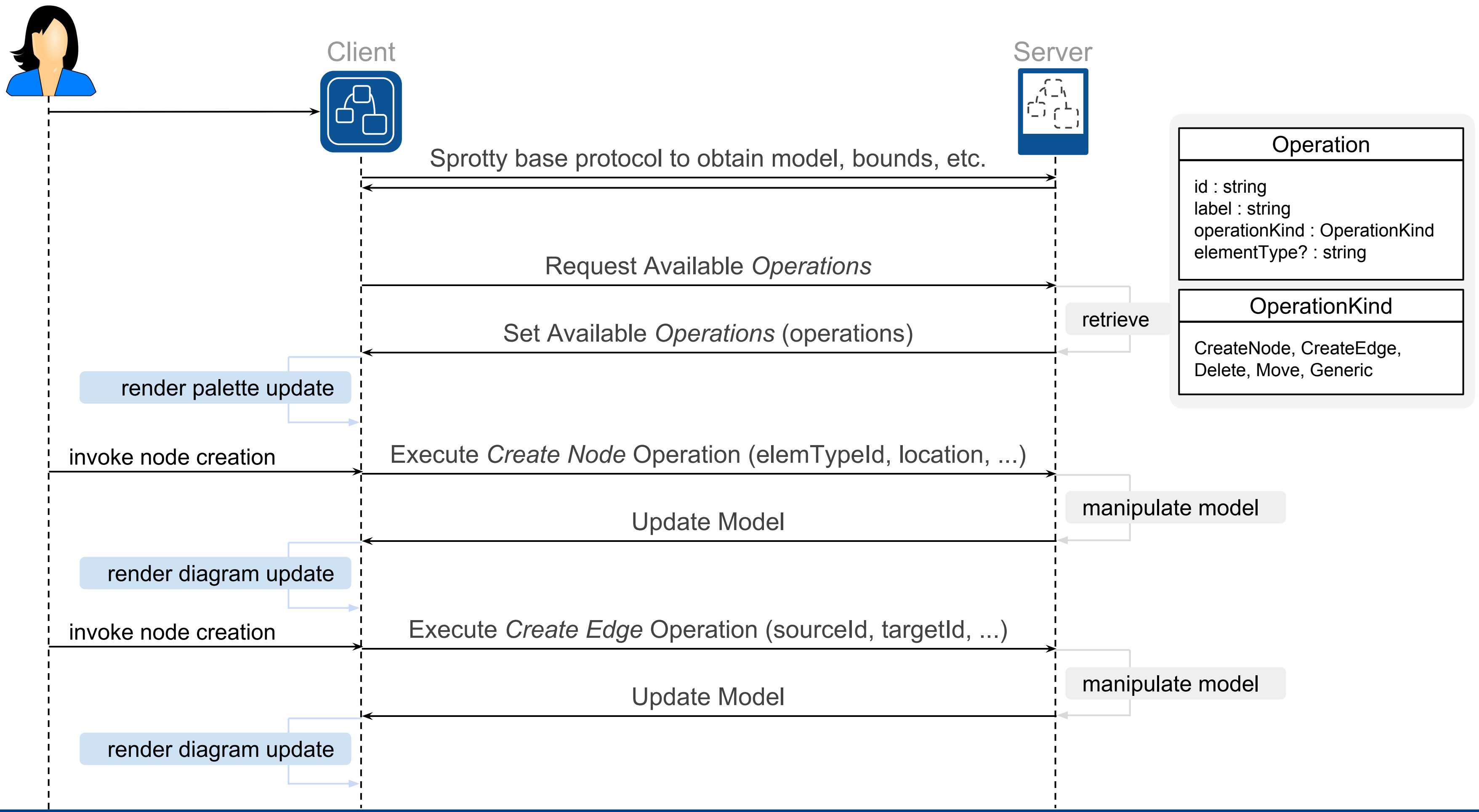


2 Towards a Graphical Language Server Protocol

- Transferring, updating, and navigating the diagram
  - Already exists in Sprotty
  - Re-used as a base protocol
- Goals
  - Encapsulate modeling language “smarts” on the server
  - Minimize client-server round-trips (esp. on UI interactions)
- Extensions for editing support
  - Available editing operations
  - Request execution of operation
  - Graphical move and resize
  - Drag and drop hints
- Extensions for additional features
  - Execute server action
  - Problem markers

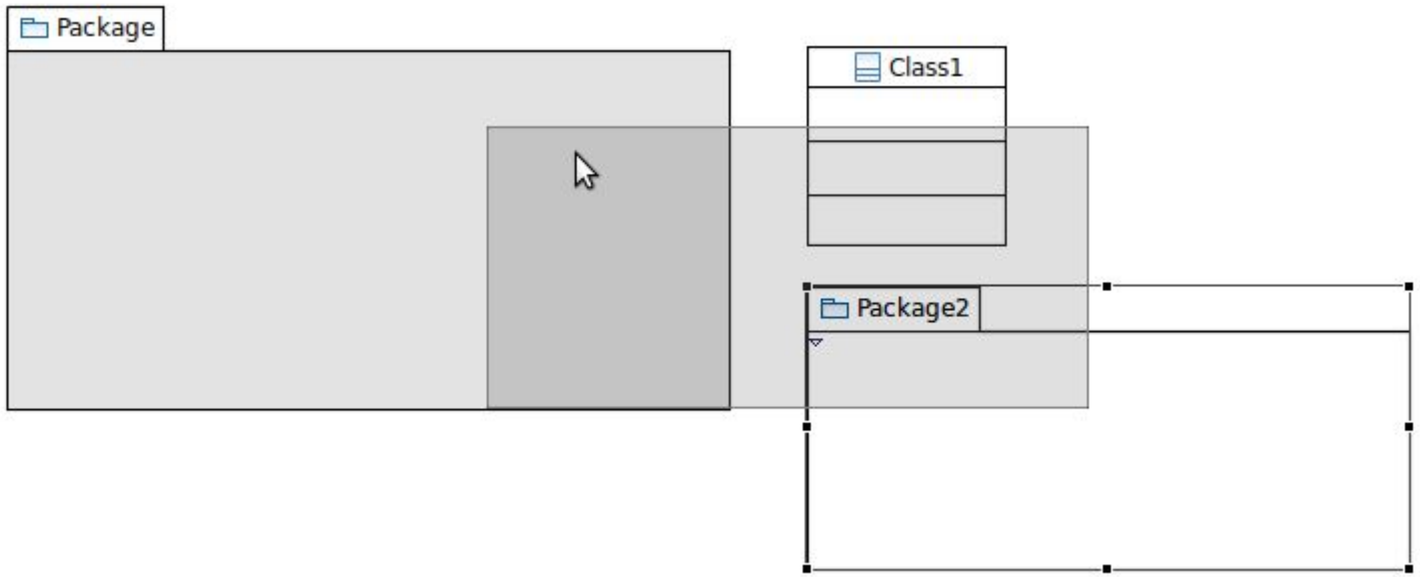
github.com/eclipsesource/GraphicalServerProtocol

2 Towards a Graphical Language Server Protocol



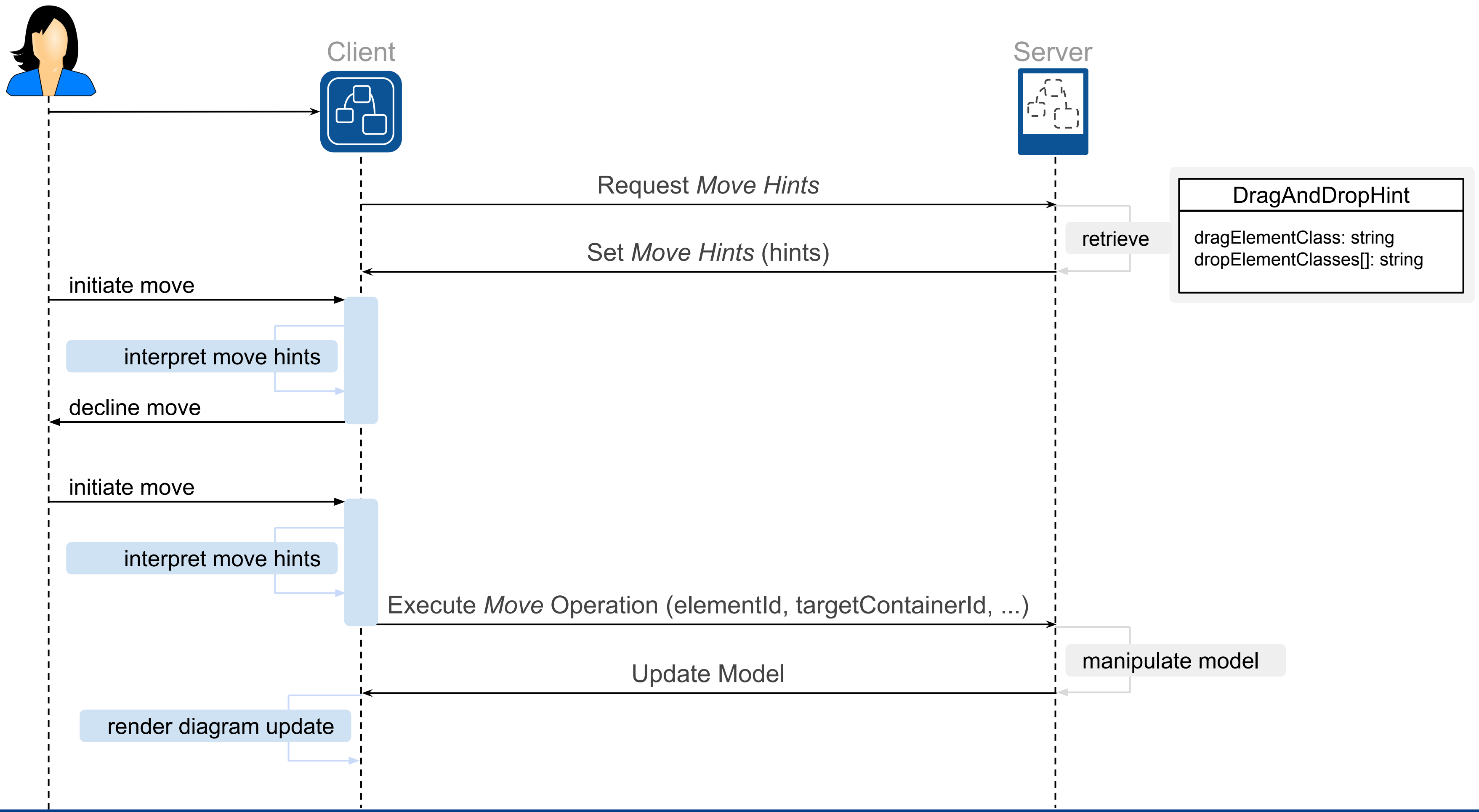
Client
Server
Sprotty base protocol to obtain model, bounds, etc.
Request Available *Operations*
retrieve
Set Available *Operations* (operations)
render palette update
invoke node creation
Execute *Create Node* Operation (elemTypeId, location, ...)
manipulate model
Update Model
render diagram update
invoke node creation
Execute *Create Edge* Operation (sourceId, targetId, ...)
manipulate model
Update Model
render diagram update
Operation
id : string
label : string
operationKind : OperationKind
elementType? : string
OperationKind
CreateNode, CreateEdge, Delete, Move, Generic

2 Towards a Graphical Language Server Protocol

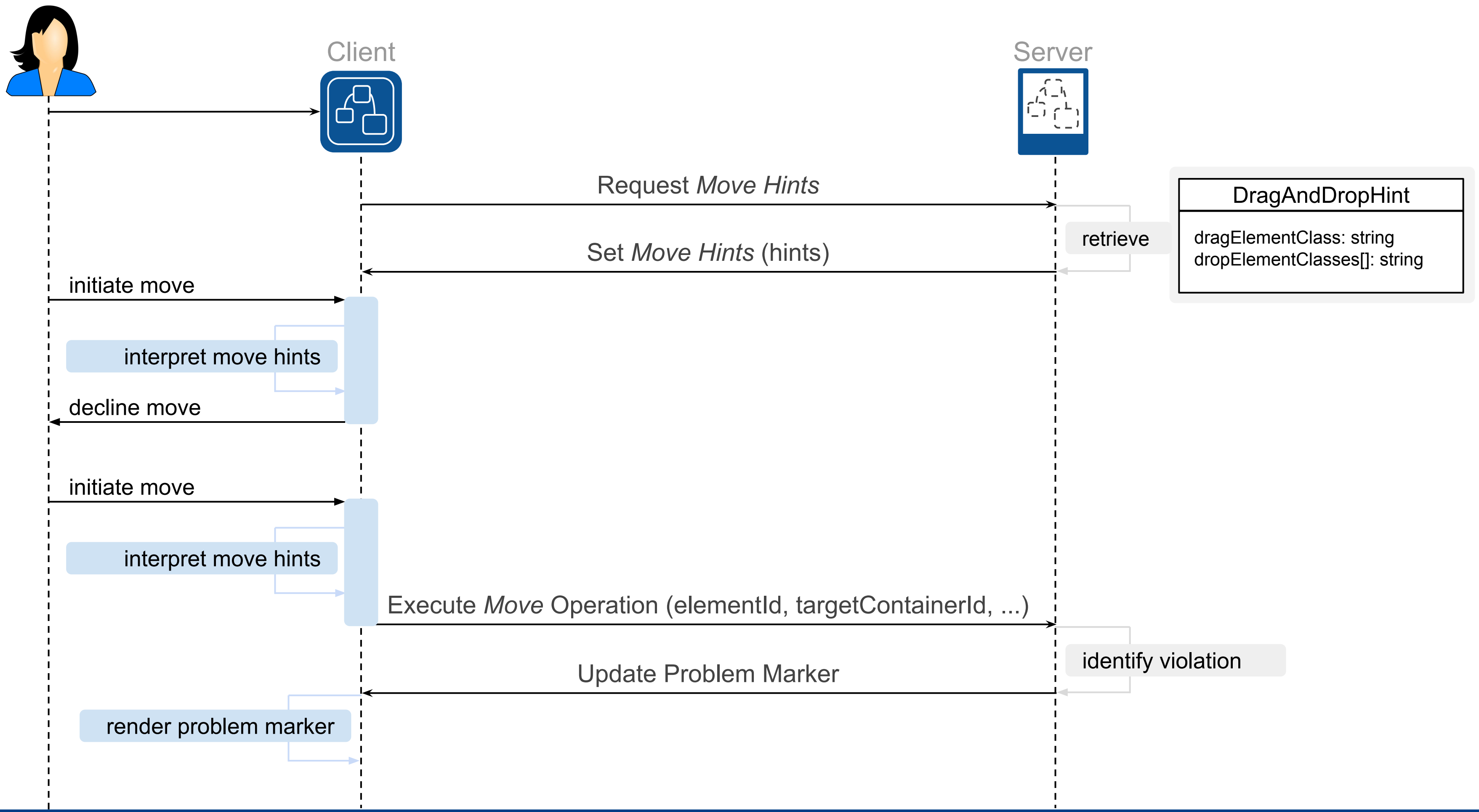


Avoiding server-roundtrip on direct user interaction!

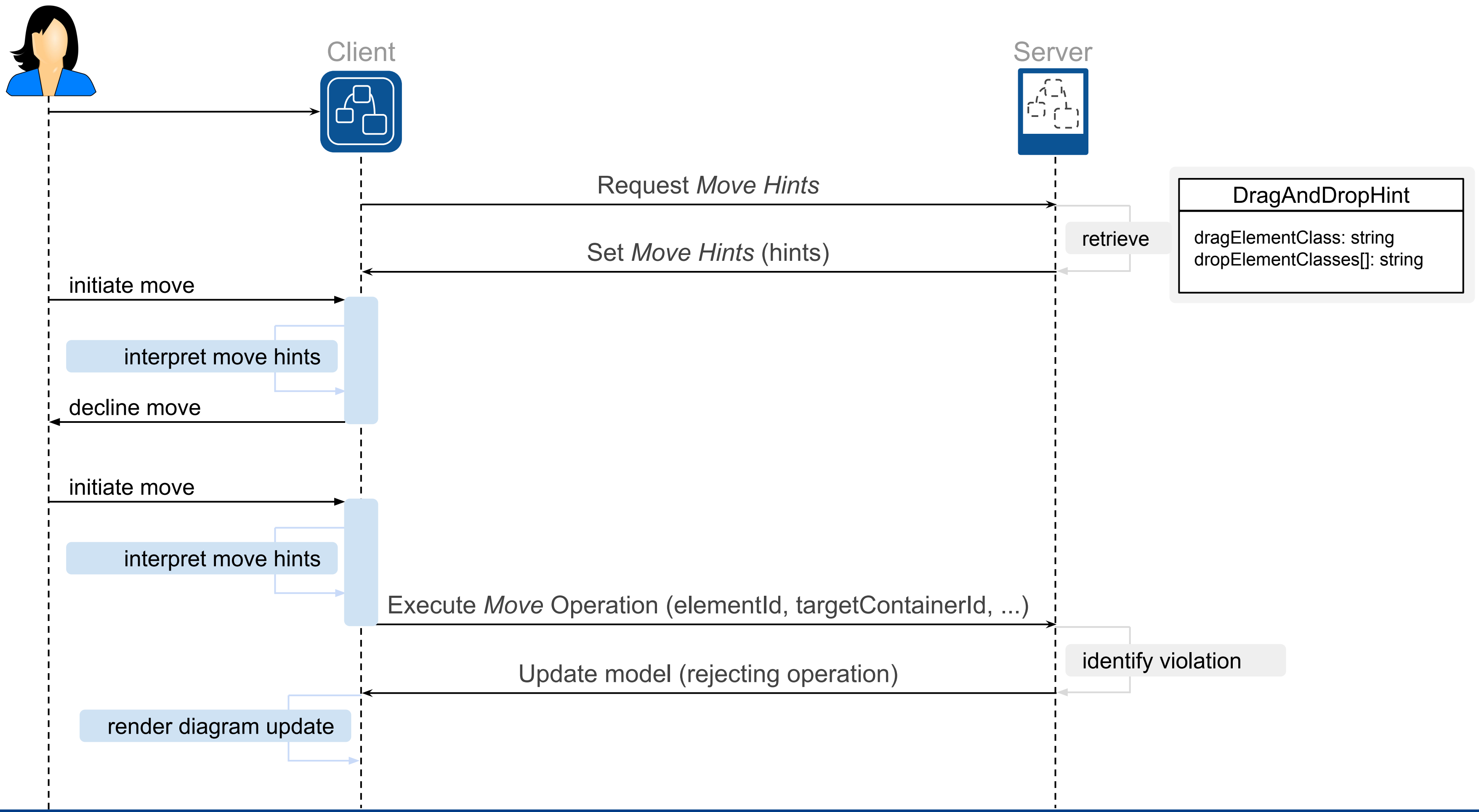
2 Towards a Graphical Language Server Protocol



Towards a Graphical Language Server Protocol



2 Towards a Graphical Language Server Protocol



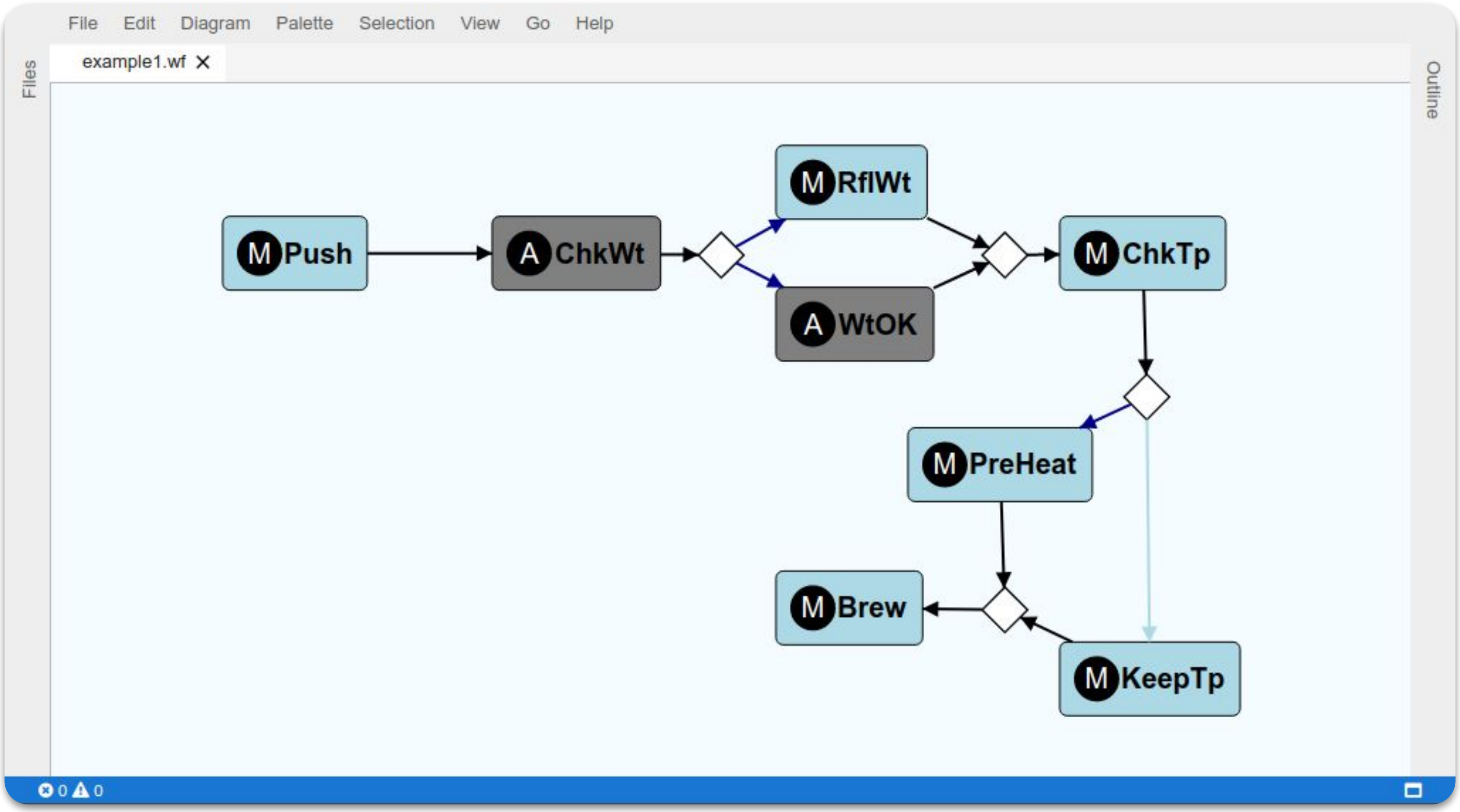
3 Client Framework

- Client-side GLSP Diagram Server implementation
  - Connects to a stand-alone GLSP server instance
  - Handles dispatching events locally or to the server

- Sprotty extensions to enable editing capabilities
  - Palette that enables editing tools (for now very simple)
  - Editing tools, e.g. for adding nodes, edges, etc.
  - Persisting diagram changes

- Editing command handlers
  - Hooking up client-side editing commands with the server
  - Sending and receiving the respective protocol messages

Demo


File
Edit
Diagram
Palette
Selection
View
Go
Help
example1.wf
Files
Outline
M Push
A ChkWt
M RflWt
A WtOK
M ChkTp
M PreHeat
M Brew
M KeepTp
0
0

Current State and Outlook

- So far
  - Focus on the server framework & protocol
  - Client extensions only as much as necessary
- Next steps
  - Enhance generic editing capabilities in the client
  - Palette, visual feedback, support for drag and drop hints, etc.
  - Problem markers, property views based on JsonForms, etc.
- Collaboration and contribution
  - Enhancement of Sprotty with TypeFox
  - Client-server protocol definition with TypeFox and Obeo
  - Hopefully with you too?

github.com/eclipsesource/graphical-lsp

github.com/eclipsesource/GraphicalServerProtocol

More on related topics at EclipseCon

- Earlier today -- catch them later on Youtube
  - If, when and how? - Strategies towards web-based tooling
  - Lucky in the Cloud With Diagrams
- Tomorrow
  - Building a Web-IDE based on Eclipse Theia for Smart Home (11:55 Bürgersaal 2)
  - EMF, JSON and I (14:45 Theater Stage)
  - Domain-Specific Languages in the Cloud – With Eclipse Technologies (16:30)
- Oct 25th
  - JSON Forms 2.0 (10:45 Theater Stage)
  - Building Web-based Modeling Tools based on Eclipse Theia (11:30 Theater Stage)

Evaluate the Sessions

Sign in and vote at **eclipsecon.org**

-1 0 +1