

SemanticSourceCodeNavigator

Files	Classes	Functions
demos/systembrowser/testFile1.js	class TestClass	update2 ()
demos/systembrowser/testFile2.js	class TestClass2	print2 ()

```
1 static update2() {  
2   var variables = _recorder_[getScopeIdForModule()];  
3  
4   console.log('something');  
5 }
```

# Semantic Source Code Navigation

Web-based Development Environments WS 17/18  
Siegfried Horschig, Theresa Zobel - 06.02.2018

Software Architecture Group 2006-present

# Agenda

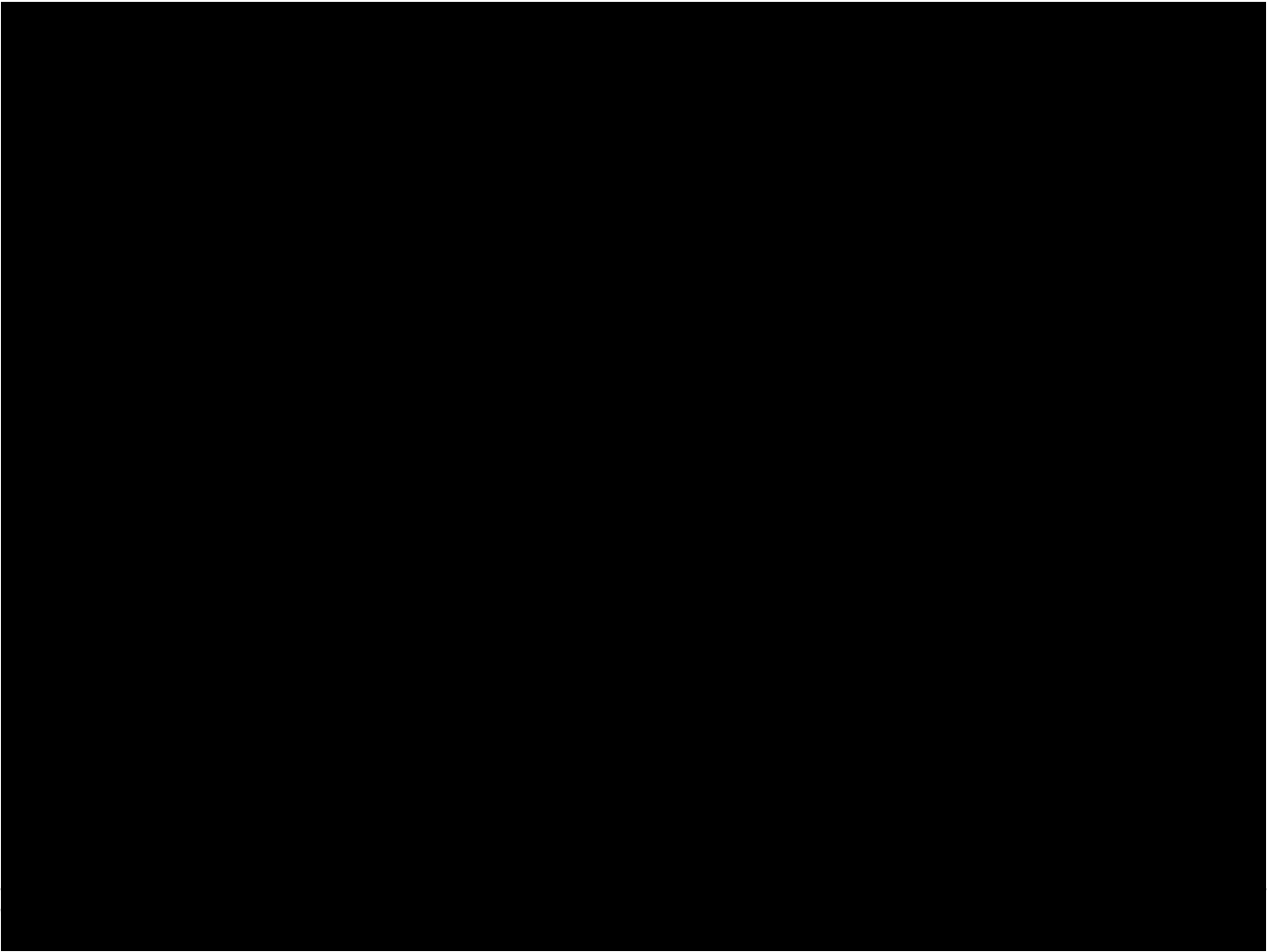
---

1. Domain
2. Motivation
3. Demo
4. Structure
5. Next Steps & Future Work

# Domain

---

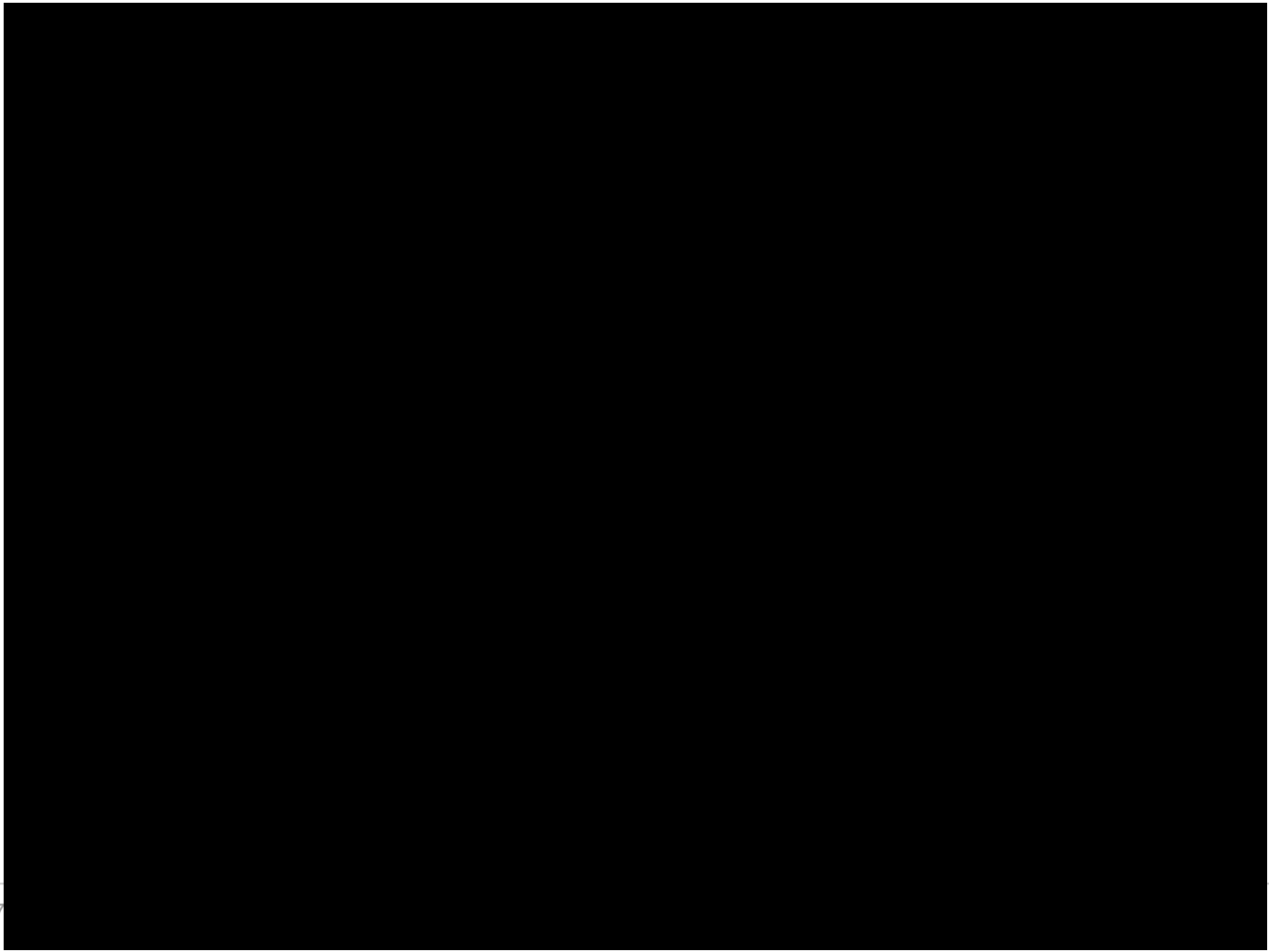
## Classic Navigation



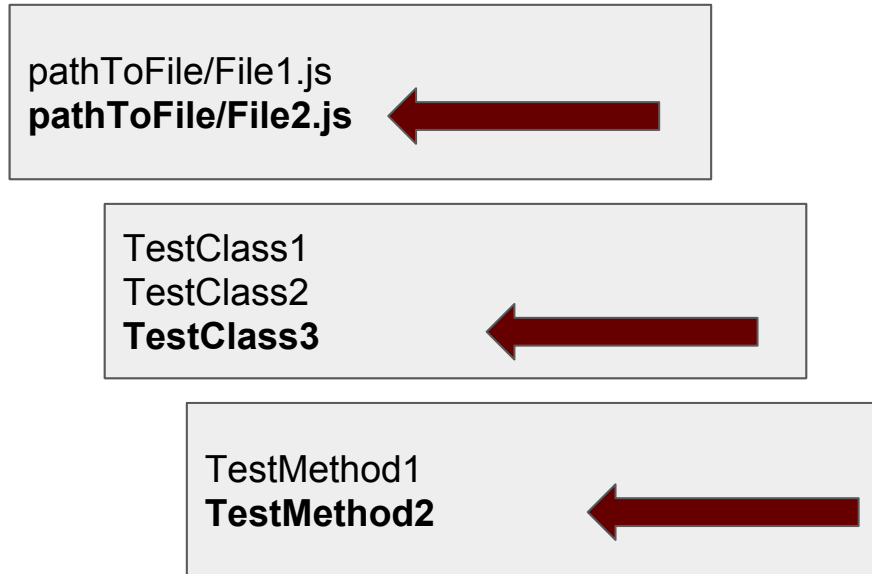
# Domain

---

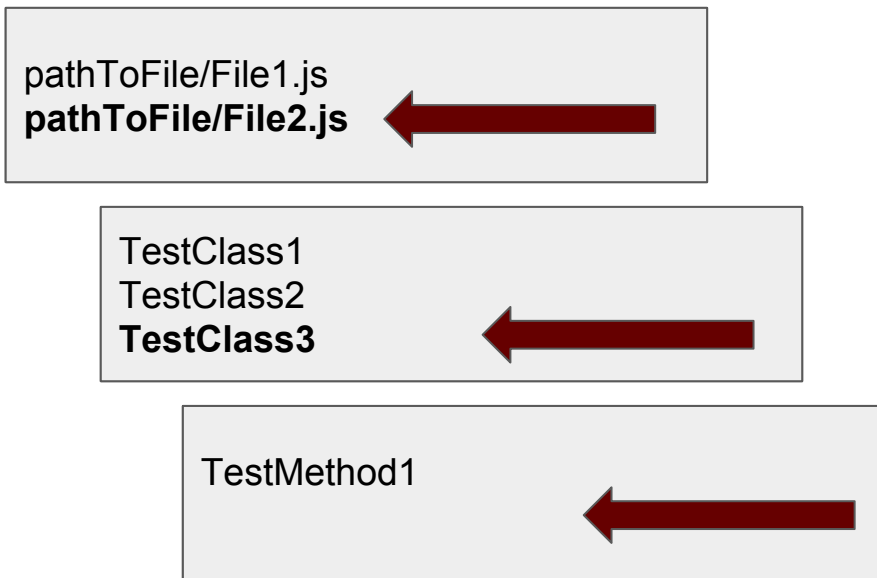
## Classic Manipulation



## Semantic Navigation



## Semantic Manipulation





# Motivation

The screenshot displays the Morph IDE interface with several overlapping windows illustrating the motivation for a specific design approach.

- System Browser: Morph**: Shows a tree view of the system structure, including categories like `KernelTests-Processes`, `Files-Kernel`, `Morphic-Kernel`, `NetworkTests-Kernel`, `Protocols-Kernel`, `SUnit-Kernel`, `Compiler-Kernel`, `ToolBuilder-Kernel`, `Traits-Kernel`, and `TraitsTests-Kernel`. The `Morph` package is selected.
- event handling**: A list of event handling mechanisms, including `events-accessing`, `events-alarms`, `events-filtering`, `events-filtering-bubbling`, `events-filtering-capturing`, `events-processing`, `events-removing`, and `*Etoys-custom-events-s`.
- mouseDown: evt**: A window showing the default response for a mouse down event: "Handle a mouse down event. The default response is to let my eventHandler, if any, handle it." It includes a code snippet: 

```
self eventHandler
ifNotNil: [self eventHandler mouseDown: evt from: self].
```
- Implementors of mouseDown: [65]**: A window listing 65 implementors of the `mouseDown` message, including `AbstractResizerMorph`, `AlternatePluggableListMorphOfMany`, `BookPageThumbnailMorph`, `BracketSliderMorph`, `ButtonProperties`, and `ChatButtonMorph`.
- Senders of mouseDown: [37]**: A window listing 37 senders of the `mouseDown` message, including `FatBitsPaint`, `InterimSoundMorph`, `MagnifierMorph`, `MenuItemMorph`, `Morph click`, `Morph handleMouseDown`, `Morph showActions`, `NumericReadoutTile`, `PasteUpMorph`, `PluggableListMorph`, `PluggableListMorphOfMany`, and `PluggableListMorphOfMany`.

# Motivation

---

## Initial Task

- Webstorm-like Shift+Click navigation for senders and implementors
- Finding the definition of a method / function / class
- Autocompletion on variables and properties over multiple files

# Motivation

---

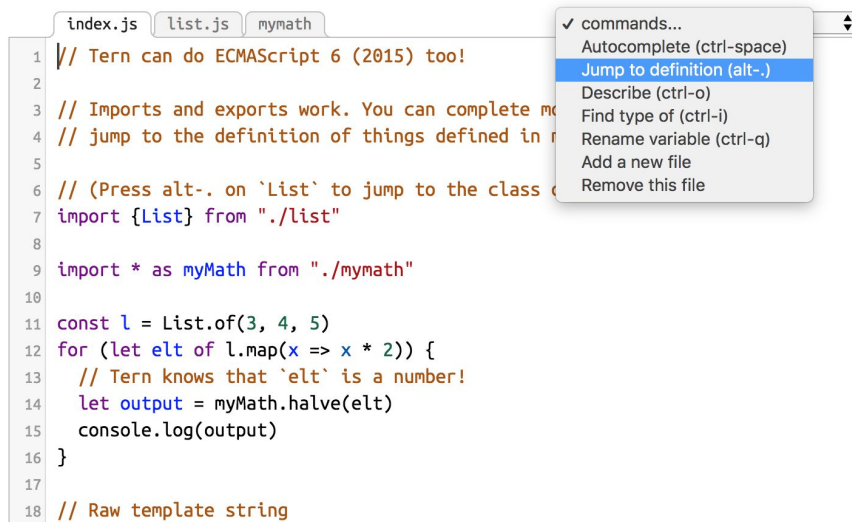
## Using Tern

- Tern is a stand-alone code-analysis engine
- Can be integrated into code editors (CodeMirror) as a plugin
- Builds abstraction with inference engine

# Motivation

## Initial Task - Using Tern

- **[...] over multiple files**
- **Problem:** Demo example does not work in Lively -> Tern is hard to debug!



The screenshot shows a code editor with three tabs: 'index.js', 'list.js', and 'mymath'. The 'index.js' tab is active, displaying the following code:

```
1 // Tern can do ECMAScript 6 (2015) too!
2
3 // Imports and exports work. You can complete m
4 // jump to the definition of things defined in
5
6 // (Press alt-. on `List` to jump to the class
7 import {List} from "./list"
8
9 import * as myMath from "./mymath"
10
11 const l = List.of(3, 4, 5)
12 for (let elt of l.map(x => x * 2)) {
13   // Tern knows that `elt` is a number!
14   let output = myMath.halve(elt)
15   console.log(output)
16 }
17
18 // Raw template string
```

A context menu is open over the code, listing the following commands:

- ✓ commands...
- Autocomplete (ctrl-space)
- Jump to definition (alt-.)
- Describe (ctrl-o)
- Find type of (ctrl-i)
- Rename variable (ctrl-q)
- Add a new file
- Remove this file

# Motivation - Project Revamp

## New Task

- System Browser (similar to Squeak)
- Parsing files and their signatures
- Visualization in component
- Edit content in component



# Demo

---



# Component

 SemanticSourceCodeNavigator   

Files	Classes	Functions
demos/systembrowser/testFile1.js	class TestClass	update2 ()
demos/systembrowser/testFile2.js	class TestClass2	print2 ()

```
1 static update2() {  
2   var variables = _recorder_[getScopeIdForModule()];  
3  
4   console.log('something');  
5 }
```



# Structure

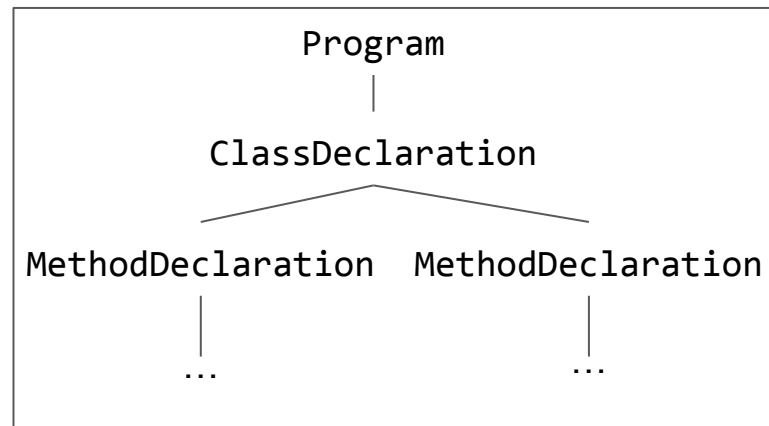
## Workflow - Populate Component

```
1 import { getScopeIdForModule } from
2 "src/external/babel-plugin-var-recorder.js";
3
4 class TestClass {
5   static update() {
6     console.log('something');
7   }
8
9   static print() {
10    console.log("somethingelse");
11  }
12 }
13 }
```

File

**BABEL**

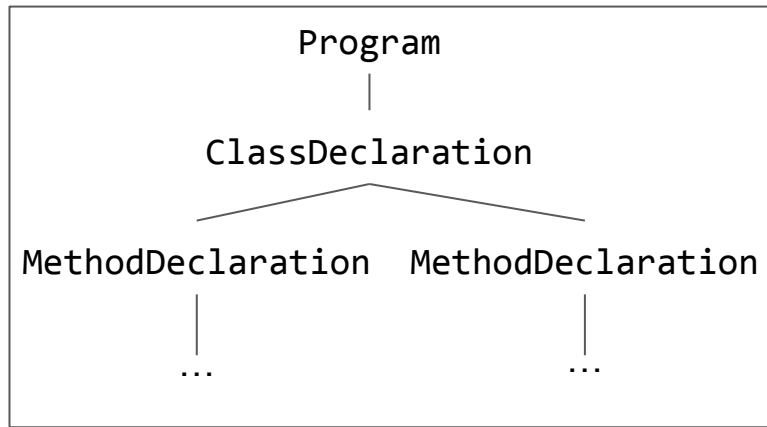
parse file



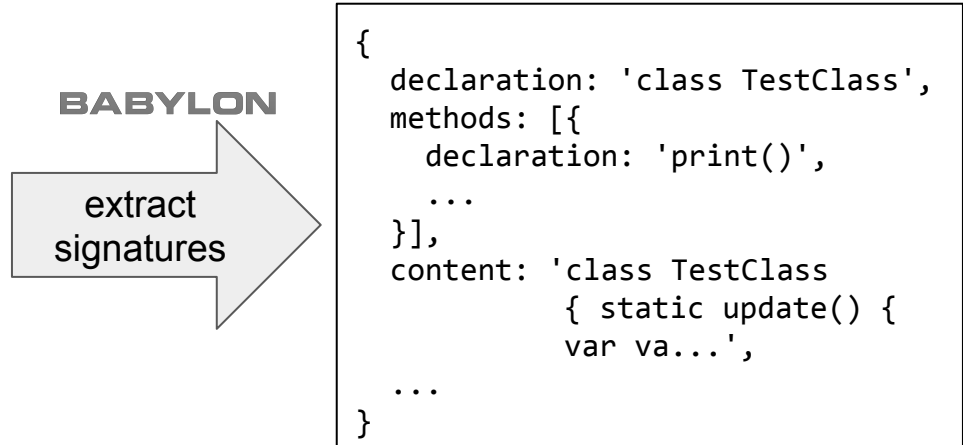
File AST

# Structure

## Workflow - Populate Component



File AST



Signature object

# Structure

## Workflow - Populate Component

```
{  
  declaration: 'class TestClass',  
  methods: [{  
    declaration: 'print()',  
    ...  
  }],  
  content: 'class TestClass  
    { static update() {  
      var va...',  
  ...  
}
```

Signature object

visualize

The screenshot shows the SemanticSourceCodeNavigator interface. It has three main panes: Files, Classes, and Functions. The Files pane lists 'demos/systembrowser/testFile1.js' and 'demos/systembrowser/testFile2.js'. The Classes pane shows 'class TestClass' and 'class TestClass2'. The Functions pane shows 'update2 ()' and 'print2 ()'. Below these panes, the source code for 'update2()' is displayed, showing a static method that logs a message.

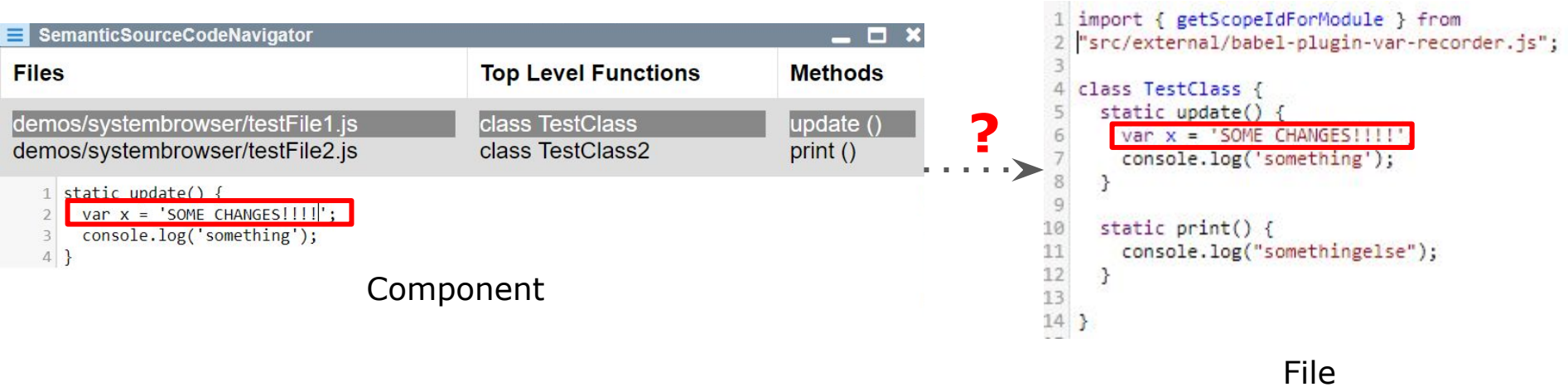
Files	Classes	Functions
demos/systembrowser/testFile1.js	class TestClass	update2 ()
demos/systembrowser/testFile2.js	class TestClass2	print2 ()

```
1 static update2() {  
2   var variables = _recorder_[getScopeIdForModule()];  
3  
4   console.log('something');  
5 }
```

Component

# Structure

## Workflow - Edit File Contents



# Structure

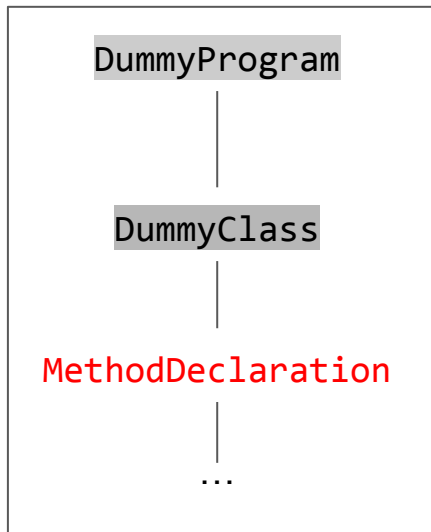
## Workflow - Edit File Contents

SemanticSourceCodeNavigator

Files	Top Level Functions	Methods
demos/systembrowser/testFile1.js	class TestClass	update ()
demos/systembrowser/testFile2.js	class TestClass2	print ()

```
1 static update() {  
2   var x = 'SOME CHANGES!!!!';  
3   console.log('something');  
4 }
```

Component



Part AST

# Structure

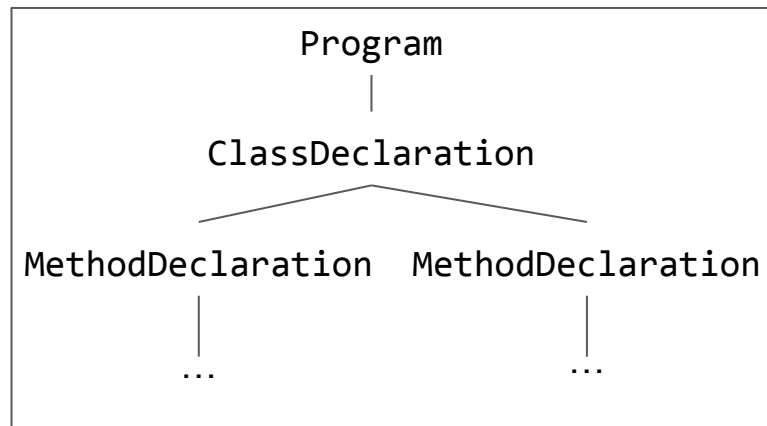
## Workflow - Edit File Contents

```
1 import { getScopeIdForModule } from
2 "src/external/babel-plugin-var-recorder.js";
3
4 class TestClass {
5   static update() {
6     var variables = _recorder_[getScopeIdForModule()];
7
8     console.log('something');
9   }
10
11   static print() {
12     console.log("somethingelse");
13   }
14 }
15
16
```

Original File

**BABEL**

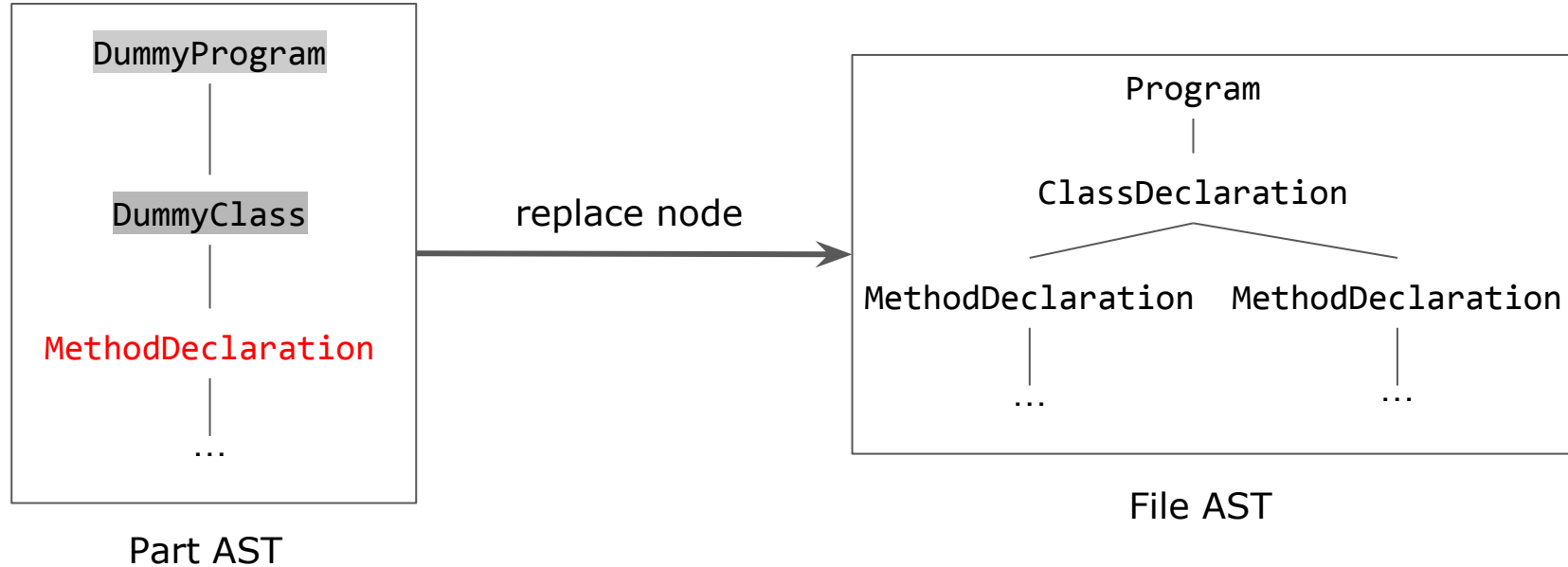
parse file



File AST

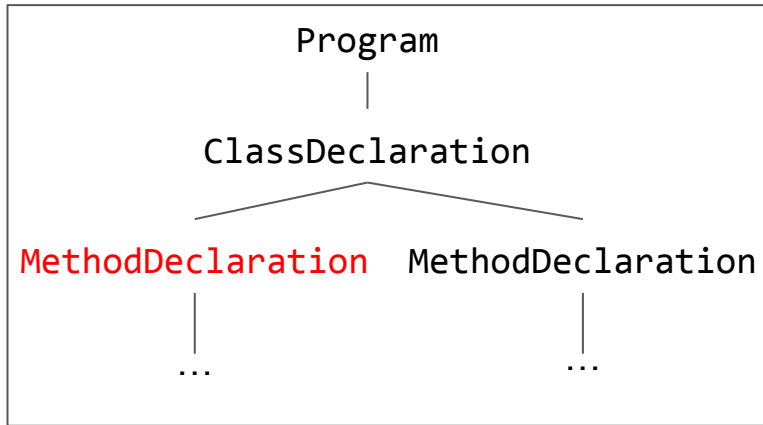
# Structure

## Workflow - Edit File Contents



# Structure

## Workflow - Edit File Contents



File AST

**BABYLON**

print

```
1 import { getScopeIdForModule } from
2 |"src/external/babel-plugin-var-recorder.js";
3
4 class TestClass {
5   static update() {
6     var x = 'SOME CHANGES!!!!'
7     console.log('something');
8   }
9
10  static print() {
11    console.log("somethingelse");
12  }
13
14 }
```

File



# Next Steps & Future Work

---

- Editing file-level
- Integrate IndexedDB (Dexie)
- Variable bulk declarations
- Replace static file list with directory browser

```
(var x, y, z;)
```

SemanticSourceCodeNavigator		
Files	Classes	Functions
demos/systembrowser/testFile1.js	class TestClass	update2 ()
demos/systembrowser/testFile2.js	class TestClass2	print2 ()
<pre>1 static update2() { 2   var variables = _recorder_[getScopeIdForModule()]; 3 4   console.log('something'); 5 }</pre>		

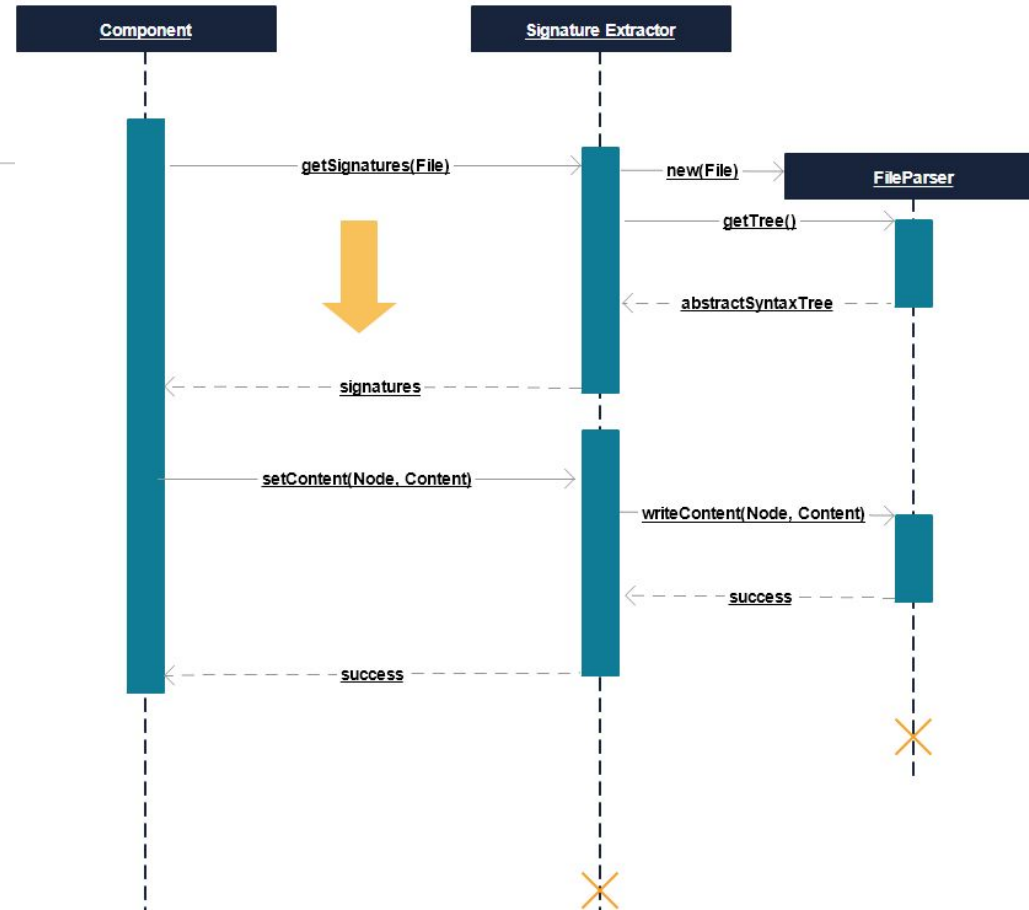
# Happy Navigation!

Web-based Development Environments WS 17/18  
Siegfried Horschig, Theresa Zobel - 06.02.2018

Software Architecture Group 2006-present

# Structure

## Behaviour



# Structure

## Workflow

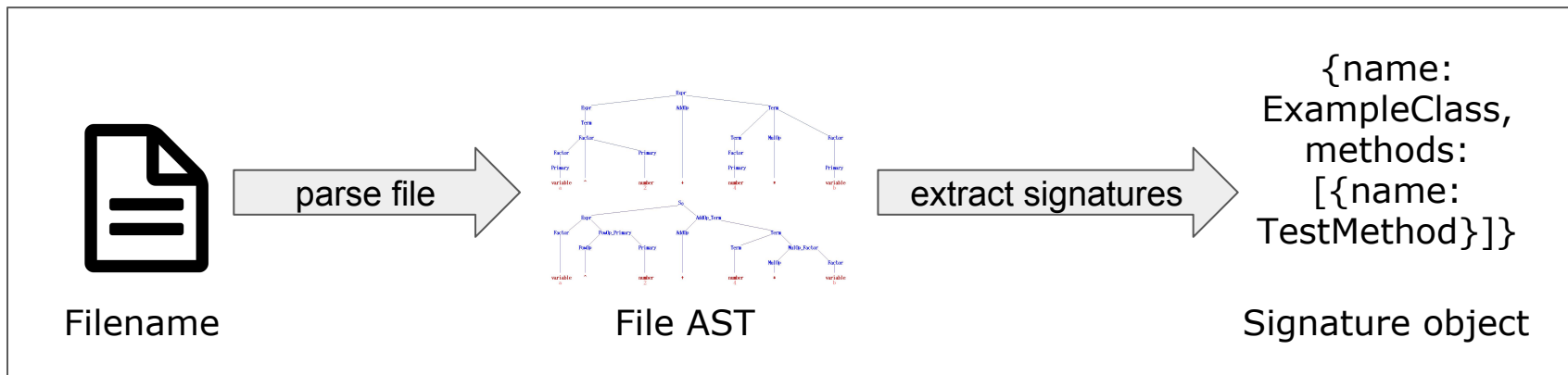
Files	Classes	Functions
demos/systembrowser/testFile1.js	class TestClass	update2 ()
demos/systembrowser/testFile2.js	class TestClass2	print2 ()

```
1 static update2() {  
2   var variables = _recorder._getScopeIdForModule();  
3  
4   console.log('something');  
5 }
```

*filenames*

Component

*signatures*



## Workflow

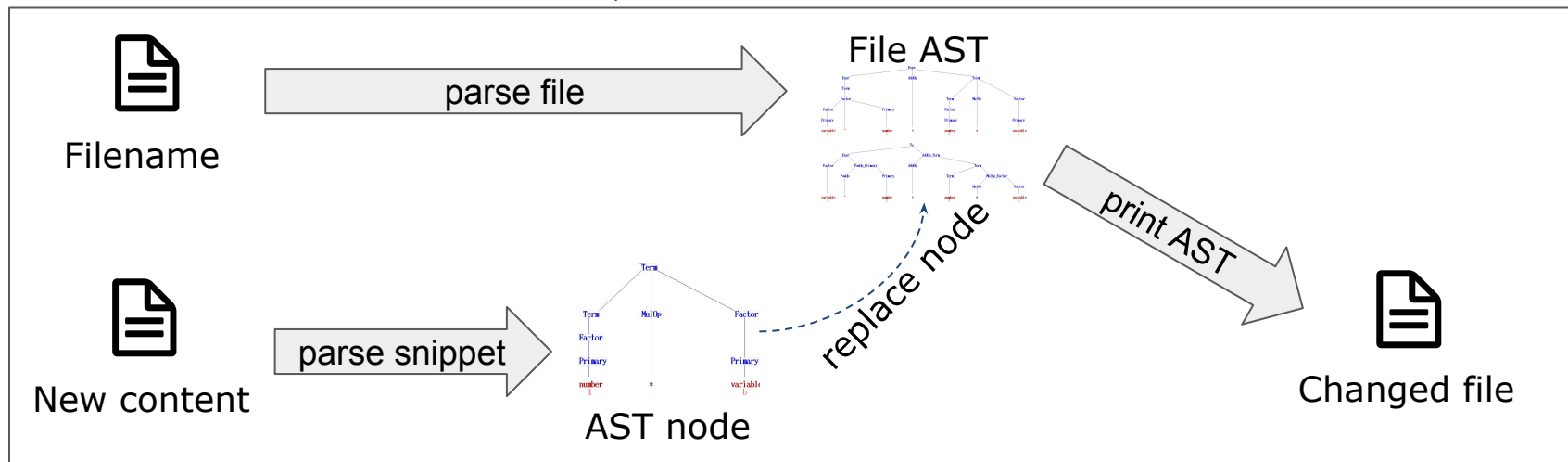
Files	Classes	Functions
demos/systembrowser/testFile1.js	class TestClass	update2 ()
demos/systembrowser/testFile2.js	class TestClass2	print2 ()

```

1 static update2() {
2   var variables = _recorder_[getScopeIdForModule()];
3
4   console.log('something');
5 }

```

*content changes* ↓ Component



# Challenges

- First idea for parsing: Esprima
- **Problem:** newest proposals (ES7) used, like `::` bind-operator

```
> var esprima = require('esprima');  
> var program = 'const answer = 42';  
> esprima.parse(program);  
{ type: 'Program',  
  body:  
    [ { type: 'VariableDeclaration',  
        declarations: [Object],  
        kind: 'const' } ],  
  sourceType: 'script' }
```

=> use Babel



# Challenges

- First idea for AST printing: ESCodeGen
- **Problem:** Babel AST differs from ESTree specification

```
escodegen.generate({  
  type: 'BinaryExpression',  
  operator: '+',  
  left: { type: 'Literal', value: 40 },  
  right: { type: 'Literal', value: 2 }  
});
```

=> use Babylon (Babel built-in AST printer)

# Challenges

- First idea for AST printing: ESCodeGen
- **Problem:** Babel AST differs from ESTree specification

```
escodegen.generate({  
  type: 'BinaryExpression',  
  operator: '+',  
  left: { type: 'Literal', value: 40 },  
  right: { type: 'Literal', value: 2 }  
});
```



# Challenges

---

- Using Babylon as AST printer
- **Problem:** Very 'picky' with AST structure, can't parse AST-parts

BABYLON

=> Create wrapper objects