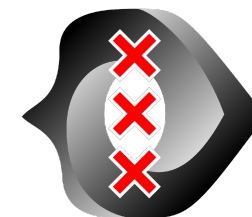
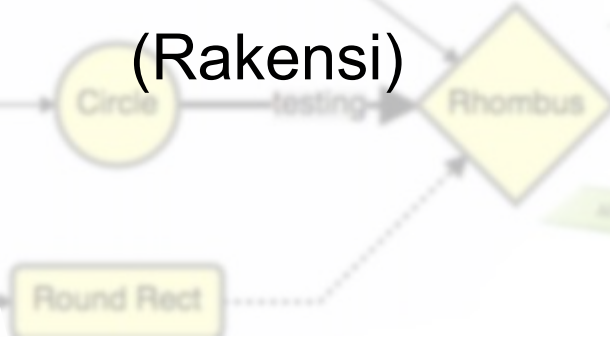


Declarative Diagrams with Mermaid

Nico Verwer
(Rakensi)



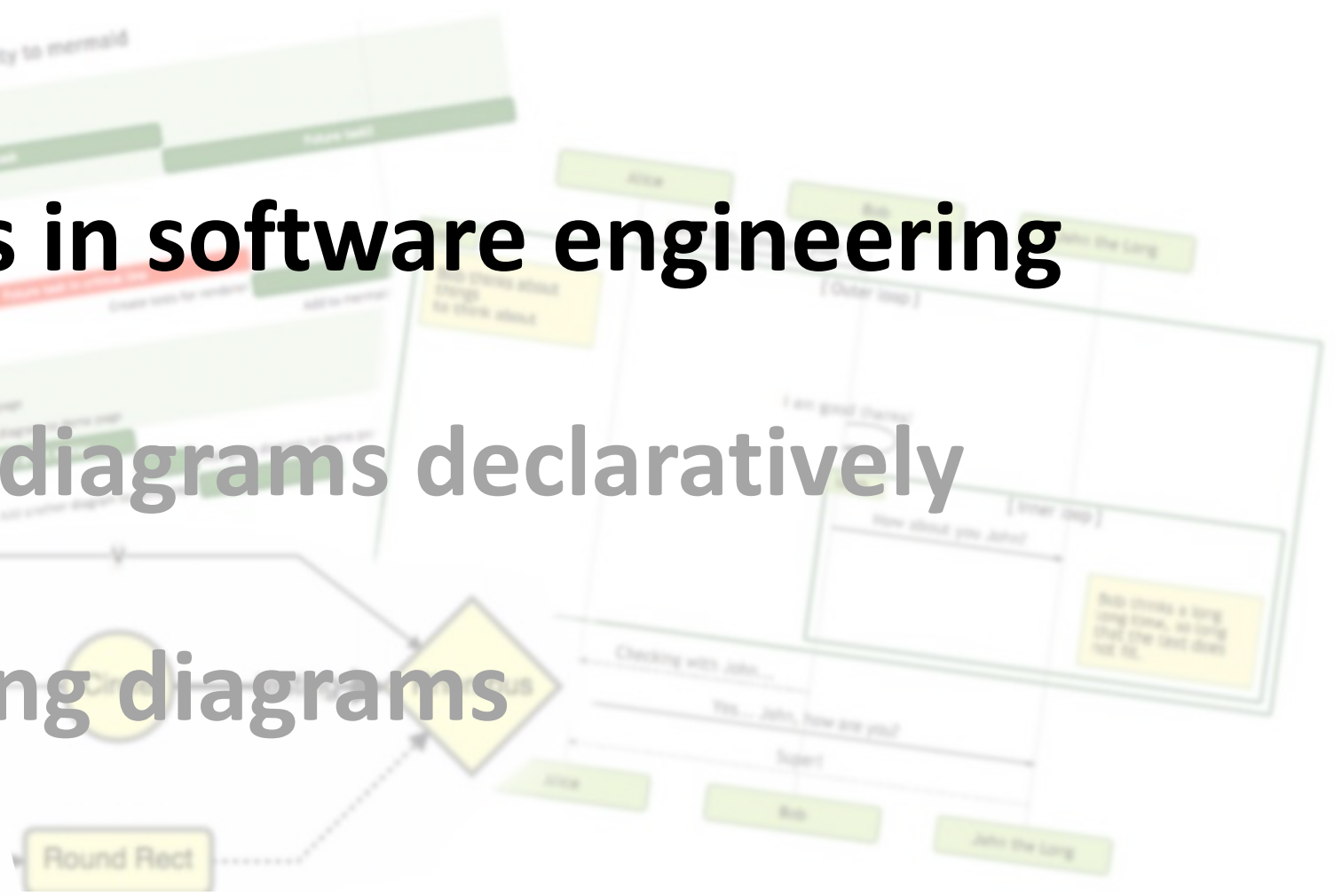
Declarative
Amsterdam

Part 1: Diagrams in software engineering

Part 2: Drawing diagrams declaratively

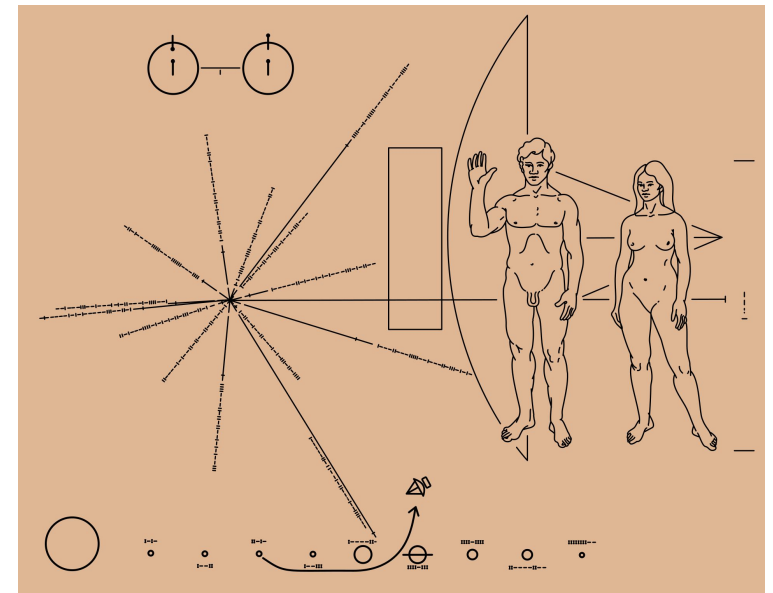
Part 3: Generating diagrams

```
41 loadOptionsFromFile --> setOptions
42 loadOptionsFromFile --> setOptions
43 loadOptionsFromFile --> setOptions
44 nextclick[[nextclick]] --> setOptions
45 optionChangedHandler --> updateOptionsProperties
46 optionChangedHandler --> updateOptionsProperties
47 optionChangedHandler --> updateOptionsProperties
48 prevclick[[prevclick]] --> setFilteredOptions
49 resetSearch --> setFilteredOptions
50 resetSearch --> setFilteredOptions
51 setFilteredOptions --> fillOptionList
52 setFilteredOptions --> updateFooter
53 setFilteredOptions --> updateFooter
54 setOptions --> setInputFieldInfo
55 setOptions --> validateInput
56 setOptions --> validateInput
57 showOptionsList --> setFilteredOptions
58 showOptionsList --> setFilteredOptions
59 showOptionsList --> setFilteredOptions
```



Why use diagrams?

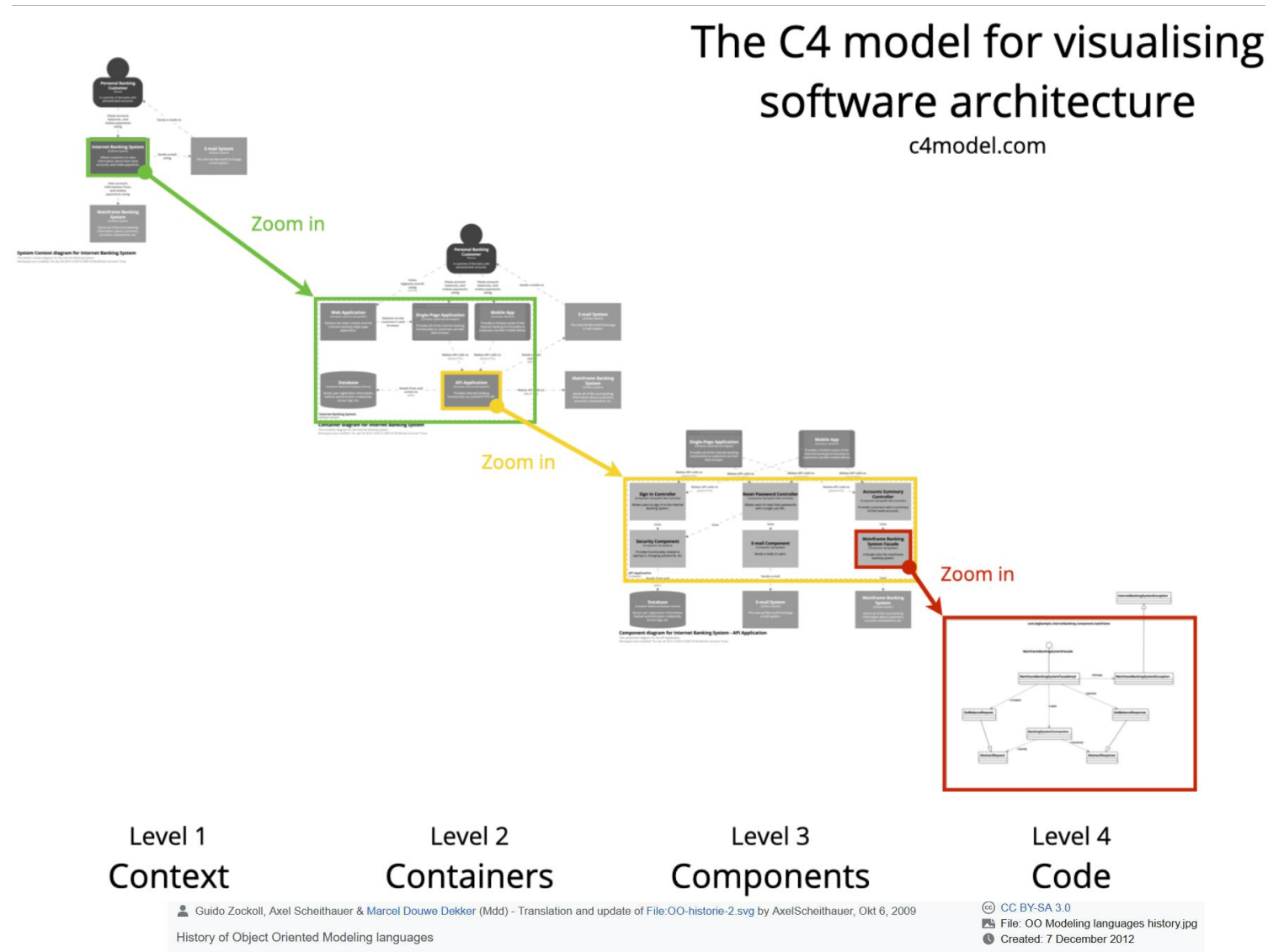
- A diagram is a symbolic representation of information using visualization techniques.
 - “Diagrams are simplified figures, caricatures in a way, intended to convey essential meaning.” – Bert S. Hall
- Overview of the structure of one aspect of a complex system.
 - Understand how complex systems work.
 - Diagrams may help to solve problems.
- Help to communicate.
 - Symbols are more universal than language.
 - Enable users to provide feedback.



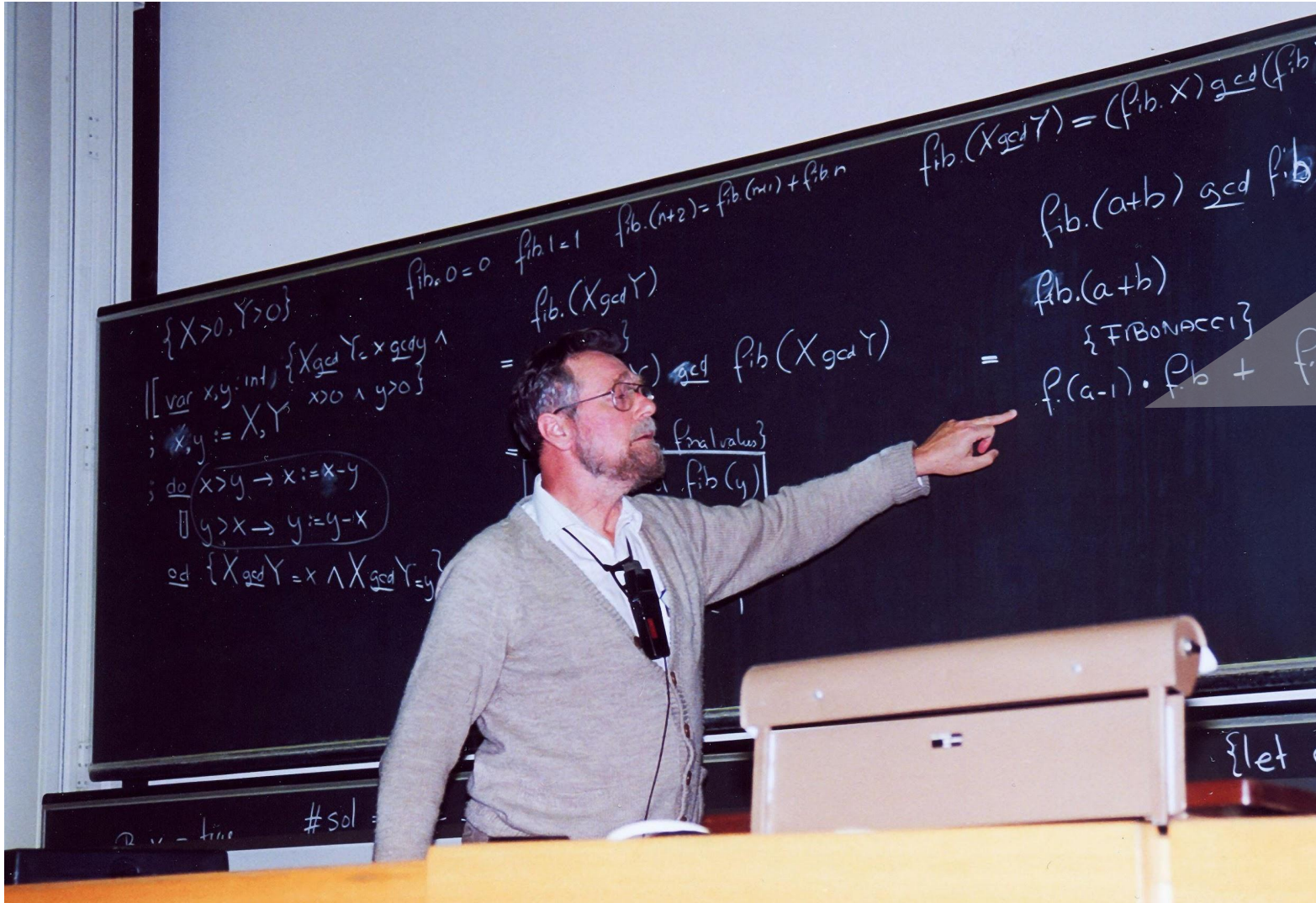
Diagrams for software engineering

- Flowcharts (1921)
- Nassi-Shneiderman (1972)
- ERD (1976)
- UML (1994)
- C4 (2006)

- Control flow / data flow
- Workflow
- Syntax diagram
- ...



Diagrams don't mean *much*



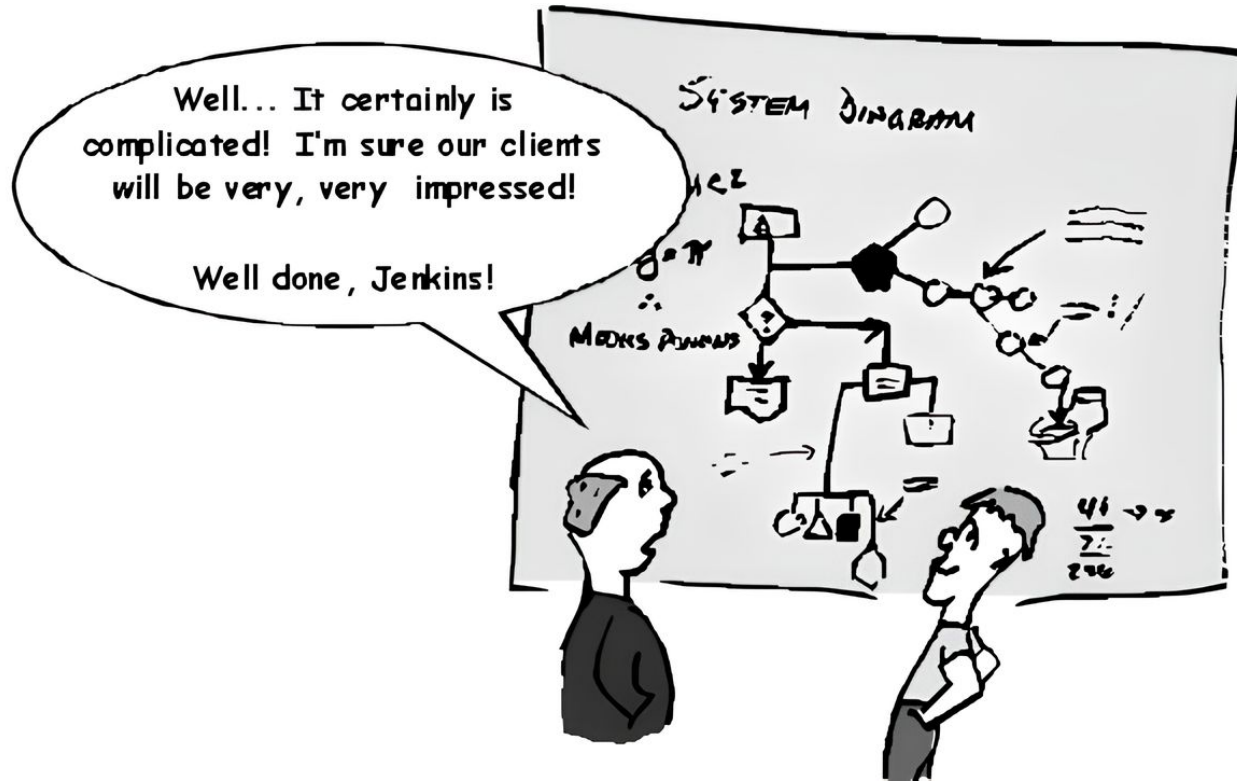
“A picture may be worth a thousand words, a formula is worth a thousand pictures.”

Edsger W. Dijkstra, 1996
(EWD1239: A first exploration of effective reasoning)

“A picture is worth 10K words - but only those to describe the picture. Hardly any sets of 10K words can be adequately described with pictures.”

Alan Perlis, 1982
(Epigrams on Programming)

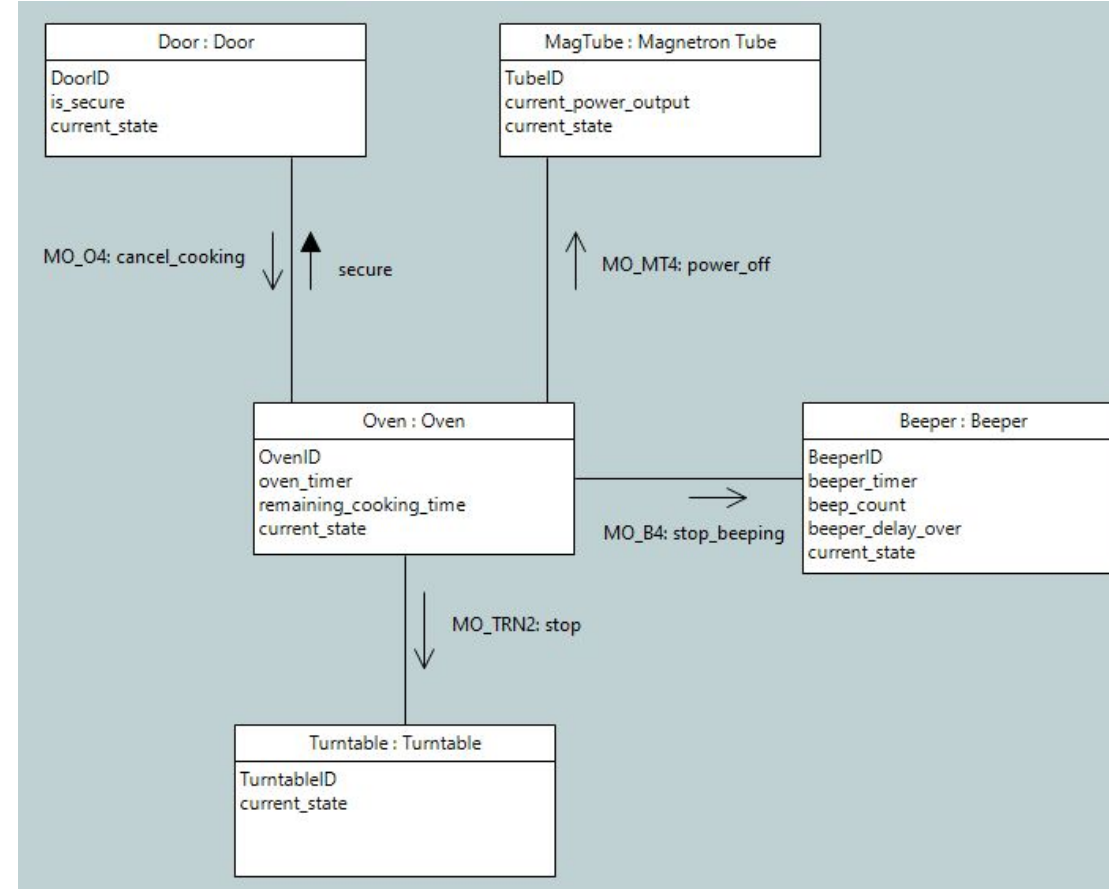
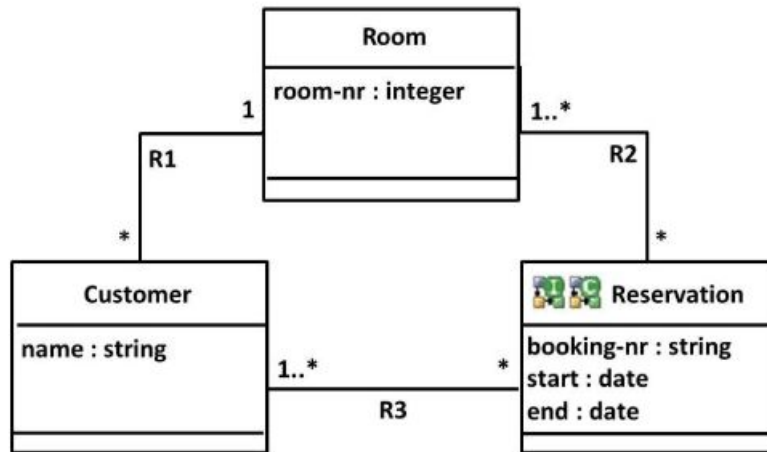
Complexity, information content, ...



PowerPointless Presentation

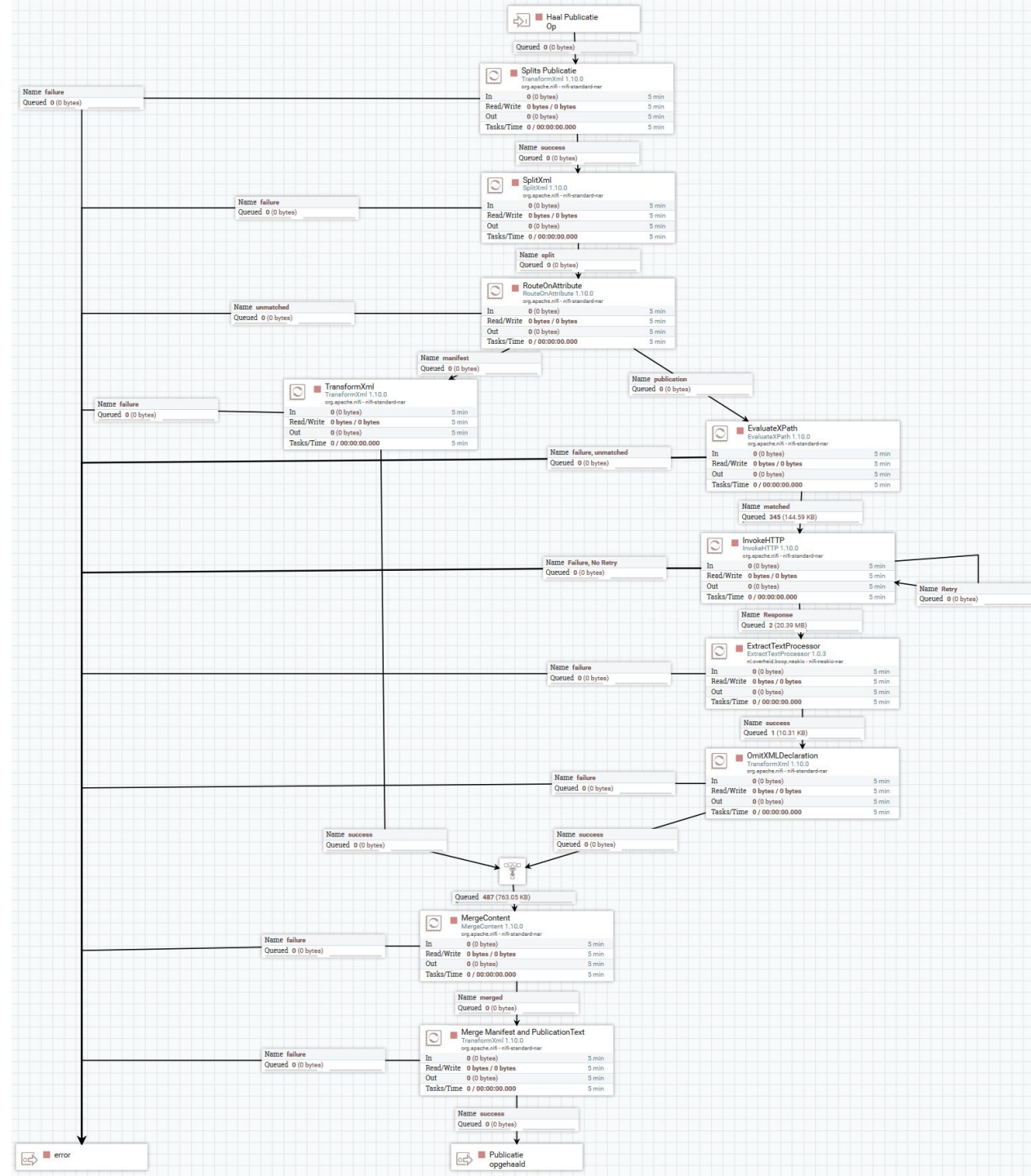
Executable diagrams

- Shlaer–Mellor (1988) □ Executable UML (2002).
- xtUML.org (open source executable and translatable UML modeling).



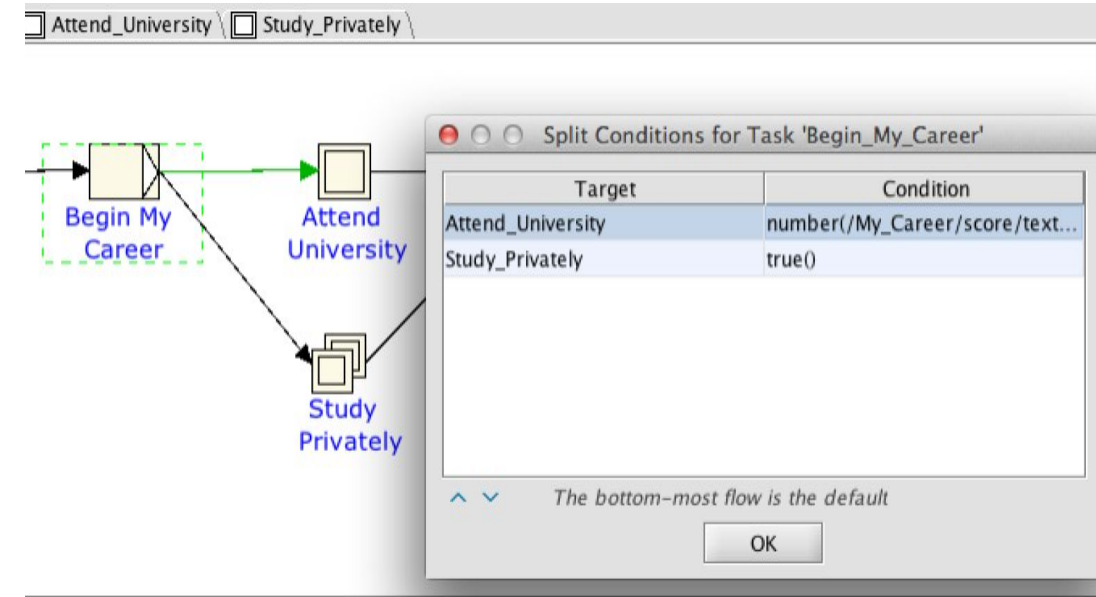
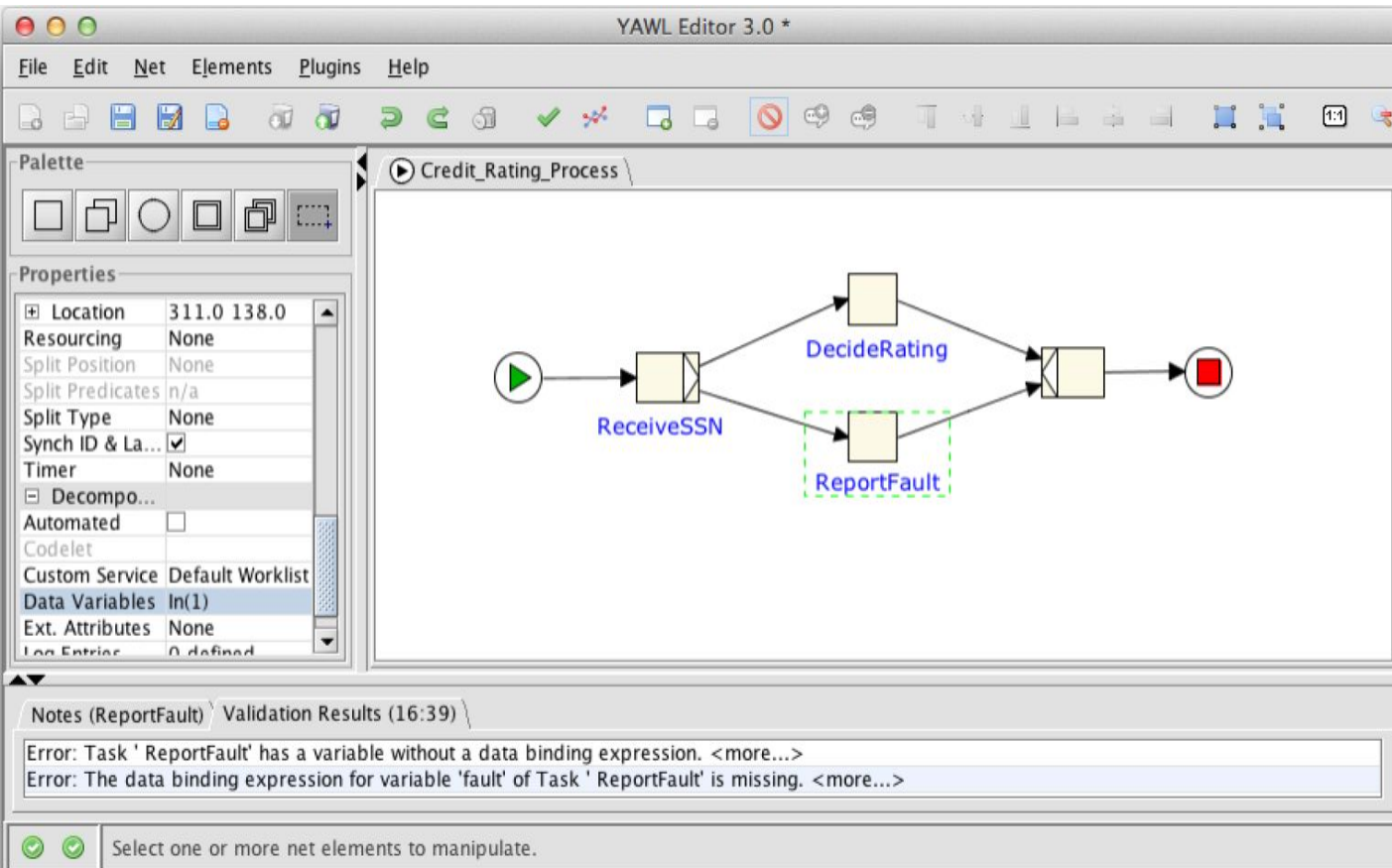
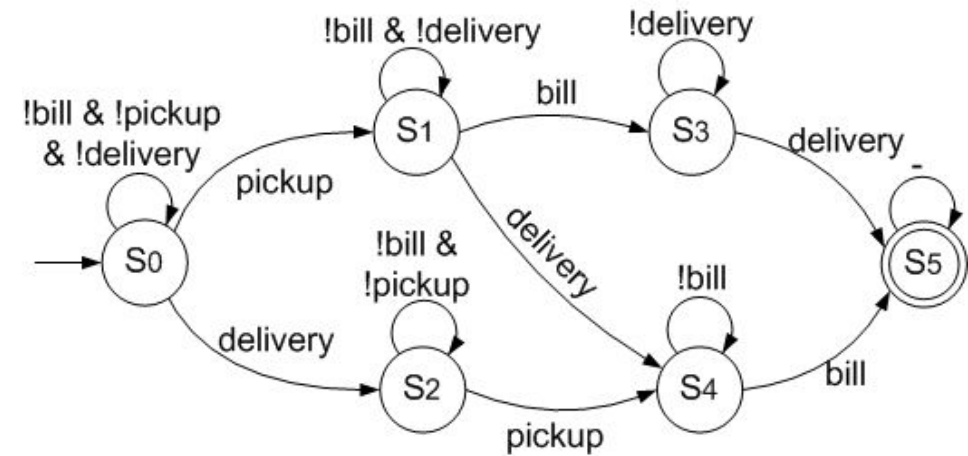
Executable diagrams

- Apache Nifi
- Directed graphs of data routing, transformation, and logic.
- Data pipelines as diagrams.



Executable diagrams

- YAWL BPM
 - Uses Xquery.

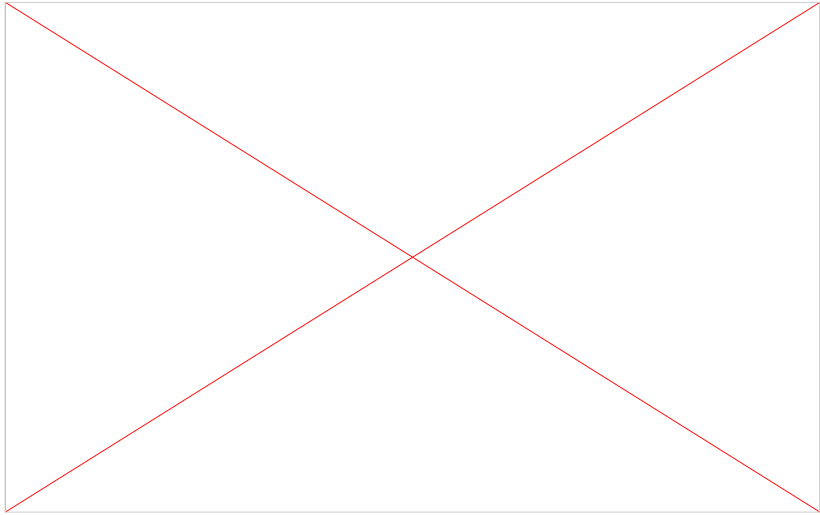
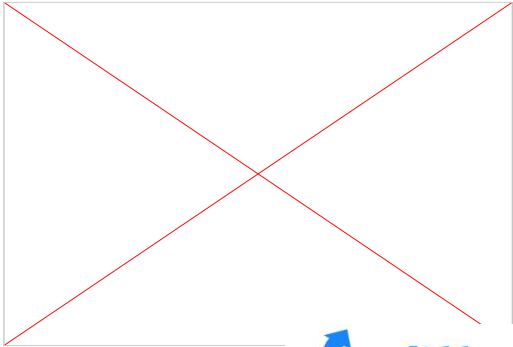
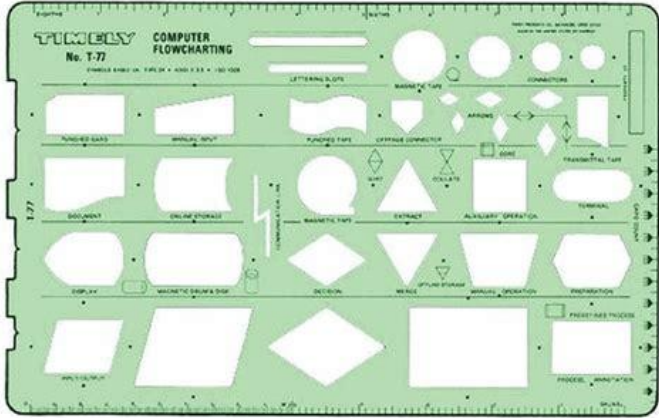
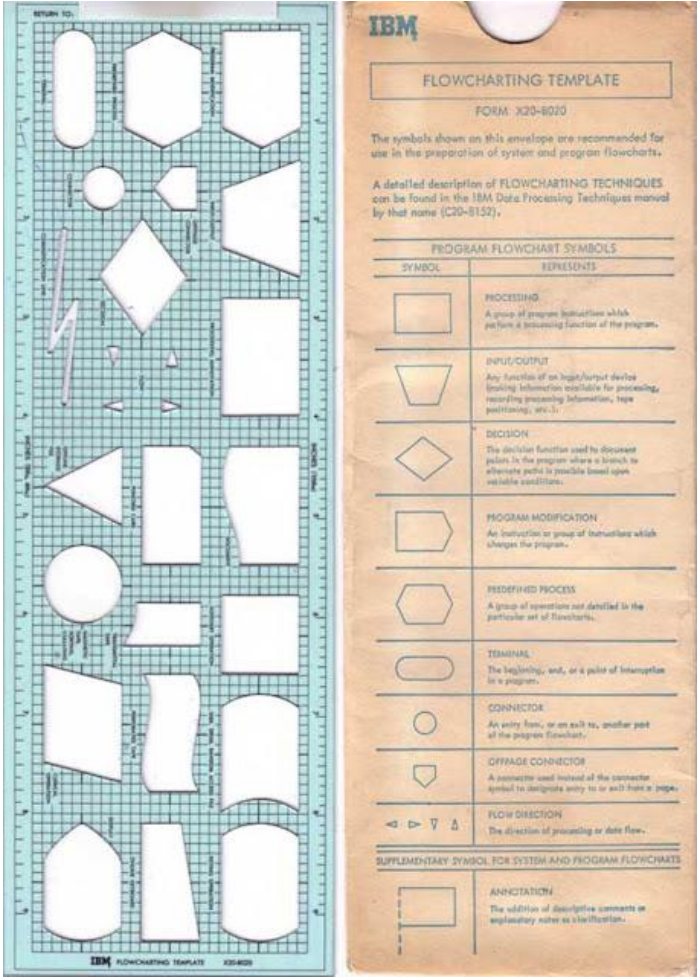


Part 1: Diagrams in software engineering

Part 2: Drawing diagrams declaratively

Part 3: Generating diagrams

Tools to make diagrams



Declarative drawing

- Drawing boxes and lines: How.
- Specifying as code: What.
- Many implementations.
 - textografo, flowchart.js, plantUML, kroki, mingrammer, structurizr, yuml, mermaid
- Here we will use Mermaid.
 - <https://mermaid.js.org/>
 - <https://mermaid.live/>



Demo

- Example 1 (flow chart) mermaid.live VS code
- Example 2 (flow chart) VS code
- Example 3 (flow chart) VS code
- Example 4 (class diagram) VS code
- Example 5 (sequence diagram) VS code
- Example 6 (ER diagram) VS code
- Example 7 (other diagrams) VS code

Advantages of declarative diagram drawing

- Declarative, specify what to draw, not how.
- Maintenance: No re-arranging of boxes all the time.
- Interactive real-time updates, see the results immediately.
- Work on design with customer.

- Generate diagrams. (next)

- Execute diagrams?

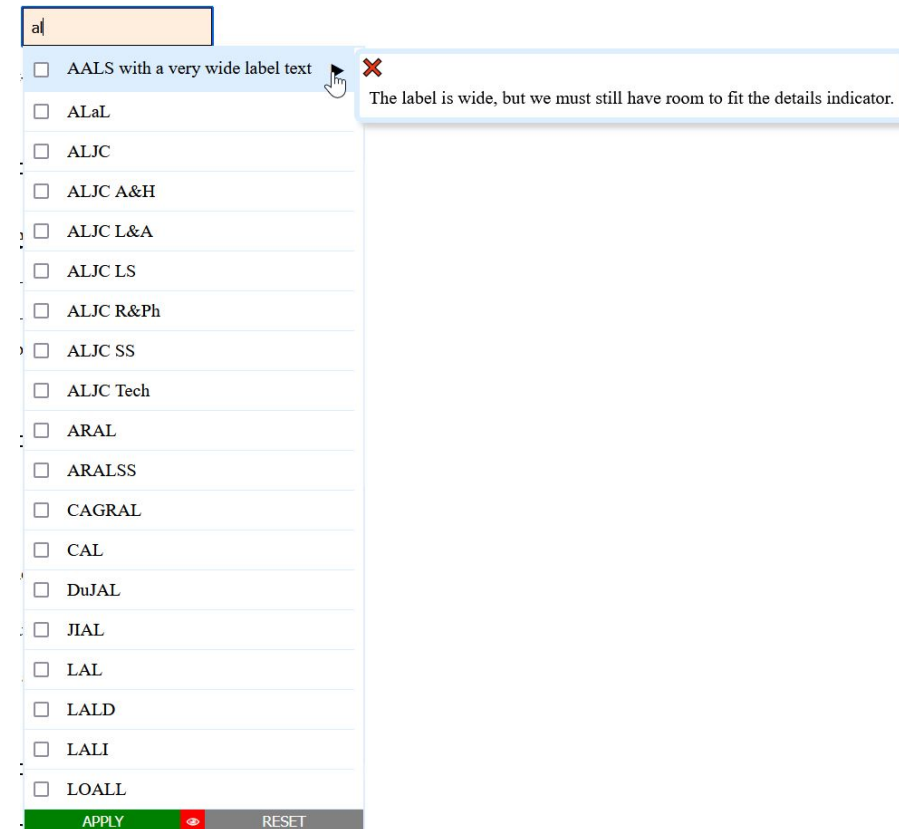
Part 1: Diagrams in software engineering

Part 2: Drawing diagrams declaratively

Part 3: Generating diagrams

Diagrams for code analysis

- Used in development of a combo-select web component for Benjamins.
- Web components respond to many events.
- More than 40 functions, close to 1000 lines.
- Many different code paths can be followed.
- The component became slow.
- Some functions were called 2 or 3 times when certain events occurred.



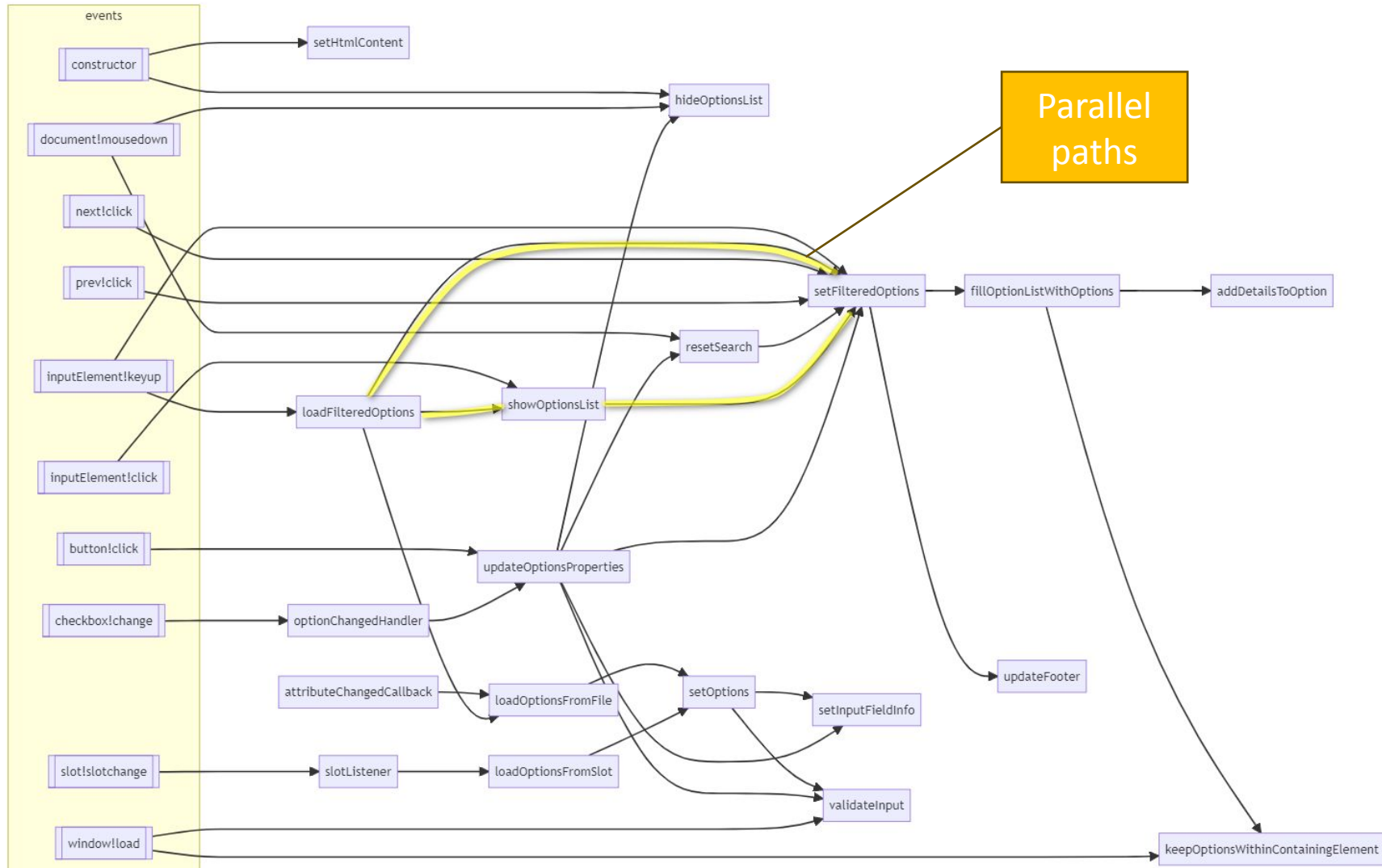
Find functions that are called too often

```
562 loadFilteredOptions() {
563   const src = encodeURI(this.src.replace('${search}', encodeURIComponent(this.optionsFilter)));
564   this.debug && console.info(`jb-combo-select (${this.name}): loadFilteredOptions ${src}`);
565   if (this.optionsFilter.length < this.minsearchlength) {
566     //this.setOptions([]); // this overwrites the search input
567     this.shadow.querySelector('.jb-option-footer').innerHTML =
568     | `search for at least ${this.minsearchlength} characters`;
569   } else {
570     this.shadow.querySelector('.jb-option-footer').innerHTML = 'loading options...';
571     this.loadOptionsFromFile(src)
572     .then(result => {
573       this.setFilteredOptions();
574       this.showOptionsList();
575     });
576   }
577 }
```

```
613 showOptionsList() {
614   this.debug && console.info(`jb-combo-select (${this.name}): showOptionsList`);
615   this.optionContainer.style.visibility = 'visible';
616   this.setFilteredOptions();
617 }
```

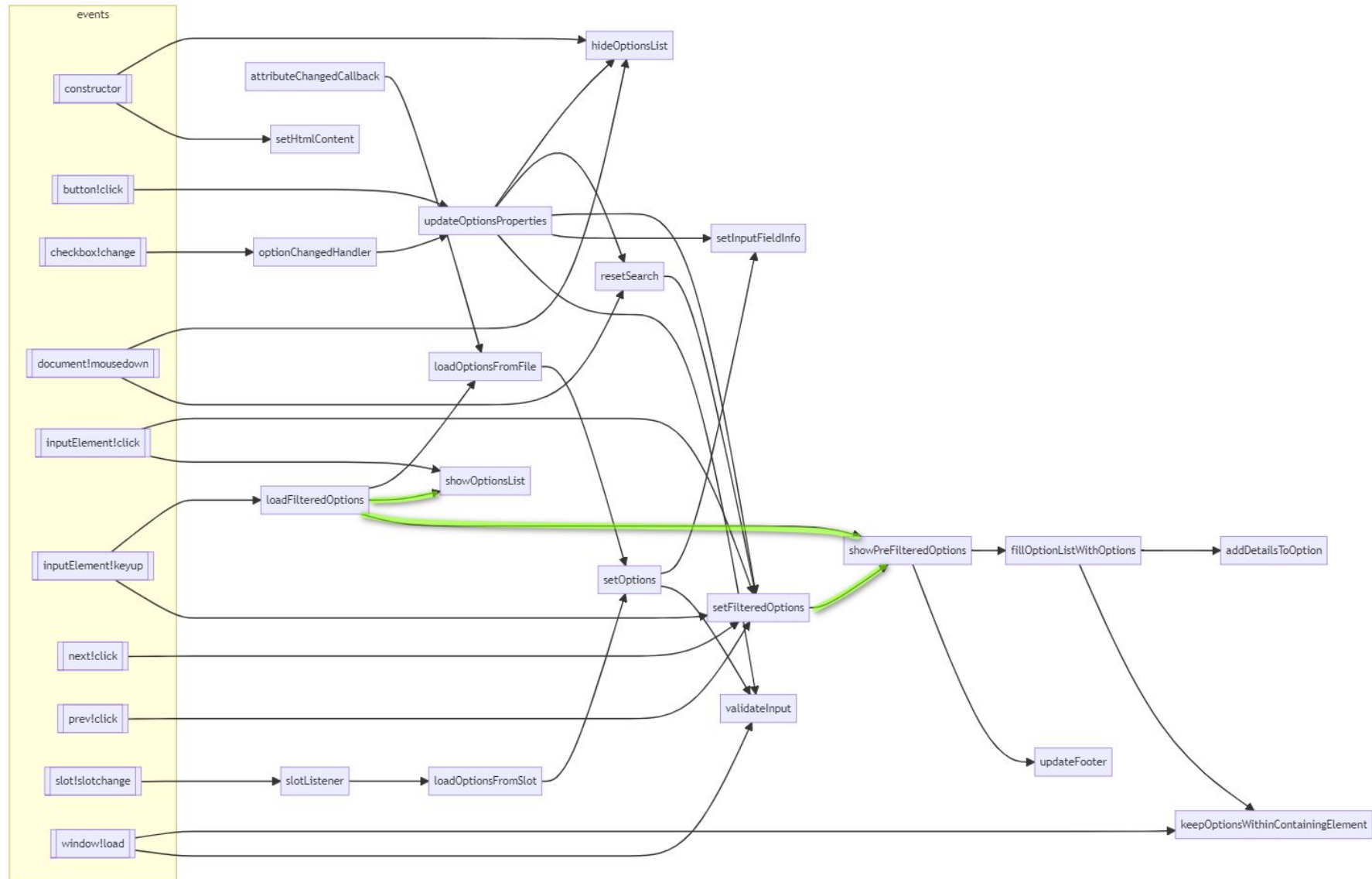
The call graph as a Mermaid diagram

Preview jb-combo-select0411a.md



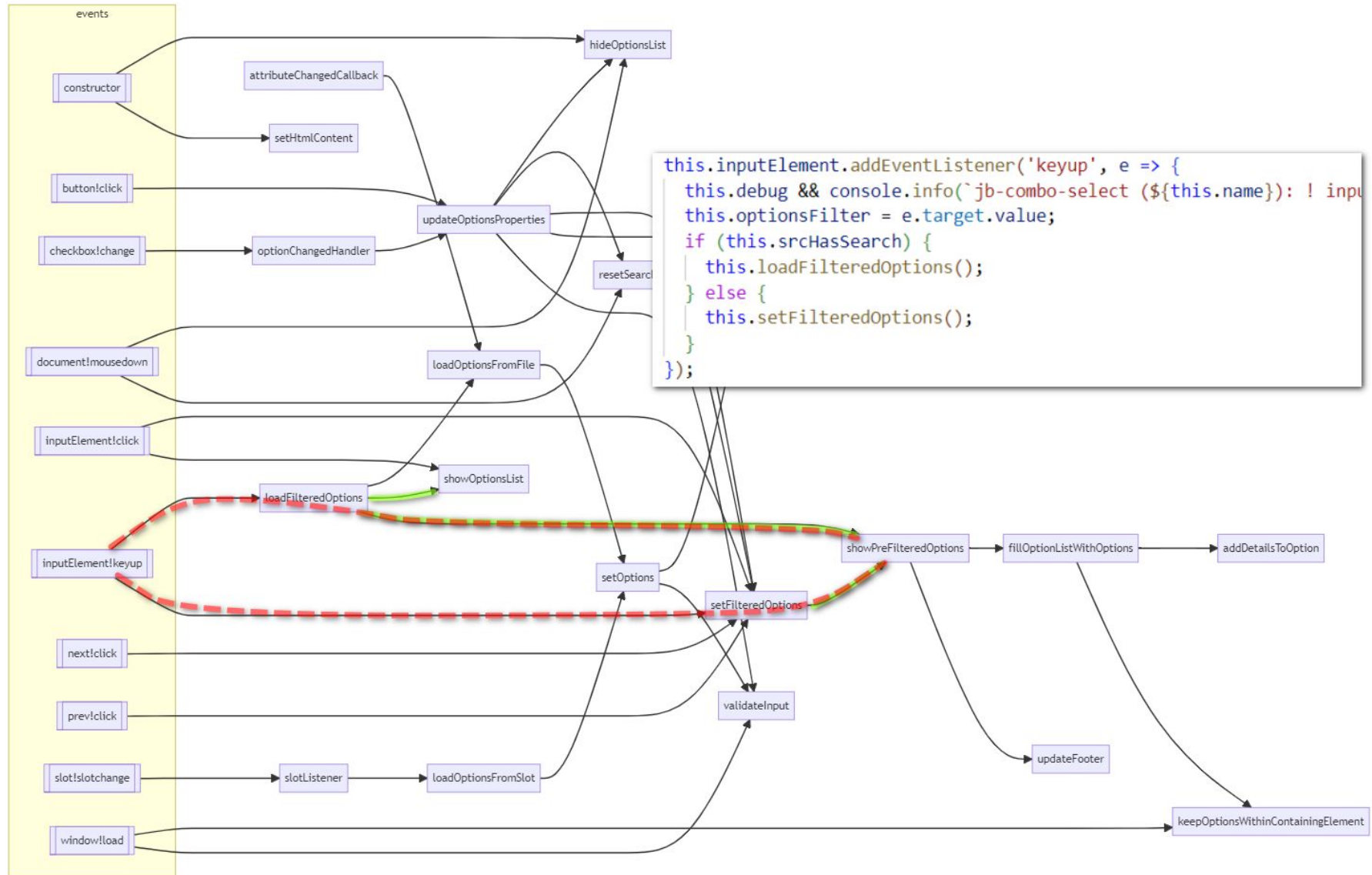
Call graph for the revised code

Preview jb-combo-select0411b.md ×



One call graph is not enough

Preview jb-combo-select0411b.md ×



Generating Mermaid code for a call graph

- Specific for web components.
 - All paths start with the constructor or an event.
 - <https://github.com/nverwer/DA2023>
- To do:
 - Highlight parallel paths.
 - Generalize to other types of code.
 - Split diagrams at if – else branches.
- Future:
 - Use iXML to parse Mermaid syntax and generate code?

Finding parallel paths

- Use an algorithm (in the original meaning of the word).
 - Use *widest path* problem (Schulze method, Floyd-Warshall, Dijkstra).
- Use AI / LLM / GPT
 - The “ultimate declarative programming language”,
 - or a hallucinating stochastic parrot.
- Maybe just a glorified search engine.
- phind.com

phind

☰ ☰ Nico Verwer

Your AI search engine and pair programmer.

Describe your task in detail. What are you stuck on?

▼ 🔍

Pair Programmer

GPT-4 ▼

Using Phind (ChatGPT) to find parallel paths

- (loadFilteredOptions, setFilteredOptions)
- (loadFilteredOptions, showOptionsList)
- (loadOptionsFromFile, setOptions)
- (updateOptionsProperties, hideOptionsList)
- (updateOptionsProperties, resetSearch)
- (updateOptionsProperties, setFilteredOptions)
- (updateOptionsProperties, setInputFieldInfo)
- (updateOptionsProperties, validateInput)
- (window!load, keepOptionsWithinContainingElement)
- (window!load, validateInput)

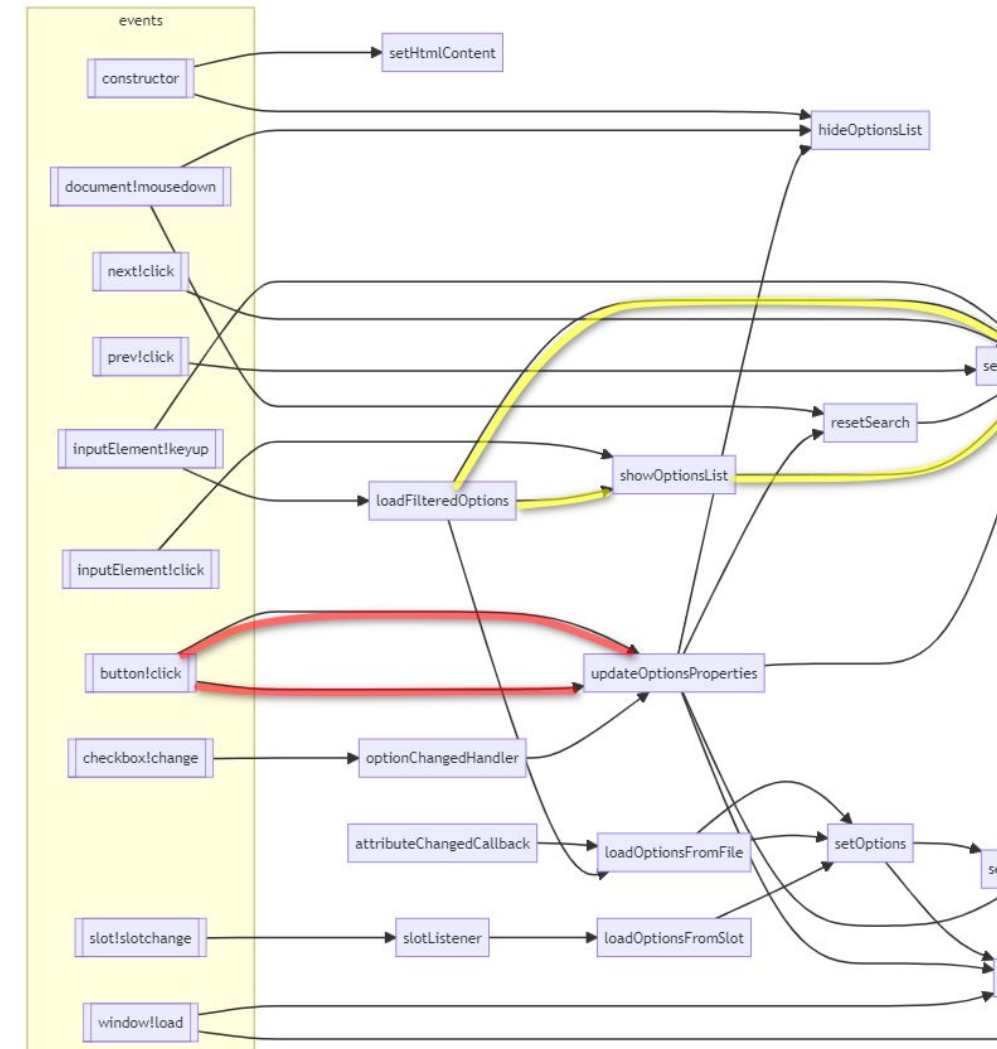
Note that in Mermaid diagrams, an arrow (`-->` or `--->`) represents a path from one node to another mermaid.js.org. **Therefore, any pair of nodes that have more than one arrow connecting them can be considered to have more than one path from A to B.**

For instance, consider the pair (constructor, hideOptionsList). The diagram shows two arrows (`--->`) leading from the 'constructor' node to the 'hideOptionsList' node indicating that there are two paths from 'constructor' to 'hideOptionsList'.

Similarly, for the pair (button!click, updateOptionsProperties), there are two arrows (`--->`) leading from the 'button!click' node to the 'updateOptionsProperties' node, indicating two paths from 'button!click' to 'updateOptionsProperties'.

This process is repeated for all the pairs listed above.

Preview jb-combo-select0411a.md ×



Trying to help Phind (ChatGPT)

Can you include paths that consist of more than one arrow? This is called the transitive closure. For example, if there is an arrow $A \rightarrow C$ and an arrow $C \rightarrow B$, that counts as a path from A to B . ^

ANSWER | GPT-4 MODEL

To include paths that consist of more than one arrow (also known as the transitive closure), we need to consider all possible paths that can be taken from node A to node B , not just direct connections. This process involves looking at the entire network of nodes and arrows to see if there is a sequence of arrows that leads from A to B .

From the given Mermaid diagram, the following pairs of nodes have more than one path from A to B , including those that consist of more than one arrow:

- (constructor, hideOptionsList) ❌
- (constructor, setHtmlContent) ❌
- (button!click, updateOptionsProperties) ❌
- (document!mousedown, hideOptionsList)
- (document!mousedown, resetSearch)
- (inputElement!keyup, loadFilteredOptions)
- (inputElement!keyup, setFilteredOptions) ✓
- (loadFilteredOptions, loadOptionsFromFile) ❌
- (loadFilteredOptions, setFilteredOptions)

To identify these paths, we can utilize an algorithm for computing the transitive closure of a graph like the Floyd-Warshall algorithm or Warshall's algorithm [geeksforgeeks.org](https://www.geeksforgeeks.org/warshalls-algorithm/), [math.libretexts.org](https://math.libretexts.org/Bookshelves/Discrete_Mathematics/Book%3A_Discrete_Mathematics_(Lehman%2C_Ling%2C_Lovink)/Chapter10%3A_Graph_Theory/10.3%3A_Reachability_in_Graphs). These algorithms use the concept of reachability to determine if there is a path from one node to another.

Consider the first Mermaid diagram I gave, can you list the pairs (A, B) of nodes that are connected by more than one path, plus for each path the seq... ^

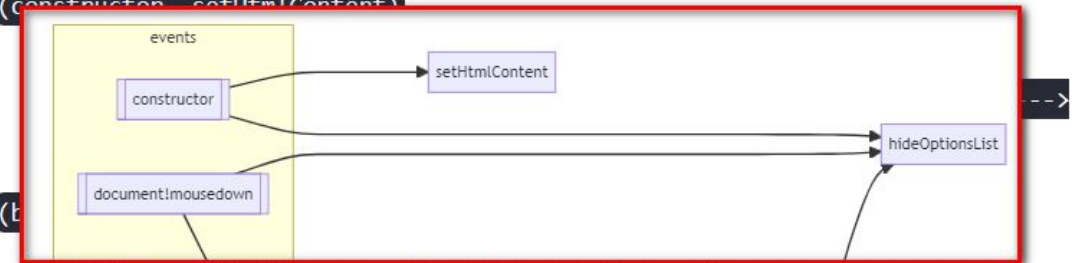
ANSWER | GPT-4 MODEL

To list pairs of nodes (A, B) that are connected by more than one path, and to include each path's sequence of arrows, we can trace the paths from the given Mermaid diagram. Here are the pairs:

1. (constructor, hideOptionsList)

- Path 1: constructor \rightarrow hideOptionsList
- Path 2: constructor \rightarrow setHtmlContent \rightarrow updateOptionsProperties \rightarrow hideOptionsList

2. (constructor, setHtmlContent)



3. (button!click, updateOptionsProperties)

- Path 2: button!click \rightarrow updateOptionsProperties \rightarrow hideOptionsList \rightarrow updateOptionsProperties

4. (document!mousedown, hideOptionsList)

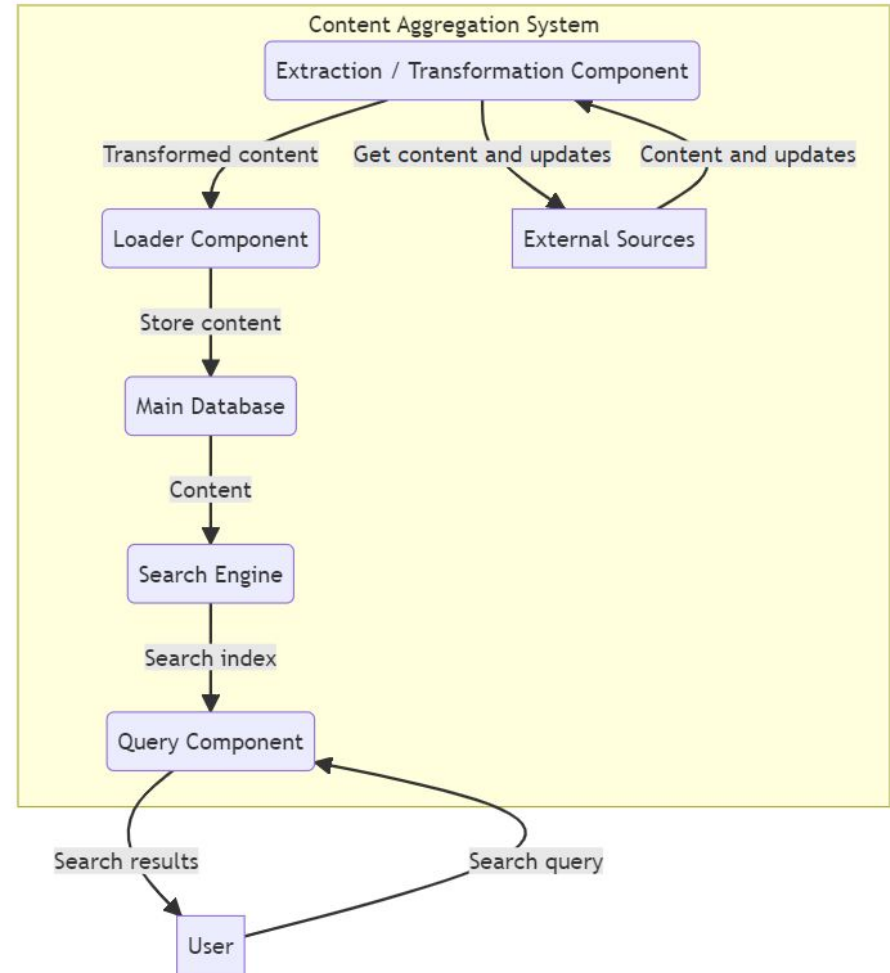
- Path 1: document!mousedown \rightarrow hideOptionsList
- Path 2: document!mousedown \rightarrow resetSearch \rightarrow setFilteredOptions \rightarrow fillOptionsListWithOptions \rightarrow addDetailsToOption \rightarrow setOptions \rightarrow hideOptionsList

Generating diagrams from descriptions

A content aggregation system consist of the following components:

1. A number of connectors to external sources.
2. An extraction / transformation component that gets content and updates from the external sources.
3. A loader component that stores the transformed content in the main database.
4. The main database stores the extracted and transformed content.
5. A search engine that creates and updates a search index for the content in the main database.
6. A query component that allows users to search the aggregated content.

Please create a C4 context diagram for this using Mermaid syntax.

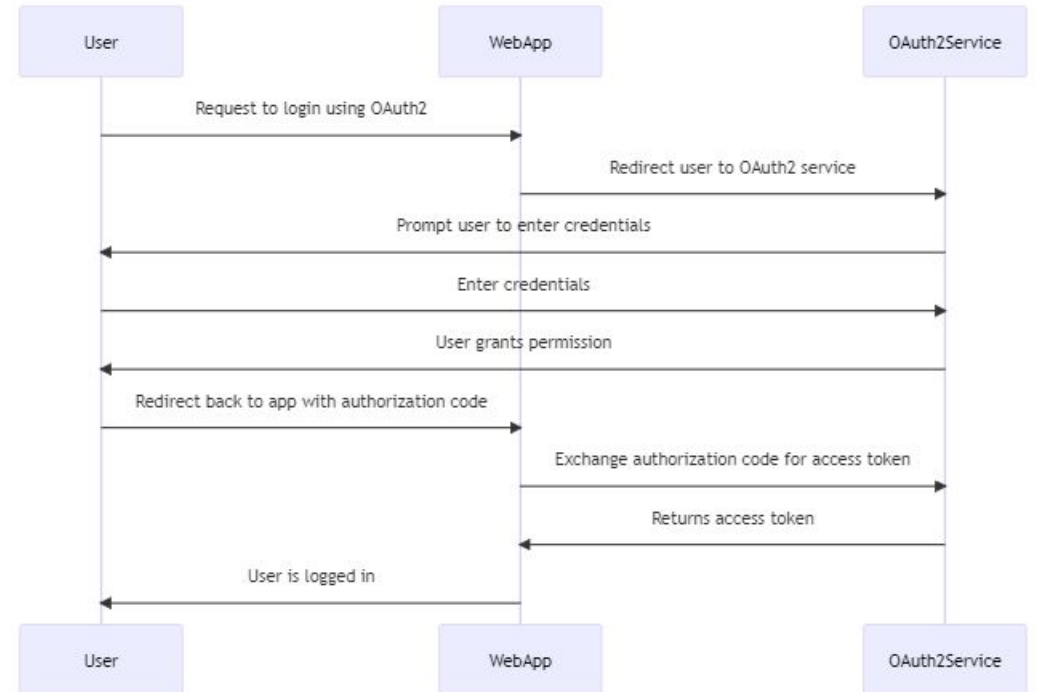
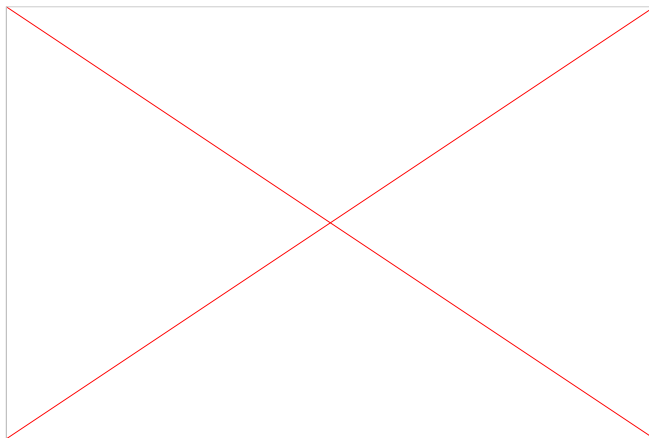


Generate something more common

“How can a user interact with an webapp and login using OAuth2?”

Please generate a sequence diagram in Mermaid syntax to explain.”

The result looks convincing, but is slightly wrong.



Conclusion

- Using declarative syntax for diagrams can save time and effort.
- Fine-tuning layout is not possible
 - But in many cases you should not want that.
- Generating diagrams from code can provide insights.
 - But a real code analyzer may be more useful.
- Ai is not going to take our jobs yet.
 - But the JSON generation should be worried?

THANK YOU
Any questions?

