

Platform-Independent UI Models: Extraction from UI Prototypes and rendering as W3C Web Components

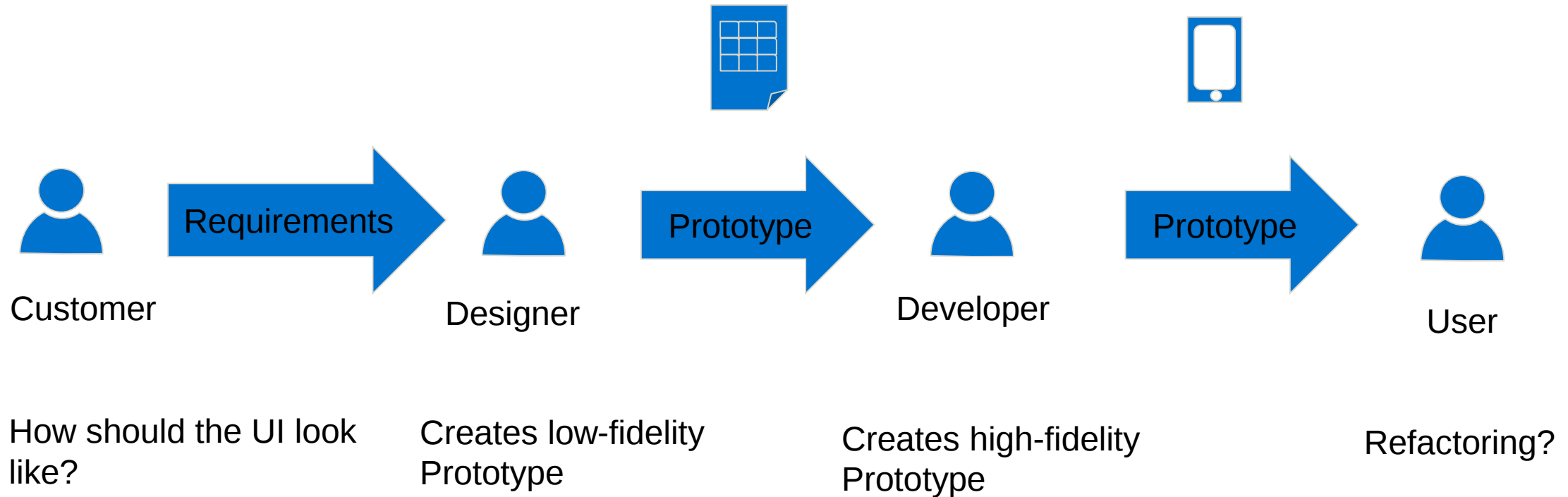
Marvin Aulenbacher, 13.11.2017, München

Chair of Software Engineering for Business Information Systems (sebis)
Faculty of Informatics
Technische Universität München
www.matthes.in.tum.de

1. Introduction
 - Motivation
 - Problem Description
2. Platform Independent View Model
 - View Model
 - Proposed Process
3. Design & Implementation
 - Development Approach
 - Example Walk-Through
4. Live Demo
5. Evaluation
 - Use Cases
 - Limitations
 - Future Work

Motivation

Prototype driven development in web application development



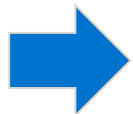
Problem Description

Communication Issue between Designer and Developer

Designer



Developer



Do not share common domain specific language

- Support the developer by generating web components from an UI prototype
- Standardize in- and output file formats(SVG, JSON)
- Use view models as domain specific language

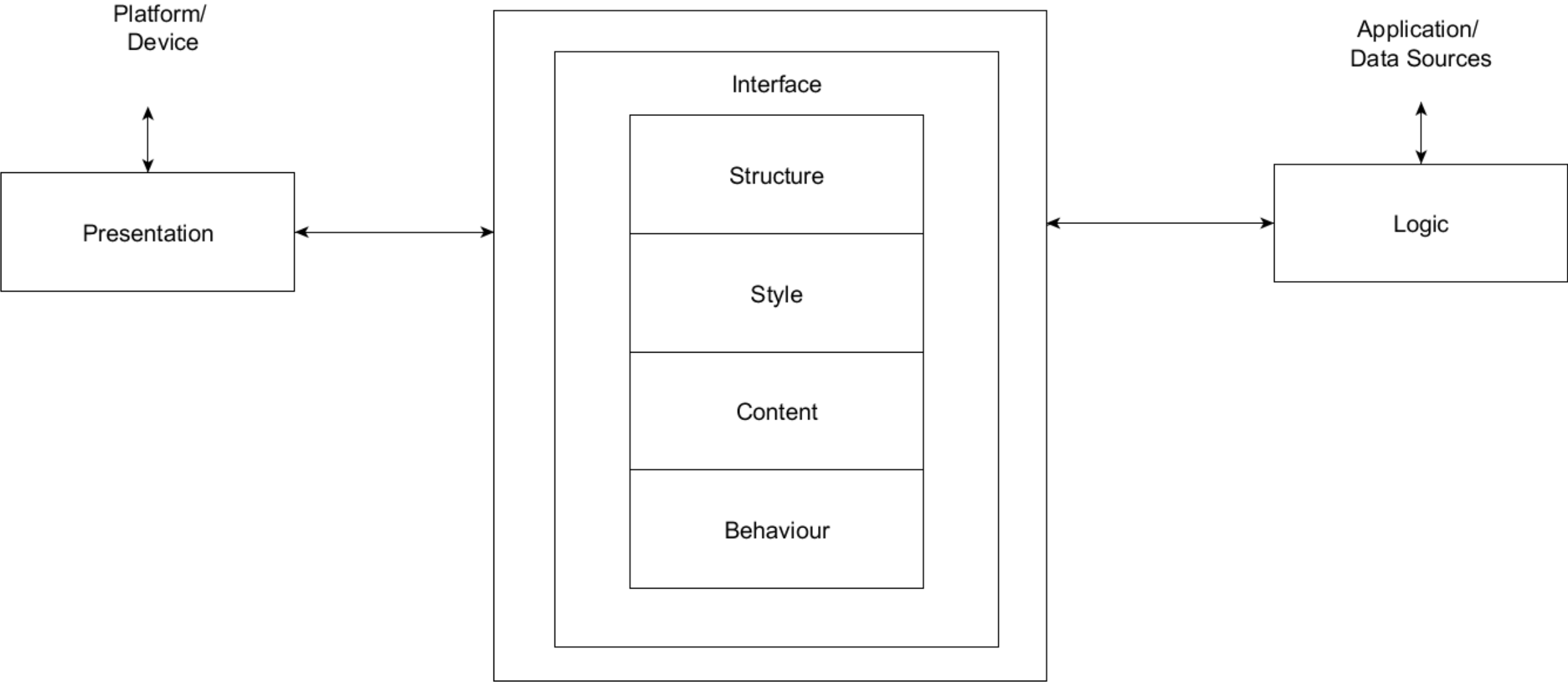
1. View Model

- Separation of concerns
- View model adaption

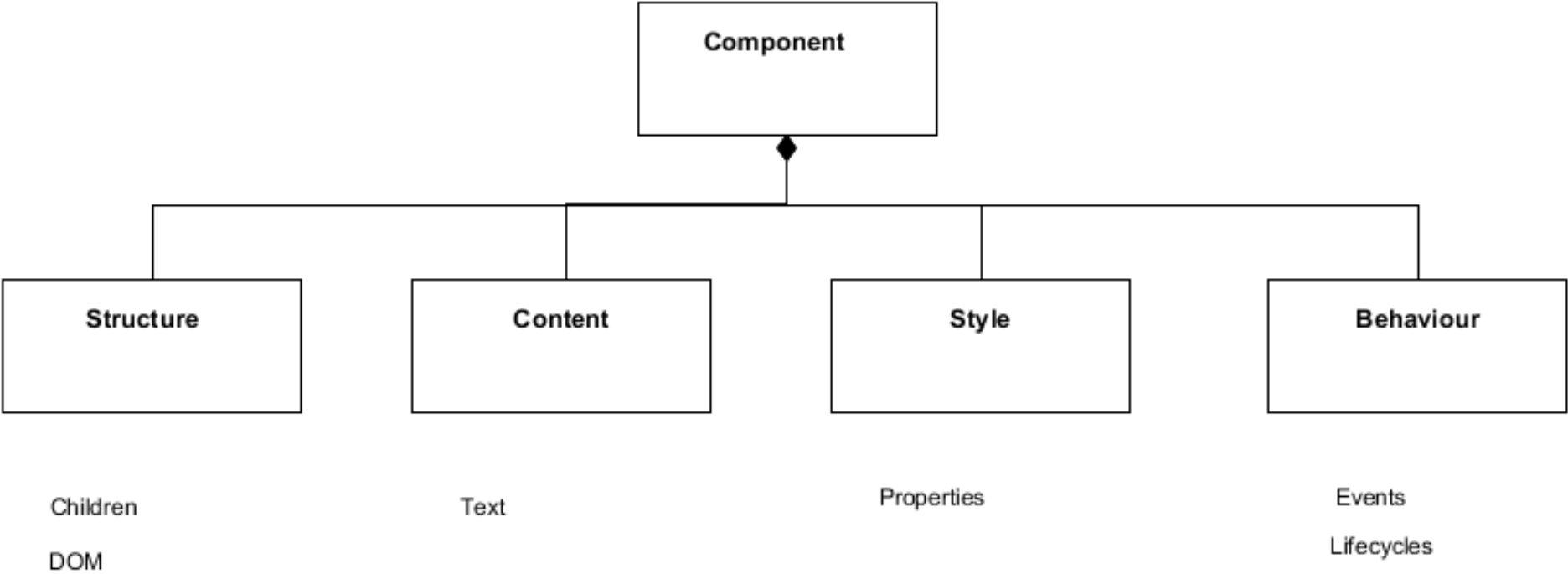
2. Proposed process

- Parser
- Extraction process
- View model generation
- Component generation

Separation of Concerns



Source: <https://www.oasis-open.org/committees/download.php/28457/uiml-4.0-cd01.pdf>



Adapted from UIML 4.0 specification, specification, <http://docs.oasis-open.org/uiml/v4.0/cd01/uiml-4.0-cd01.pdf>, 2008

View Model in JSON Schema

```
{
  "componentName": "",
  "type": "object",
  "properties": {
    "content": {
      "type": "object",
      "properties": {
        "text": {
          "type": "string"
        }
      }
    },
    "structure": {
      "type": "object",
      "required": [
        "className"
      ],
      "properties": {
        "className": {
          "type": "string"
        }
      }
    },
    "style": {
      "type": "object",
      "properties": {
        "property": {
          "type": "object"
        }
      }
    },
    "children": {
      "type": "array"
    }
  }
}
```

Labels, text

HTML class name

Colors, fonts, fills

Component's children

Proposed Process

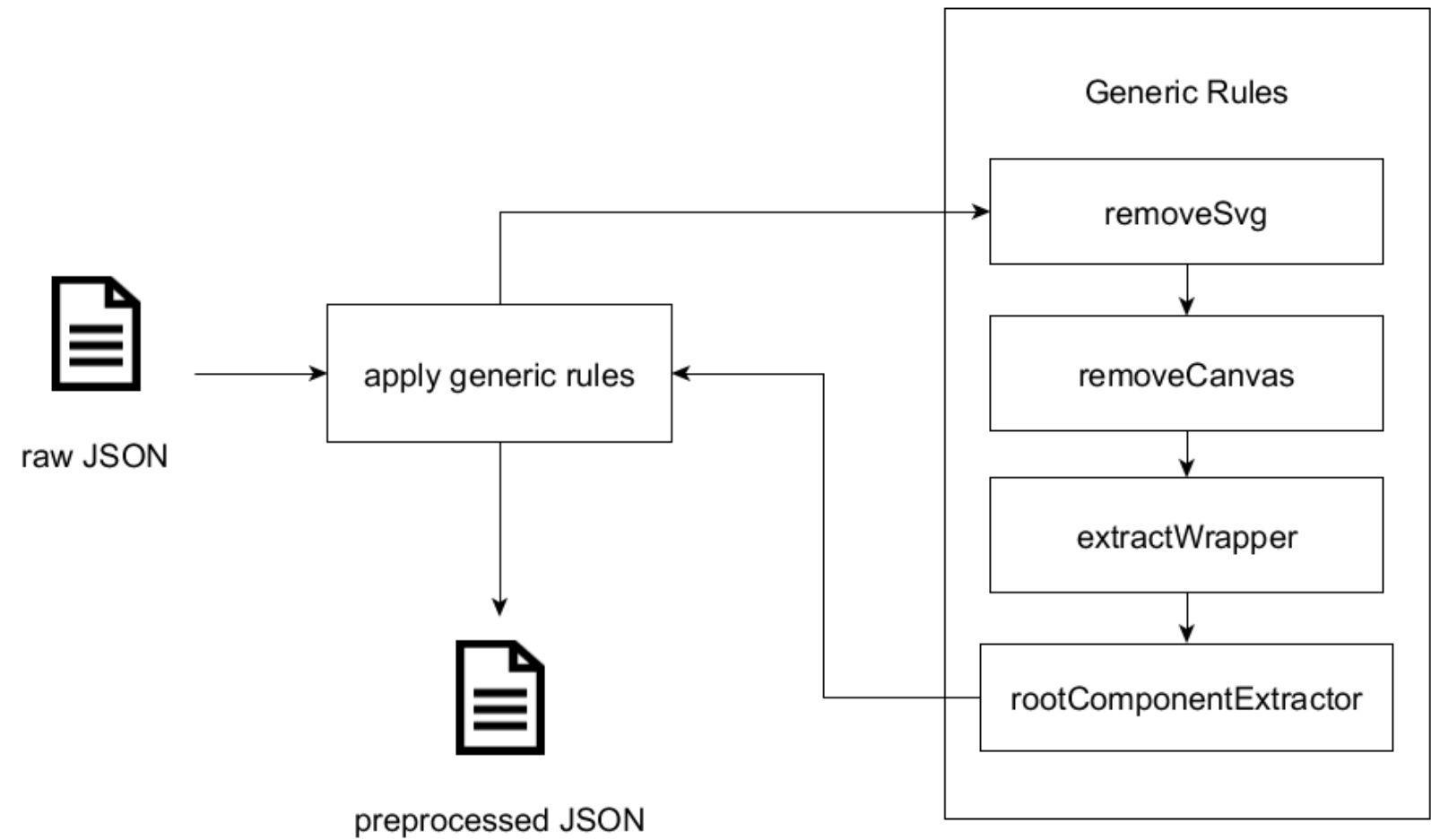
1. Parser
2. Extraction process
 - First phase
 - Second phase
3. DOM abstraction
4. View model generation
5. Component generation

- Pure functions
- Responsible for:
 - Identifying the component
 - Validating the component
 - Extracting the component`s information
- Reusable, extensible
- Rule set is sequentially applied to the input object

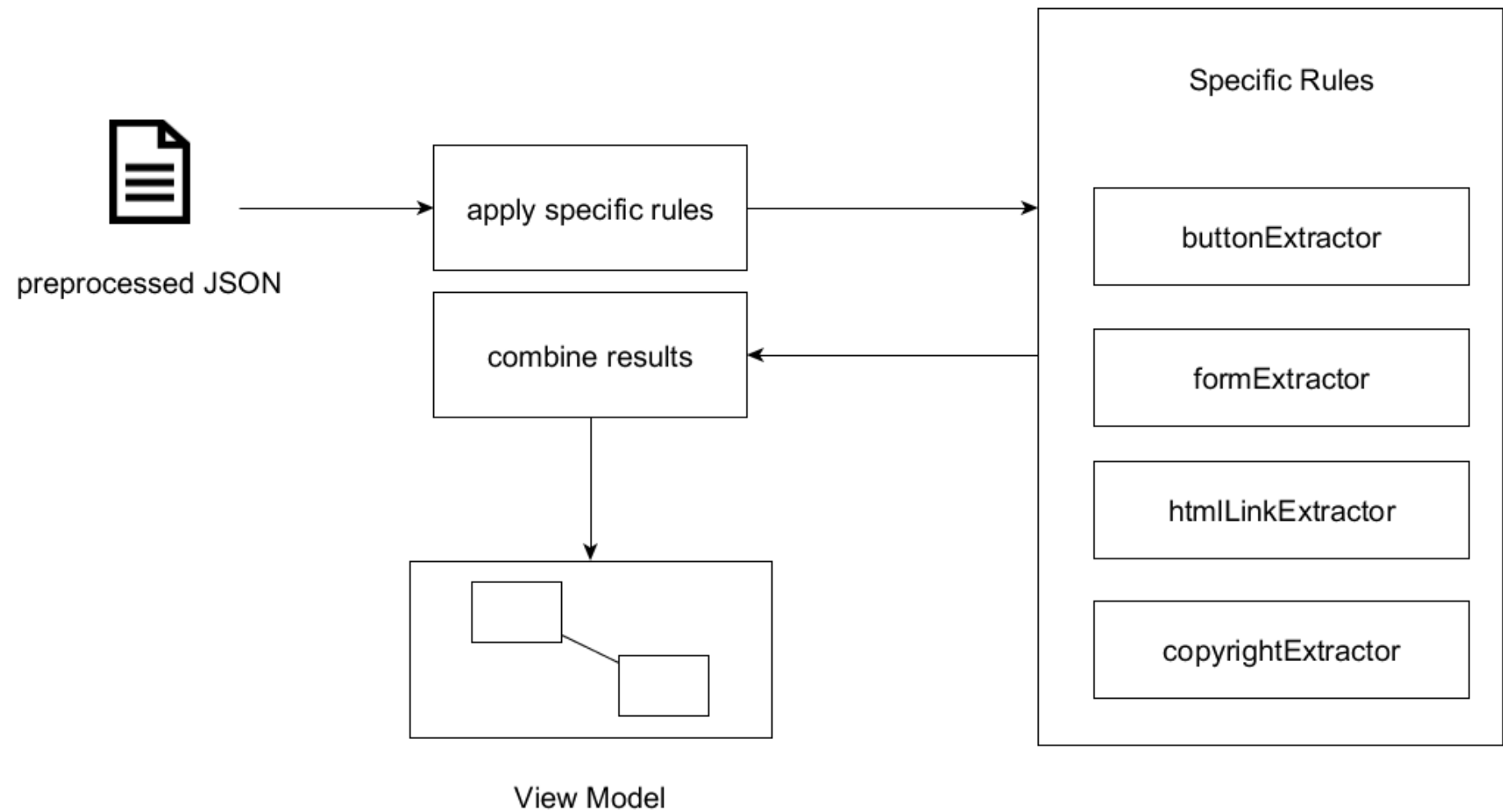


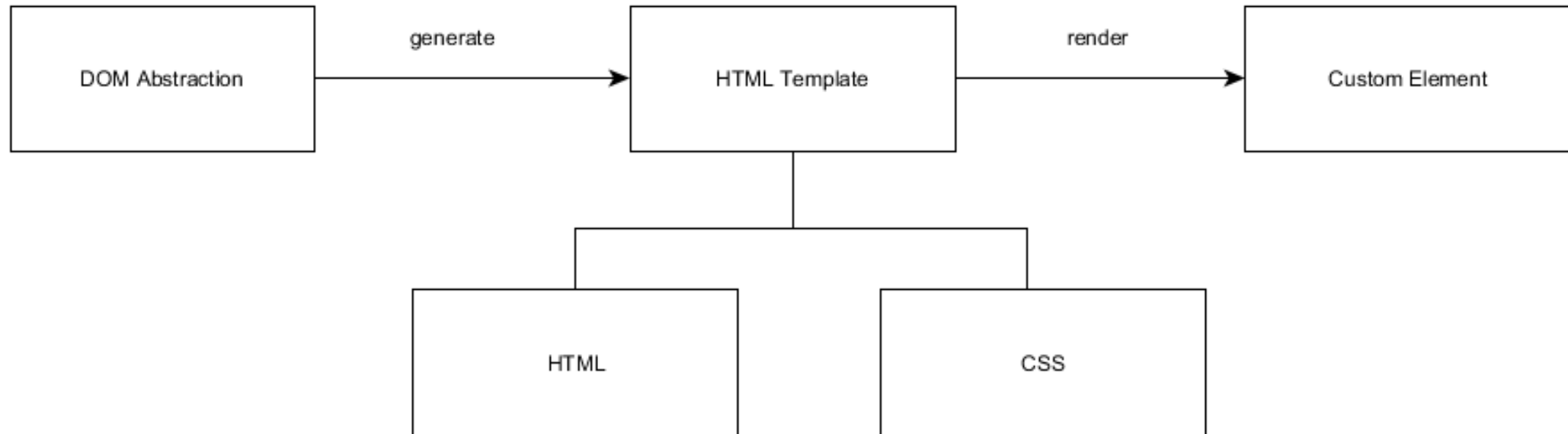
xml2Json framework taken from: <https://github.com/buglabs/node-xml2json>

First Phase Extraction Process



Second Phase Extraction Process: View Model Generation

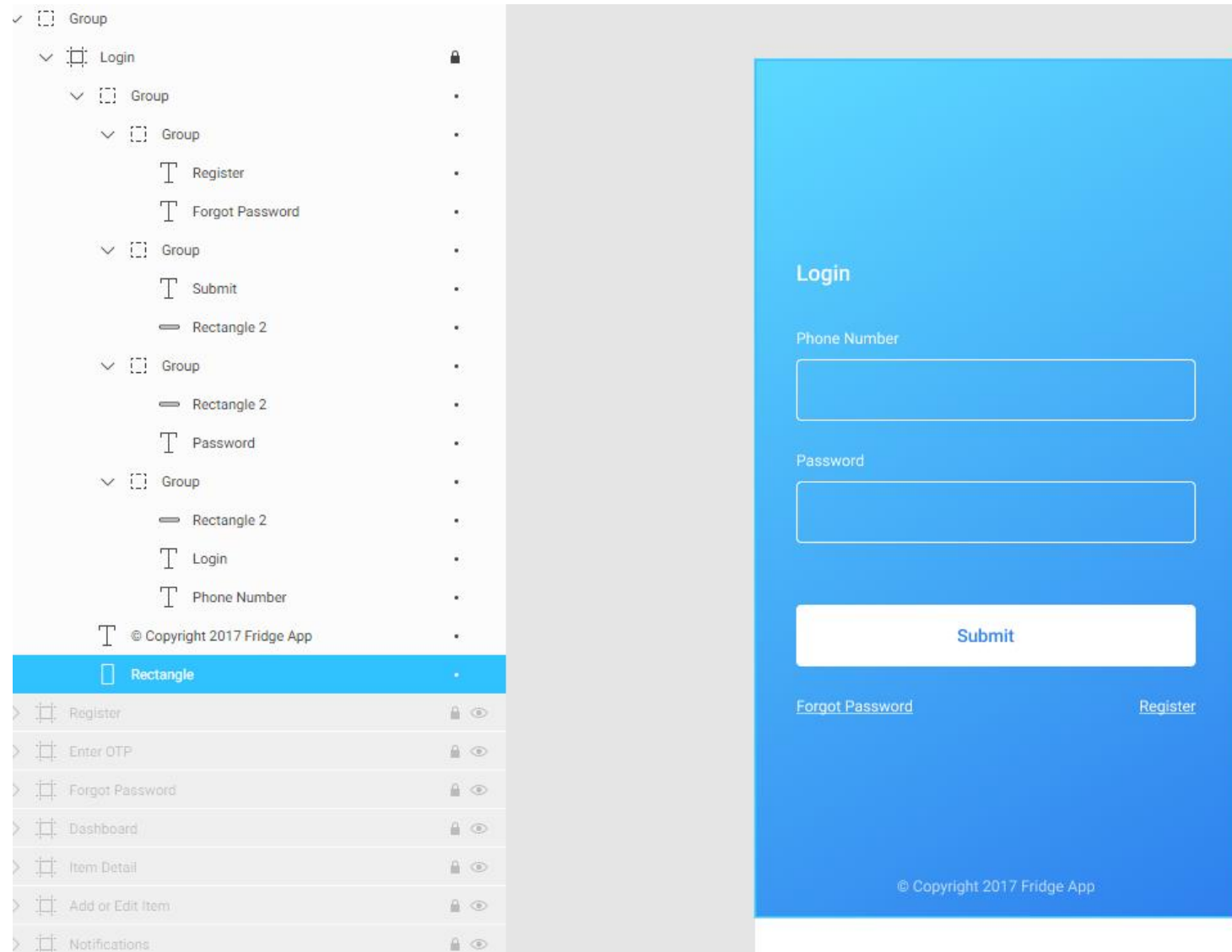




Development Approach:

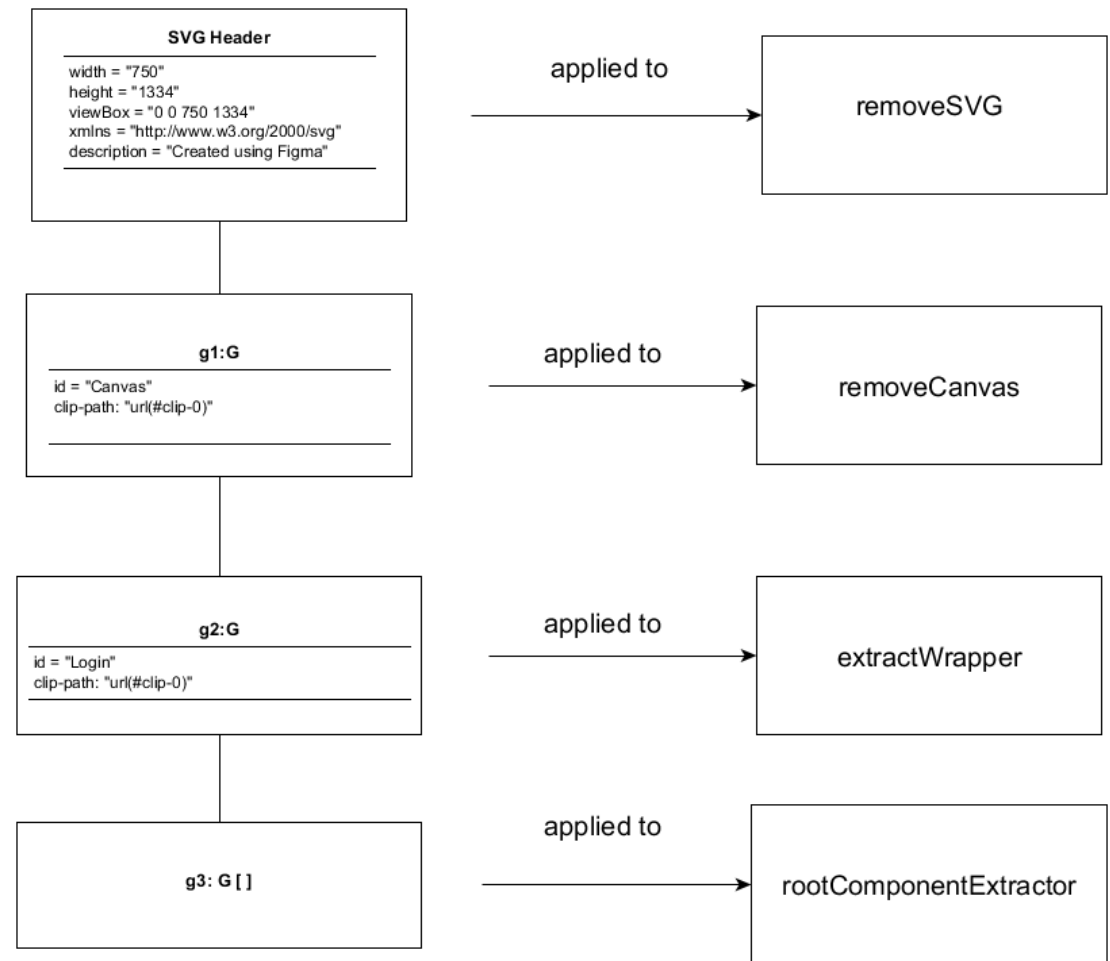
- Test-Driven-Development
- Jest Test Framework was used
- Unit tests for each rule and their composition
- Achieved test coverage for rules ~ 86 %
- Polymer starter kit used for visualizing generated components

Example Walk-Through

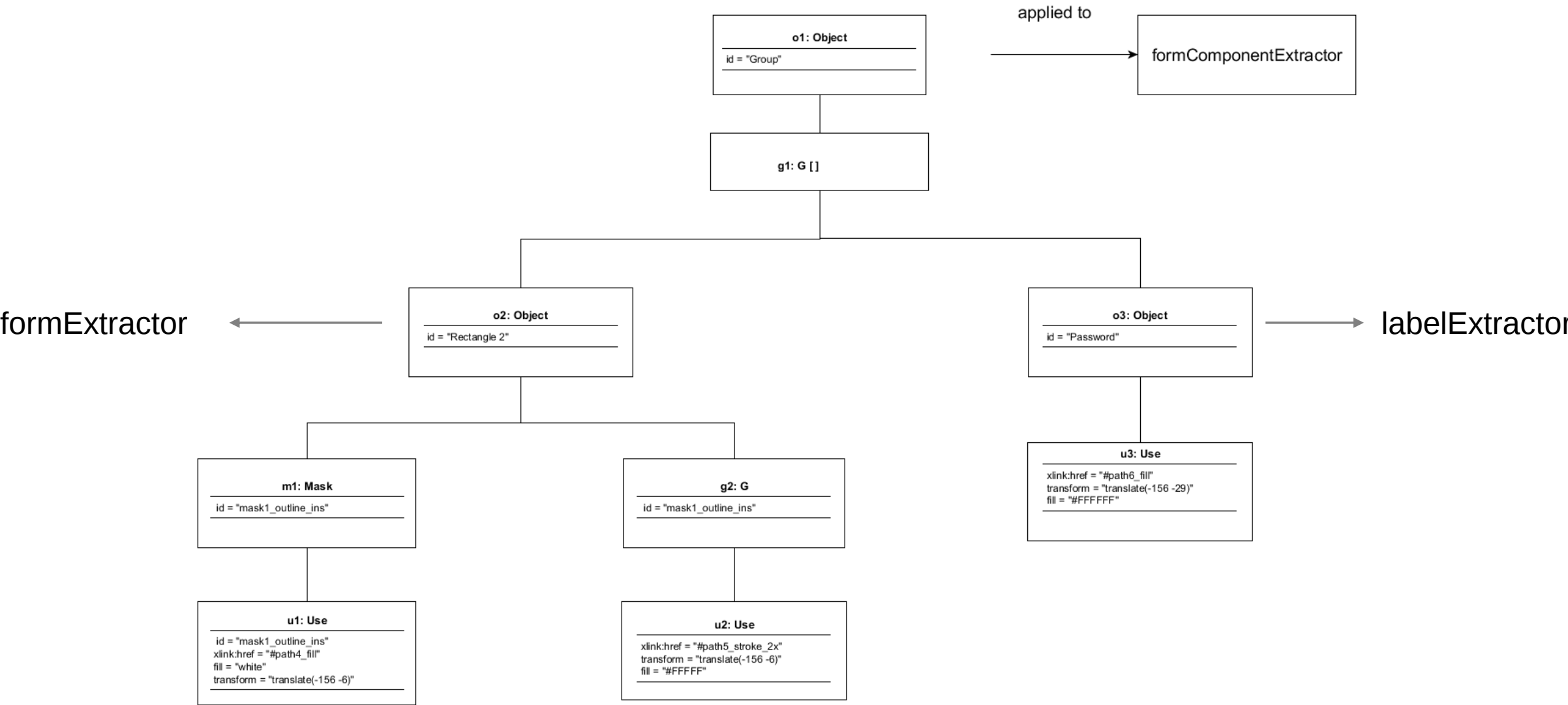


Taken from: <https://www.uplabs.com/posts/fridge>

Parser & First Phase Extraction Process



Second Phase Extraction Process



View Model Form Component

```
{
  formComponentWrapper: {
    id: "passwordFormWrapper",
    structure: {
      className: "div"
    },
    children: [
      {
        labelComponent: {
          id: "passwordLabel",
          style: {
            properties: {
              transform: "translate(-156 -29)",
              color: "#FFFFFF"
            }
          },
          structure: {
            className: "label"
          },
          content: {
            "text": "Password"
          }
        },
        formComponent: {
          id: "passwordForm",
          style: {
            properties: {
              fill: "white",
              transform: "translate(-156 -6)"
            }
          },
          structure: {
            className: "form"
          }
        }
      ]
    ]
  }
}
```



Wrapping div



Label part



Form part

```
"id": "rootComponent",
"structure": {
  "className": "div"
},
"children": [
  {
    "CopyrightComponent": {
      "id": "CopyrightComponent",
      "structure": {
        "className": "p"
      },
      "content": {
        "text": "&copy; Copyright"
      },
      "style": {
        "properties": {
          "fill": "#FFFFFF",
          "fill-opacity": "0.7"
        }
      }
    }
  }
],
....
```

mapped to

```
DOMElement {
  tagName: 'ROOTCOMPONENT',
  nodeName: 'ROOTCOMPONENT',
  className: 'div',
  dataset: {},
  childNodes:
    [
      VirtualNode {
        tagName: 'CopyrightComponent',
        properties: [
          { className: 'p' },
          { color: '#FFFFFF',
            fill-opacity: '0.7'
          }
        ],
        children: [
          VirtualText
            { text: '&copy; Copyright' } ],
        key: undefined,
        namespace: null,
        count: 1,
        hasWidgets: false,
        hasThunks: false,
        hooks: undefined,
        descendantHooks: false,
        parentNode: [Circular]
      }
    ]
  ...
}
```

```
<link rel="import"
href="../../bower_components/polymer/polymer-element.html">
<dom-module id="my-passwordformwrapper">
  <template>
    <style>
      label{
        transform: translate(-156, -29);
        color: #FFFFFFF;
      }
      form{
        fill: white;
        transform: translate(-156, -6);
      }
    </style>
    <div id="passwordFormWrapper">
      <label id="passwordLabel">Password</label>
      <form id="passwordForm"> <input> </form>
    </div>
  </template>
  <script>
    class MyPasswordformwrapper extends Polymer.Element {
      static get is() {
        return 'my-passwordformwrapper';
      }
    }
    window.customElements.define(
      MyPasswordformwrapper.is,
      MyPasswordformwrapper
    );
  </script>
</dom-module>
```

→ HTML Import

→ HTML Template definition

→ Scoped styles

→ Render the template

→ Retrieve dom-module and clone its content to the element's shadow dom

→ Define the class as custom element

Live Demo

Pros

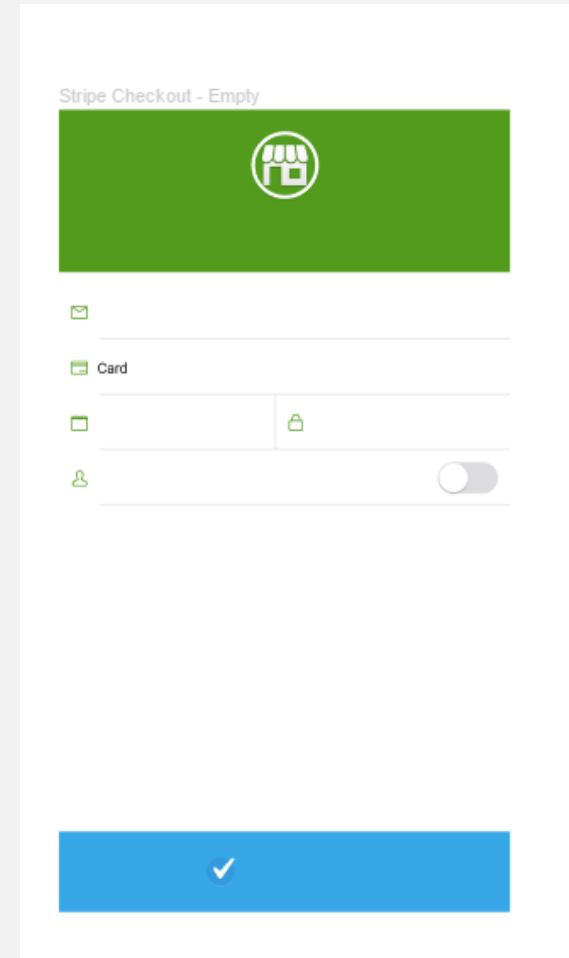
- ~ 80 % component identification
- Decent extraction quality
- Extracted components conform to the W3C Web Components Standard
- Extensible and reusable components

Cons

- Partially problematic structure hindering generic applicability
- Assumptions concerning structure needed to be made because of the lack of documentation about prototyping tools
- Tool was built around use case
- Already limitations extraction components

Problems identified

- Completely different structure
- Pre-processing failed
- Identification failed
-> therefore extraction also failed
- Identifiers are used for component identification, not structure



- Structural assumptions were not valid in general
 - Tool needs to be modified for each use case
 - Identification fails -> Mapping and rest of the process will fail too
- First phase of the extraction process is prone to unexpected SVG structure -> which leads to complete failure
- Comparing both use cases, no useable structural patterns could be identified
- Some components can only be identified by best practices
 - E.g. commonly used button descriptions (submit, OK, Cancel)
- Style mapping problems
 - Component identification necessary in order to correctly map SVG's fill to respective component style properties

- Make first phase of the extraction process more flexible
- Develop more rules and enhance existing ones
- Design Guidelines
- Components Annotations



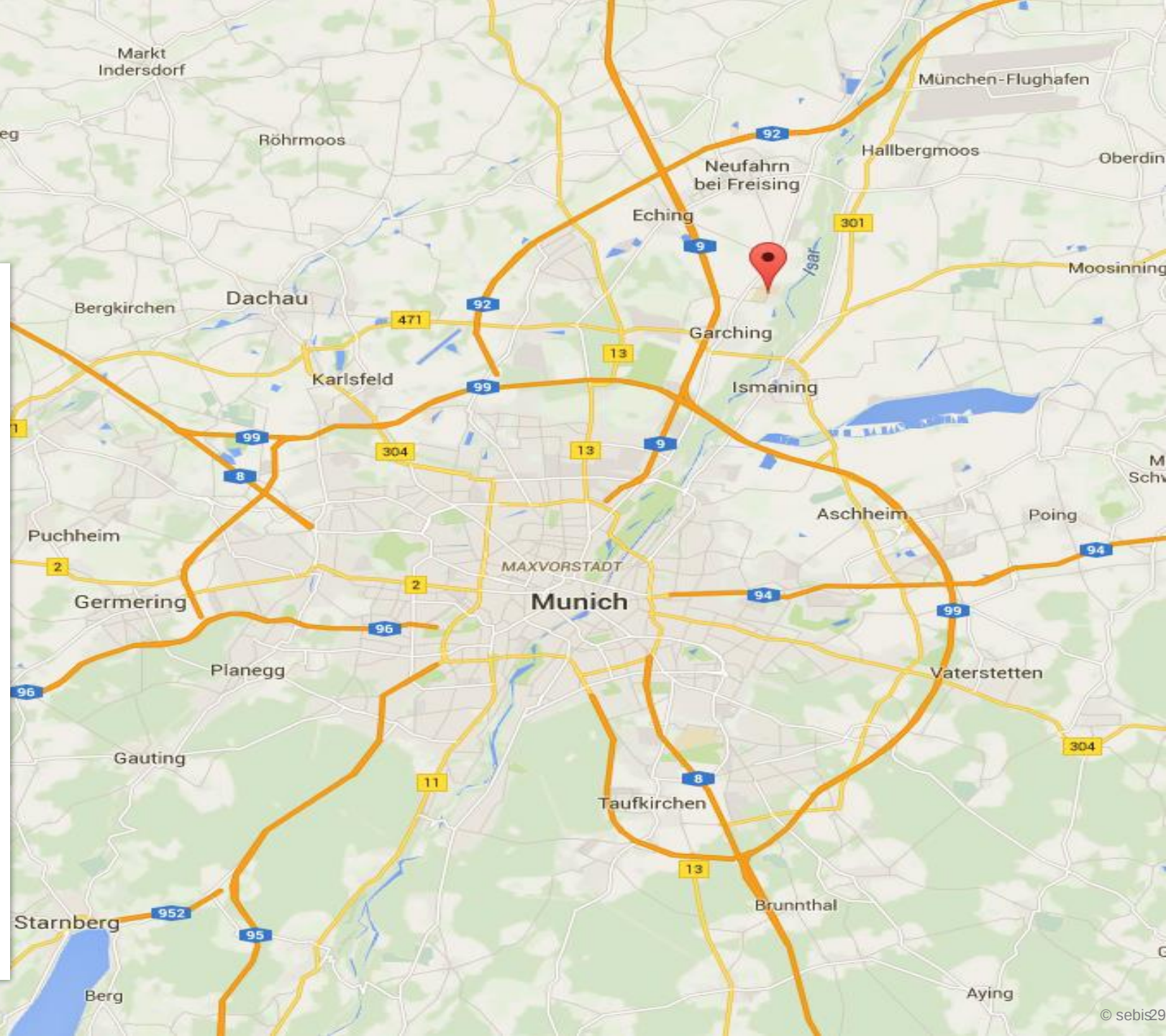
B. Sc. Information Systems
Marvin Aulenbacher

Technische Universität München
Faculty of Informatics
Chair of Software Engineering for Business
Information Systems

Boltzmannstraße 3
85748 Garching bei München

Tel +49.89.289. marvin.aulenbacher@tum
Fax +49.89.289.17136.de

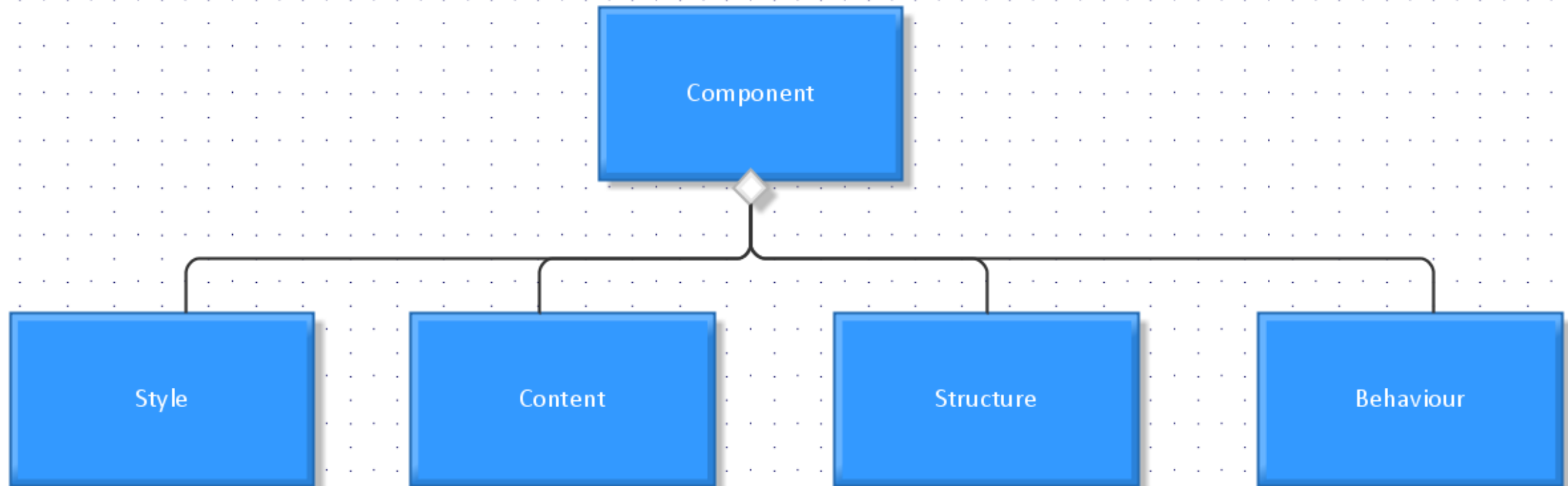
www.matthes.in.tum.de



Backup

Marvin Aulenbacher, 13.11.2017, Munich

Chair of Software Engineering for Business Information Systems (sebis)
Faculty of Informatics
Technische Universität München
www.matthes.in.tum.de



- Backward compability via polyfills
- Custom Elements supported by modern browsers
- Early version support of Shadom Dom and Custom Elements in Chrome, Opera, Firefox
- Edge has neither started to implement Shadow Dom nor Custom Elements

Hollstic HTML Tempate DOM Node

```
<link rel="import"
  href="../../bower_components/polymer/polymer-element.html">
<dom-module id="my-component">
  <template>
    <style>
      className{
        style.properties
      }
    </style>
    <className id="component">content.text</className>
  </template>
  <script>

  class MyComponent extends Polymer.Element {
    static get is() {
      return "my-component";
    }
  }
  window.customElements.define(MyComponent.is,
  MyComponent)
  </script>
</dom-module>
```

Falsy Identified Form Component

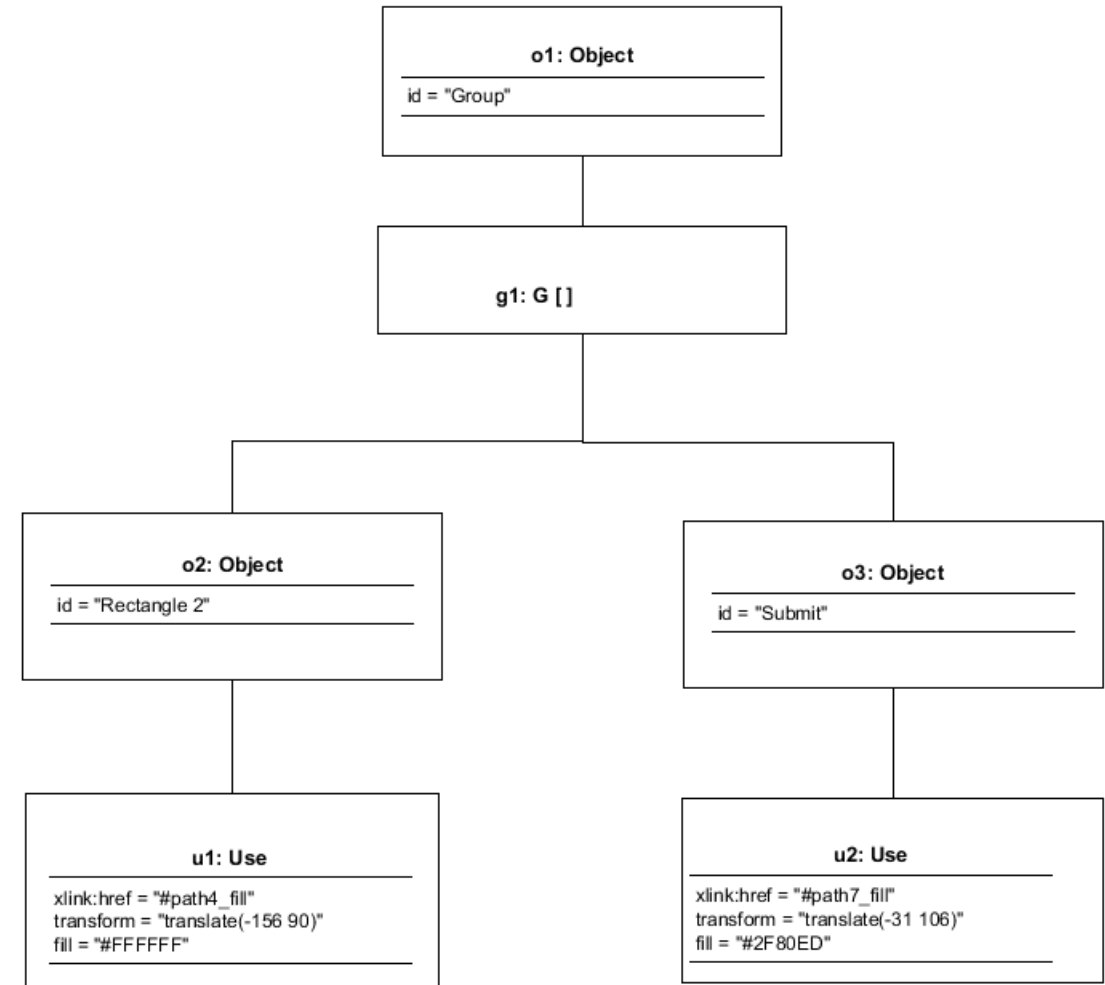
```
formComponentWrapper: {
  id: "loginFormWrapper",
  structure: {
    className: "div"
  },
  children: [
    {
      labelComponent: {
        id: "loginLabel",
        style: {
          properties: {
            transform: "translate(-156 -178)",
            color: "#FFFFFF"
          }
        },
        structure: {
          className: "label"
        },
        content: {
          text: "Login"
        }
      },
    },
    {
      formComponent: {
        id: "loginForm",
        style: {
          properties: {
            transform: "translate(-156 -101)",
            fill: "#FFFFFF"
          }
        },
        structure: {
          className: "form"
        }
      },
    }
  ]
}
```

Header „Login“ was identified as label part

-> leads to invalid mapping

Button Component

```
<link rel="import"
      href="../../bower_components/polymer/polymer-element.html">
<dom-module id="my-button1">
  <template>
    <style>
      button{
        transform: translate(-156, 90);
        color: #2F80ED;
        background-color: #FFFFFF;
      }
    </style>
    <button id="button1">Submit</button>
  </template>
  <script>
    class MyButton1 extends Polymer.Element {
      static get is() {
        return "my-button1";
      }
    }
    window.customElements.define(MyButton1.is,
    MyButton1)
  </script>
</dom-module>
```

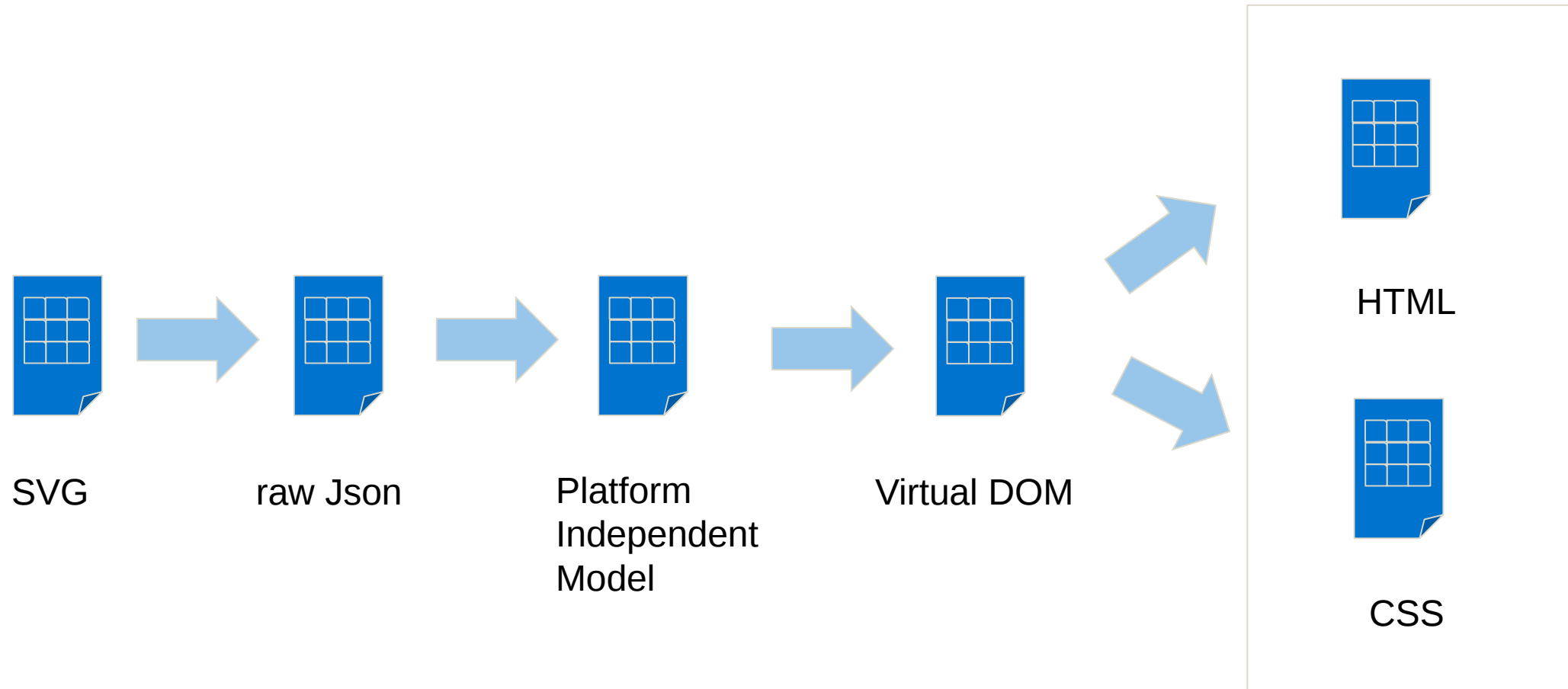


	Correct	Incorrect	Missing	Total	Rate
Identified components	6	0	0	6	100 %
Extracted components	5	1	0	6	83 %
Extracted properties	19	0	5	26	73 %

Evaluation Login View

	Correct	Incorrect	Missing	Total	Rate
Identified components	4	0	1	5	80 %
Extracted components	4	0	1	5	80 %
Extracted properties	10	0	5	15	67 %

Evaluation Add Item View



All files

86.58% Statements 387/447 **76.36%** Branches 197/258 **81.43%** Functions 57/70 **86.43%** Lines 382/442

File ▲		Statements ▾	
src		43.18%	19/44
src/keys		100%	2/2
src/rules		85.82%	115/134
src/rules/componentRules		93.68%	237/253
test		100%	14/14