openCPQ
–
A React-Based Product-Configuration Toolkit

Tim Geisler, Heribert Schütz

webXcerpt Software GmbH

tg@webxcerpt.com, hs@webxcerpt.com

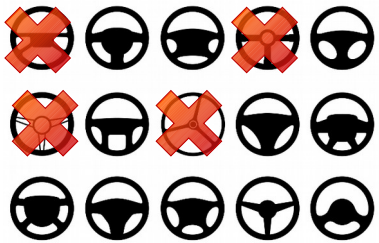
MunichJS Meetup, 2015-05-13

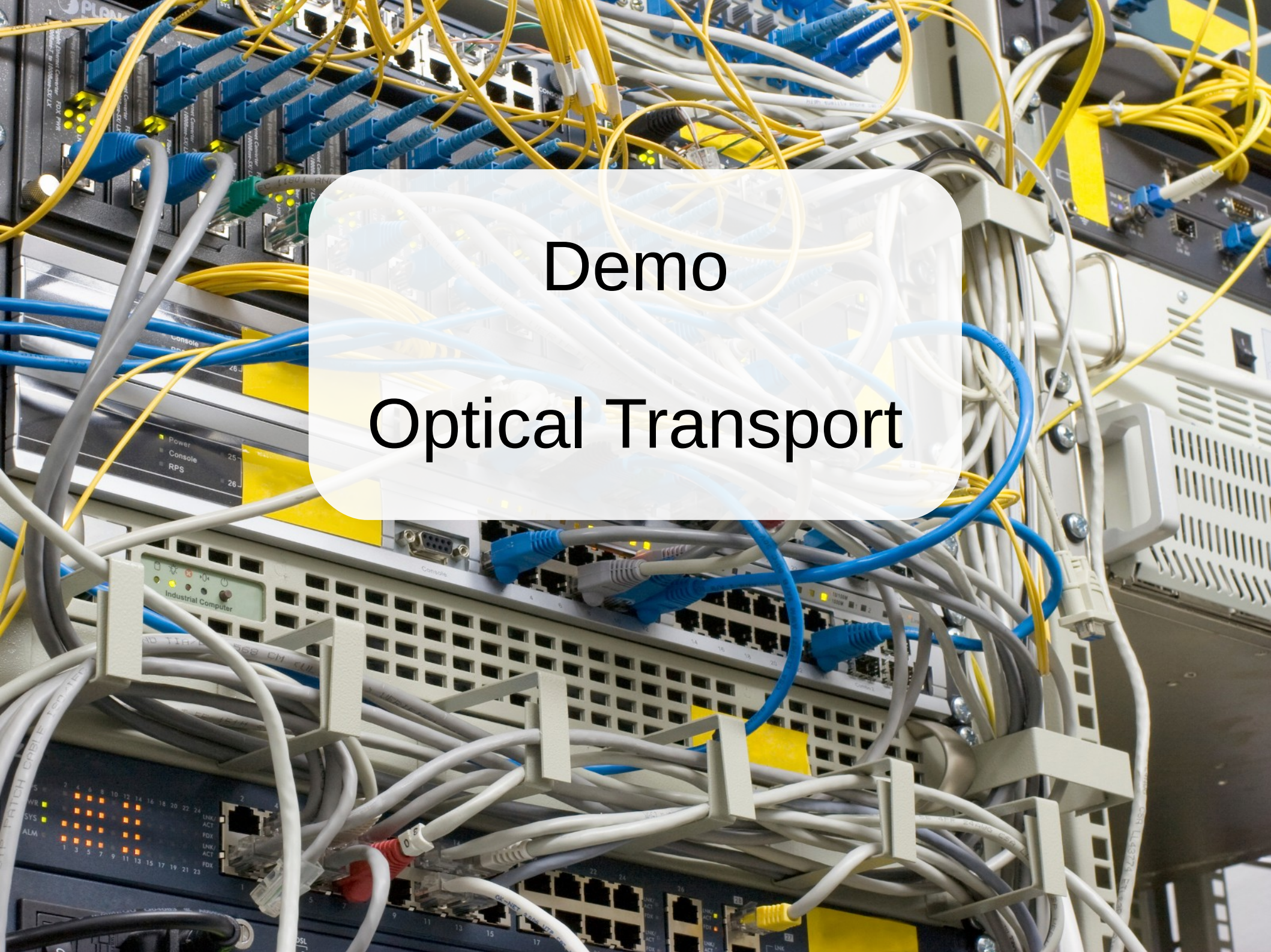
webXcerpt
manage your information

Product Configuration

Variants

Parameters and Domains, Dependencies



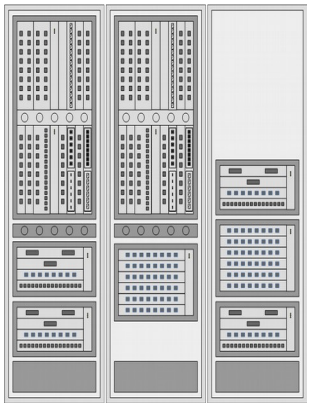


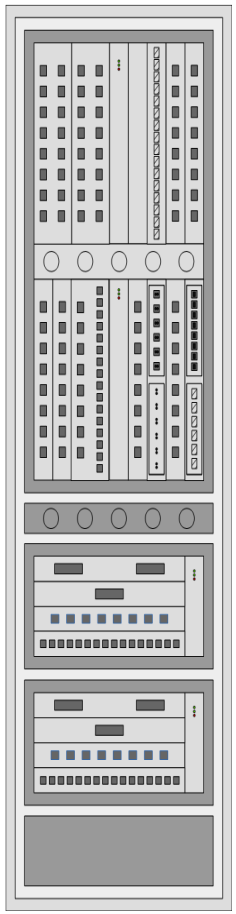
Demo

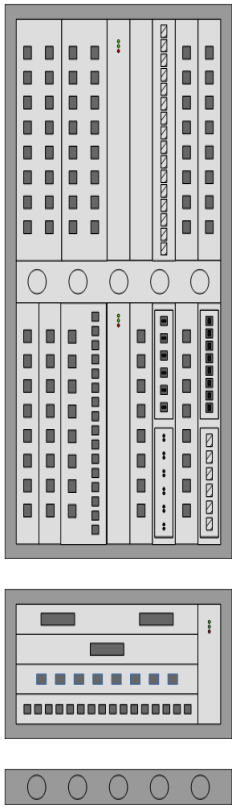
Optical Transport

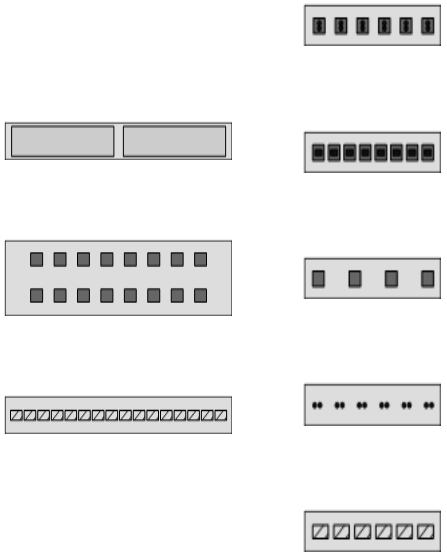
Demo Example: Hierarchical Configuration

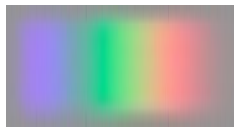
Solution Rack Switch Board (Module) Transceiver (Wavelength)











Demo

http://opencpq.webxcerpt.com/examples/optical-transport/

opencpq.webxcerpt.com/examples/optical-transport/

undo | redo | reset | save | restore | Datei auswählen | Keine ausgewählt | import | export

Configuration

Solution

Project Settings

Release

Rel. 1.0

Rack Type

ANSI

Uninterruptible Power Supply (default for each rack)

Racks

#	Rack
1	Uninterruptible Power Supply Switches

#	Product
2	Optical Switch OS6 Slot 1 16 x 10 G board

Contents

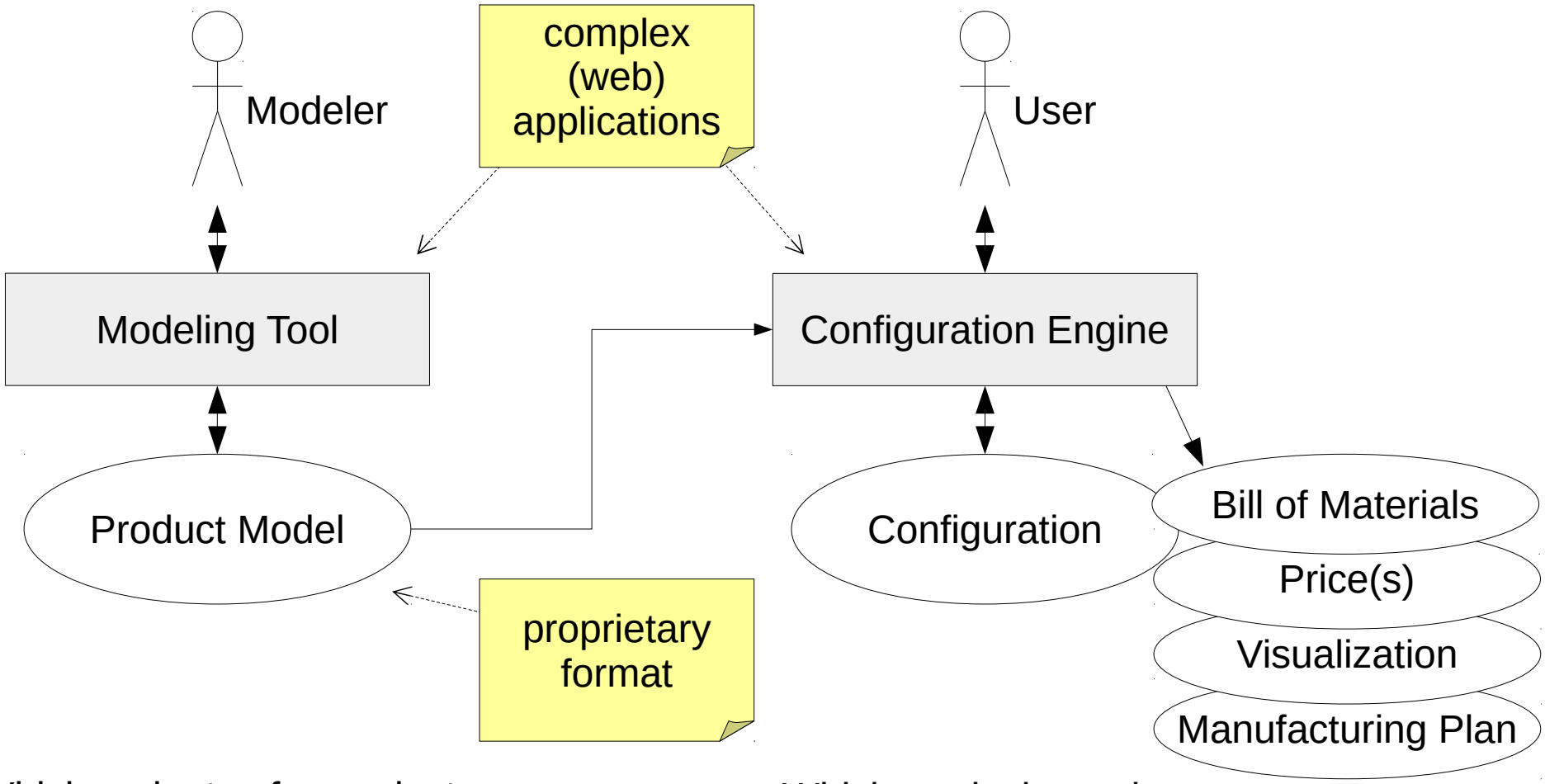
- Project Settings
- #1: Rack
  - #1: 2 × OS6
  - #2: OS6
- #2: Rack
  - #1: OS4
- Network Management
- Services

Bill of Materials

export as CSV

#	Material No.	Description	Price (€)
3	SFP+:SR	SFP+ SR (850 nm, up to 300 m)	1.000,00
3	B:16x10	16 x 10 G board	25.000,00
16	B:FP	Blank faceplate for board slots	20,00

Business Processes



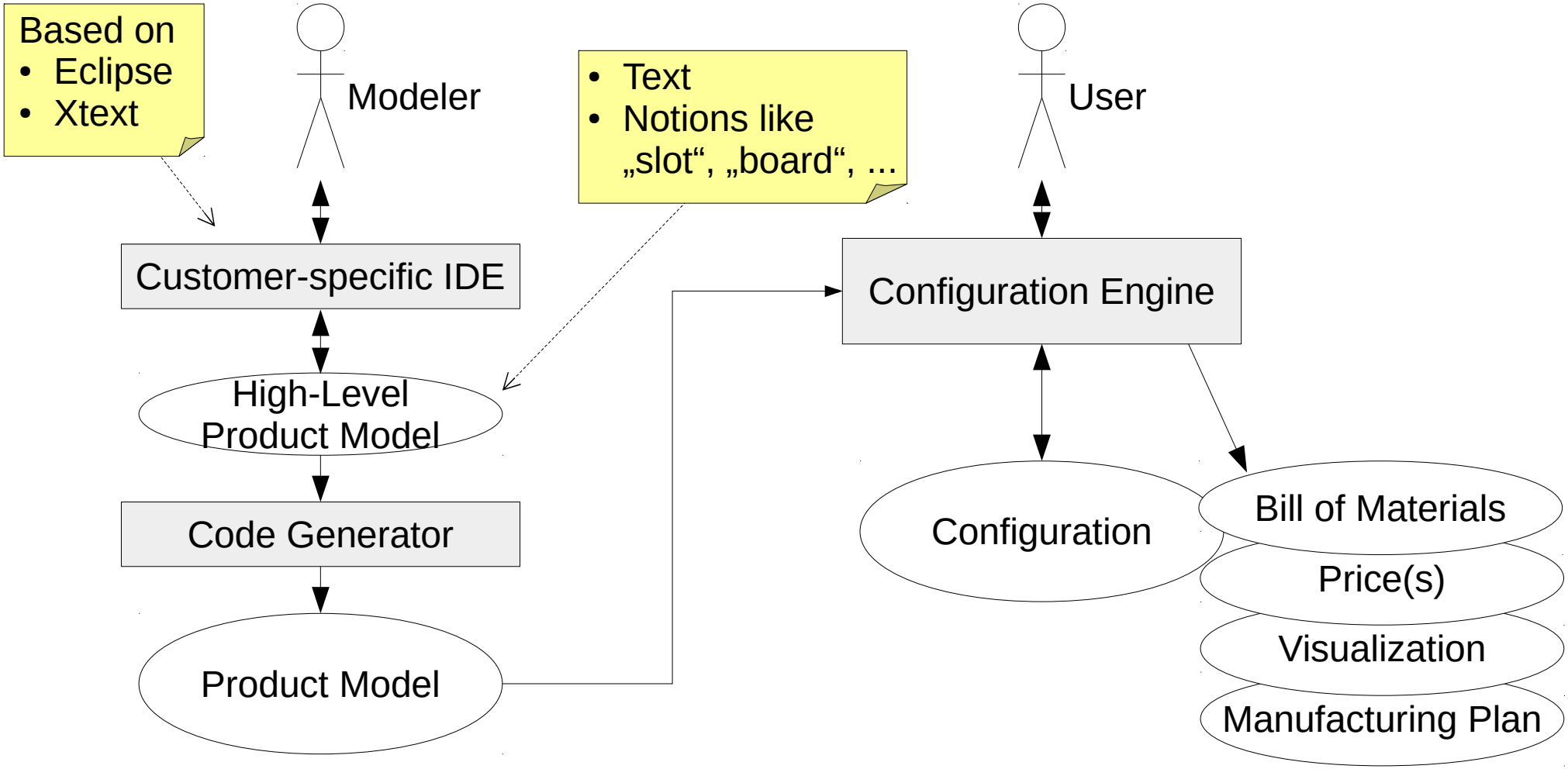
Which variants of a product are available?

Which particular variant do I want to buy/sell?

...

Problem 1

Customer-Specific Modeling Language



Which variants of a product are available?

Which particular variant do I want to buy/sell?

Product Models

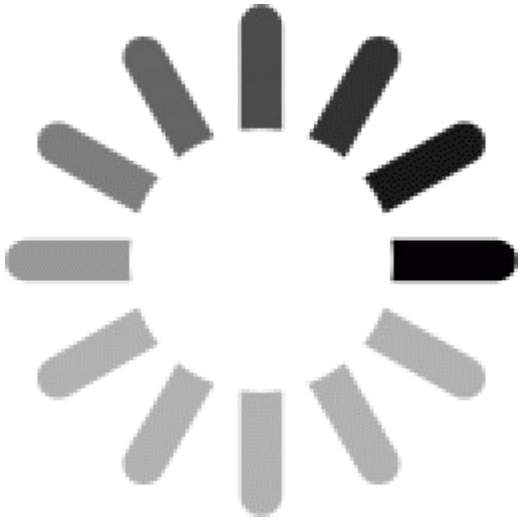
- Product parameters
  - Data types
  - Ranges
- Components
- Dependencies between parameters/components
- Calculation of additional output

Models are programs!

Modeling as Programming

- Abstractions, data structures
- Programming tools
  - Editors/IDEs
  - Debuggers and profilers
  - Revision control
  - Test and CI frameworks
- General purpose tools and languages
  - Maturity
  - Re-usable knowledge, may already be available
  - Large communities and „ecosystems“

Problem 2

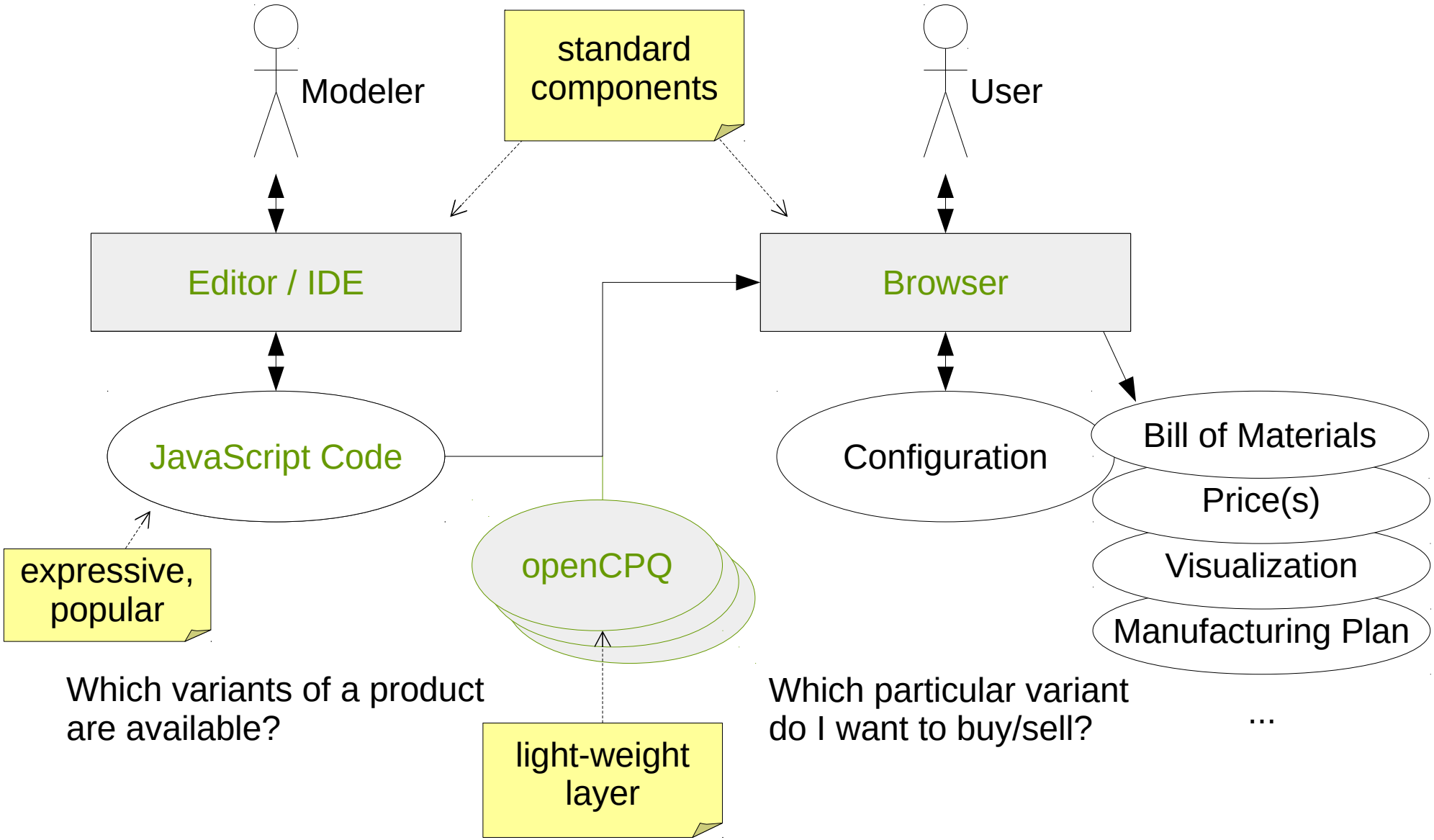


Configuring in the Browser

Implement configurators in JavaScript.

JavaScript is also
a reasonable choice for modeling.

Processes with openCPQ



open – a Configurator Toolkit in JS

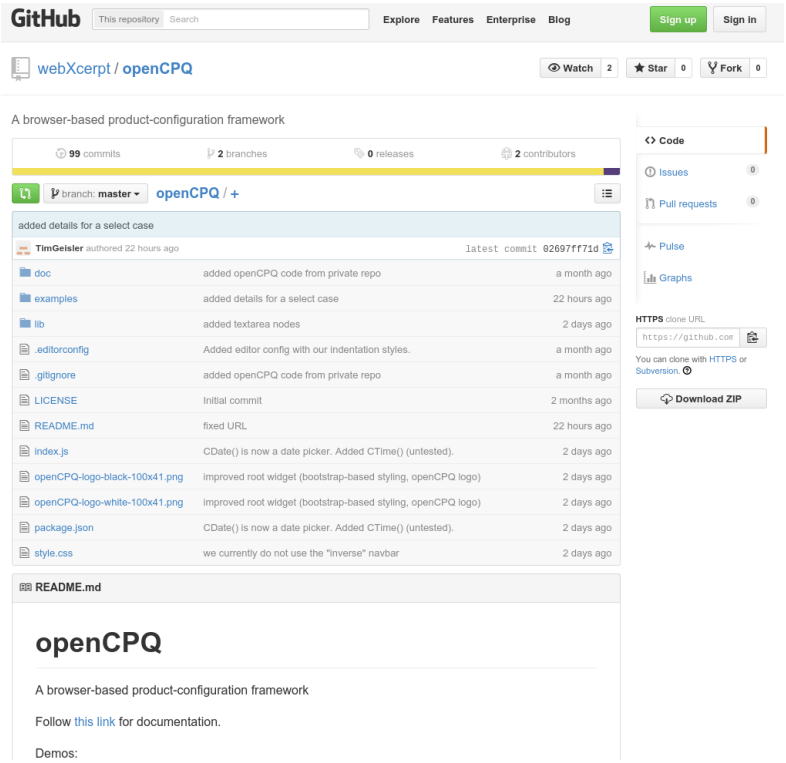
- Building-block library
  - Components
  - Dependencies
- Combine building blocks with JavaScript
- Add application-specific building blocks
- A light-weight layer based on React and Bootstrap

– an Open-Source Project

Source code and links to live demos available on Github:
https://github.com/webXcerpt/openCPQ

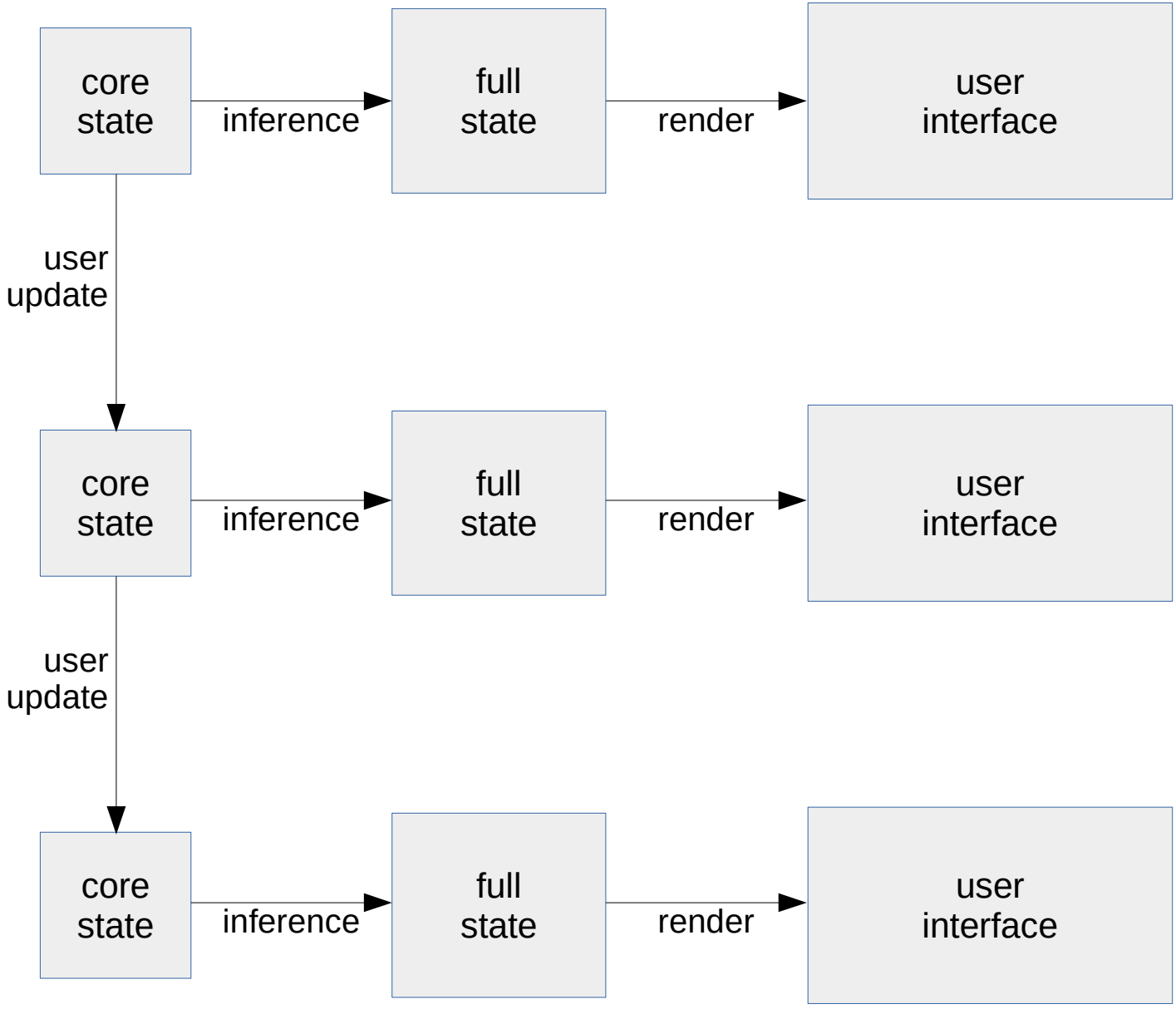
Liberal MIT license

Use, adapt,
integrate, contribute!

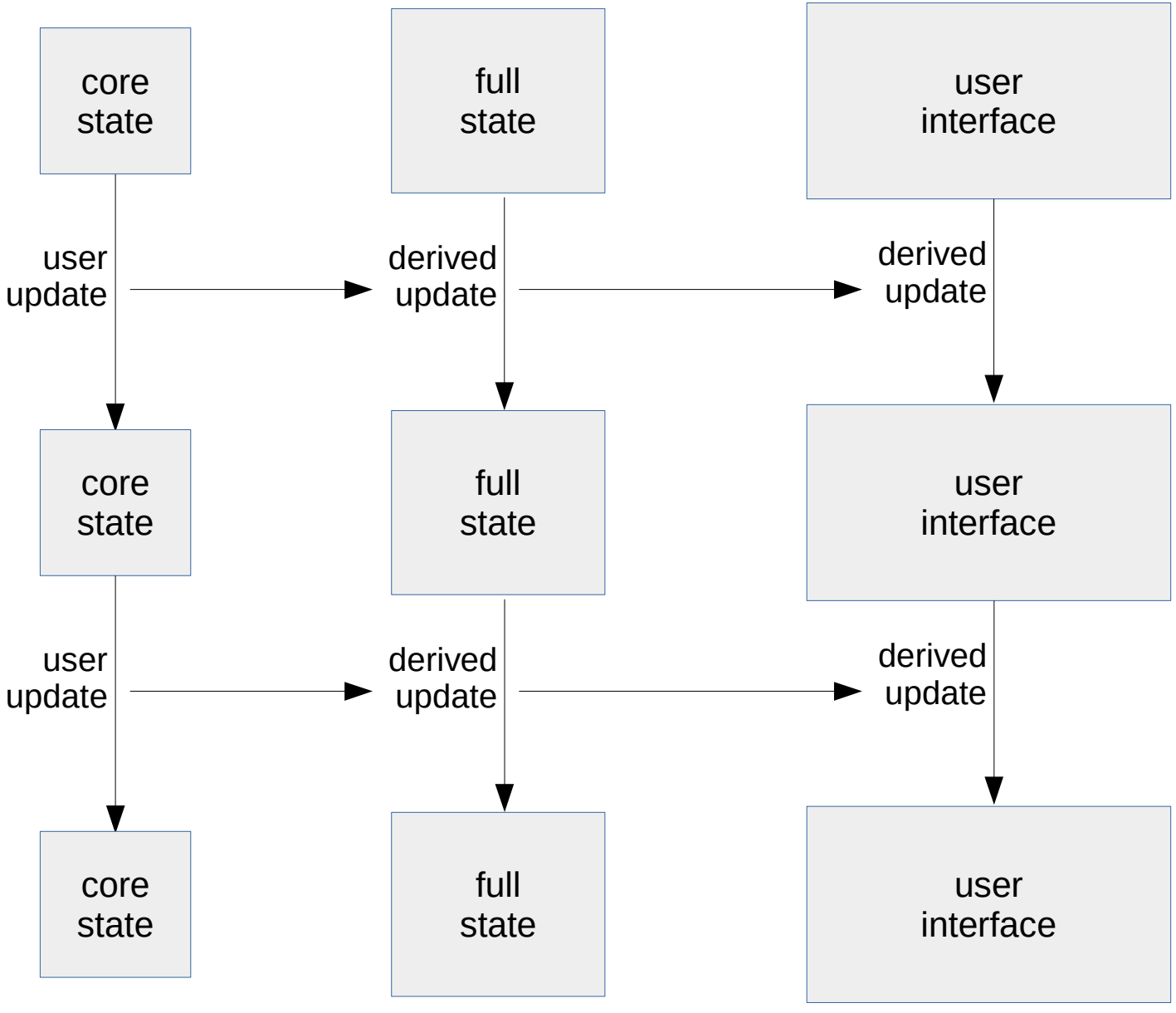


How it Works

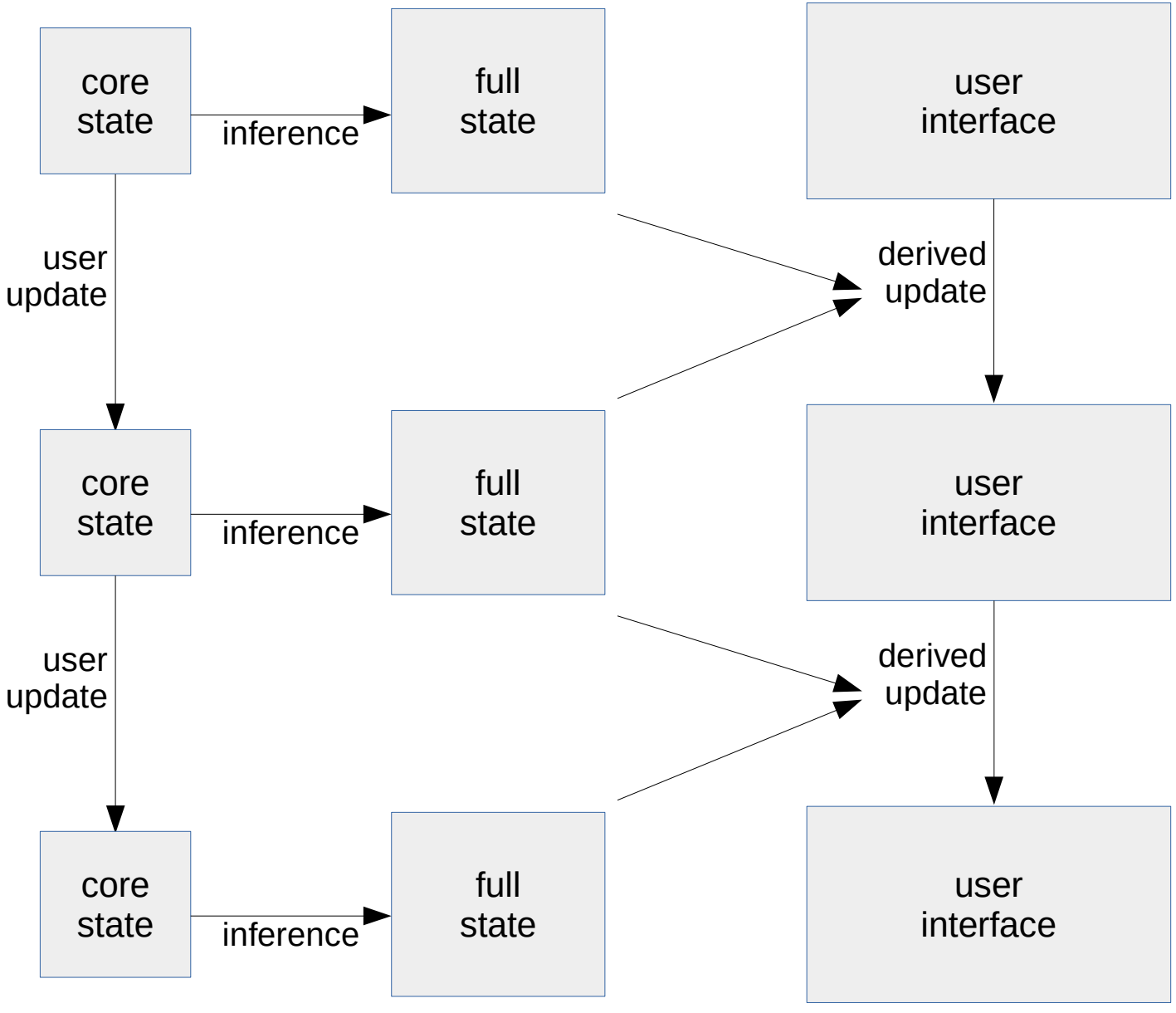
Change Propagation: Non-Incremental



Change Propagation: Incremental



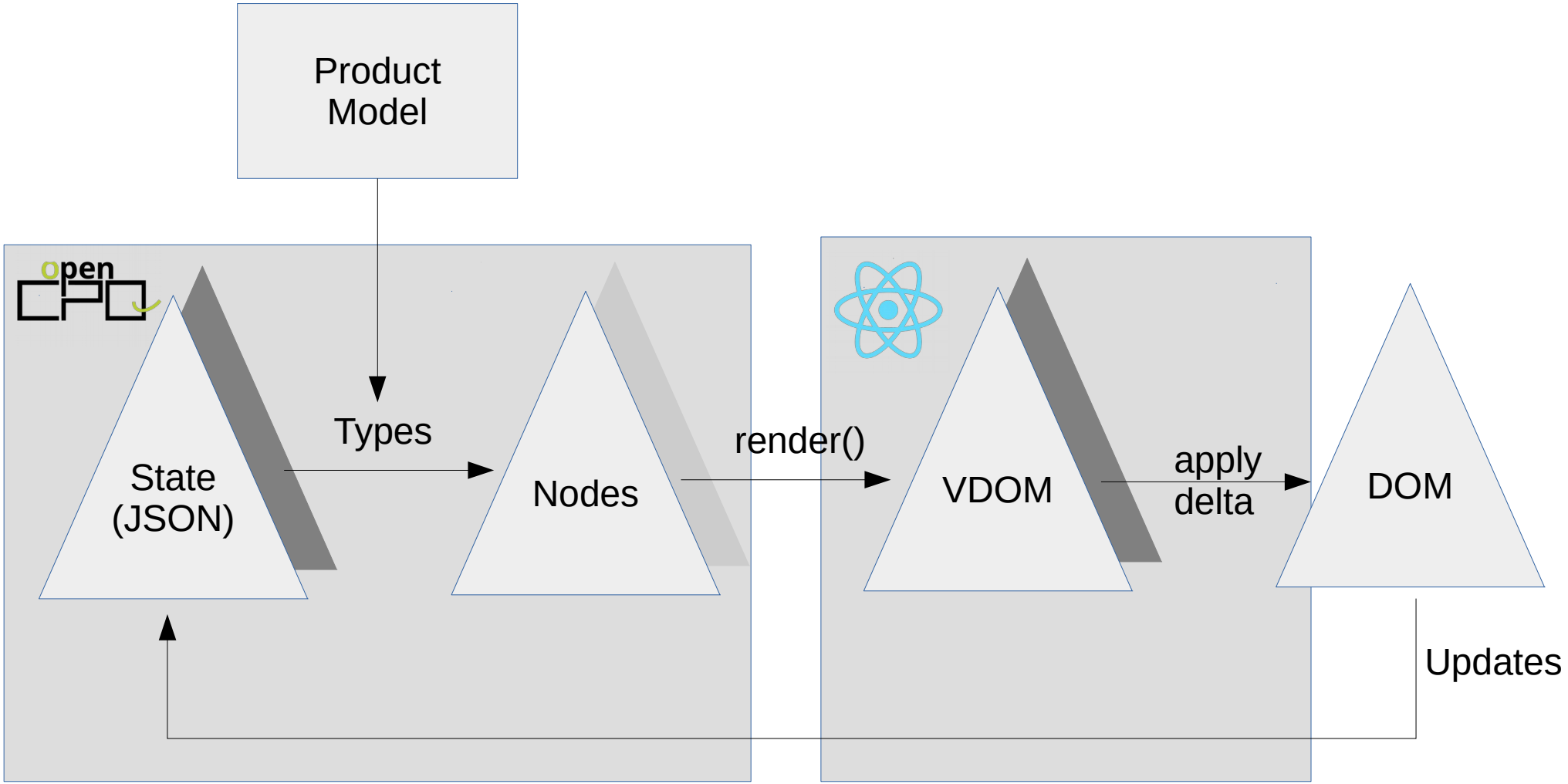
Change Propagation: Mixed



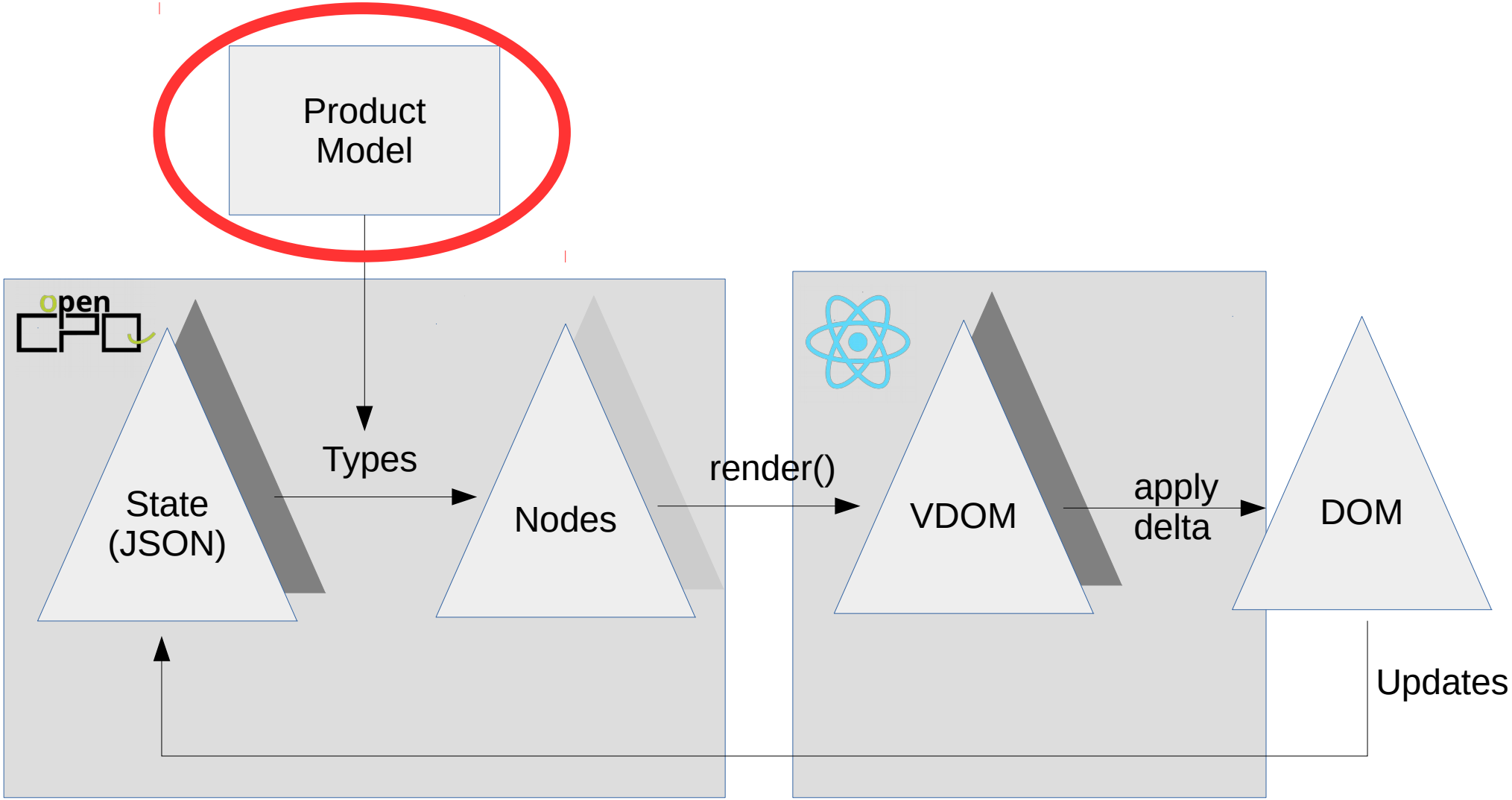
React:
A JavaScript library for building user interfaces

- Unique approach:
  - not a widget library
  - not an MVC framework
- Virtual DOM ("VDOM"):
  - Representation of the DOM tree as a JavaScript data structure (cheap!)
- Upon each update:
  - User code
    - generates VDOM from your model
    - possibly using XML templating integrated into JavaScript ("JSX")
  - React
    - diffs the VDOM with the previous VDOM
    - applies only the diff to the actual DOM

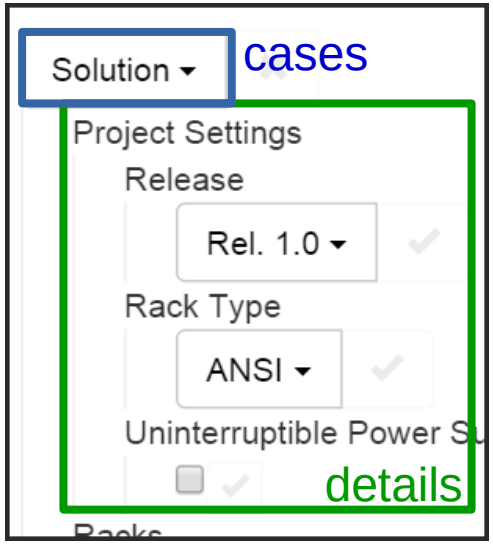
Architecture



Example Code: Product Model



Product Model: Cases with Details




```
var configuration = CSelect([
    unansweredCase("Select Configuration Mode"),
    ccase("Switches", "Optical Switches",
        CQuantifiedList({}, "Optical Switch",
                        opticalSwitches)),
    ccase("Rack",     "Racks",
        CQuantifiedList({}, "Rack",
                        rack)),
    ccase("Solution", "Solution",
          solution),
]);
```

Compare to pseudocode:

```
function configuration({caseId, details}) {
    switch (caseId) {
        case "unanswered": /* do nothing */ break;
        case "Switches":
            CQuantifiedList({}, "Optical Switch",
                            opticalSwitches)(details);
            break;
        case "Rack": ...; break;
        case "Solution":
            solution(details);
            break;
    }
}
```

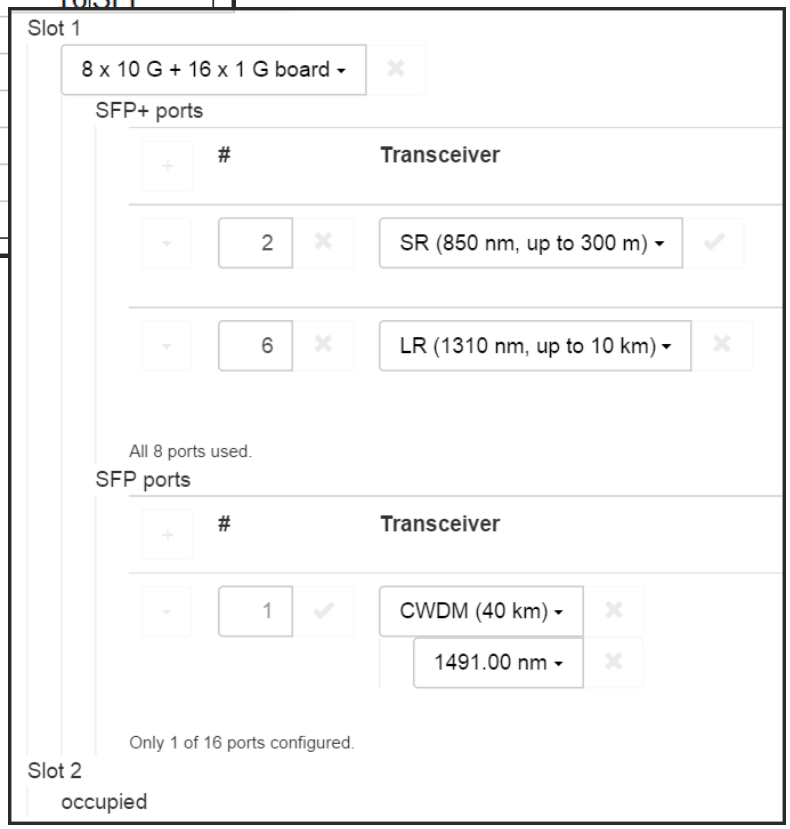
Data-Driven Product Modeling

Boards

Name	Label	Double width	Power	Ports		
				Label	Count	Type
B:FP	unequipped					
B:8x10_16x1	8 x 10 G + 16 x 1 G board	y	45	SFP+ ports	8	SFP+
				SFP ports	16	SFP
B:8x10	8 x 10 G board		30	SFP+ ports		
B:16x10	16 x 10 G board	y	50	SFP+ ports		
B:16xE1_75	16 x E1 electrical board (75 Ohm)		40			
B:16xE1_120	16 x E1 electrical board (120 Ohm)		40			
B:2x40	2 x 40 G board		60	QSFP+ ports		
B:1x100	1 x 100 G board		60	CFP ports		

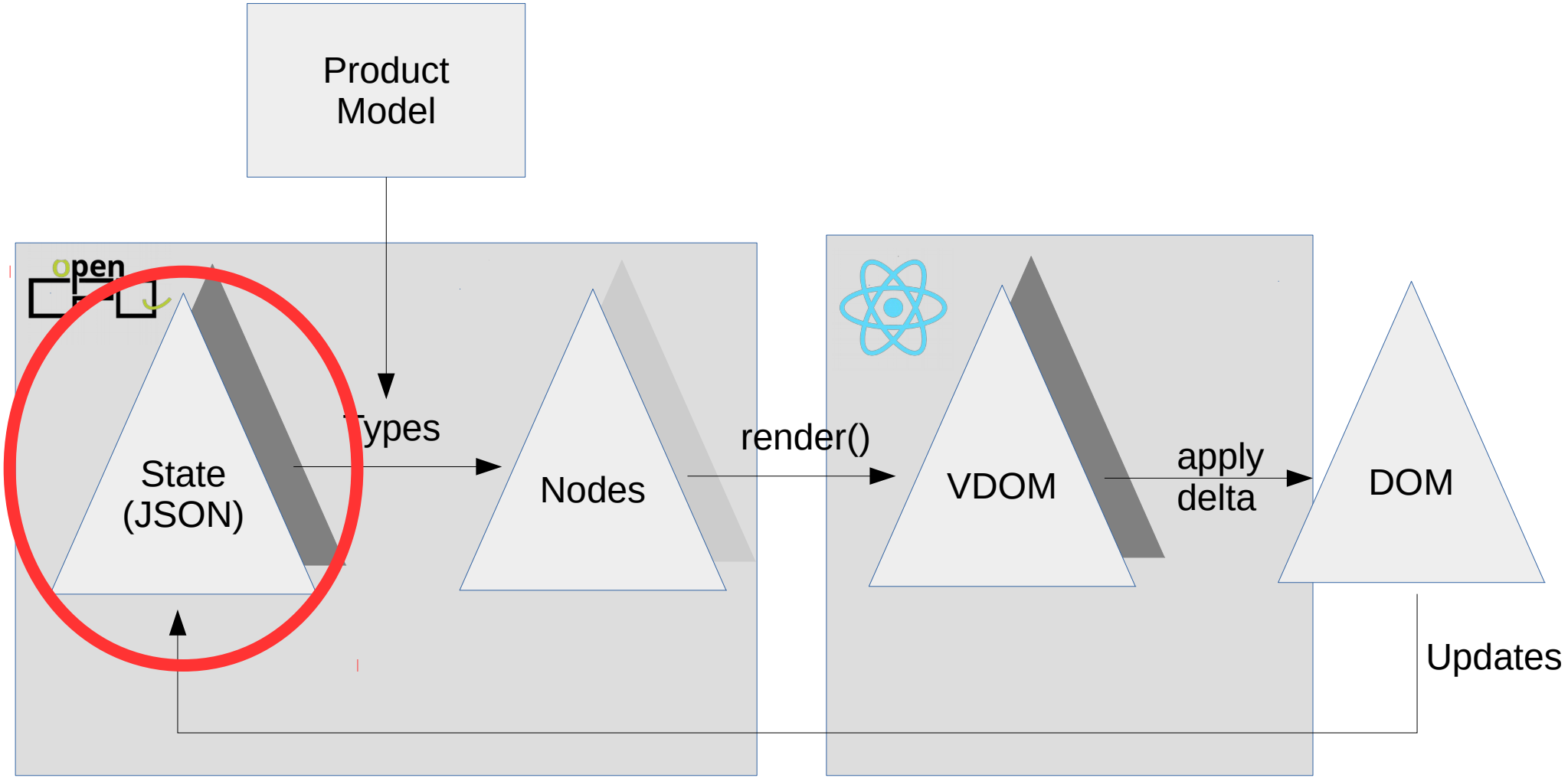
[for (… of …) if (…) …]
array comprehension

```
function boards(isDoubleWidthSlot) {
    return CSelect([
        for (b of components.boards)
            if (!b.doubleWidth || isDoubleWidthSlot)
                ccaseBOM(b.name, b.label,
                    aggregate("power", b.power,
                        ports(b.ports)))
    ]);
}
```

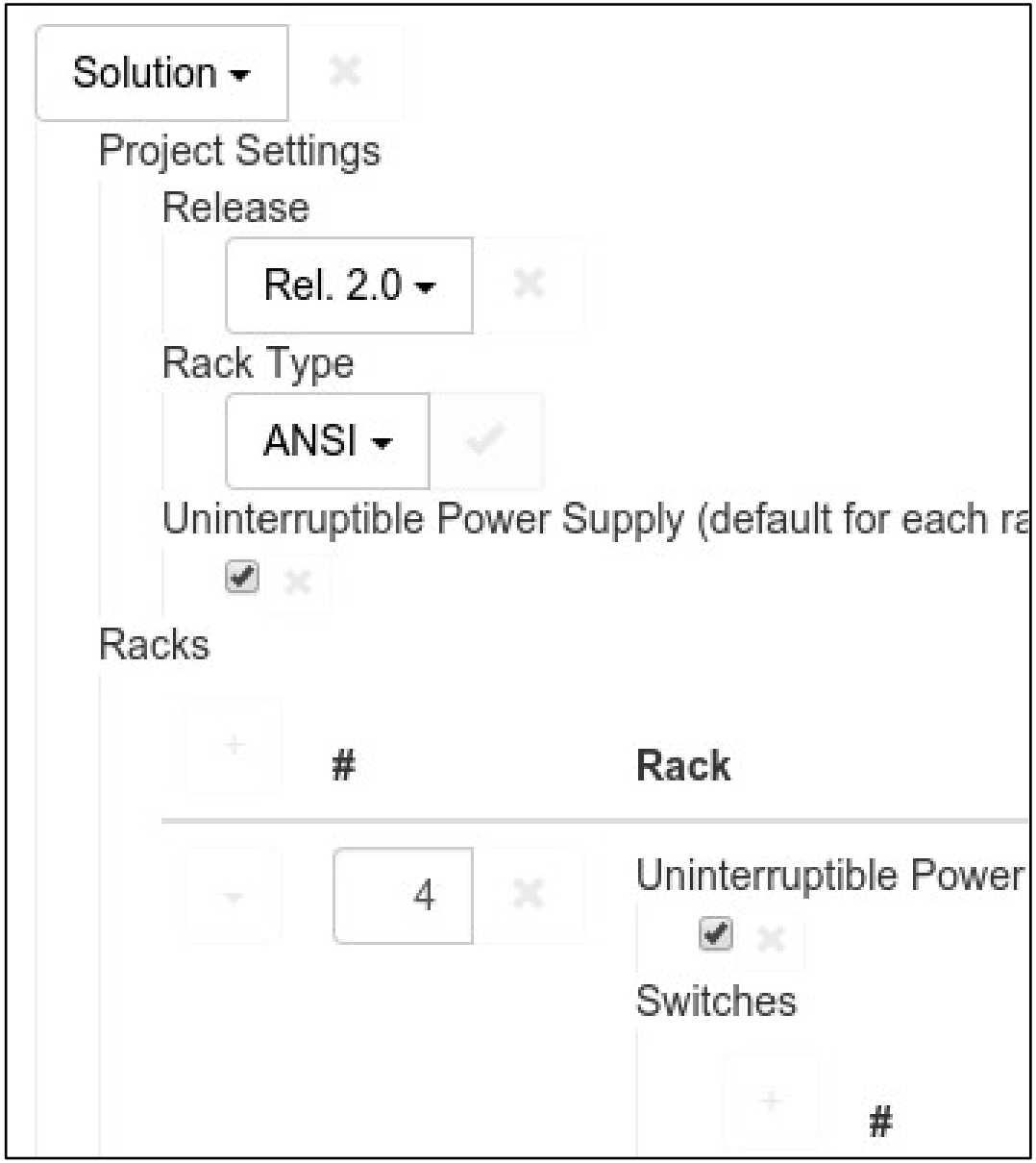


Concise specification of complex models

Example Data: State

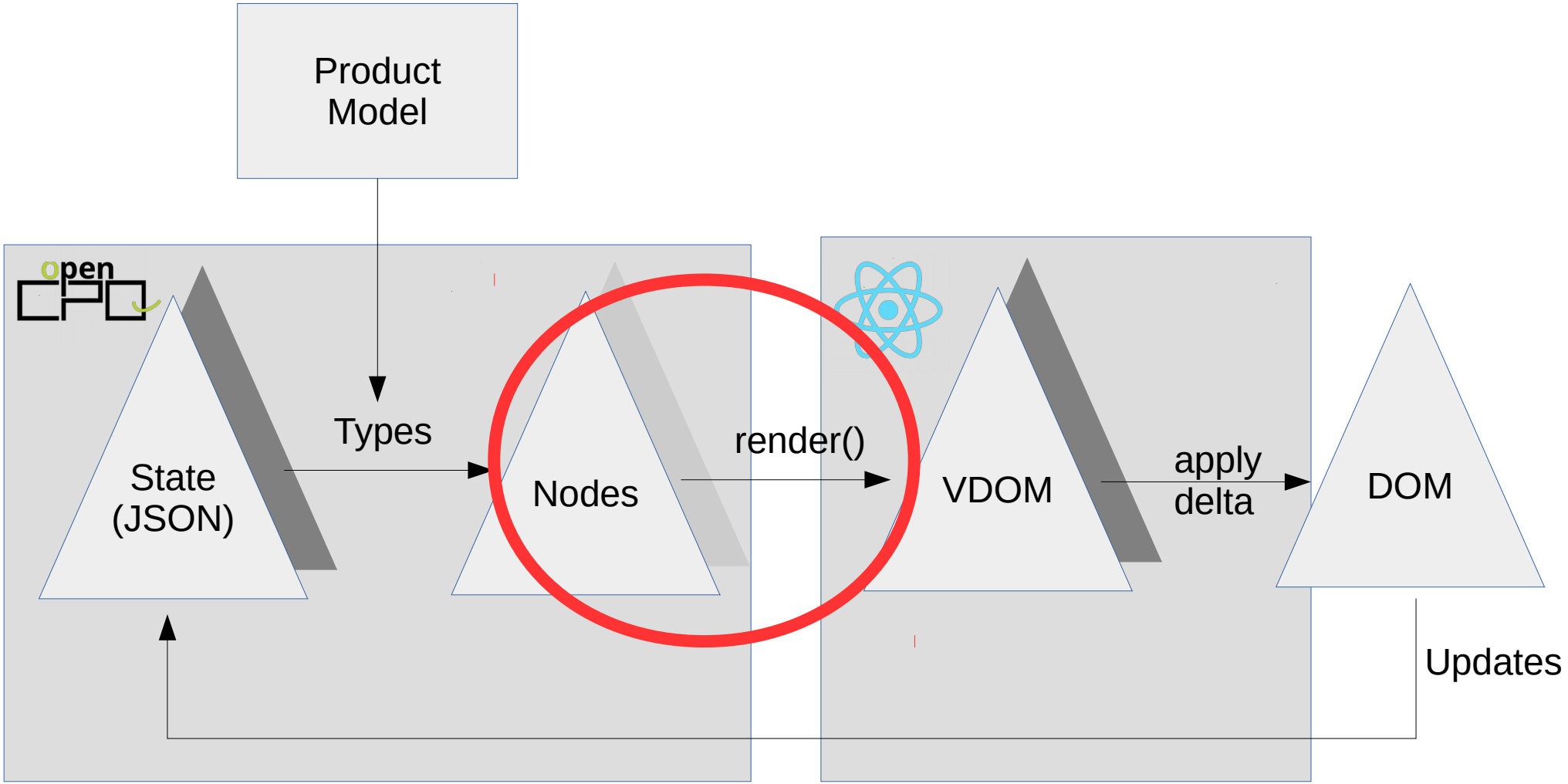


Configuration State




```
{
    "caseId": "Solution",
    "detailValue": {
        "project": {
            "release": {
                "caseId": "R2.0"
            },
            "UPS": true
        },
        "racks": [
            {
                "quantity": "4",
                "value": {
                    "UPS": true,
                    "switches": [
                        …
                    ]
                }
            }
        ]
    }
}
```

Example Code: Node Rendering



Selection Node (simplified)

```
class SelectNode extends Node {

  //constructor(options) { this.__options = options; }

  render() {
    var {cases, currentCase, detailNode, updateTo} = this.__options;
    return (
      <div>
        <DropdownButton title={currentCase.label}>
          {[
            for ({id, label} of cases)
              <MenuItem onSelect={() => updateTo({caseId: id})}>
                {label}
              </MenuItem>
          ]}
        </DropdownButton>
        {detailNode.render()}
      </div>
    );
  }
}
```

Inherited constructor

Unpack constructor parameters.

Create a VDOM tree.

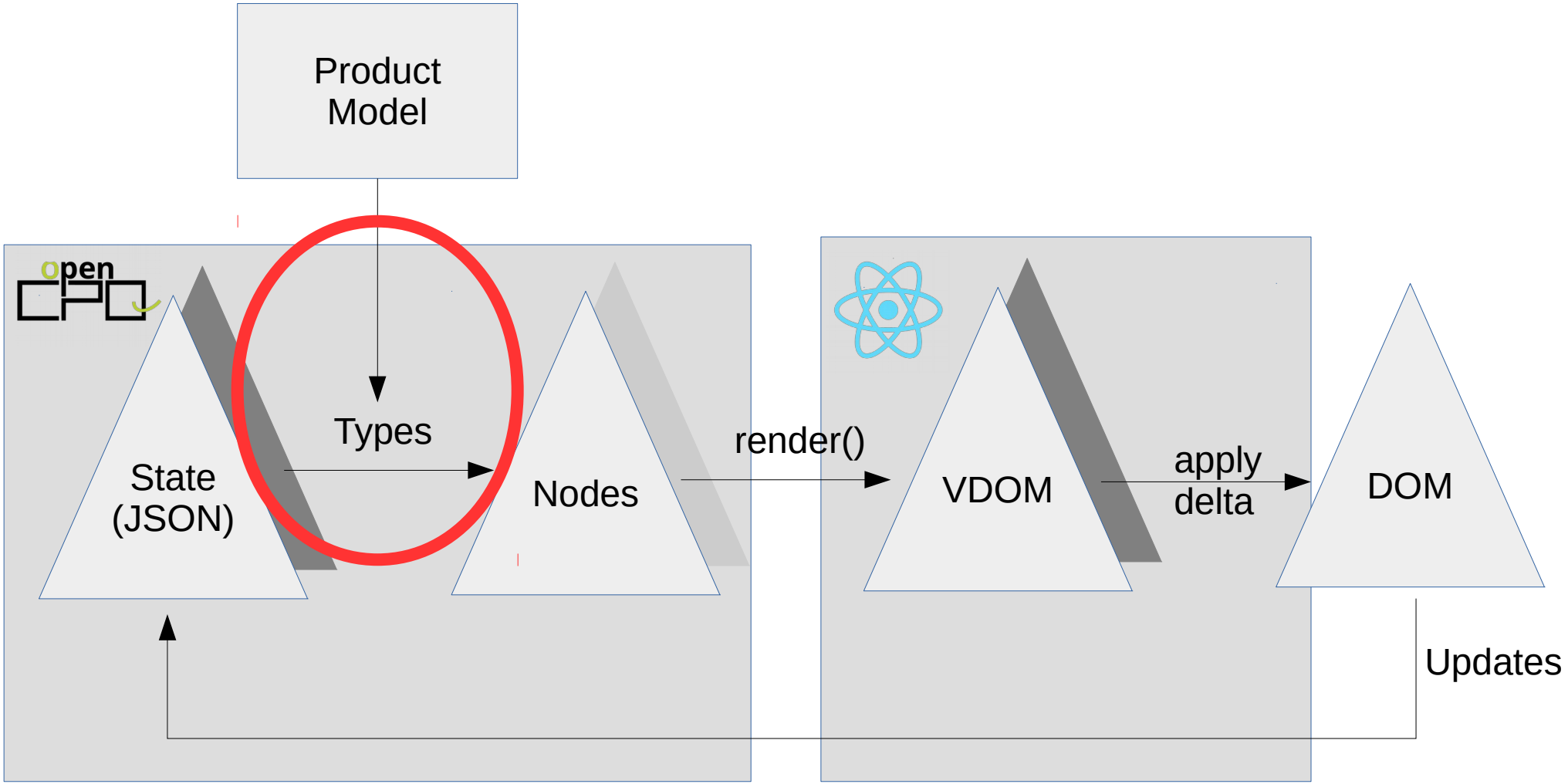
JSX: HTML **templates** in JavaScript

… also with „higher-level“ XML elements (from react-bootstrap)

Interpolate JavaScript expressions with {...}

array comprehension

Example Code: Types



Selection Type (simplified)

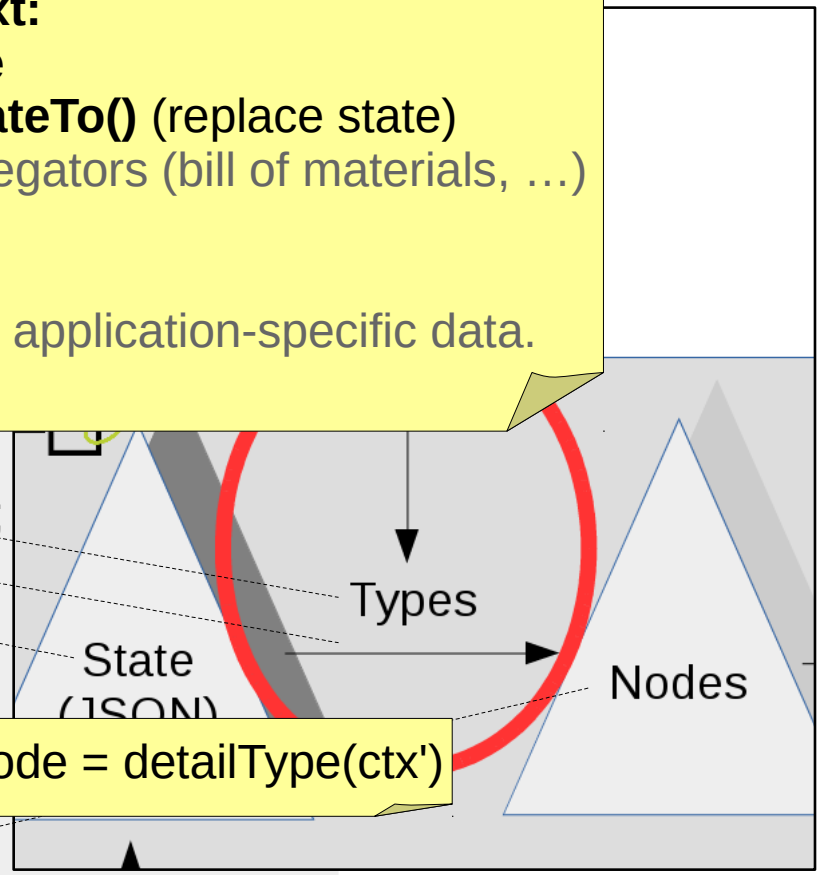
```
function ccase(id, label, type = CUnit()) {
  return {id, label, type};
}

function CSelect(cases) {
  return {
    makeNode(ctx) {
      var {state, updateTo} = ctx;
      var {caseId, detailState} = state || {caseId: cases[0].id};
      var currentCase = cases.find(x => x.id === caseId);
      var detailNode = currentCase.type.makeNode({
        ...ctx,
        state: detailState,
        updateTo(newDetail) {
          updateTo({caseId, detailState: newDetail});
        }
      });
      return new SelectNode({cases, currentCase, detai
    }
  };
}
```

Nothing to configure

Context:
- **state**
- **updateTo()** (replace state)
- aggregators (bill of materials, …)
- ...

Injects application-specific data.



detailNode = detailType(ctx')

updateTo() for detail node:
- **do not modify** surrounding state
- send **new state** to parent's updateTo()

=> easy **undo/redo**

Tools

- react.js
- bootstrap with less (Sass or stylus?)
- react-bootstrap, react-widgets
- BABEL (react JSX/esprima; TypeScript?)

- webpack (Gulp/ browserify)
- io (node), npm
- jQuery write less, do more.

Summary

Take advantage of modern **browser technology** for product configuration.

Powerful **modeling** based on JavaScript, React, and openCPQ.

Flexible and fast **user interface**.

Use, adapt, integrate, contribute!
https://github.com/webXcerpt/openCPQ

Issues to Discuss

- Use cases
  - product configuration, software configuration
  - questionnaires
  - …?
- Technologies
- Cooperation
  - Extensions: Integrations (SAP, Salesforce, ...), Visualization, ...
  - Student projects
  - Application development

webXcerpt
manage your information