

Configuring in the Browser, Really!

Tim Geisler, Heribert Schütz

webXcerpt Software GmbH

tg@webxcerpt.com, hs@webxcerpt.com

CWG 2015, Prague 2015-04-28



Our History

- Product configuration since 2002,
with SAP since 2007
- Built and maintained
 - Models
 - Modeling environments
 - Configuration frameworks

Problem 1: Modeling

- Framework-specific modeling tools
- Lack of abstraction features and data structures
 - Loops, functions
 - Arrays, objects (with methods)
- Models not represented as human-readable text
 - Edit, search & replace
 - Discuss, annotate
 - Compare, manage revisions

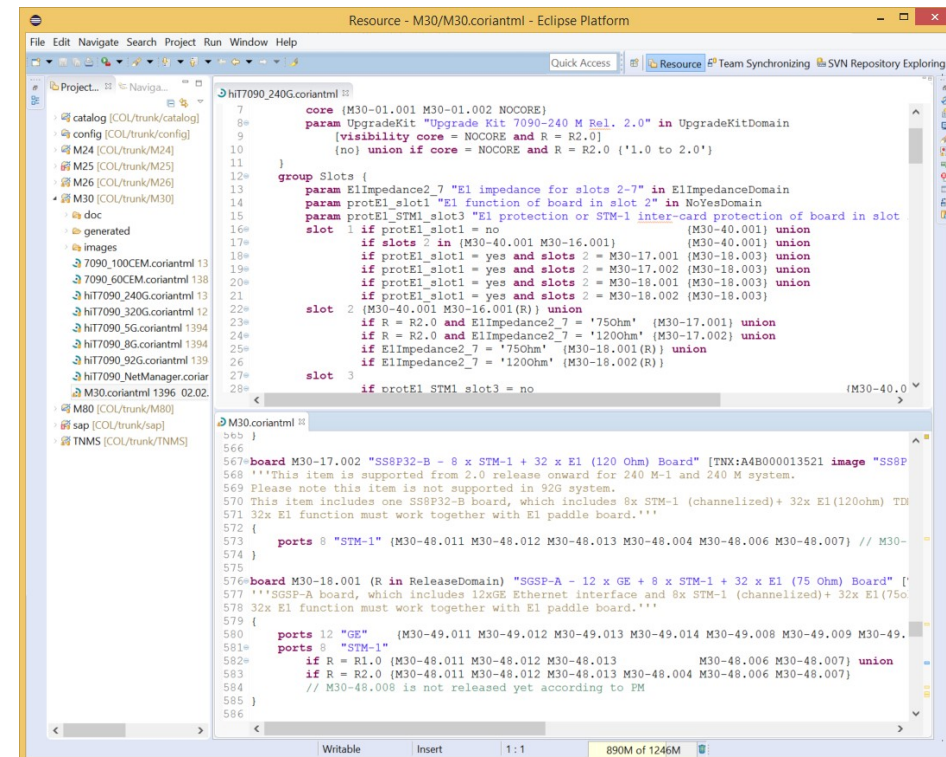
Problem 1: Modeling – Solution A



Problem 1: Modeling – Solution B

Our solution so far:

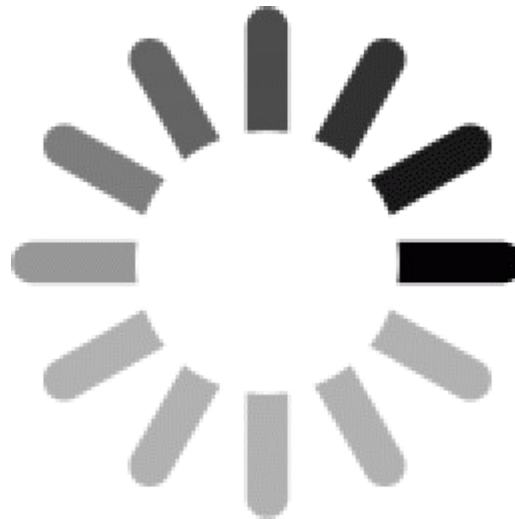
- Customer-specific modeling languages
- Modeling environments based on Eclipse and Xtext
- Automated generation of model representation for target framework
- See also CWG talks
 - Vienna 2010:
ConfigModeler and VClipse -
languages and IDEs
for product modeling
 - Cologne 2011:
Domain-Specific Languages
for Product Modeling
 - Berlin 2012:
How to Build Your Own
Product-Modeling Environment?



Problem 1: Modeling – Solution C

- Use a programming language
 - For application-specific inferencing
 - But also to build up the model
- Use 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: User Experience



Problem 2: User Experience

- Performance
 - Client-server round trips
- Rigid UI
 - UI structure imposed by framework
 - High costs for application-specific UI
- Need to be online

Increasing gap:

Configurators ↔ Modern web applications

In the meantime ...



In the meantime ...

Client hardware improved

- CPU
- Memory
- Even on mobiles



... but the speed of light remained the same.

In the meantime ...

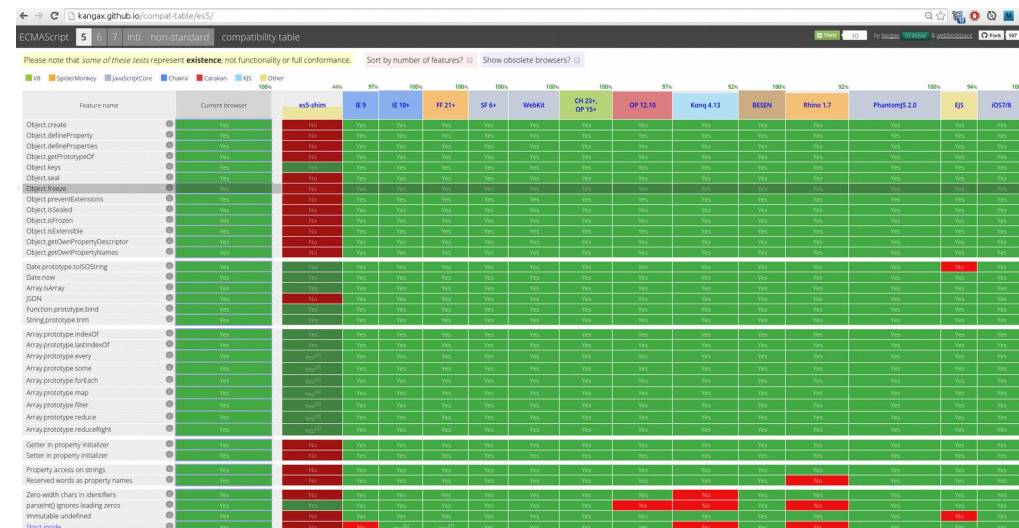
Browser improvements:

- JavaScript performance
- Standardized features



- UI extensions
- Offline applications
- Local storage
- ...

- Improved compatibility



The screenshot shows a web browser displaying the ECMAScript compatibility table for ES5. The table lists various features and their support status across different browsers. The features listed include:

- Object.create
- Object.defineProperty
- Object.defineProperties
- Object.getPrototypeOf
- Object.keys
- Object.seal
- Object.freeze
- Object.preventExtensions
- Object.isExtensible
- Object.isFrozen
- Object.isSealed
- Object.getPrototypeOf
- Object.getPrototypeOf
- Date.prototype.toJSON
- Date.now
- Array.isArray
- JSON
- Function.prototype.bind
- String.prototype.trim
- Array.prototype.indexOf
- Array.prototype.lastIndexOf
- Array.prototype.every
- Array.prototype.some
- Array.prototype.forEach
- Array.prototype.map
- Array.prototype.filter
- Array.prototype.reduce
- Array.prototype.reduceRight
- Getter in property initializer
- Setter in property initializer
- Property access on strings
- Reserved words as property names
- Zero-width chars in identifiers
- parsed() ignores leading zeros
- Immutable undefined

The table shows support status for various browsers, including:

- es5-shim
- IE 9
- IE 10
- FF 21+
- SF 6+
- WebKit
- CH 29+ OP 15+
- OP 12.10
- Kang 4.13
- BSLN
- Rhino 1.7
- PhantomJS 2.0
- ES
- IOS78

The table uses a color-coded system to indicate support: green for 'Yes', red for 'No', and yellow for 'Partial'.

In the meantime ...

A software ecosystem for web applications flourished:

- Web-application frameworks
- Preprocessors for JavaScript/HTML/CSS
- Libraries
- Build tools



In the meantime ...

Web browsers have become a serious application platform.

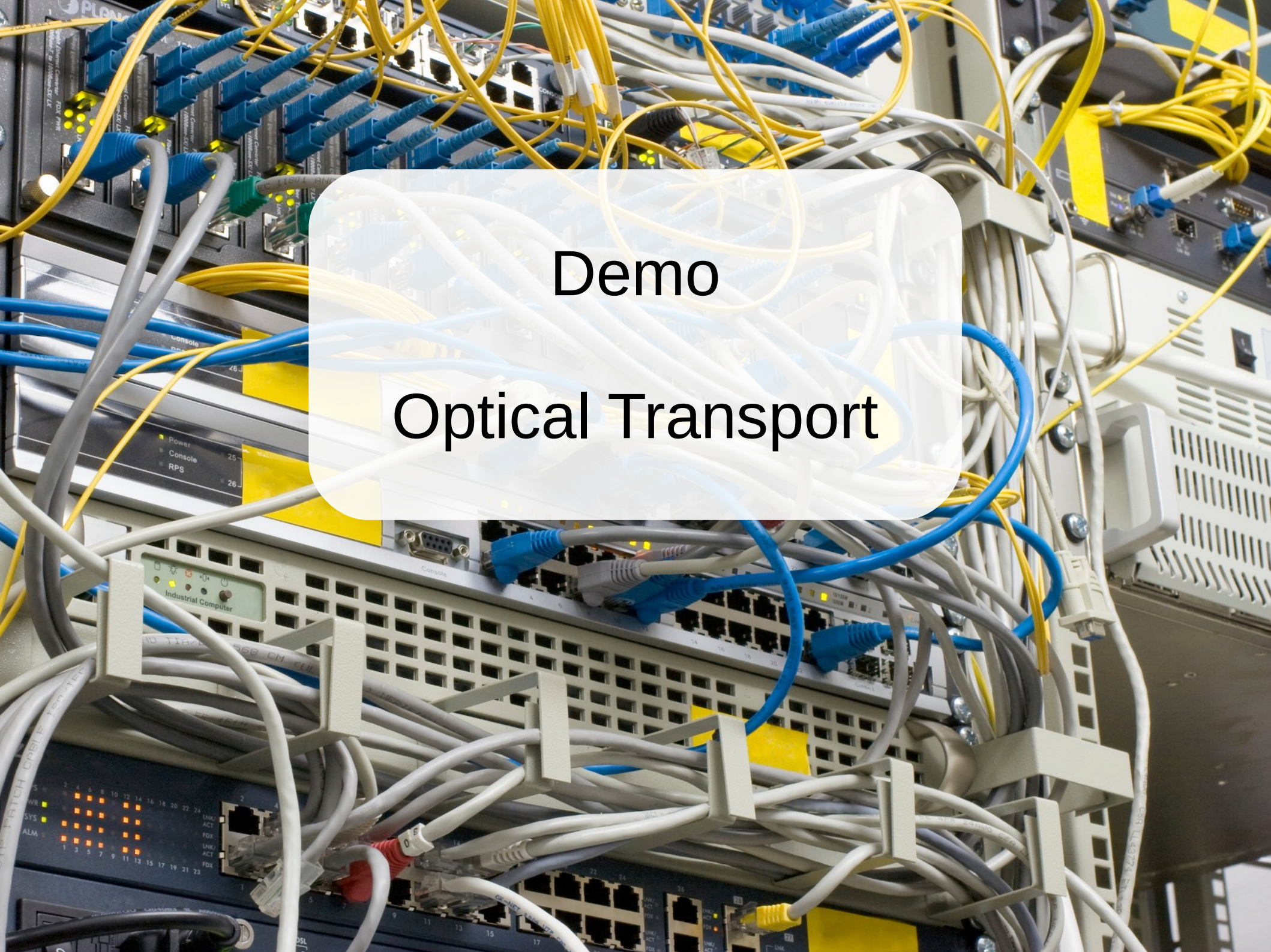
Even for the business logic.

And they are getting better and better.

Configuring in the Browser:

Implement configurators in JavaScript.

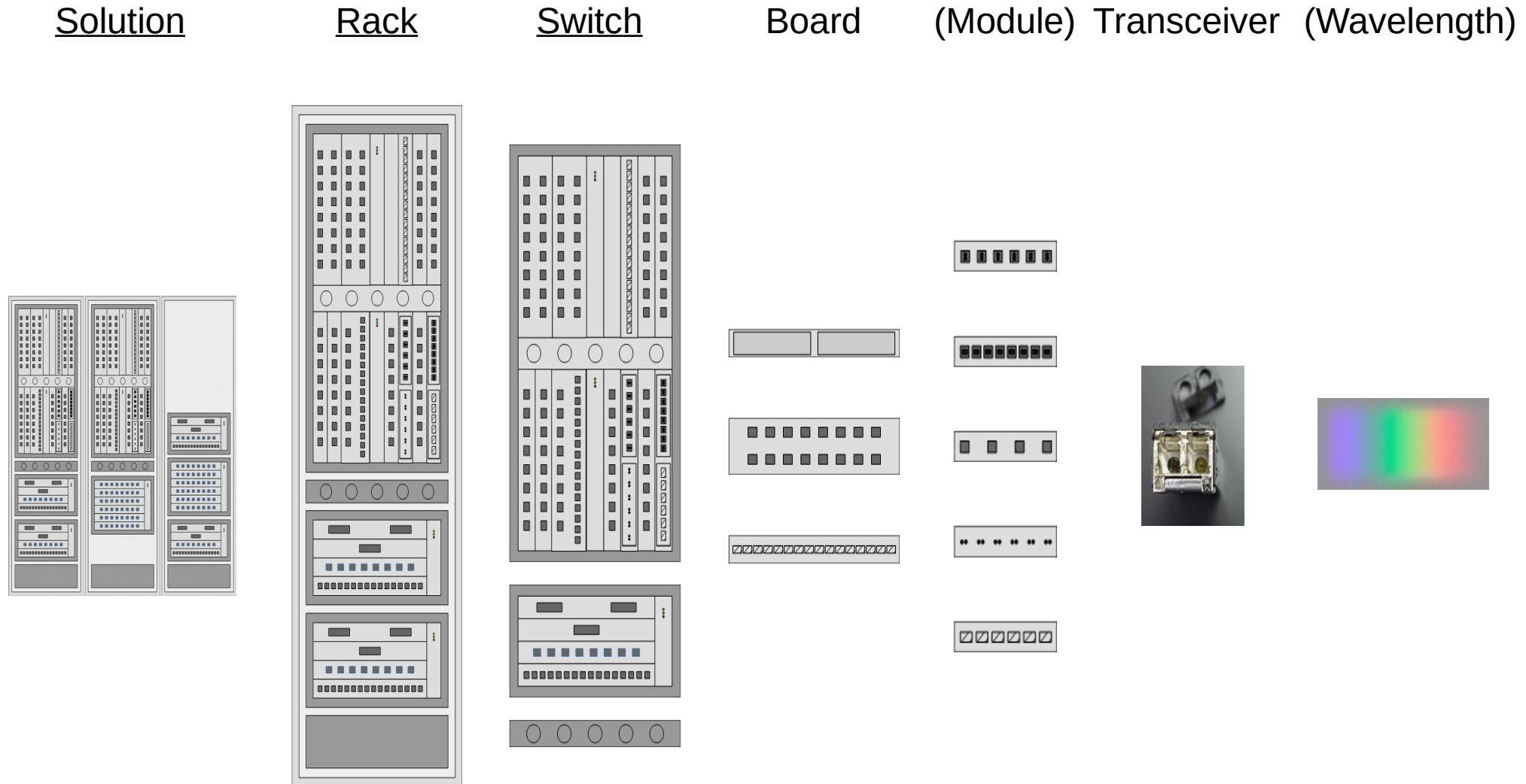
JavaScript is also
a reasonable choice for modeling.



Demo

Optical Transport

Demo Example: Hierarchical Configuration



Demo

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

The screenshot displays the OpenCPQ web application interface. The top navigation bar includes a browser address bar with the URL `opencpq.webxcerpt.com/examples/optical-transport/` and a toolbar with buttons for undo, redo, reset, save, restore, file selection, import, and export. The main interface is divided into two primary sections: Configuration and Contents.

Configuration Panel:

- Solution:** A dropdown menu with a close button.
- Project Settings:**
 - Release:** A dropdown menu set to "Rel. 1.0" with a checkmark.
 - Rack Type:** A dropdown menu set to "ANSI" with a checkmark.
 - Uninterruptible Power Supply (default for each rack):** A checkbox that is checked.
- Racks:** A table with columns for a plus icon, a number, and a rack name.

	#	Rack
	1	Uninterruptible Power Supply
- Switches:** A table with columns for a plus icon, a number, and a product name.

	#	Product
	2	Optical Switch OS6
- Slot 1:** A table with columns for a dropdown arrow, a number, a name, and a product name.

	2	Slot 1	16 x 10 G board

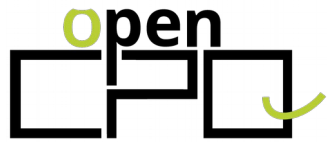
Contents Panel:

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

Bill of Materials Panel:

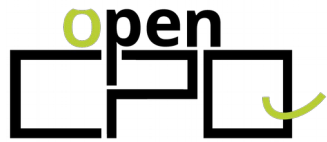
export as CSV

	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



– a JavaScript-based Configurator Framework

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



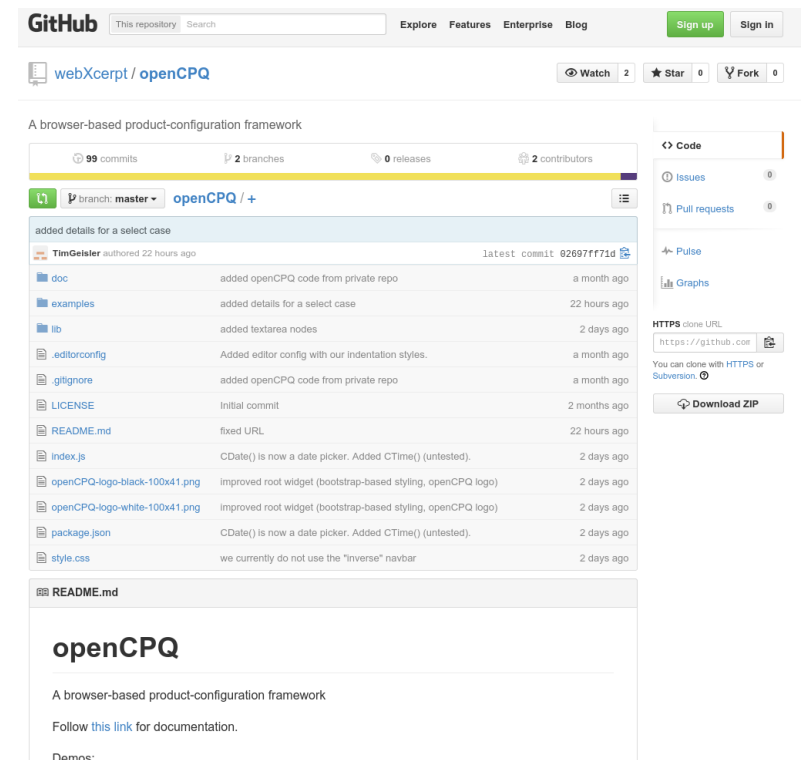
– an Open-Source Project

Source code and links to live demos
available on Github:

<https://github.com/webXcerpt/openCPQ>

Liberal MIT license

Use, adapt,
integrate, contribute!



Modeling with openCPQ: Cases with Details

Solution ▼ **cases**

Project Settings

Release

Rel. 1.0 ▼ ✓

Rack Type

ANSI ▼ ✓

Uninterruptible Power Supply

☐ ✓ **details**

Racks

```
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),
]);
```

Slot 1

16 x 10 G board ▼ **cases**

SFP+ ports

+	#	Transceiver
▼	1 ✓	CWDM (40 km) ▼ 1471.00 nm ▼ ✓

⚠ Only 1 of 16 ports configured. **details**

Slot 2
occupied

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

Data-Driven Modeling with openCPQ

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 SFP ports	8 16	SFP+ 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		

```
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)))
    ]);
}
```

Slot 1

8 x 10 G + 16 x 1 G board ▾ ✕

SFP+ ports

#	Transceiver
2 ✕	SR (850 nm, up to 300 m) ▾ ✓
6 ✕	LR (1310 nm, up to 10 km) ▾ ✕

All 8 ports used.

SFP ports

#	Transceiver
1 ✓	CWDM (40 km) ▾ ✕ 1491.00 nm ▾ ✕

Only 1 of 16 ports configured.

Slot 2
occupied



Concise specification of complex models

Modeling with openCPQ: Application-specific Abstractions

Configuration Type

New Configuration ▾ ✕

Server

New Configuration

Connected clients ✕

Server size ✕

Redundant server ☒ ✕

Configuration Type

Upgrade / Extension ▾ ✕

Server

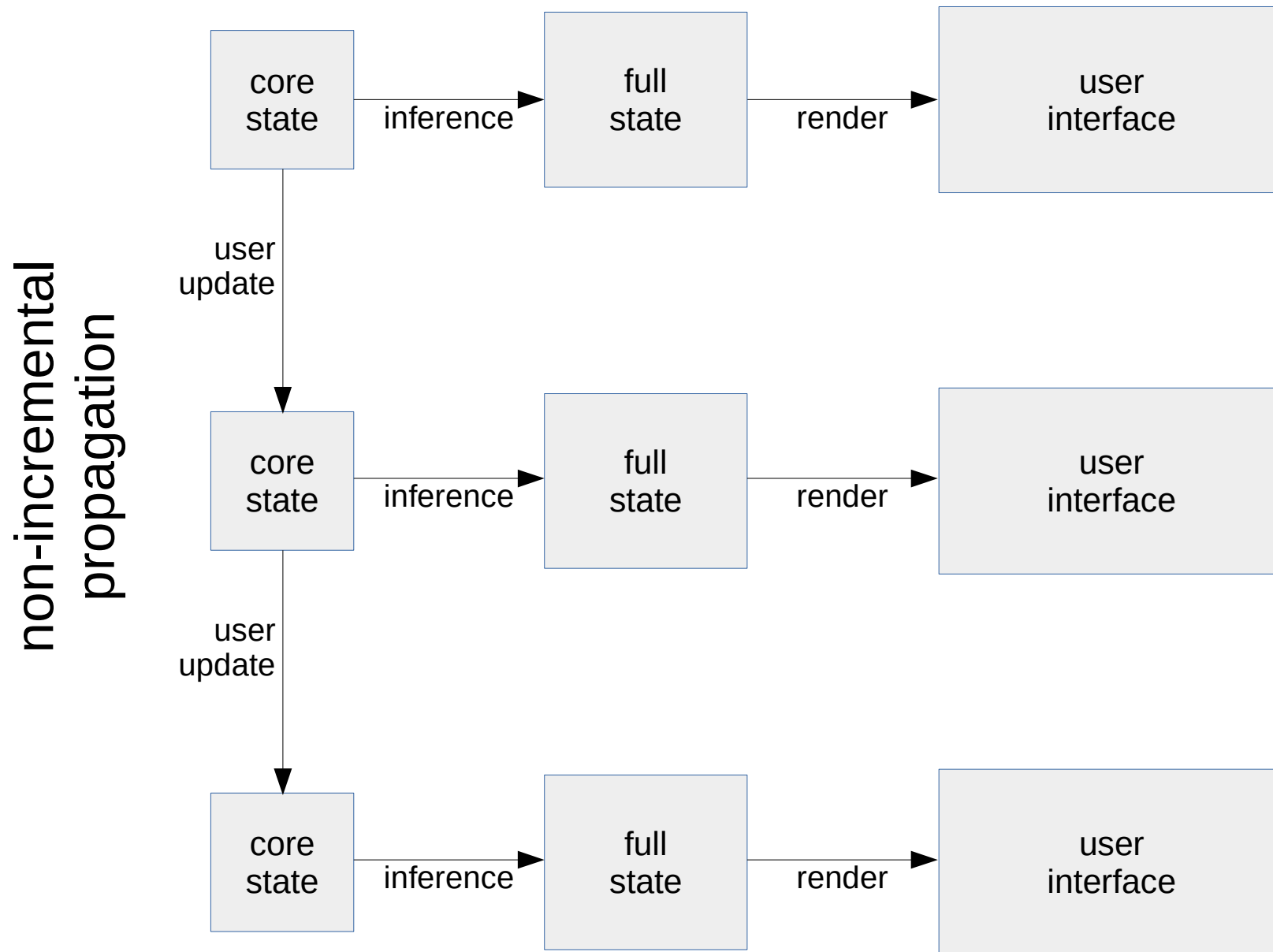
	Existing Configuration	Planned Configuration
Connected clients	<input type="text" value="20"/> ✕	<input type="text" value="20"/> ✓
Server size	<input type="text" value="medium"/> ✕	<input type="text" value="small"/> ✕ ⚠ Downgrade not supported
Redundant server	<input type="checkbox"/> ✓	<input checked="" type="checkbox"/> ✕

```
CNamespace("props", CGroup([
  cmember("ConfigType", "Configuration Type",
    CNamed("props", "ConfigType", {valueAccessor: n => n.value}, CSelect([
      ccase("NEW", "New Configuration"), ccase("EXT", "Upgrade / Extension"),
    ]))),
  cmember("Server", "Server", ep.table([
    ep.rowInteger("clients", "Connected clients"),
    crow("Size", "Server size", ({props}) => props.ConfigType === "EXT"
      ? [ep.eCell("Size", CSelect([for (s of serverSizes) ccase(s)])),
        () => ep.pCell("Size", CSelect([for (s of serverSizes)
          onlyIf(serverSizes.indexOf(s) >= serverSizes.indexOf(ep.E(props.Size)),
            "Downgrade not supported", [ccase(s)]))]
        : [ep.pCell("Size", CSelect([for (s of serverSizes) ccase(s)]))]
      ],
    ep.rowBoolean("redundancy", "Redundant server"),
  ])),
]))),
```

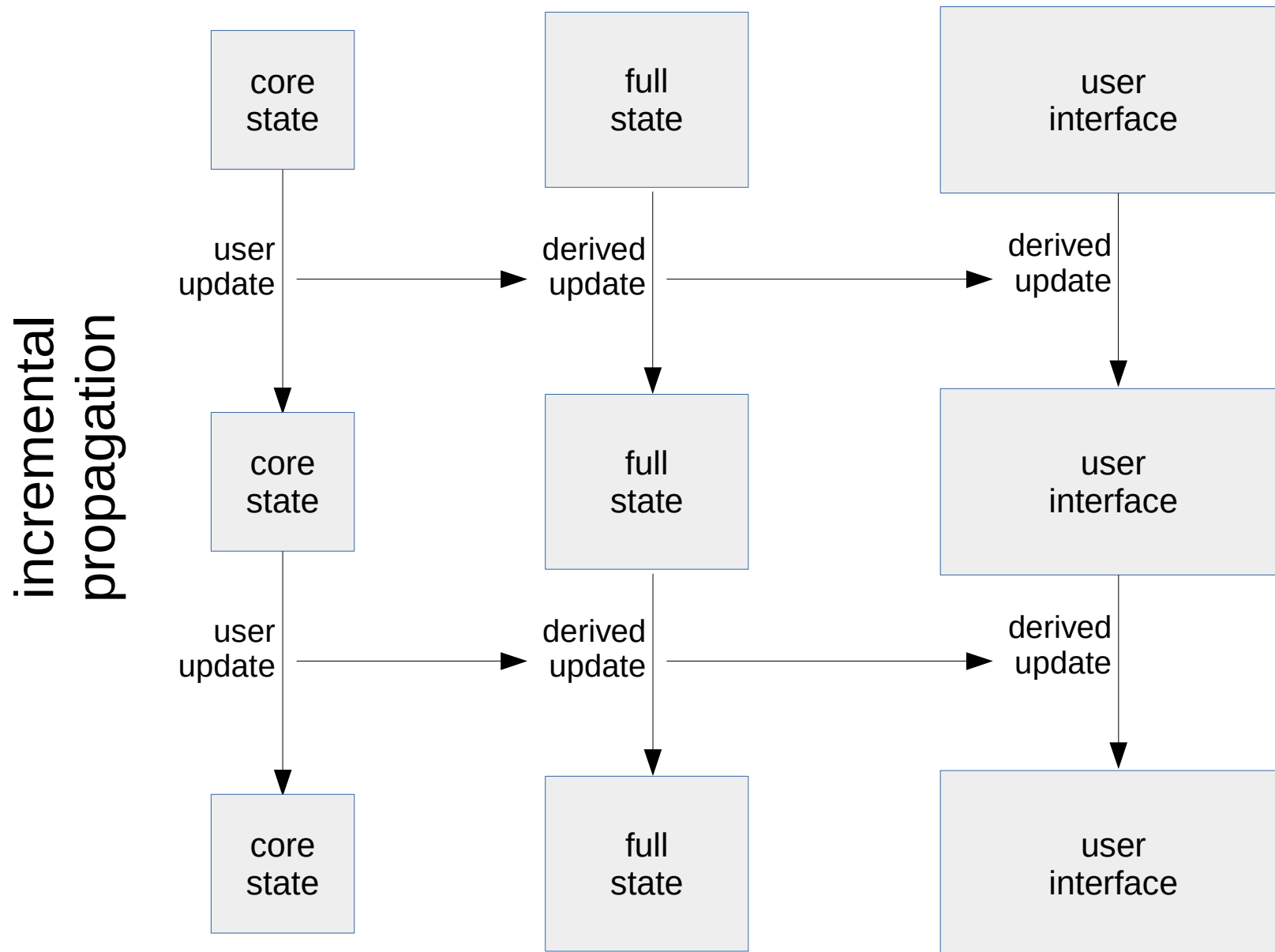



How it Works

Change Propagation



Change Propagation

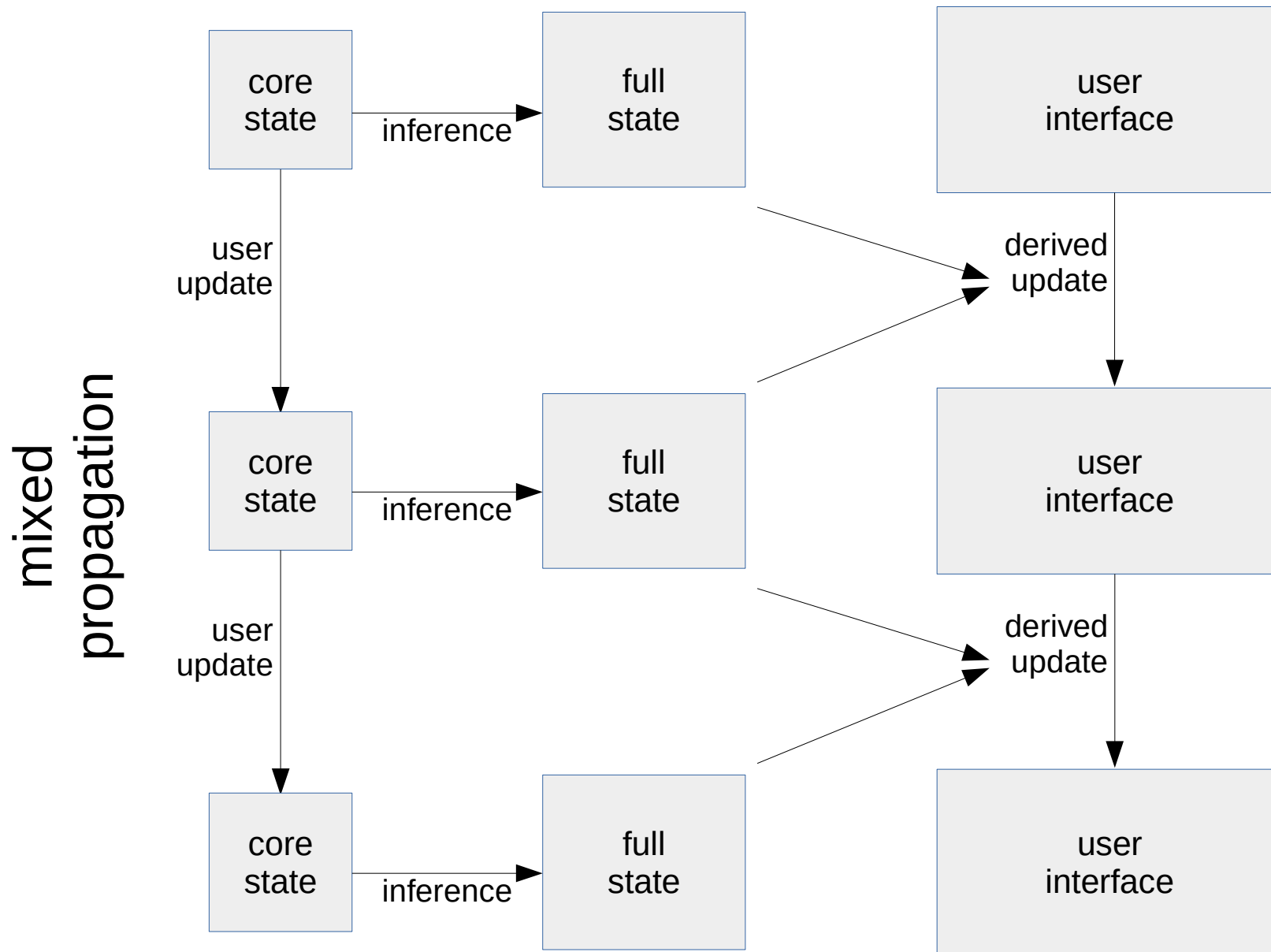


Change Propagation

Trade-off:

- Non-incremental propagation:
 - Redo inference steps
 - CPU consumption
 - Redo rendering
 - CPU consumption
 - Flicker, loss of UI state (focus, scroll, selection), ...
- Incremental propagation:
 - Keep track of dependencies
 - Error-prone (unless completely shielded from the modeler)
 - Consumes memory and CPU

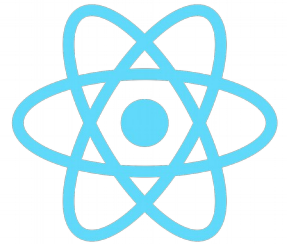
Change Propagation



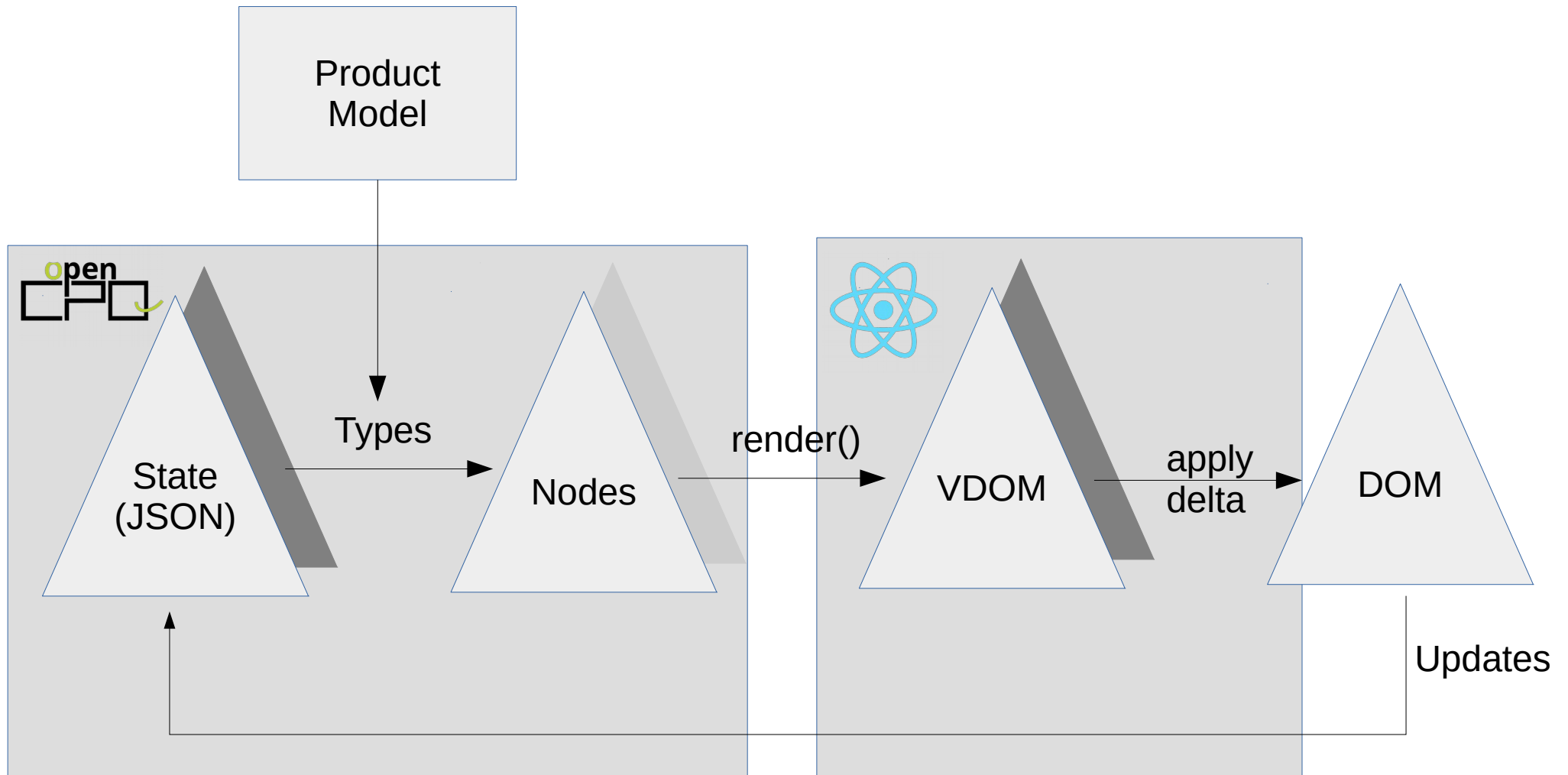
React:

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



SAP Integration

- Models
 - Conversion of LO-VC and IPC models to openCPQ
 - Schema, basic logic: automatable with Vclipse extension
 - Complex logic: manual conversion
 - Model storage and management
 - Just static resources
 - App server not needed (but can be used)

SAP Integration

- Data
(e.g. materials with classification information)
 - Live vs. pre-exported
 - Bundling with application vs. loading on demand
- Runtime
 - Loading and saving configurations
 - External configurator API
 - Mimic IPC

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>



Our Offer

Discuss:

- Use cases, modeling challenges, ...
- Integrations

Cooperate:

- Professional services, training, ...
- For end users or integrators

