

Glenn Cavarlé

August 2016

Lab-STICC - University of Brest, France

Dynamic Round-Trip Engineering in the context of FOMDD

Glenn Cavarlé

August 2016

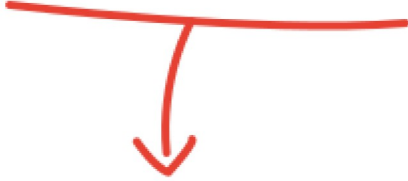
Lab-STICC - University of Brest, France

FOMDD

Feature-Oriented Model Driven Development

FOMDD

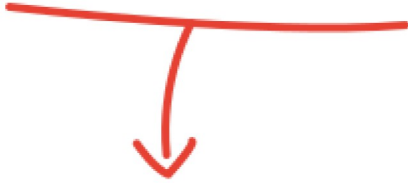
Feature-Oriented Model Driven Development



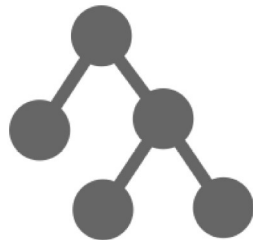
**Breaking down software systems
in terms of the features they provide**

FOMDD

Feature-Oriented Model Driven Development

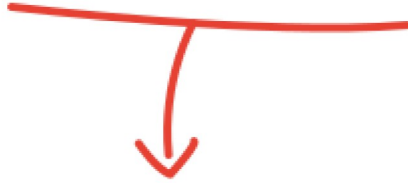


Breaking down software systems
in terms of the features they provide



FOMDD

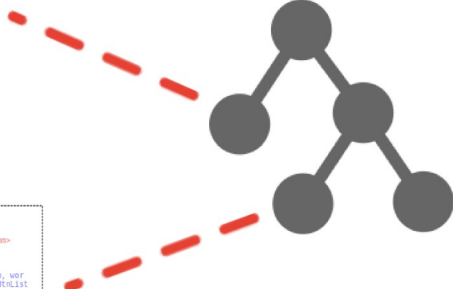
Feature-Oriented Model Driven Development



Breaking down software systems
in terms of the features they provide



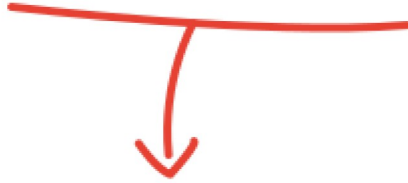
```
/**
 * Simple HelloButton() method.
 * @version 1.0
 * @author John Doe <doe.j@example.com>
 */
HelloButton()
{
    JButton hello = new JButton("hello, wor
    hello.addActionListener( new HelloList
    // use the JFrame type until support for t
    // new component is finished
    JFrame frame = new JFrame("Hello Button"
    Container pane = frame.getContentPane();
    pane.add( hello );
    frame.pack();
    frame.show(); // display the fra
```



```
/**
 * Simple HelloButton() method.
 * @version 1.0
 * @author John Doe <doe.j@example.com>
 */
HelloButton()
{
    JButton hello = new JButton("hello, wor
    hello.addActionListener( new HelloList
    // use the JFrame type until support for t
    // new component is finished
    JFrame frame = new JFrame("Hello Button"
    Container pane = frame.getContentPane();
    pane.add( hello );
    frame.pack();
    frame.show(); // display the fra
```

FOMDD

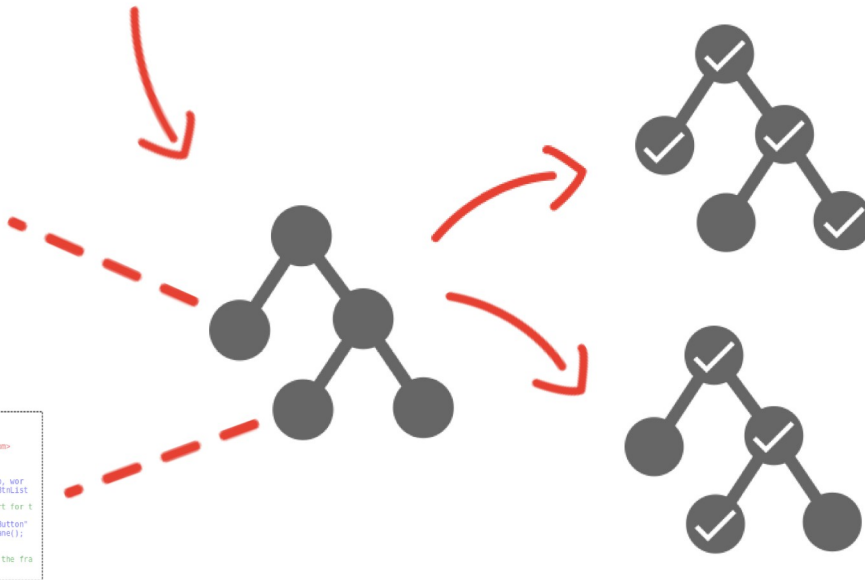
Feature-Oriented Model Driven Development



Breaking down software systems
in terms of the features they provide

```
/**
 * Simple HelloButton() method.
 * @version 1.0
 * @author John Doe <doe.j@example.com>
 */
HelloButton()
{
    JButton hello = new JButton("Hello, wor
    hello.addActionListener( new HelloList
// use the JFrame type until support for t
// new component is finished
JFrame frame = new JFrame("Hello Button"
Container pane = frame.getContentPane();
pane.add( hello );
frame.pack();
frame.show(); // display the fra
```

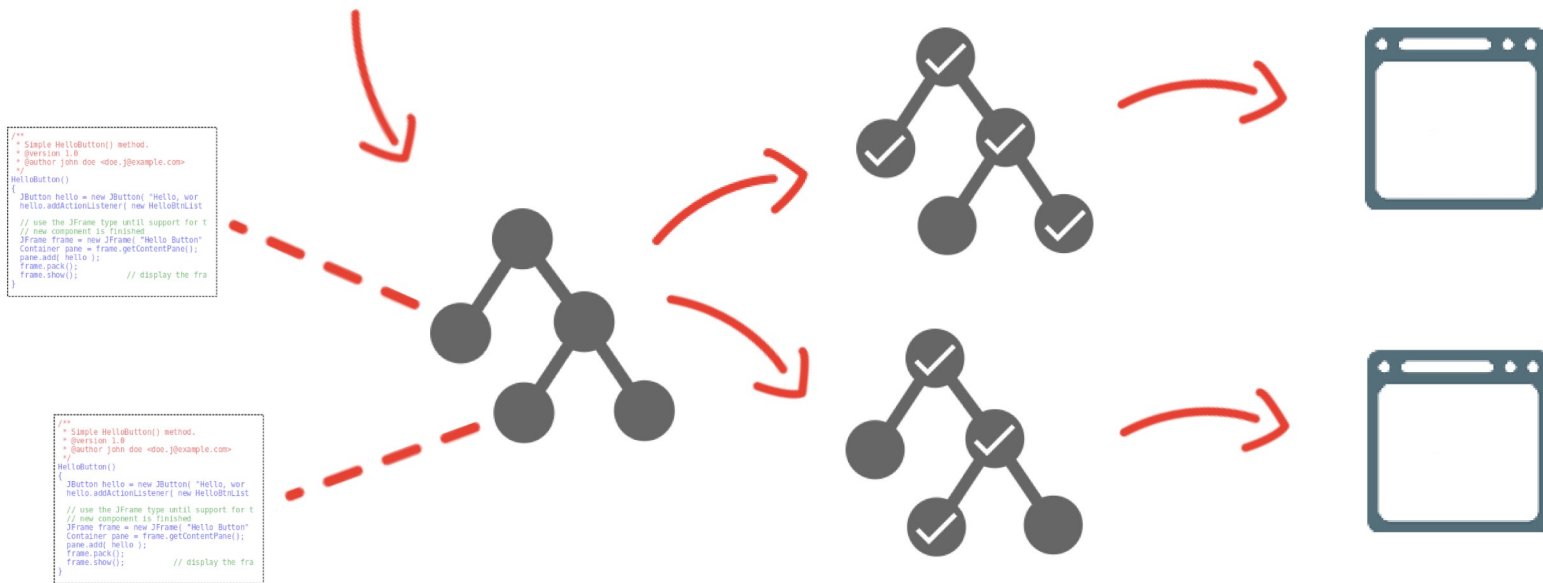
```
/**
 * Simple HelloButton() method.
 * @version 1.0
 * @author John Doe <doe.j@example.com>
 */
HelloButton()
{
    JButton hello = new JButton("Hello, wor
    hello.addActionListener( new HelloList
// use the JFrame type until support for t
// new component is finished
JFrame frame = new JFrame("Hello Button"
Container pane = frame.getContentPane();
pane.add( hello );
frame.pack();
frame.show(); // display the fra
```



FOMDD

Feature-Oriented Model Driven Development

Breaking down software systems
in terms of the features they provide



FOMDD

Feature-Oriented Model Driven Development



**Building abstract representations
of software systems**

FOMDD

Feature-Oriented Model Driven Development



Building abstract representations
of software systems

```
from urllib2 import StringIO
import os
import urllib
import urllib2
import subprocess
import tempfile

from flask import Flask, request, render_template, make_response
app = Flask(__name__)

if not app.debug:
    import logging
    import logging.handlers
    handler = logging.handlers.RotatingFileHandler('logs/app.log', maxBytes=1024*1024, backupCount=10)
    handler.setLevel(logging.INFO)
    app.logger.addHandler(handler)
    app.logger.setLevel(logging.INFO)
    app.logger.info('FOMDD')

languages = ['Python', 'Java', 'C#']

@app.route('/')
def index():
    return render_template("index.html", source=app.config['base_path'], readID, languages=languages)

@app.route('/reader', methods=['POST'])
def reader():
    lang = request.form['lang']
    source = request.form['source']
    assert lang in languages
    src = tempfile.mkdtemp()

    requestBody = render_template("source-listing.txt", lang=lang, source=source)
    url = StringIO(requestBody)
    standardize = os.path.join(src, 'standardize')
    src = os.chdir(src)
    cmd = "python %s %s %s" % (standardize, url, lang)
    p = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, cwd=src)
    stdout, stderr = p.communicate()

    p_src = subprocess.Popen("ls -la %s" % src, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, cwd=src)
    stdout, stderr = p_src.communicate()

    ppg = subprocess.Popen("cat %s" % src, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, cwd=src)
    stdout, stderr = ppg.communicate()

    app.logger.info("Showing %s" % lang)
    return render_template("reader.html", lang=lang, source=source, stdout=stdout, stderr=stderr)

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```

FOMDD

Feature-Oriented Model Driven Development



Building abstract representations
of software systems

```
from stringIO import StringIO
import os
import urllib
import urllib2
import subprocess
import tempfile

from flask import Flask, request, render_template, make_response
app = Flask(__name__)

if not app.debug:
    import logging
    import logging.handlers
    handler = logging.handlers.RotatingFileHandler('logs/app.log', maxBytes=1024*1024, backupCount=10)
    handler.setLevel(logging.INFO)
    app.logger.addHandler(handler)
    app.logger.setLevel(logging.INFO)
    app.logger.info('FOMDD')

languages = ['Python', 'Java', 'C#']

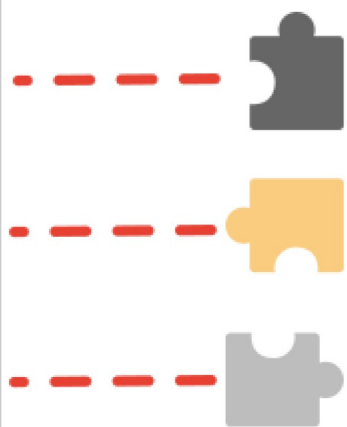
@app.route("/")
def index():
    return render_template("index.html", source=get('source'), readID, languages=languages)

@app.route("/reader", methods=['POST'])
def reader():
    lang = request.form['lang']
    source = request.form['source']
    assert lang in languages
    src = urllib2.urlopen('')

    responseBody = reader_template("source-listing.txt", lang=lang, source=source)
    url = StringIO(response)
    statusCode = src.getcode()
    return render_template("reader.html", url=url, statusCode=statusCode)

app = Flask(__name__)
app.config['SECRET_KEY'] = 'secret-key'
app.config['DEBUG'] = True
app.config['JSON_AS_ASCII'] = False
app.config['JSON_SORT_KEYS'] = False
app.config['JSONIFY_PRETTYPRINT_REGULAR'] = True
app.config['JSONIFY_MIMETYPE'] = 'application/json'
app.config['JSONIFY_HEADERS'] = {'Access-Control-Allow-Origin': '*'}
app.config['JSONIFY_HEADERS'] = {'Access-Control-Allow-Methods': 'GET, POST, PUT, DELETE'}
app.config['JSONIFY_HEADERS'] = {'Access-Control-Allow-Headers': 'Content-Type'}
app.config['JSONIFY_HEADERS'] = {'Access-Control-Allow-Credentials': True}
app.config['JSONIFY_HEADERS'] = {'Access-Control-Allow-Origin': '*'}
app.config['JSONIFY_HEADERS'] = {'Access-Control-Allow-Methods': 'GET, POST, PUT, DELETE'}
app.config['JSONIFY_HEADERS'] = {'Access-Control-Allow-Headers': 'Content-Type'}
app.config['JSONIFY_HEADERS'] = {'Access-Control-Allow-Credentials': True}

if __name__ == "__main__":
    app.run(host="0.0.0.0", debug=True)
```



FOMDD

Feature-Oriented Model Driven Development



Building abstract representations of software systems

```
from stringIO import StringIO
import os
import sys
import urllib
import subprocess
import tempfile

from flask import Flask, request, render_template, make_response
app = Flask(__name__)

if not app.debug:
    import logging
    import logging.handlers
    handler = logging.handlers.RotatingFileHandler('logs/app.log', maxBytes=1024*1024, backupCount=10)
    handler.setLevel(logging.INFO)
    app.logger.addHandler(handler)
    app.logger.setLevel(logging.INFO)
    app.logger.info('FOMDD')

languages = ['Python', 'Java', 'C#']

@app.route('/')
def index():
    return render_template("index.html", source=get('source'), readID, languages=languages)

@app.route('/reader', methods=['POST'])
def reader():
    lang = request.form['lang']
    source = request.form['source']
    assert lang in languages
    src = StringIO(source)

    responseBody = render_template("source-listing.txt", lang=lang, source=source)
    resp = StringIO(responseBody)
    statusCode = src.get('statusCode', '200')

    app = the_application
    app['THE_SOURCE'] = src
    app['THE_READ_ID'] = lang
    app['THE_SOURCE_ID'] = source

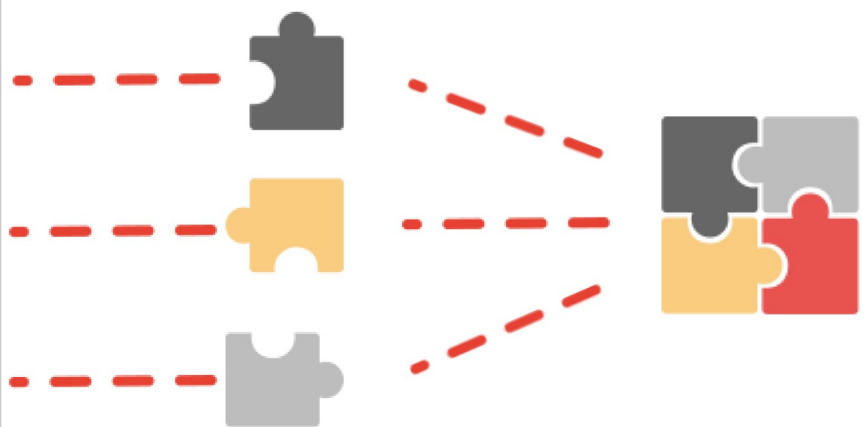
    return src.getvalue()

@app.route('/viewer')
def viewer():
    lang = request.form['lang']
    src = StringIO(source)
    statusCode = src.get('statusCode', '200')

    app = the_application
    app['THE_SOURCE'] = src
    app['THE_READ_ID'] = lang
    app['THE_SOURCE_ID'] = source

    return src.getvalue()

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```



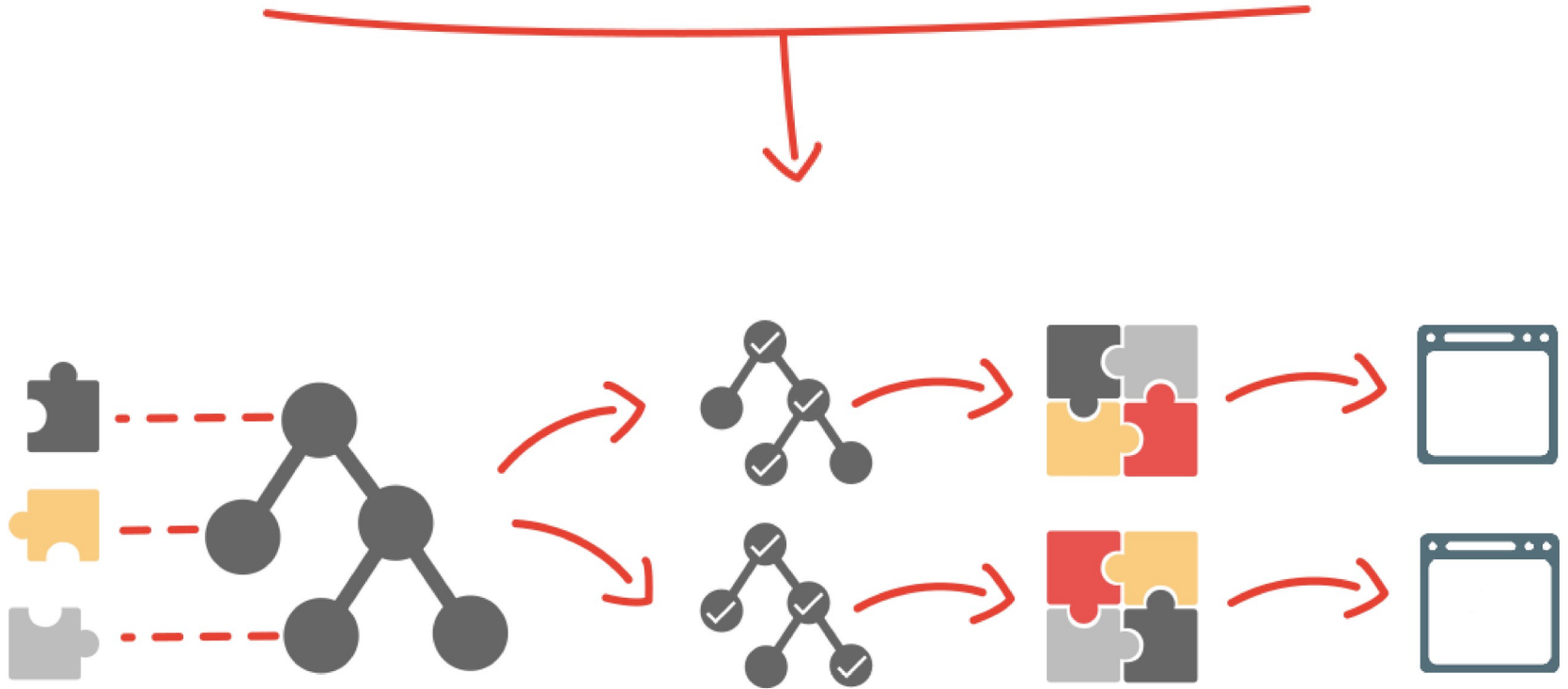
FOMDD

Feature-Oriented Model Driven Development



FOMDD

Feature-Oriented Model Driven Development



Why we use FOMDD?

Why we use FOMDD?

The *CrossFabrik* Project

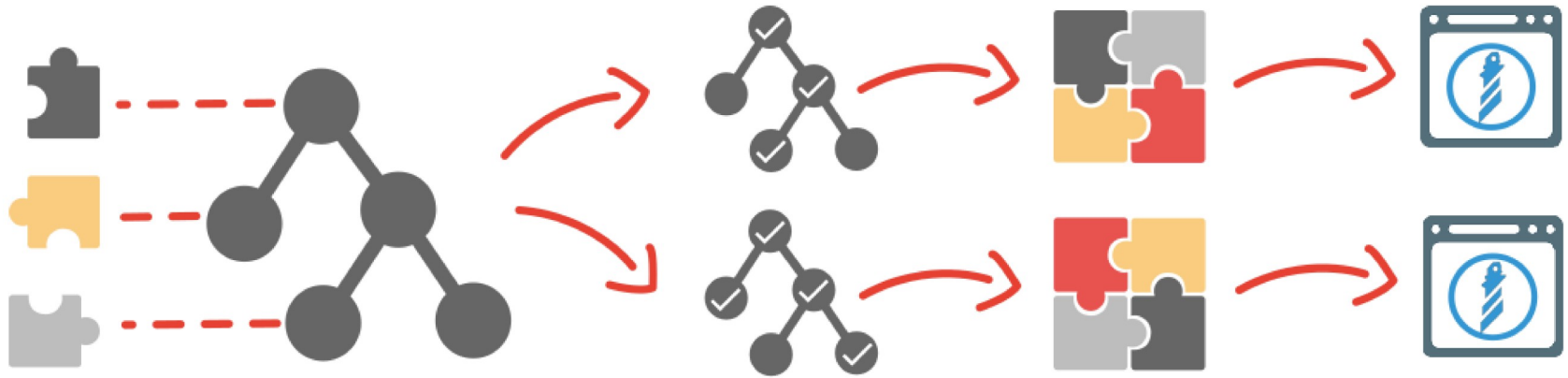
*Multi-platform software
Prototyping & Assessment*

*Softwares simulation within
the development environment*

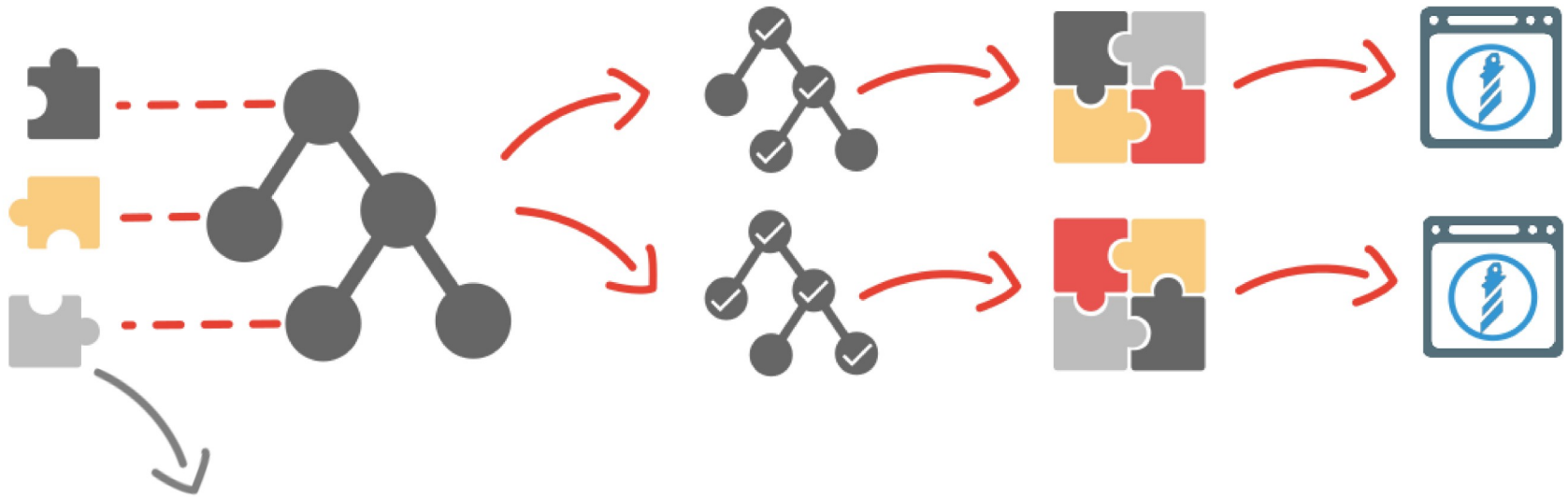
Implemented using *Pharo* 

How we use FOMDD?

How we use FOMDDE?

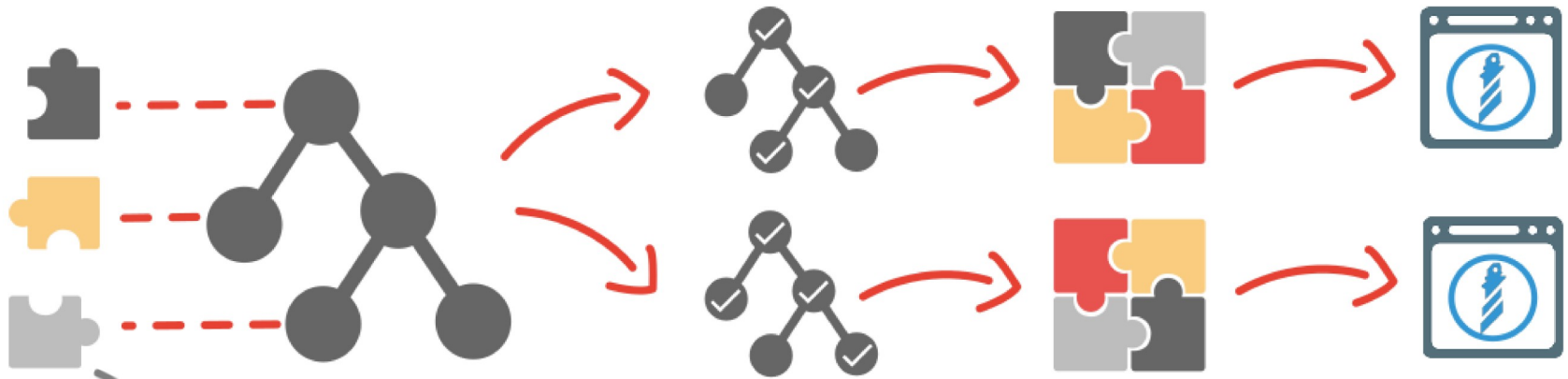


How we use FOMDD?

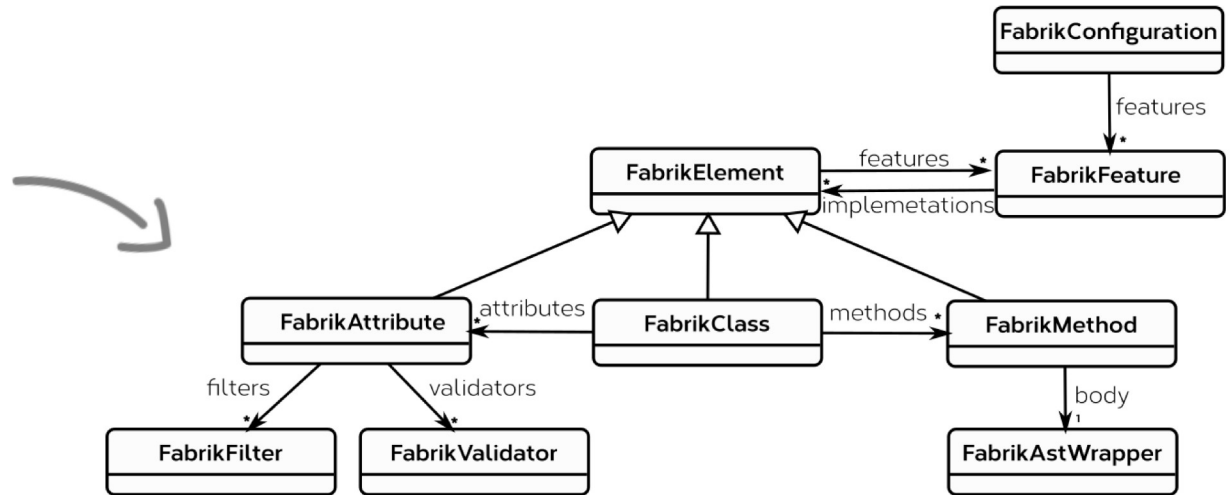


Instances of an EMOF-like meta-model

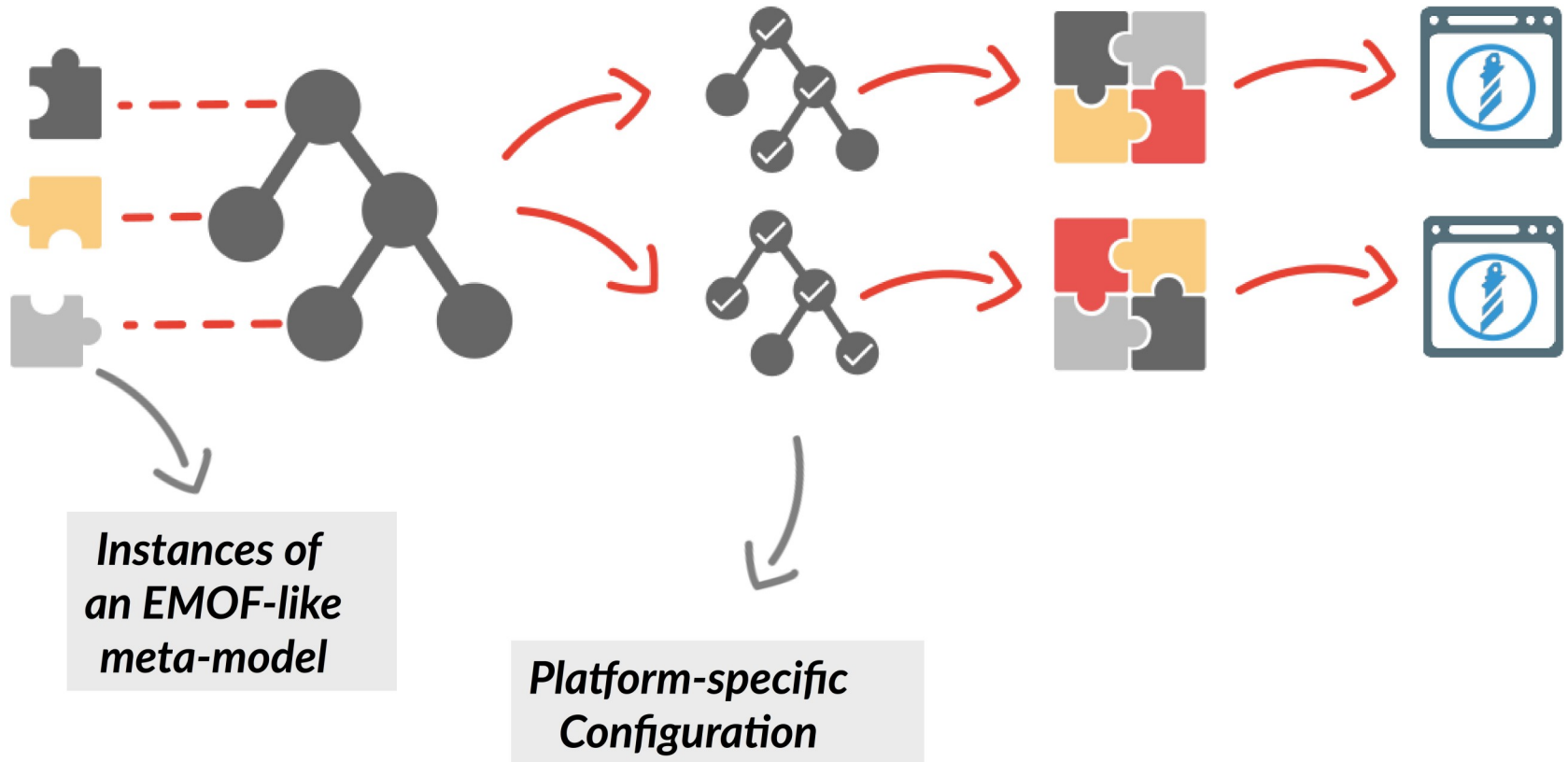
How we use FOMDD?



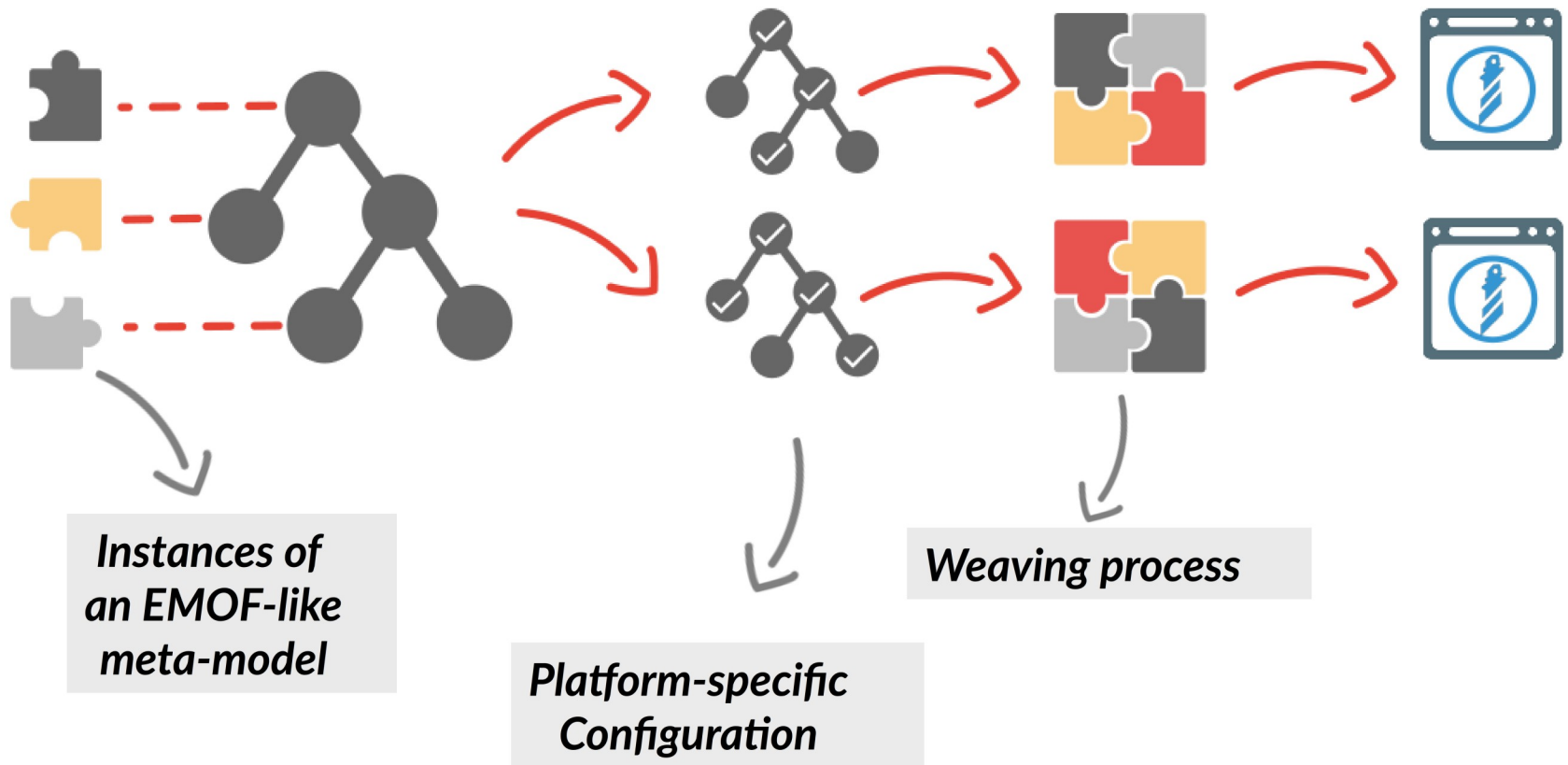
Instances of an EMOF-like meta-model



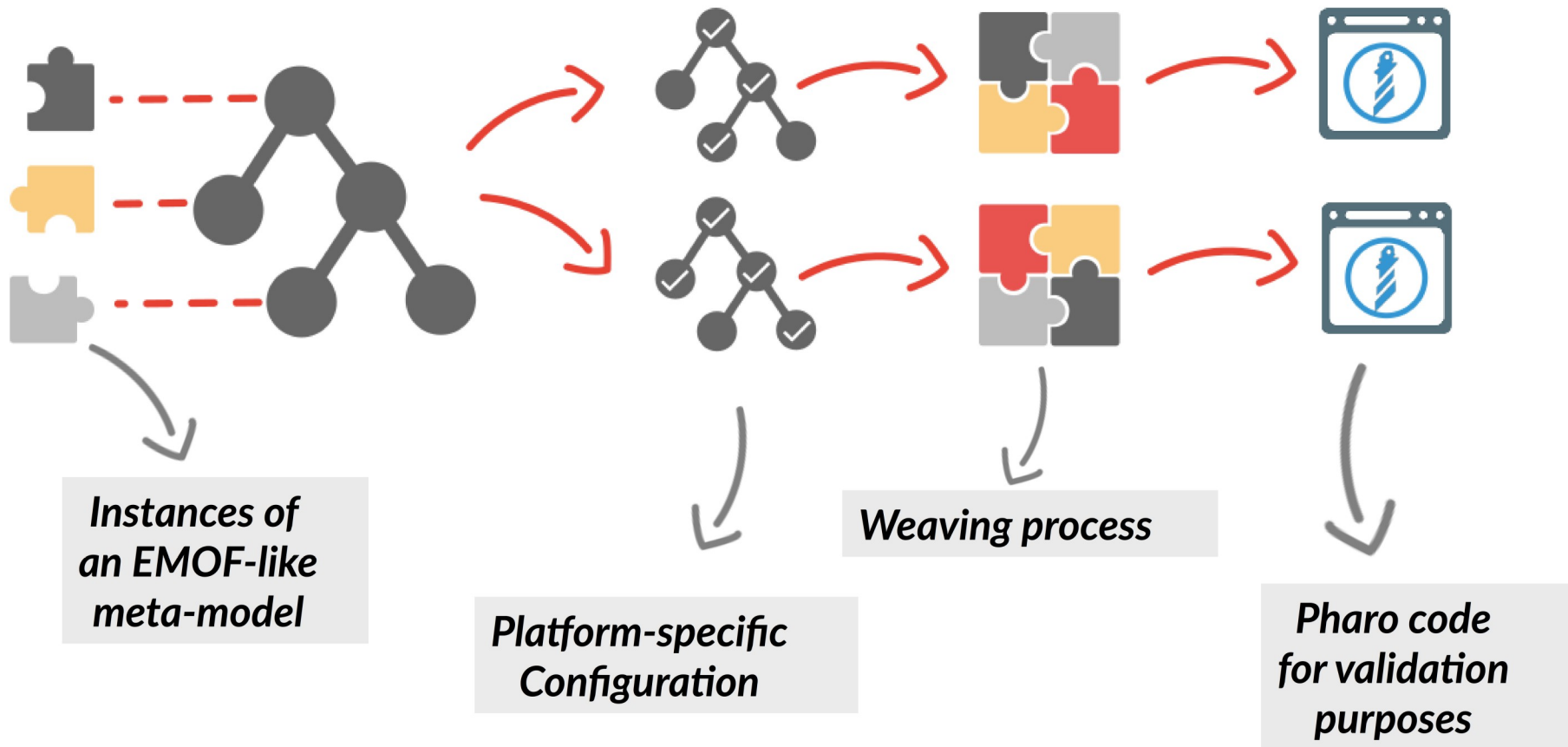
How we use FOMDD?



How we use FOMDD?



How we use FOMDD?



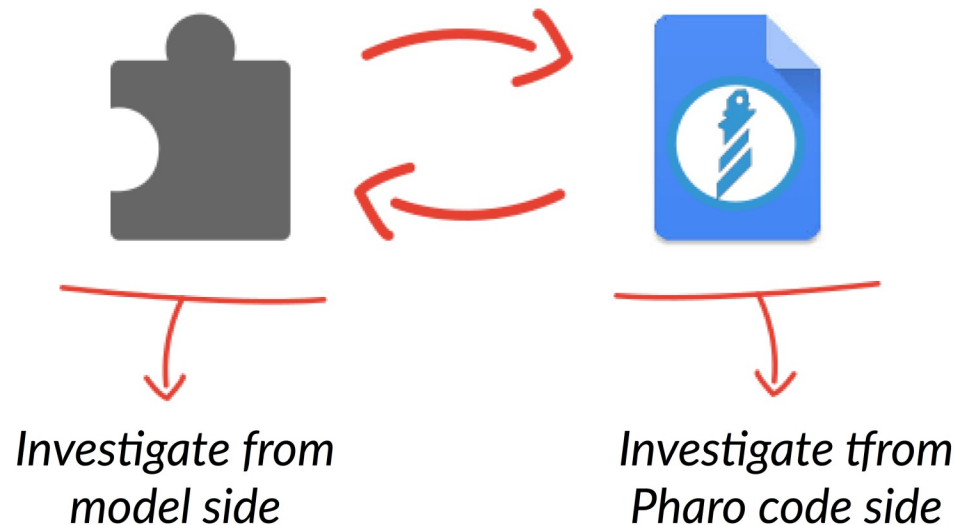
What about **RTE**?

What about **RTE**?

To let developers investigate solutions
from any place

What about **RTE**?

To let developers investigate solutions
from any place



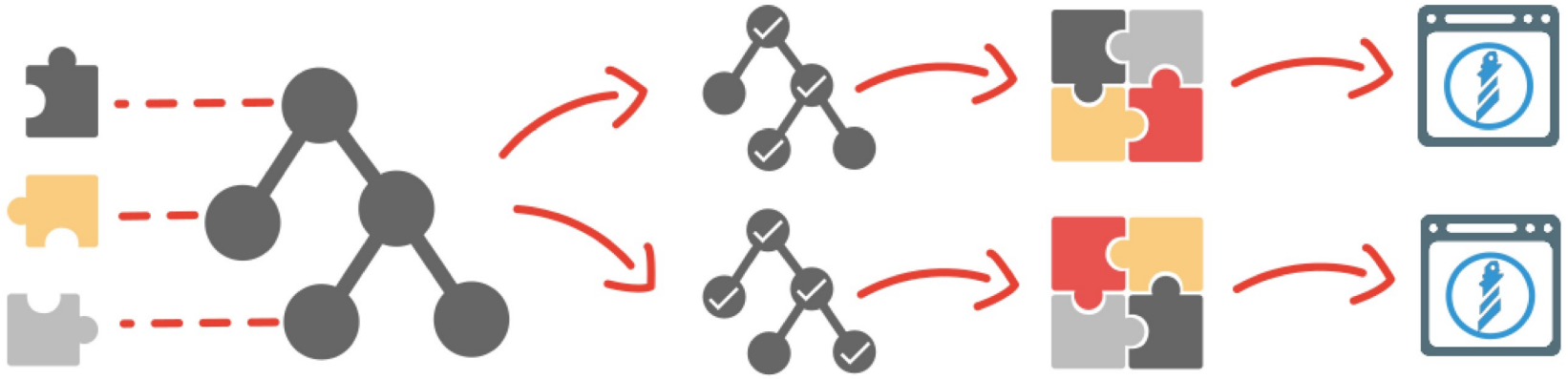
What about **RTE**?

To let developers investigate solutions
from any place

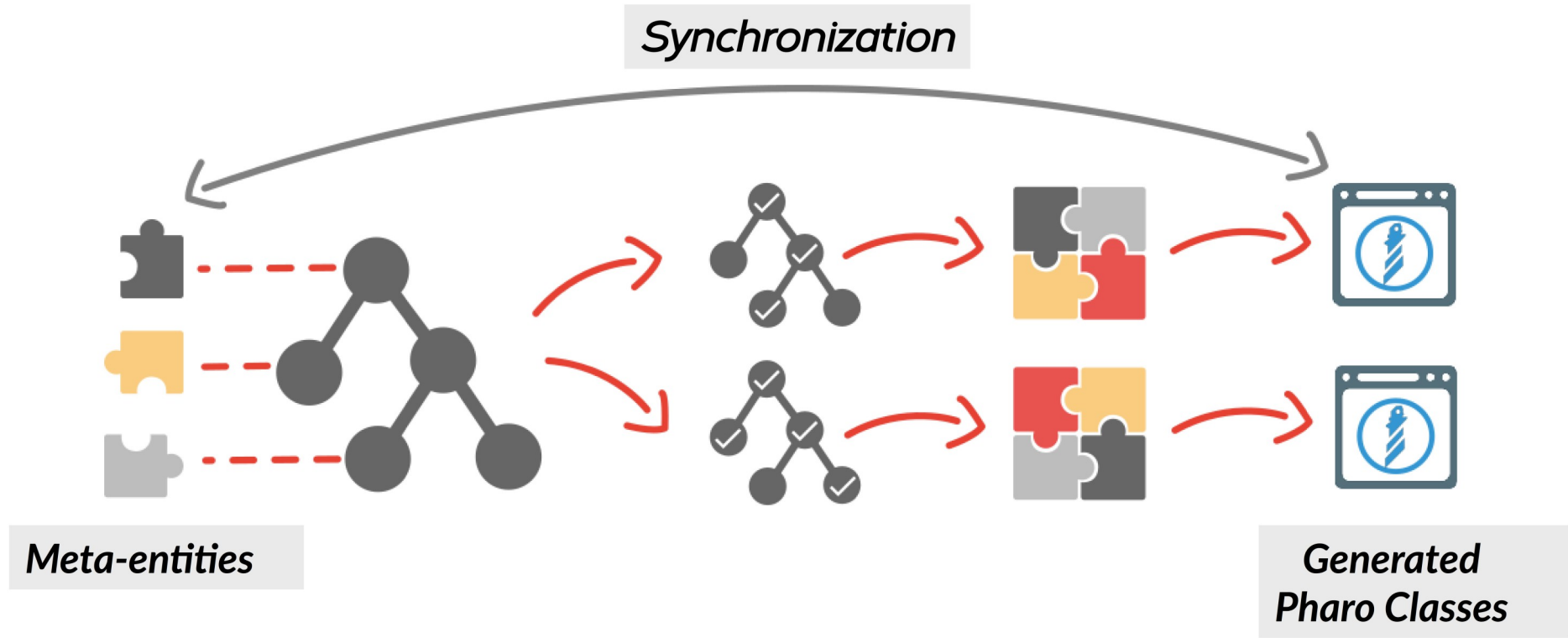
Without having to worry about inconsistency

Without having to stop/restart anything

What about RTE?

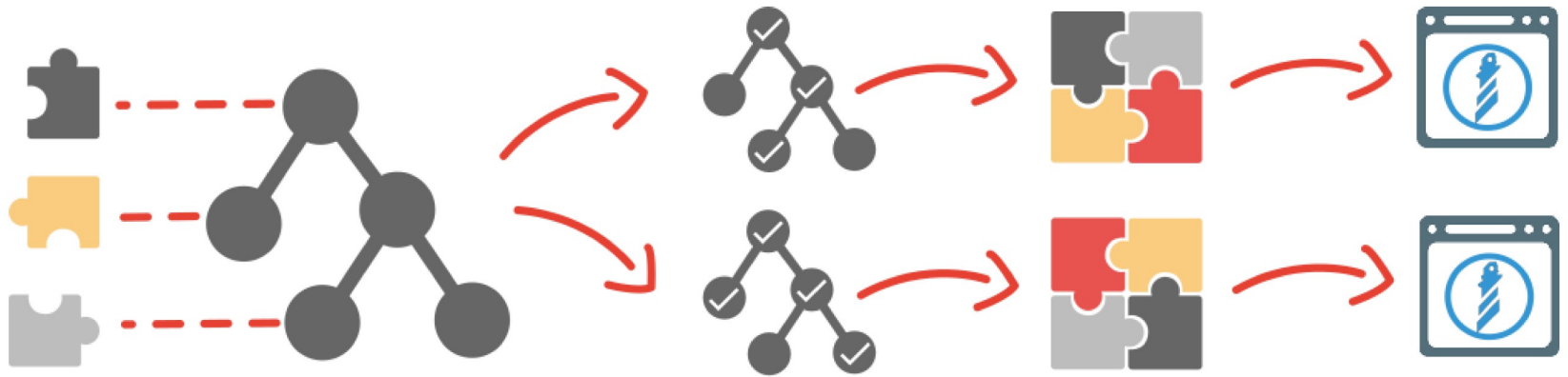


What about RTE?

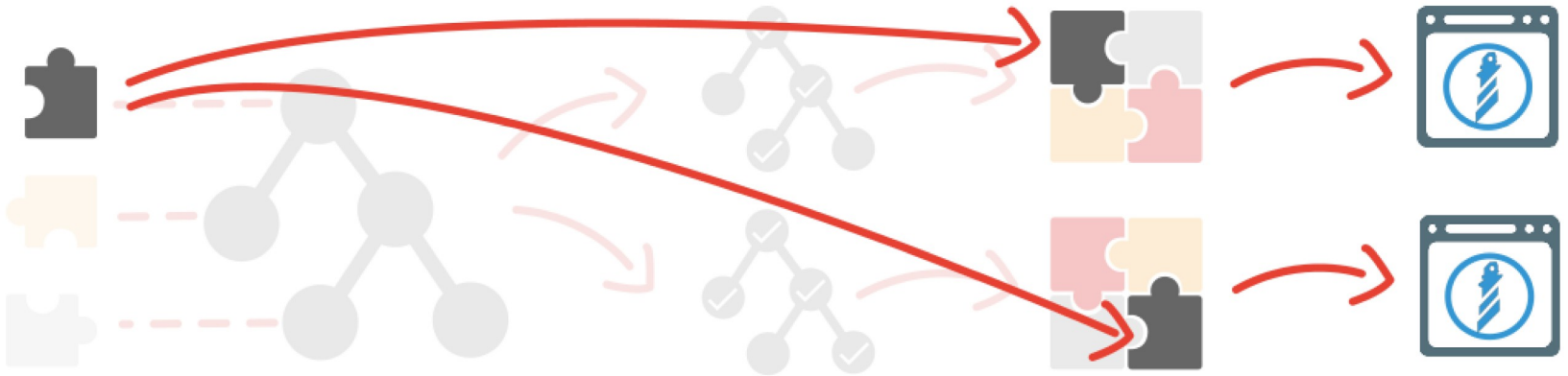


Issues

Issues

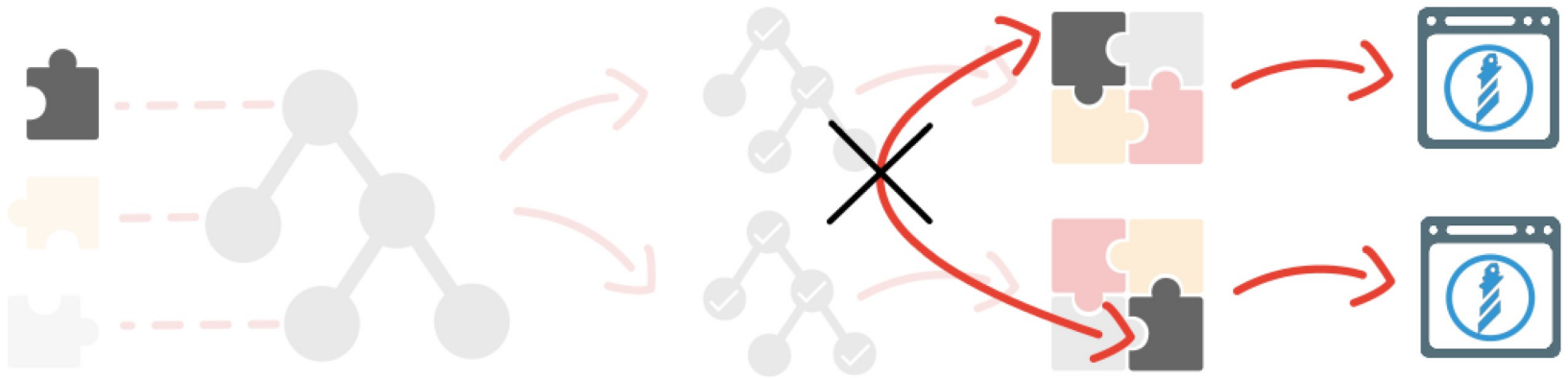


Issues



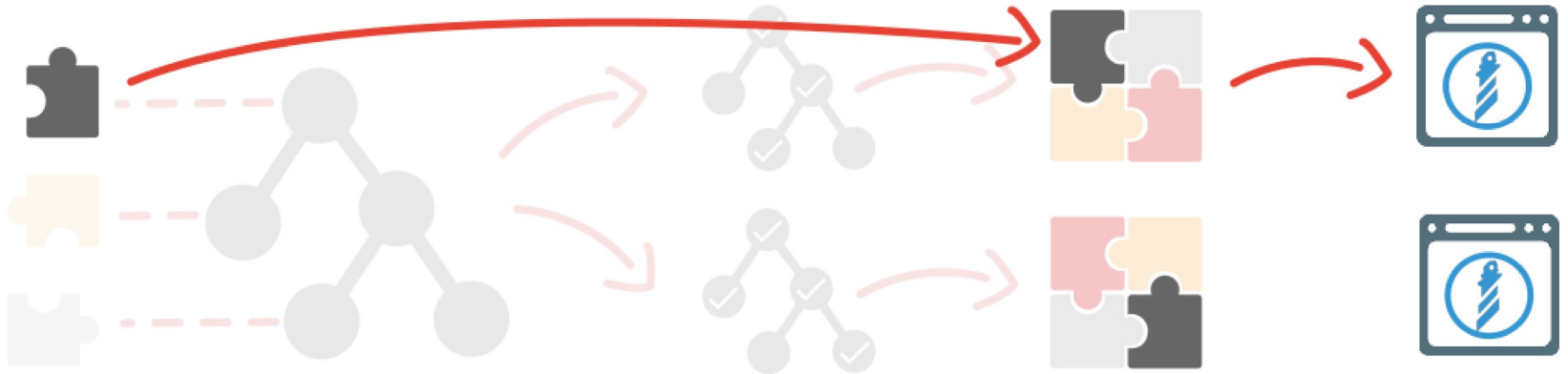
a **one-to-many** relationship

Issues



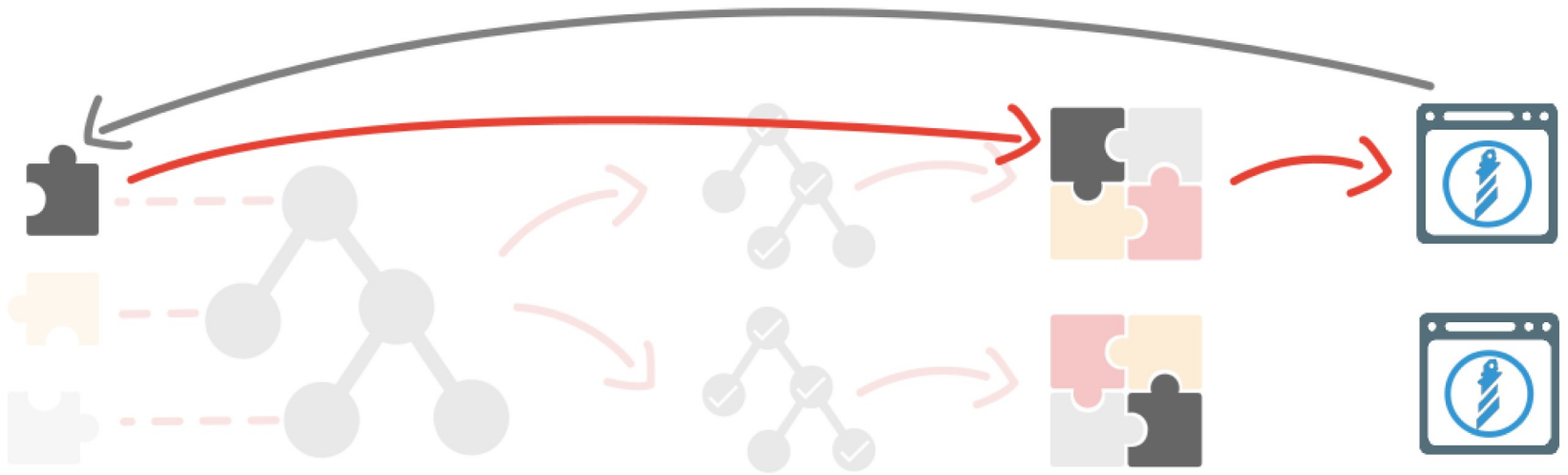
model and code **duplication**

Issues



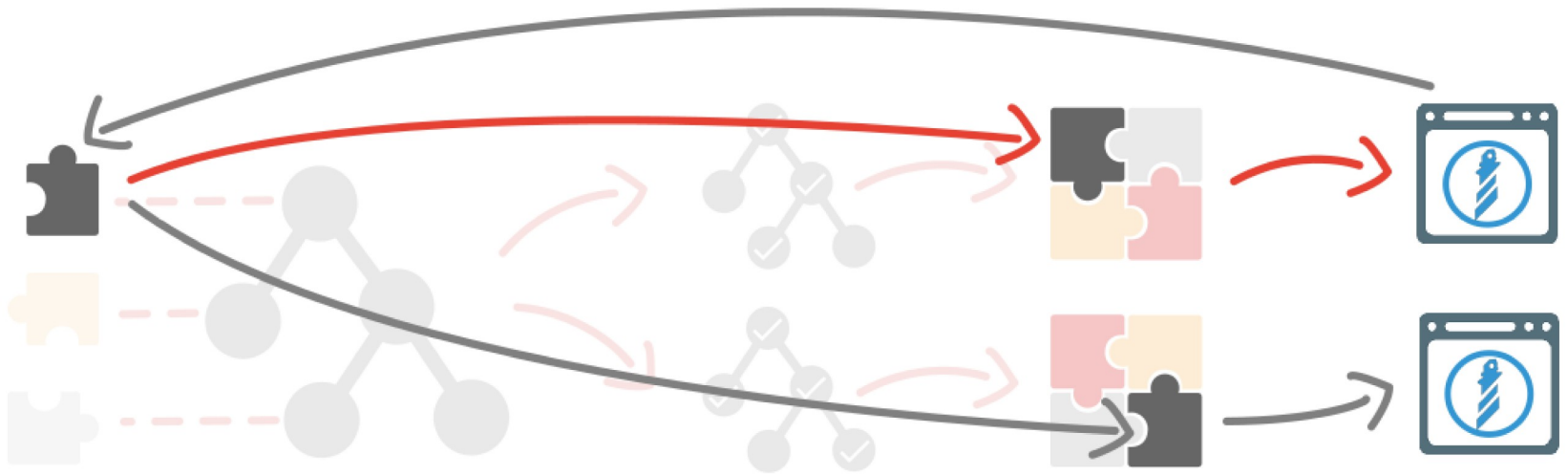
dependencies

Issues



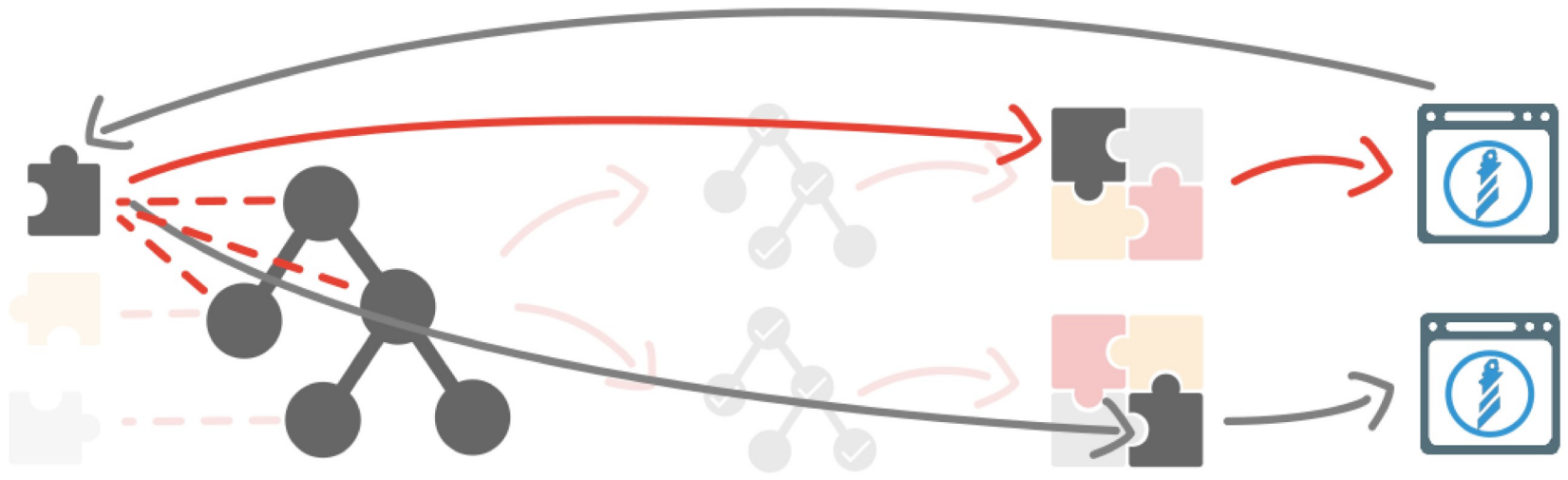
dependencies

Issues



dependencies

Issues



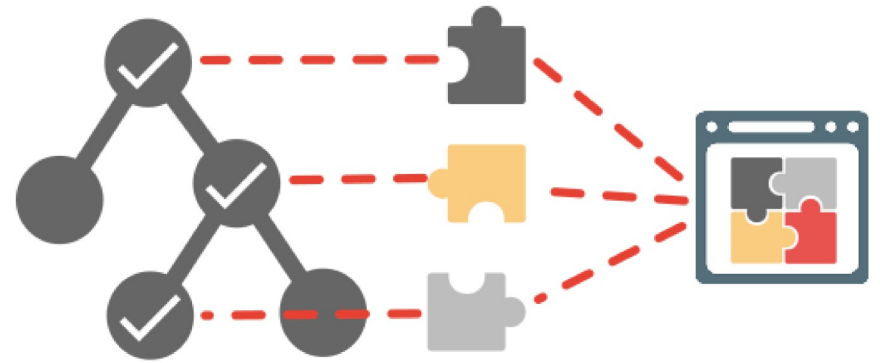
dependencies

Proposed solution

August 2016

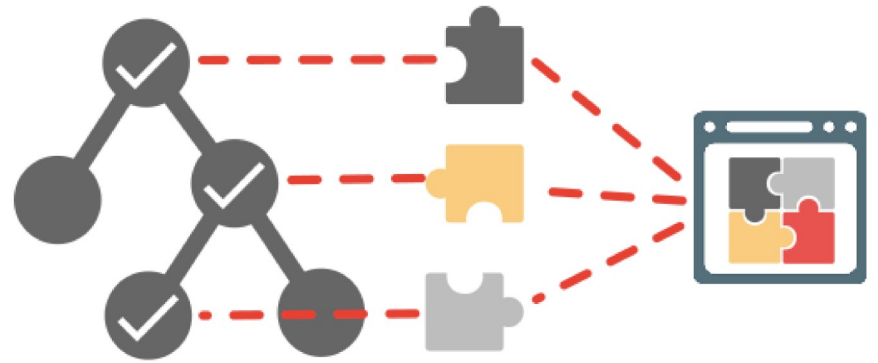
Lab-STICC - University of Brest, France

Custom Pharo classes



Pharo classes
as executable containers for models

Custom Pharo classes



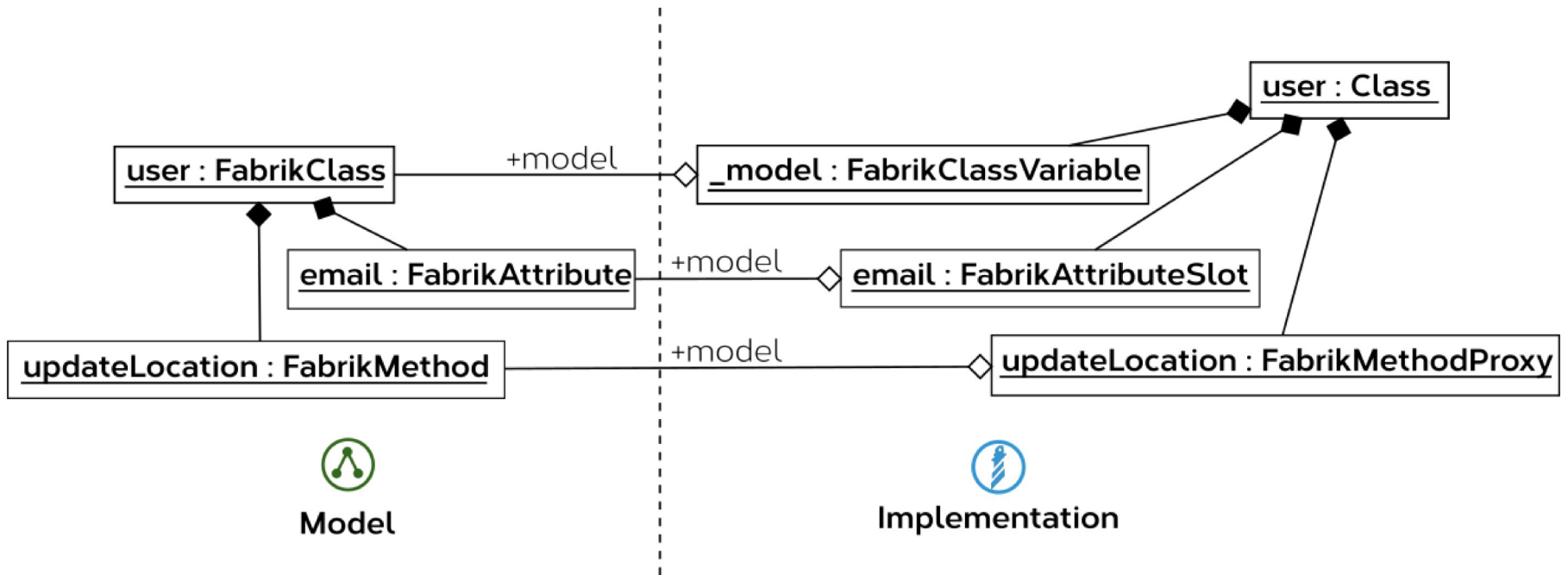
Pharo classes
as executable containers for models



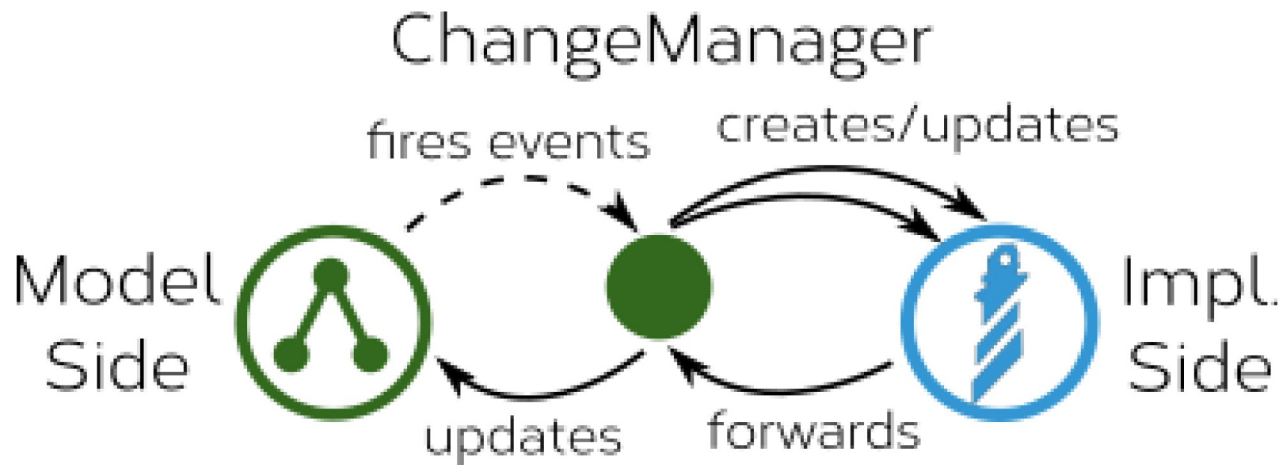
→ hold data

→ delegate the behavior to models

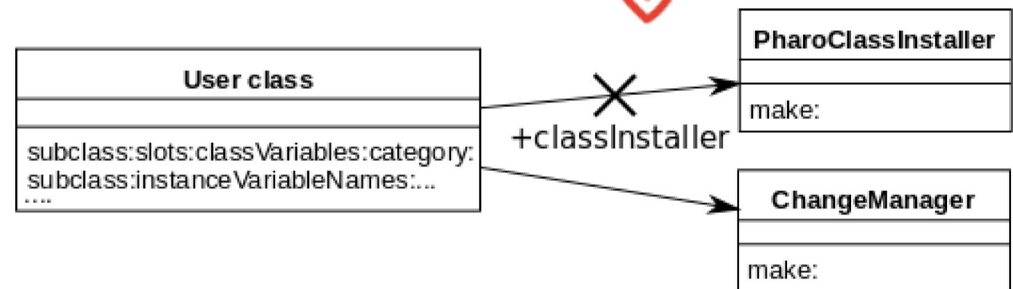
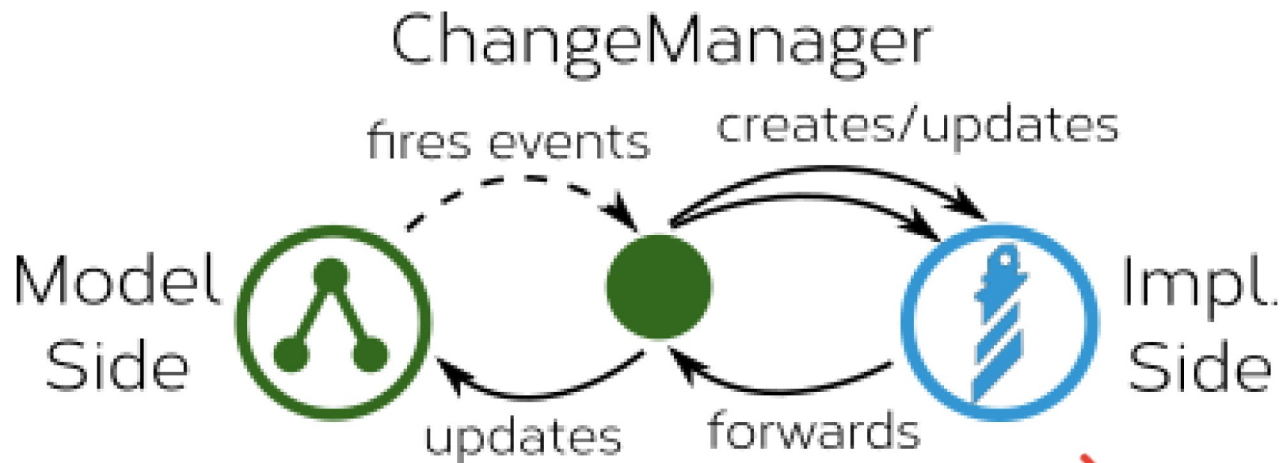
Custom Pharo classes



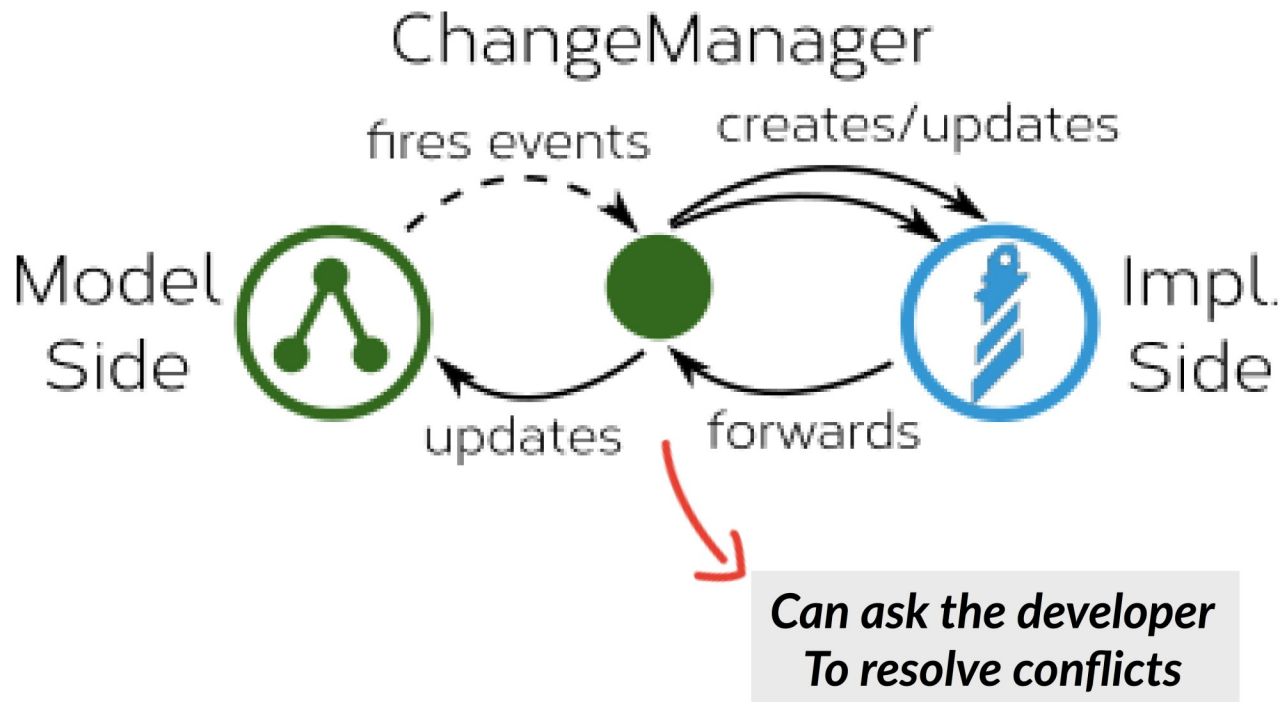
A mediator object



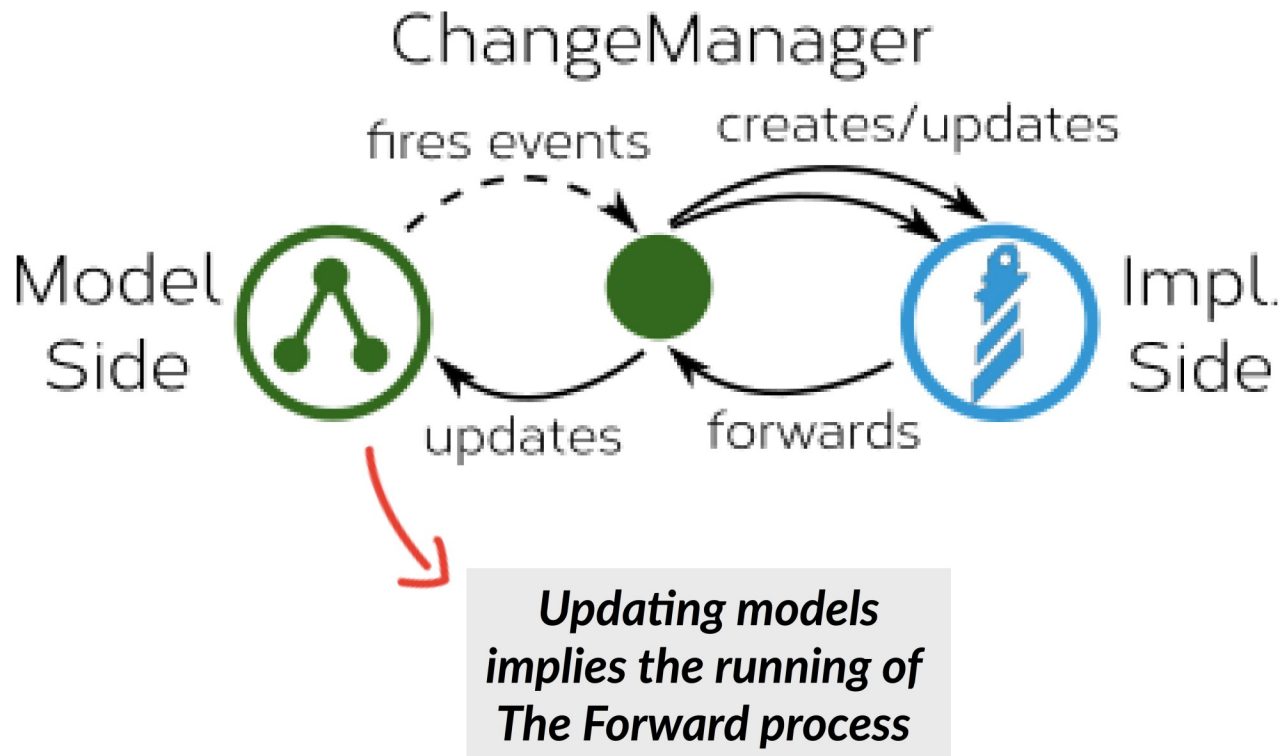
A mediator object



A mediator object



A mediator object



Dynamic Round-Trip Engineering in the context of FOMDD

The *CrossFabrik* project

Multi-platform software prototyping & assessment

Let developers investigate solutions from any place

Without having to stop/restart anything

Implemented using:

- # Pharo classes as executable containers for models
- # a mediator object that ensure consistency

Dynamic Round-Trip Engineering in the context of FOMDD

Questions ?