17-708 SOFTWARE PRODUCT LINES: CONCEPTS AND IMPLEMENTATION

GENERATORS, DSLS, MDD

CHRISTIAN KAESTNER
CARNEGIE MELLON UNIVERSITY
INSTITUTE FOR SOFTWARE RESEARCH

READING ASSIGNMENT NOV 16

No class on Wednesday!

Garvin, Brady J., and Myra B. Cohen. "Feature interaction faults revisited: An exploratory study." In *Software Reliability Engineering (ISSRE), 2011 IEEE 22nd International Symposium on*, pp. 90-99. IEEE, 2011.

Tartler, Reinhard, Daniel Lohmann, Christian Dietrich, Christoph Egger, and Julio Sincero. "Configuration coverage in the analysis of large-scale system software." In *Proceedings of the 6th Workshop on Programming Languages and Operating Systems*, p. 2. ACM, 2011.

LEARNING GOALS

Explain the difference between classic product-line variability and variability supported in generators

Understand mechanisms to generate code, beyond composing or removing pieces based on boolean decisions

Understand the relevance of domain-specific notations

Gain an overview of different implementation strategies at different binding times

FINITE CONFIGURATION SPACES?

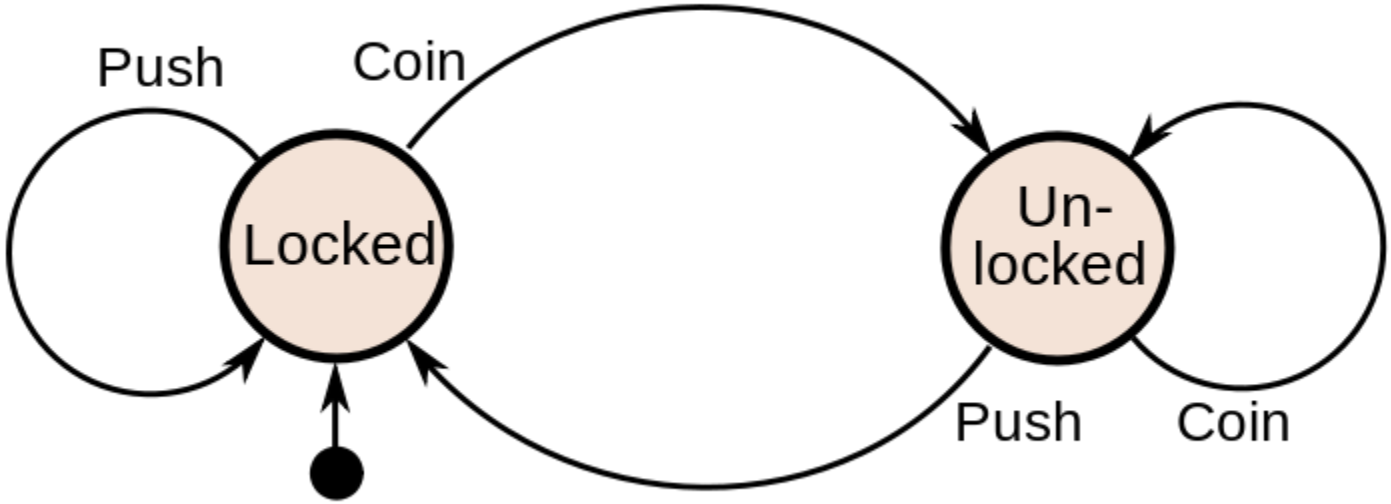
Enabling/disabling features

Parameters/attributes in feature models

Further customization

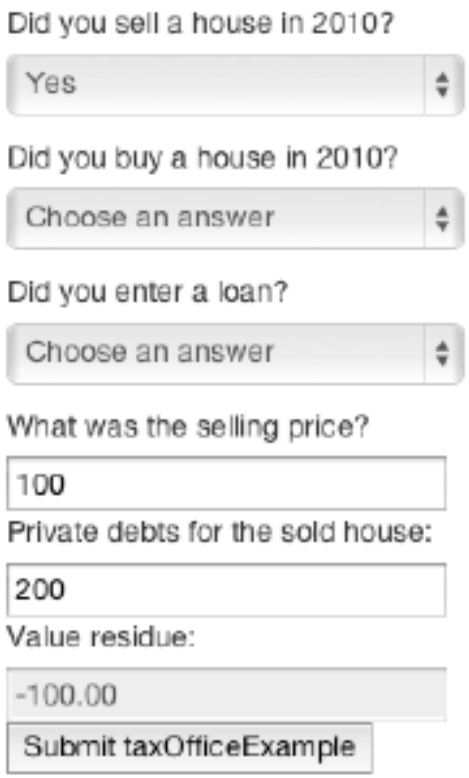
- **Logos, pictures**
- **Workflows, algorithms, menus**
- **…**

```
if the patient
  has a Weight Height ratio of less than 46
    and
  a cholesterol LDL level below 100 and does not take LDL medications
    and
  the systolic blood pressure level is less than 125
            and does not take blood pressure medication
    and
  the hemoglobin A1c test is equal or greater than 6.5
            and does not take glucose medication
then <advice according to diabetes plan>.
```



https://en.wikipedia.org/wiki/Finite-state_machine

```
form taxOfficeExample {
  "Did you sell a house in 2010?"
    boolean hasSoldHouse
  "Did you buy a house in 2010?"
    boolean hasBoughtHouse
  "Did you enter a loan?"
    boolean hasMaintLoan
  if (hasSoldHouse) {
    "What was the selling price?"
      money sellingPrice
    "Private debts for the sold house:"
      money privateDebt
    "Value residue:"
      money valueResidue =
        (sellingPrice - privateDebt)
  }
}
```



Erdweg, Sebastian, Tijs van der Storm, Markus Völter, Meinte Boersma, Remi Bosman, William R. Cook, Albert Gerritsen et al. "The state of the art in language workbenches." In *Software language engineering*, pp. 197-217. Springer International Publishing, 2013.

- Open GL for high level 3D graphics
- Postscript for low level graphics
- VHDL for hardware description
- Lex and Yacc for lexing and parsing
- Latex for document layout
- HTML for document markup

FRAMEWORKS/ COMPONENTS

Extension points with custom implementations

GENERATORS

Varis.php

```php
<?php
include("header.php");

echo '<form method="' . $_GET['method'] . '" name="searchform">';
if ($ajax)
      $input = '<input ... onkeyup="update()" />';
else
      $input = '<input ... onkeyup="update()" />' .
               '<input type="submit" />';
echo $input;
echo '</form>';
?>

<script type="text/javascript">
<?php if ($ajax) { ?>
      function update() { ...
      }
      </script>
<?php } else { ?>
      function update() { ...
      }
      </script>
<?php } ?>

<?php include("footer.php"); ?>
```

METAPROGRAMMING

```
#!/bin/sh
echo '#!/bin/sh' >program
for I in $(seq 992)
do
  echo "echo $I" >> program
done
chmod +x program
```

https://en.wikipedia.org/wiki/Metaprogramming

DOMAIN-SPECIFIC LANGUAGES

target platform

internal vs external

interpreter vs compiler

concrete syntax, semantic

technical DSLs vs application domain DSLs

	GPLs	DSLs
Domain	large and complex	smaller and well-defined
Language size	large	small
Turing completeness	always	often not
User-defined abstractions	sophisticated	limited
Execution	via intermediate GPL	native
Lifespan	years to decades	months to years (driven by context)
Designed by	guru or committee	a few engineers and domain experts
User community	large, anonymous and widespread	small, accessible and local
Evolution	slow, often standardized	fast-paced
Deprecation/incompatible changes	almost impossible	feasible

Source and suggested further reading: M. Voelter. DSL engineering. 2013

MODEL-DRIVEN (SOFTWARE) DEVELOPMENT (MDD)

Prescriptive, not descriptive

Executable (domain-specific) models / generators

Often graphical notation

(Compare UML)

	Modeling	Programming
Define your own notation/language	Easy	Sometimes possible to some extent
Syntactically integrate several langs	Possible, depends on tool	Hard
Graphical notations	Possible, depends on tool	Usually only visualizations
Customize generator/compiler	Easy	Sometimes possible based on open compilers
Navigate/query	Easy	Sometimes possible, depends on IDE and APIs
View Support	Typical	Almost Never
Constraints	Easy	Sometimes possible with Findbugs etc.
Sophisticated mature IDE	Sometimes, effort-dependent	Standard
Debugger	Rarely	Almost always
Versioning, diff/merge	Depends on syntax and tools	Standard

DOMAIN SPECIFIC!

Generative programming

Domain analysis

Domain-specific generation

Domain-specific analysis

Domain-specific optimization

```
fountain
  basin sensor s
        nozzle n1
        nozzle n2
  pump p
  if s.full && p.rpm > 0 then p.rpm = 0
```

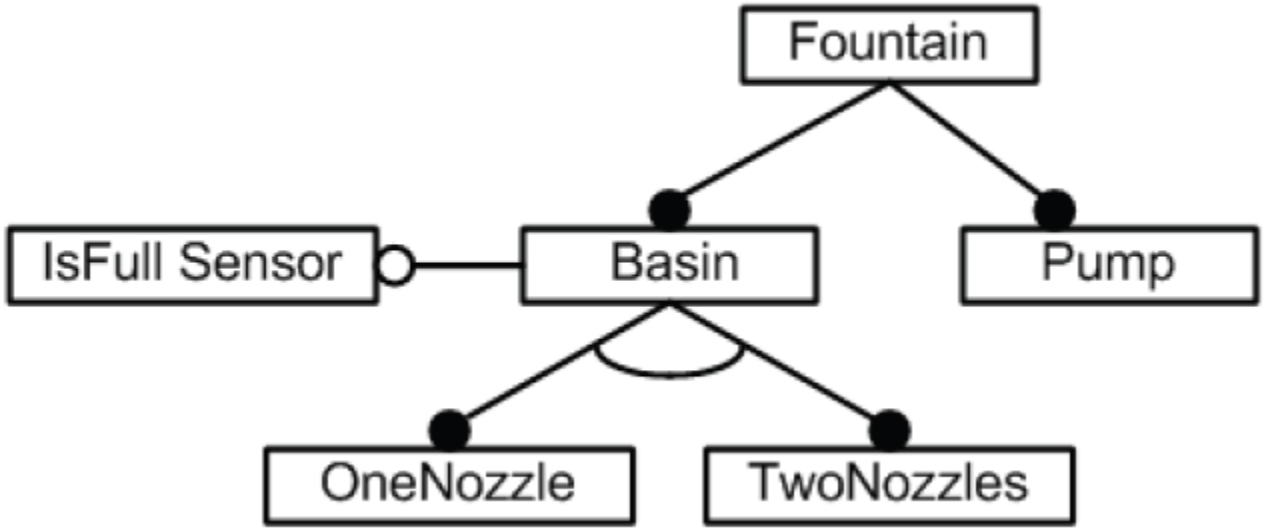
```
Fountain               -> "fountain" Basin Pump Behavior
Basin                  -> "basin" IsFullSensor Nozzle*
Behavior               -> Rule*

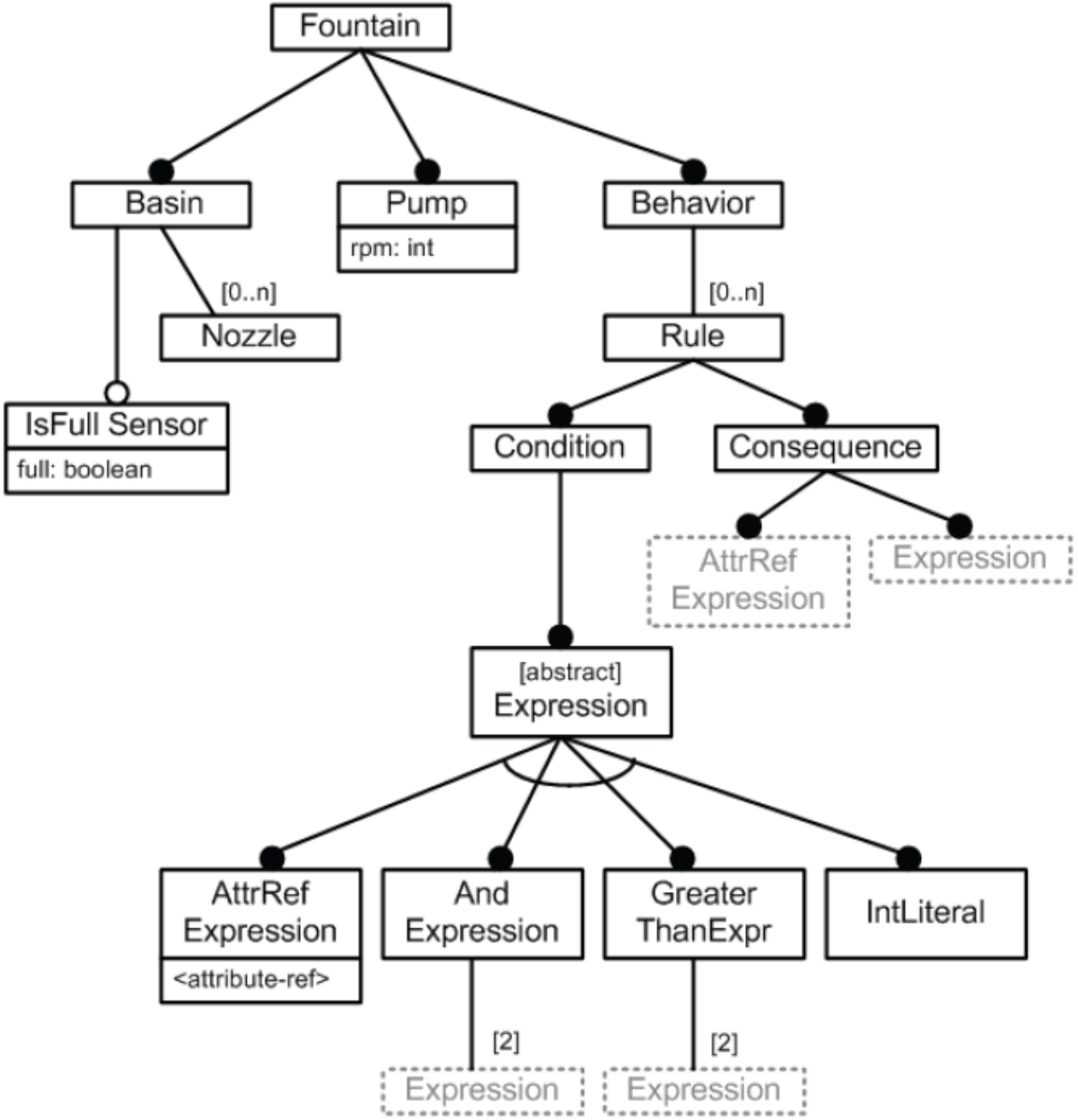
Rule                   -> "if" Condition "then" Consequence
Condition              -> Expression
Expression             -> AttrRefExpression | AndExpression |
                          GreaterThanExpression | IntLiteral;

AndExpression          -> Expression "&&" Expression
GreaterThanExpression  -> Expression ">" Expression
AttrRefExpression      -> <attribute-ref-by-name>
IntLiteral             -> (0..9)*

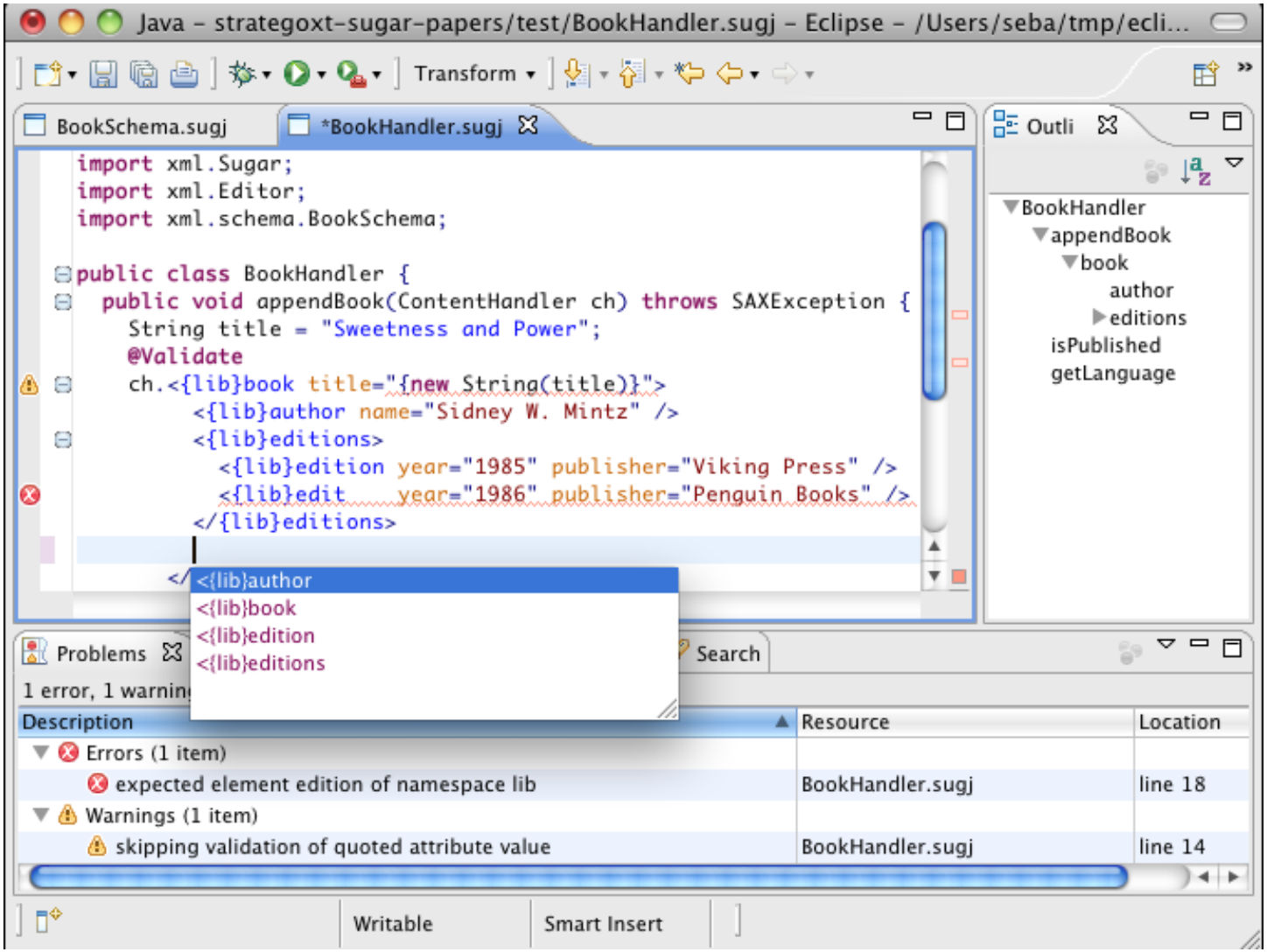
Consequence            -> AttrRefExpression "=" Expression

IsFullSensor           -> "sensor" ID (full:boolean)?
Nozzle                 -> "nozzle" ID
Pump                   -> "pump" ID (rpm:int)?
```


Fountain
IsFull Sensor
Basin
Pump
OneNozzle
TwoNozzles

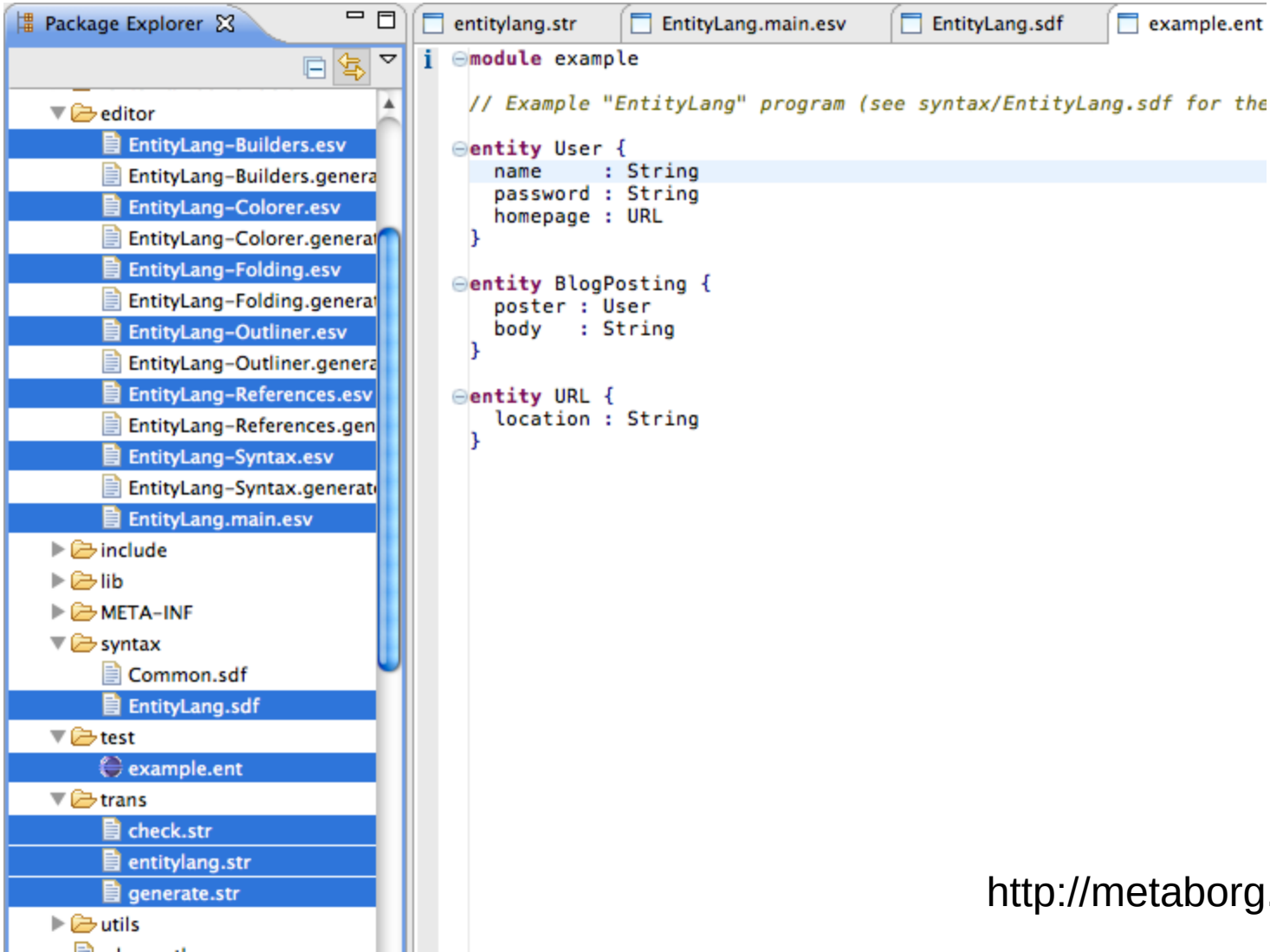

Fountain
Basin
Pump
rpm: int
Behavior
[0..n]
Nozzle
[0..n]
Rule
IsFull Sensor
full: boolean
Condition
Consequence
AttrRef Expression
Expression
[abstract] Expression
AttrRef Expression
<attribute-ref>
And Expression
Greater ThanExpr
IntLiteral
[2]
Expression
[2]
Expression

EXTENSIBLE LANGUAGES



sugarj.org

LANGUAGE WORKBENCHES



http://metaborg.org/spoofax/



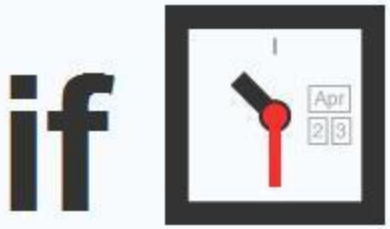
then



Send ifttt any text message from [redacted]

Create a link note in your notes

created about 1 hour ago
last triggered 41 minutes ago
triggered 1x



then



Every month on the 3rd at 02:00 PM

Add file from URL to Josh Haas's dropbox

created about 1 hour ago
last triggered Never



then



Send trigger@ifttt.com an email tagged [redacted] from [redacted]

Send me a text message at [redacted]

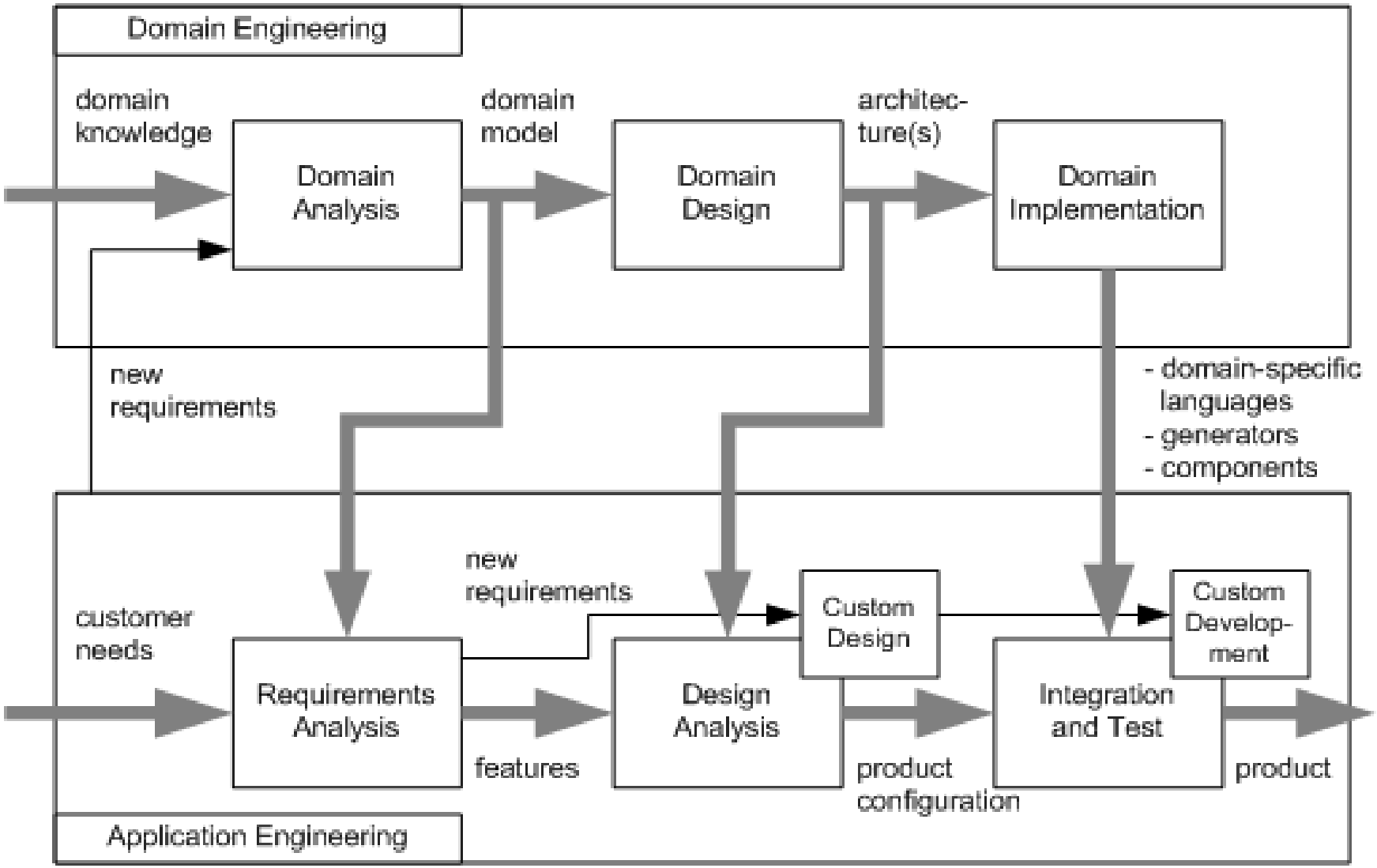
created about 2 hours ago
last triggered about 2 hours ago
triggered 4x

source: http://blog.joshhaas.com/2011/10/self-experimentation-using-ifttt-and-a-dash-of-python/

XVCL

```
<x-frame name="Notepad">
import java.awt.*;
class Notepad extends JPanel {
    Notepad() {
        super();
        …
    }
    public static void main(String[] args) {
        JFrame frame = new JFrame();
        frame.setTitle("<value-of expr="?@TITLE?"/>");
        frame.setBackground(
            Color.<value-of expr="?@BGCOLOR?"/>);
        frame.show();
    }
    <adapt x-frame="Editor.XVCL"/>
    <adapt x-frame="Menubar.XVCL"/>
    <adapt x-frame="Toolbar.XVCL"/>
    …
}
</x-frame>
```

```
<x-frame name="Toolbar">
<set-multivar="ToolbarBtns" value="New,Open,Save"/>
private Component createToolbar() {
      JToolBar toolbar = new JToolBar();
      JButton button;
<while using-items-in="ToolbarBtns">
  <select option="ToolbarBtns">
      <option value="-">
      toolbar.add(Box.createHorizontalStrut(5));
      </option>
      <otherwise>
      button = new JButton(new ImageIcon(
          "<value-of expr="?@Gif@ToolbarBtns?"/> "));
      toolbar.add(button);
      </otherwise>
  </select>
</while>
      toolbar.add(Box.createHorizontalGlue());
      return toolbar;
}
</x-frame>
```


Domain Engineering
domain knowledge
Domain Analysis
domain model
Domain Design
architec-ture(s)
Domain Implementation
new requirements
- domain-specific languages
- generators
- components
customer needs
Requirements Analysis
new requirements
Custom Design
Design Analysis
Custom Develop-ment
Integration and Test
features
product configuration
product
Application Engineering

MBEDDR

Unit types

```
unit V :=       for voltage
unit A :=       for Amps
unit Ω := V·A^-1 for resistance

uint16/Ω/ resistance(uint16/V/ u, uint16/A/[] i, uint8 ilen) {
    uint16/A/ avg_i = (sum p = 0 .. ilen : i[p]) / ilen;
    return avg_i / u;
}
```

Error: type uint16 /V^(−1) · A/ is not a subtype of uint16 /Ω/

resistance (function)

Voelter, Markus, et al. "Using C language extensions for developing embedded software: a case study." *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*. ACM, 2015.

MBEDDR

state machine extension

```
statemachine FrameParser initial = idle {
  var uint8 idx = 0
  in event dataReceived(uint8 data)
  state idle {
    entry { idx = 0; }
    on dataReceived [data == LEADING_BYTE] -> wakeup
  }
  state wakeup {
    on dataReceived [data == START_FLAG]
       -> receivingFrame { idx++; }
  }
  state receivingFrame { .. }
}
```

Voelter, Markus, et al. "Using C language extensions for developing embedded software: a case study." *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*. ACM, 2015.