17-708 SOFTWARE PRODUCT LINES: CONCEPTS AND IMPLEMENTATION

GENERATORS, DSLS, MDD

CHRISTIAN KAESTNER
CARNEGIE MELLON UNIVERSITY
INSTITUTE FOR SOFTWARE RESEARCH
No class on Wednesday!


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>.
Finite-state machine

Locked → Unlocked
Push → Coin

Locked → Unlocked
Push → Coin
• 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

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

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"); ?>
```
#!/bin/sh

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**

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

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)

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

Basin

Pump
  rpm: int

Nozzle

[0..n]

Behavior

Rule

IsFull Sensor
  full: boolean

Condition

Consequence

AttrRef Expression
  Expression

And Expression

Greater ThanExpr

IntLiteral

[abstract] Expression

Attribute-Ref

Expression

[2]

Expression

[2]

Expression
EXTENSIBLE LANGUAGES
// Example "EntityLang" program (see syntax/EntityLang.sdf for the

entity User {
    name : String
    password : String
    homepage : URL
}

entity BlogPosting {
    poster : User
    body : String
}

entity URL {
    location : String
}
if SMS then Create a link note in your notes

Send ifttt any text message from

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

if Clock then Add file from URL to Josh Haas's dropbox

Every month on the 3rd at 02:00 PM

created about 1 hour ago
last triggered Never

if Email then Send me a text message at

Send trigger@ifttt.com an email tagged from

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

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();
    }

    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?"/>")
                        .getImageIcon());
                    toolbar.add(button);
                </otherwise>
            </select>
        </while>
        toolbar.add(Box.createHorizontalGlue());
        return toolbar;
    }

    ...
</x-frame>
Domain Engineering

- Domain knowledge
  - Domain Analysis
    - Domain model
  - Domain Design
    - Architecture(s)
  - Domain Implementation
    - Domain-specific languages
    - Generators
    - Components

Application Engineering

- Customer needs
  - Requirements Analysis
    - New requirements
  - Design Analysis
    - New requirements
      - Features
  - Custom Design
    - Product configuration
  - Integration and Test
    - Product
MBEDDR

Unit types

```c
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];
    return \frac{avg_i}{u};
    // Error: type uint16 /V\wedge(-1) \cdot A/ is not a subtype of uint16 /Ω/
}
```

MBEDDR

state machine extension

```c
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 { .. }
}
```