Software Variability and Artificial Intelligence

Mathieu Acher (Associate Professor)

https://www.mathieuacher.com
https://teaching.variability.io
https://varyvary.github.io/

@acherm
Disclaimer

• Slides for the EJCP 2019 course
  – ~French summer school for PhD candidates in programming, verification, software engineering, etc.

  • Abstract: Most modern software systems are subject to variation or come in many variants. Web browsers like Firefox or Chrome are available on different operating systems, in different languages, while users can configure 2000+ preferences or install numerous 3rd parties extensions (or plugins). Web servers like Apache, operating systems like the Linux kernel, or a video encoder like x264 are other examples of software systems that are highly configurable at compile-time or at run-time for delivering the expected functionality and meeting the various desires of users. Variability ("the ability of a software system or artifact to be efficiently extended, changed, customized or configured for use in a particular context") is therefore a crucial property of software systems. Organizations capable of mastering variability can deliver high-quality variants (or products) in a short amount of time and thus attract numerous customers, new use-cases or usage contexts. A hard problem for end-users or software developers is to master the combinatorial explosion induced by variability: Hundreds of configuration options can be combined, each potentially with distinct functionality and effects on execution time, memory footprint, quality of the result, etc. The first part of this course will introduce variability-intensive systems, their applications and challenges, in various software contexts. We will use intuitive examples (like a generator of LaTeX paper variants) and real-world systems (like the Linux kernel). A second objective of this course is to show the relevance of Artificial Intelligence (AI) techniques for exploring and taming such enormous variability spaces. In particular, we will introduce how (1) satisfiability and constraint programming solvers can be used to properly model and reason about variability; (2) how machine learning can be used to discover constraints and predict the variability behavior of configurable systems or software product lines.

• http://ejcp2019.icube.unistra.fr/
• I had 90 minutes + 90 minutes (~ 3 hours)
• Some results have not been published yet, but some preprints/datasets/Github repos are available (ask me!)
An incomplete family of researchers that have something in common and some variability (thanks!)

http://www.diverse-team.fr/
How to configure frama-c?

Primary Options

- option **-eva-slevel**: allows Eva to explore \( n \) separated paths before joining them
- option **-eva-slevel-function**: same as previous, but for a particular function
- annotation **loop unroll** \( n \): for considering \( n \) iterations of a loop separately

For specialists only

- option **-eva-ilevel**: maximum number of elements in the set before conversion into intervals (default = 8)
- option **-eva-plevel**: maximum number of distinct array cells (default = 200)
How to ensure that all Linux kernel configurations build?

Enormous configurations space eg Linux has 15K+ options, tri-state values {y, n, m}. A build takes 15 minutes on average on a recent machine.
“According to several studies, configuration failures represent one of the most common types of software failures”
A Universe of Options

# Variants (independent Boolean options)

2^{33}  

2^{320}
How to master configuration space?
(with machines and humans)
How to master configuration space?
(for machines and humans)
How to master configuration space?

AI (here) = variability modeling / automated reasoning + statistical machine learning
Software Variability and EJCP

- Empirical Software Engineering
  - understanding real-world variability (data)
  - Developing effective techniques on real-world systems

- Constraint Programming
  - SAT/SMP/CP solvers to reason about variability

- Coccinelle and the Linux kernel: a challenging case study for software variability

- Formal verification: many papers on verifying software variants (Thuem et al. ACM Computing Survey 2014)
Vary\LaTeX: Learning Paper Variants That Meet Constraints

Mathieu Acher
Paul Temple
Jean-Marc Jézéquel
Univ Rennes, Inria, CNRS, IRISA
Rennes, France
mathieu.acher@irisa.fr

José A. Galindo
University of Sevilla
Sevilla, Spain
jagalindo@us.es

Jabier Martinez
Tewfik Ziadi
Sorbonne University UPMC
Paris, France
jabier.martinez@lip6.fr

Successfully submitted for VaMoS’18
(on time and meeting formatting instructions)
and then accepted
(live demonstration)
Two case studies

• FSE paper (see demonstration)
  – Page limit: 4
  – Accuracy: ~85% with 40 papers in the training set (there are 73,440 valid configurations)

• Curriculum vitae generation
  – 18 pages limit; 5 Boolean options; full generation, only 32 papers (not need to learn here)
AI#1 Logic, satisfiability, constraints, reasoning, solving

Variability annotations and modeling

LaTeX source files

// Boolean options (features)
fmlatex = FM (VARY_LATEX : BREF BIB [PL_FOOTNOTE] [ACK] JS_STYLE [LONG_AFFILIATION] ;
JSSTYLE : (JS_SCRIPTSIZE I JS_TINY I JS_FOOTNOTESIZE); // mutually exclusive
ACK : [LONG_ACK] (BOLD_ACK I PARAGRAPH_ACK); // LONG_ACK is optional
LONG_AFFILIATION : [EMAIL]; )
// numerical options (attributes)
real BIB.vspace_bib: [1.0..5.0] precision 1 // 1 decimal digit precision
real BREF.bref_size: [0.7..1.0] precision 1 // either 0.7 0.8 0.9 or 1.0
real cserver_size : [0.6..0.9] precision 1 // either 0.6 0.7 0.8 or 0.9
// specific constraints can be added a priori if needs be
### AI#2: Statistical, supervised machine learning (classification problem)

**Paper variants building and measurements**

<table>
<thead>
<tr>
<th>LONG_ACK</th>
<th>LONG_AFFILIATION</th>
<th>PARAGRAPH_ACK</th>
<th>PL FOOTNOTE</th>
<th>VARY_LATEX</th>
<th>bref_size</th>
<th>cserver_size</th>
<th>vspace_bib</th>
<th>nbPages</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0.7</td>
<td>0.9</td>
<td>4.0</td>
<td>4</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0.8</td>
<td>0.6</td>
<td>2.2</td>
<td>4</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0.9</td>
<td>0.6</td>
<td>2.3</td>
<td>4</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>0.7</td>
<td>0.8</td>
<td>1.1</td>
<td>4</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>0.8</td>
<td>0.9</td>
<td>1.8</td>
<td>5</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0.7</td>
<td>0.8</td>
<td>2.8</td>
<td>5</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0.7</td>
<td>0.8</td>
<td>2.9</td>
<td>5</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0.9</td>
<td>0.7</td>
<td>4.9</td>
<td>4</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>1.0</td>
<td>0.7</td>
<td>1.7</td>
<td>5</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>1.0</td>
<td>0.6</td>
<td>1.8</td>
<td>5</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>0.7</td>
<td>0.6</td>
<td>2.8</td>
<td>4</td>
</tr>
</tbody>
</table>
Specialization of the variability model

// same original variability model
fmLaTeX = FM (VARY_LATEX ... )
// ...
real cserver_size: [0.6..0.9] precision 1
// constraints (^ is AND, ! is NOT, => is IMPLIES)
// we negate the paths leading to class "5" (non-acceptable)
// !(JS_SCRIPTSIZE ^ cserver_size >= 0.65) or more readable:
(JS_SCRIPTSIZE => cserver_size < 0.65) ^
// !(JS_SCRIPTSIZE ^ cserver_size < 0.65 ^ PARAGRAPH_ACK)
// equivalent to
(JS_SCRIPTSIZE => (cserver_size < 0.65 => !PARAGRAPH_ACK)) ^
(!(JS_SCRIPTSIZE ^ cserver_size >= 0.9 ^ bref_size >=0.9)

https://github.com/FAMILIAR-project/varylatex/
Variability and LaTeX source files

(a) Variability annotations and excerpt of some possible paper variants

\lstdefinestyle{JavaScript}{
keywords={typeof, new, true, false, catch, function, return, null, catch, switch, var, if, in, while, do, else, case, break},
keywordstyle={color{blue}\bfseries},
basicstyle={ttfamily\#{if JS_SCRIPTSIZE}\scriptsize\#{if JS_TINY}\tiny\#{if JS_FOOTNOTESIZE}\footnotesize\#{if /}},
}

\#{if PL_FOOTNOTE}\footnote{We are considering "product lines" in a broad sense,

(b) Users can vary the font size of a code snippet, activate a footnote, vary the font size of a figure, etc.

Acknowledgements. We thank anonymous reviewers for their valuable feedbacks. We thank Pierre Laperdrix for the newspaper example.

4. REFERENCES

Acknowledgements. We thank anonymous reviewers for their valuable feedbacks.

4. REFERENCES

...
Classification tree

JS_SCRIPTSIZE = false

cserver_size < 0.85

bref_size < 0.85

PARAGRAPH_ACK = false

cserver_size < 0.65

true

true

>= 0.65

false

true

>= 0.85

false

true

4

4

5

5

5

5

4

4
Agenda

• Software Variability: An Overview
  – VaryLaTeX
  – Linux, video generator, 3D printing, etc.
  – Testing 26K+ configurations of JHipster

• AI1: Modeling and Reasoning about Variability
  – Feature models: syntax, semantics, and logics

• AI2: Learning Variability
  – Statistical supervised machine learning

• AI for fitting Software Variability
VaryLaTeX

an instance of a more general problem

(and solution based on artificial intelligence and software engineering techniques)
Sampling, Measuring, Learning

Learning Software Configuration Spaces: A Systematic Literature Review
Juliana Alves Pereira, Hugo Martin, Mathieu Acher, Jean-Marc Jézéquel, Goetz Botterweck, Anthony Ventresque
Variability

• “the **ability** of a software system or artifact to be efficiently extended, changed, customized or configured for use in a particular context” (Svahnberg et al. 2005)
  – software -*customization* perspective

• Terminology/applicability
  – Software product lines, configurable systems, variability-intensive systems, dynamic adaptive systems, generators
  – Options ≈ features, flags, parameters, variation points, etc.
Software Variability

- Configurable system
- Configuration options (aka software features) template variables of a LaTeX file
- Variants
- LaTeX source and PDF variants (papers)
- Large variability spaces
  73,440 possible variants
Software Variability

• Configurable system

Linux operating system

• Configuration options (aka software features)

conditional compilation (#ifdef) in C files

• Variants

Linux kernel variants

• Large variability spaces

16,000 options (~“yes”, “no”, “module”)

Linux Kernel
Software Variability

• Configurable system
Firefox web browser
• Configuration options (aka software features)
  feature flags (about:config)
• Variants
Firefox behavior (e.g., security)
• Large variability spaces
  2000+ options (Boolean, categorical, numeric)
Software Variability

- Configurable system

Scikit

- Configuration options (aka software features)

Hyper-parameters

- Variants

Machine learning algorithm behavior

- Large variability spaces

Dozens of options (Boolean, categorical, numerical)
Software Variability

• Configurable system
x264 video encoder
• Configuration options (aka software features)
command line parameters
• Variants
x264 behavior (different outputs, execution time, etc.)
• Large variability spaces
Dozens of options (Boolean, categorical, numeric)
Exercice

• Give examples of real-world systems with (large | complex)+ variability spaces
• What’s the underlying problems for developers?
• What’s the underlying problems for users?
• Can we apply the VaryLaTeX approach?
x264 --no-progress
   --no-asm
   --rc-lookahead 60
   --ref 9
-o trailer_480p24.x264
trailer_2k_480p24.y4m

?? seconds

Guo et al. ASE 2013, Apel et al. ASE’15, Czarnecki et al. SPLC’15, Siegmund et al.
FSE’15, Kastner et al. ASE’17, Menzies et al. FSE’17, Batory et al. FSE’17
NEW KANGOO VAN RANGE

OPTIONS

COMFORT
- Central storage console & armrest between seats £50.00

DRIVING
- Electric door mirrors £0.00

SAFETY & SECURITY
- ESC (Electronic Stability Control) with traction and understeer control £200.00

"Reverse Engineering Web Configurators" Ebrahim Khalil Abbasi, Mathieu Acher, Patrick Heymans, and Anthony Cleve. In 17th European Conference on Software Maintenance and Reengineering (CSMR'14)
LE PLIAGE PERSONNALISÉ

LE PLIAGE CUIR
LE PLIAGE TOILE

VOTRE PERSONNALISATION

Porte-monnaie Toile : 9 x 7 x 5 cm
Couleur recto : Garance
Couleur verso : Malabar
Bouclerie : Bronze

35,00 € AJOUTER AU PANIER

Infos Partager J'aime
Feature Model Extraction from Large Collections of Informal Product Descriptions

Jean-Marc Davril, Edouard Delfosse, Negar Hariri, Mathieu Acher, Jane Cleland-Huang, Patrick Heymans
(ESEC/FSE ’13)

Variability Model

« Extraction and Evolution of Architectural Variability Models in Plugin-based Systems »
Mathieu Acher, Anthony Cleve, Philippe Collet, Philippe Merle, Laurence Duchien, Philippe Lahire
ECSA/SoSyM’14
<table>
<thead>
<tr>
<th>Brand</th>
<th>Model name</th>
<th>Sensor size</th>
<th>Effective megapixels</th>
<th>Lens mount</th>
<th>Viewfinder type</th>
<th>Viewfinder coverage (% of the frame)</th>
<th>Metering zones</th>
<th>Focus points</th>
<th>Lowest ISO</th>
<th>Highest ISO</th>
<th>DxOMark sensor score</th>
<th>DxO ISO performance^{[1]}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon</td>
<td>1D X</td>
<td>Full frame</td>
<td>18.1</td>
<td>EF</td>
<td>Pentaprism</td>
<td>100</td>
<td>252</td>
<td>61</td>
<td>50</td>
<td>204800</td>
<td>62</td>
<td>2786</td>
</tr>
<tr>
<td>Canon</td>
<td>1Ds Mark III</td>
<td>Full frame</td>
<td>21.1</td>
<td>EF</td>
<td>Pentaprism</td>
<td>100</td>
<td>63</td>
<td>45</td>
<td>50</td>
<td>3200</td>
<td>80</td>
<td>1563</td>
</tr>
<tr>
<td>Canon</td>
<td>1D Mark IV</td>
<td>APS-H</td>
<td>16.1</td>
<td></td>
<td></td>
<td></td>
<td>63</td>
<td>45</td>
<td>50</td>
<td>102400</td>
<td>74</td>
<td>1320</td>
</tr>
<tr>
<td>Canon</td>
<td>5D Mark III</td>
<td>Full frame</td>
<td>22.3</td>
<td></td>
<td></td>
<td></td>
<td>63</td>
<td>61</td>
<td>50</td>
<td>102400</td>
<td>81</td>
<td>2293</td>
</tr>
<tr>
<td>Canon</td>
<td>5D Mark II</td>
<td>Full frame</td>
<td>21.1</td>
<td></td>
<td></td>
<td></td>
<td>35</td>
<td>9</td>
<td>50</td>
<td>25600</td>
<td>79</td>
<td>1815</td>
</tr>
<tr>
<td>Canon</td>
<td>6D</td>
<td>Full frame</td>
<td>20.2</td>
<td></td>
<td></td>
<td></td>
<td>63</td>
<td>11</td>
<td>100</td>
<td>102400</td>
<td>62</td>
<td>2340</td>
</tr>
<tr>
<td>Canon</td>
<td>7D</td>
<td>APS-C</td>
<td>18.0</td>
<td></td>
<td></td>
<td></td>
<td>63</td>
<td>19</td>
<td>100</td>
<td>12800</td>
<td>66</td>
<td>854</td>
</tr>
<tr>
<td>Canon</td>
<td>70D</td>
<td>APS-C</td>
<td>20.2</td>
<td>EF, EF-S</td>
<td>Pentaprism</td>
<td>95</td>
<td>63</td>
<td>19</td>
<td>100</td>
<td>25600</td>
<td>68</td>
<td>926</td>
</tr>
<tr>
<td>Canon</td>
<td>60D</td>
<td>APS-C</td>
<td>18.0</td>
<td>EF, EF-S</td>
<td>Pentaprism</td>
<td>95</td>
<td>63</td>
<td>9</td>
<td>100</td>
<td>12800</td>
<td>66</td>
<td>813</td>
</tr>
<tr>
<td>Canon</td>
<td>50D</td>
<td>APS-C</td>
<td>15.1</td>
<td>EF, EF-S</td>
<td>Pentaprism</td>
<td>95</td>
<td>35</td>
<td>9</td>
<td>100</td>
<td>12800</td>
<td>63</td>
<td>696</td>
</tr>
<tr>
<td>Canon</td>
<td>40D</td>
<td>APS-C</td>
<td>10.1</td>
<td>EF, EF-S</td>
<td>Pentaprism</td>
<td>95</td>
<td>35</td>
<td>9</td>
<td>100</td>
<td>3200</td>
<td>64</td>
<td>703</td>
</tr>
<tr>
<td>Canon</td>
<td>30D</td>
<td>APS-C</td>
<td>8.2</td>
<td>EF, EF-S</td>
<td>Pentaprism</td>
<td>95</td>
<td>35</td>
<td>9</td>
<td>100</td>
<td>3200</td>
<td>59</td>
<td>736</td>
</tr>
<tr>
<td>Canon</td>
<td>20D</td>
<td>APS-C</td>
<td>8.2</td>
<td>EF, EF-S</td>
<td>Pentaprism</td>
<td>95</td>
<td>35</td>
<td>9</td>
<td>100</td>
<td>3200</td>
<td>62</td>
<td>721</td>
</tr>
</tbody>
</table>

José A. Galindo, Mauricio Alférez, Mathieu Acher, Benoit Baudry, David Benavides: A variability-based testing approach for synthesizing video sequences. ISSTA 2014:
Mathieu Acher, Benoit Baudry, Olivier Barais, Jean-Marc Jézéquel: Customization and 3D printing: a challenging playground for software product lines. SPLC 2014: 142-146
KeY is a modern theorem prover with a large community intended to verify JML-specified Java programs.

KeY has variability!

<table>
<thead>
<tr>
<th>Assumption</th>
<th>Parameter</th>
<th>Hypothesis</th>
<th>First option</th>
<th>Second option</th>
<th>Requirement</th>
<th>Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Stop at</td>
<td>Default</td>
<td>Unclosable</td>
<td>P</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Stop at</td>
<td>Default</td>
<td>Unclosable</td>
<td>VE</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>One step simplification</td>
<td>Enabled</td>
<td>Disabled</td>
<td>P</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>One step simplification</td>
<td>Enabled</td>
<td>Disabled</td>
<td>VE</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Proof splitting</td>
<td>Delayed</td>
<td>Free</td>
<td>P</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Proof splitting</td>
<td>Delayed</td>
<td>Free</td>
<td>VE</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Proof splitting</td>
<td>Off</td>
<td>Free</td>
<td>P</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Proof splitting</td>
<td>Off</td>
<td>Free</td>
<td>VE</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Loop treatment</td>
<td>Invariant</td>
<td>Loop scope invariant</td>
<td>P</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Loop treatment</td>
<td>Invariant</td>
<td>Loop scope invariant</td>
<td>VE</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Dependency contracts without accessible-clauses</td>
<td>On</td>
<td>Off</td>
<td>P</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Dependency contracts without accessible-clauses</td>
<td>On</td>
<td>Off</td>
<td>VE</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Query treatment without queries</td>
<td>On</td>
<td>Restricted</td>
<td>P</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Query treatment without queries</td>
<td>On</td>
<td>Restricted</td>
<td>VE</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Query treatment</td>
<td>Off</td>
<td>Restricted</td>
<td>P</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Query treatment</td>
<td>Restricted</td>
<td>On</td>
<td>P</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Expand local queries</td>
<td>On</td>
<td>Off</td>
<td>VE</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Expand local queries</td>
<td>On</td>
<td>Off</td>
<td>VE</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Arithmetic treatment</td>
<td>Basic</td>
<td>DefOps</td>
<td>P</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Arithmetic treatment</td>
<td>DefOps</td>
<td>ModelSearch</td>
<td>P</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Quantifier treatment without quantifiers</td>
<td>None</td>
<td>No splits</td>
<td>P</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Quantifier treatment without quantifiers</td>
<td>None</td>
<td>No splits</td>
<td>VE</td>
<td>&lt;&gt;</td>
<td></td>
</tr>
</tbody>
</table>
How is variability implemented?
(examples)
Case study: JHipster

● Web-apps generator
  ○ Spring-Boot
  ○ Bootstrap / AngularJS
  ○ 100 % Open Source

● Yeoman
  ○ Bower, npm
  ○ yo

● Used all over the world
  ○ Large companies (HBO, Google, Adobe)¹
  ○ Independent developers
  ○ Our students

● GitHub
  ○ 6000+ stars
  ○ 118 releases (JHipster 3.6.1, 18 Aug 2016)
  ○ 300+ contributors

¹ https://jhipster.github.io/companies-using-jhipster/
macher-wifi: getting-started macher1$ yo jhipster

I'm all done. Running `npm install & bower install` for you to install the required dependencies.

HIPSTER STACK FOR JAVA DEVS

Welcome to the JHipster Generator v2.17.0

? (1/15) What is the base name of your application? jhipster
? (2/15) What is your default Java package name? com.mycompany.myapp
? (3/15) Do you want to use Java 8? Yes (use Java 8)
? (4/15) Which *type* of authentication would you like to use? (Use arrow keys)
> HTTP Session Authentication (stateful, default Spring Security mechanism)
> OAuth2 Authentication (stateless, with an OAuth2 server implementation)
> Token-based authentication (stateless, with a token)
package <%=packageName%.config;

<% if (databaseType == 'sql') { %>
import <%=packageName%.config.liquibase.AsyncSpringLiquibase;
import com.codahale.metrics.MetricRegistry;
import com.fasterxml.jackson.datatype.hibernate4.Hibernate4Module;
import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;
import liquibase.integration.spring.SpringLiquibase;<% } %><% if (databaseType == 'mongodb' & authenticationType == 'oauth2') { %>
import <%=packageName%.config.oauth2.OAuth2AuthenticationReadConverter;<% } %><% if (databaseType == 'mongodb') { %>
import com.mongodb.Mongo;
import org.mongeon.Mongoez;<% } %>
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;<% if (databaseType == 'sql') { %><% if (hibernateCache == 'hazelcast') { %>
import org.springframework.cache.CacheManager;<% } %>
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.autoconfigure.condition.ConditionalOnExpression;<% if (databaseType == 'mongodb') { %>
import org.springframework.boot.autoconfigure.mongo.MongoAutoConfiguration;
import org.springframework.boot.autoconfigure.mongo.MongoProperties;<% if (databaseType == 'sql') { %>
import org.springframework.boot.autoconfigure.jdbc.DataSourceProperties;
import org.springframework.boot.autoconfigure.liquibase.LiquibaseProperties;
import org.springframework.context.ApplicationContextException;<% } %>
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Profile;<% if (databaseType == 'mongodb') { %>
import org.springframework.context.annotation.Import;<% } %><% if (databaseType == 'sql') { %>
import org.springframework.core.env.Environment;<% if (databaseType == 'mongodb' & authenticationType == 'oauth2') { %>
import org.springframework.core.io.ClassPathResource;<% if (searchEngine == 'elasticsearch') { %>
import org.springframework.data.elasticsearch.repository.config.EnableElasticsearchRepositories;<% } %><% if (databaseType == 'mongodb') { %>
import org.springframework.data.mongodb.config.EnableMongoAuditing;<% } %><% if (databaseType == 'mongodb' & authenticationType =

} else { }
import org.springframework.data.mongodb.core.convert.CustomConversions;
import org.springframework.data.mongodb.core.mapping.event.ValidatingMongoEventListener;
import org.springframework.data.mongodb.core.repository.config.EnableMongoRepositories;
import org.springframework.validation.beanvalidation.LocalValidatorFactoryBean;<% if (databaseType == 'sql') { %>
{% if (databaseType === 'mongodb' || databaseType === 'couchbase') %}
    @Profile("!" + JHipsterConstants.SPRING_PROFILE_CLOUD)
{% endif %}

{% if (databaseType === 'mongodb') %}
    {% if (reactive) %}
        @Import(value = {MongoAutoConfiguration.class, MongoReactiveAutoConfiguration.class})
    {% else %}
        @Import(value = MongoAutoConfiguration.class)
    {% endif %}
    @EnableMongoAuditing(auditorAwareRef = "springSecurityAuditorAware")
{% endif %}

{% if (databaseType === 'couchbase') %}
    {% if (reactive) %}
        @EnableCouchbaseRepositories(repositoryBaseClass = CustomN1qlCouchbaseRepository.class, basePackages = "<%=packageName%>.repository")
    {% endif %}
    @EnableReactiveCouchbaseRepositories(repositoryBaseClass = CustomReactiveN1qlCouchbaseRepository.class, basePackages = "<%=packageName%>.repository")
    @Import(value = CouchbaseAutoConfiguration.class)
    @EnableCouchbaseAuditing(auditorAwareRef = "springSecurityAuditorAware")
{% endif %}

public class DatabaseConfiguration {

    private final Logger log = LoggerFactory.getLogger(DatabaseConfiguration.class);

    {% if (databaseType === 'sql') %}
        {% if (devDatabaseType === 'h2Disk' || devDatabaseType === 'h2Memory') %}
            private final Environment env;

            public DatabaseConfiguration(Environment env) {
                this.env = env;
            }
        {% endif %}
    {% endif %}
}
<%_ if (applicationType === 'gateway' && authenticationType === 'uaa') { _%>
  <dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
  </dependency>
<%_ } _%>
<%_ if (cacheProvider === 'hazelcast') { _%>
  <dependency>
    <groupId>com.hazelcast</groupId>
    <artifactId>hazelcast</artifactId>
  </dependency>
<%_ } _%>
<%_ if (cacheProvider === 'hazelcast' && enableHibernateCache) { _%>
  <dependency>
    <groupId>com.hazelcast</groupId>
    <artifactId>hazelcast-hibernate53</artifactId>
  </dependency>
<%_ } _%>
<%_ if (cacheProvider === 'hazelcast') { _%>
  <dependency>
    <groupId>com.hazelcast</groupId>
    <artifactId>hazelcast-spring</artifactId>
  </dependency>
<%_ } _%>
<%_ if (cacheProvider === 'infinispan') { _%>
  <dependency>
    <groupId>org.infinispan</groupId>
    <artifactId>infinispan-remote</artifactId>
  </dependency>
<%_ } _%>
Variability crosscuts all artefacts
(40 languages are used in a contemporary Web app)
Software developers:
How to ensure that all software variants are “valid”?

From: Evgeny Kuznetsov <ext-eugeny.kuznetsov@nokia.com>

Value of "irq_reg" pointer is depend on configuration and GPIO method. Potentially it may have NULL value and it is dereferenced later in code. If pointer is NULL there is some kernel issue. Warning and exit from function are added in this case. Also compilation check is added for correct architecture configuration.

Signed-off-by: Evgeny Kuznetsov <EXT-Eugeny.Kuznetsov@nokia.com>

---
arch/arm/plat omap/gpio.c | 18 +++++++++++++++++++
1 files changed, 18 insertions(+), 0 deletions(-)

diff --git a/arch/arm/plat-omap/gpio.c b/arch/arm/plat-omap/gpio.c
index c05c653..d04913c 100644
--- a/arch/arm/plat-omap/gpio.c
+++ b/arch/arm/plat-omap/gpio.c
@@ -1318,6 +1318,23 @@ static void gpio_irq_handler(unsigned int irq, struct
    } if (bank->method == METHOD_GPIO_64XX)
    irq_reg = bank->base + OMAP1_GPIO_IRQSTATUS;
@endif
    +#if 1
    +
    +#endif
    +#if defined(CONFIG_ARCH_OMAP1) &
    +#if defined(CONFIG_ARCH_OMAP15XX) &
    +#if defined(CONFIG_ARCH_OMAP16XX) &
    +#if defined(CONFIG_ARCH_OMAP730) &
    +#if defined(CONFIG_ARCH_OMAP850) &
    +#if defined(CONFIG_ARCH_OMAP2) &
    +#if defined(CONFIG_ARCH_OMAP3) &
    +#if defined(CONFIG_ARCH_OMAP4)
Software Variability

Software is working (sometimes)
  ○ yes but perhaps for one specific configuration (the default one)
  ○ is it working for all configurations?
At each modification/commit/push/release, do you test all configurations?

- No and you certainly have very good reasons
  - needs lots of resources (machines!); don’t want to burn the planet
  - needs an engineering effort to instrument testing of all configurations
  - the number of configurations is too important (eg 2^16000 for Linux)

```java
@AutoWired(required = false)
private MetricRegistry metricRegistry;

if (clusteredHttpSession == 'hazelcast' || hibernateCache == 'hazelcast') {

    FilterRegistration.Dynamic hazelcastWebFilter = servletContext.addFilter("hazelcastWebFilter", new
    SpringAwareWebFilter());

    Map<String, String> parameters = new HashMap<>();
    parameters.put("instance-name", hazelcastInstance.getName());

    // Name of the distributed map storing your web session objects
    parameters.put("map-name", "clustered-http-sessions");
```
At each modification/commit/push/release, do you test all configurations?

- No since too much resources and effort (impossible and unpractical)
- “Sampling” techniques (subset of configurations)
  - Apel et al. ICSE’16, Kaestner et al. ICSE’14 and ASE’16, Ana B. Sánchez et al. SoSyM 2017, Perrouin et al. ICST’10, Cohen et al. TSE’06, Henard et al. TSE’14, etc.
  - Many papers at SPLC, FSE, ASE, ICSE, ESE, TSE on this topic

What is the cost-effective sampling strategy to test configurations of a system?
Is it Worth testing All Configurations?

Testing with the community

ALL

Sampling
We have tested all configurations of an industrial-strength, open-source generator (Jhipster)

● 26K+ configurations, 4376 hours/machine, 8 man/month
● “Ground truth” allows us to precisely assess sampling

36% failures explained by 6 feature interactions (faults)

What is the most cost-effective sampling strategy?
○ T-wise or dissimilarity are very effective
○ with “only” 126 configurations you can detect all 6 most important faults

Software users: How to choose the configuration that fits my requirements?
Software Variability and Artificial Intelligence

• **Very large variability spaces**
  
  • **AI#1** Abstraction/languages to formally and efficiently reason about configuration spaces
    – with SAT/CSP/SMT solvers
    – Eg constrained sampling
  
  • **AI#2** Statistical machine learning to (out of a sample):
    – Understand the configuration space
    – Find the best configuration
    – Specialize the configuration space (e.g., by capturing constraints)
    – In a cost-effective way

• Humans (developers, end-users, integrator, scientists, etc.) and machines
(end of the first part)
Modeling Variability

• Very large variability spaces

• **AI#1** Abstraction/languages to formally and efficiently reason about configuration spaces
  – with SAT/CSP/SMT solvers
  – Eg constrained sampling

• Variability Models
  – Elaborated by humans
  – Reverse engineered from existing artefacts/systems
  – Promise: sound and complete representation of the configuration space
AI#1 Logic, satisfiability, constraints, reasoning, solving

Variability annotations and modeling

LaTeX source files

// Boolean options (features)
fmLaTeX = FM (VARY_LATEX : BREF BIB [PL_FOOTNOTE] [ACK] JS_STYLE [LONG_AFFILIATION];
JS_STYLE : (JS_SCRIPTSIZE I JS_TINY I JS_FOOTNOTESIZE); // mutually exclusive
ACK : [LONG_ACK] (BOLD_ACK I PARAGRAPH_ACK); // LONG_ACK is optional
LONG_AFFILIATION : [EMAIL];)
// numerical options (attributes)
real BIB.vspace_bib: [1.0..5.0] precision 1 // 1 decimal digit precision
real BREF.bref_size: [0.7..1.0] precision 1 // either 0.7 0.8 0.9 or 1.0
real cserven_size: [0.6..0.9] precision 1 // either 0.6 0.7 0.8 or 0.9
// specific constraints can be added a priori if needs be

...
Linux

KConfig file

config PRINTK
  default y
  bool "Enable support for printk" if EXPERT
  select IRQ_WORK
  help
  This option enables normal printk support. Removing it eliminates most of the message strings from the kernel image and makes the kernel more or less silent. As this makes it very difficult to diagnose system problems, saying N here is strongly discouraged.

config PRINTK_NMI
  def_bool y
  depends on PRINTK
  depends on HAVE_NMI

config BUG
  bool "BUG() support" if EXPERT
  default y
  help
  Disabling this option eliminates support for BUG and WARN, reducing the size of your kernel image and potentially quietly ignoring numerous fatal conditions. You should only consider disabling this option for embedded systems with no facilities for reporting errors. Just say Y.

config ELF_CORE
  depends on COREDUMP
  default y
  bool "Enable ELF core dumps" if EXPERT
  help
  Enable support for generating core dumps. Disabling saves about 4k.

config AIO
  bool "Enable AIO support" if EXPERT
  default y
  help
  This option enables POSIX asynchronous I/O which may be used by some high performance threaded applications. Disabling this option saves about 7k.
Simple question: what are the constraints over WORLD and BYE?

```c
#include <stdio.h>

#ifndef WORLD
    char * msg = "Hello World\n";
#endif
#ifndef BYE
    #ifdef BYE
        char * msg = "Bye bye!\n";
    #endif
#endif

main() {
    printf(msg);
}
```
Unused flexibility
Illegal variant
Feature Model

Communicative

Analytic

Generative
Feature Models
(defacto standard for modeling variability)

Hierarchy: rooted tree

Variability:
• mandatory,
• optional,
• Groups: exclusive or inclusive features
• Cross-tree constraints
Hierarchy + Variability

= set of valid configurations

configuration = set of features selected

{CarEquipment, Comfort, DrivingAndSafety, Healthing, AirConditioning, FrontFogLights}
Hierarchy + Variability

= set of valid configurations

configuration = set of features selected

{CarEquipment, Comfort, DrivingAndSafety, Healthing, AirConditioning}
Hierarchy + Variability

= set of valid configurations

configuration = set of features selected

{CarEquipment, Comfort, DrivingAndSafety, Healthing, AirConditioning, AutomaticHeadLights}
Hierarchy + Variability

= set of valid configurations

{CarEquipment, Comfort, DrivingAndSafety, Healing}
{AirConditioning, FrontFogLights}
{AutomaticHeadLights, AirConditioning, FrontFogLights}
{AutomaticHeadLights, FrontFogLights, AirConditioningFrontAndRear}
{AirConditioningFrontAndRear}
{AirConditioning}
{AirConditioningFrontAndRear, FrontFogLights}
Hierarchy + Variability

= set of valid configurations

<table>
<thead>
<tr>
<th>CarEquipment</th>
<th>Comfort</th>
<th>DrivingAndSafety</th>
<th>Heating</th>
<th>AirConditioning</th>
<th>FrontFogLights</th>
<th>AutomaticHeadLights</th>
<th>AirConditioningFrontAndRear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car2</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Car6</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Car1</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Car4</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Car5</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Car3</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
Hierarchy + Variability

= set of valid configurations

<table>
<thead>
<tr>
<th>Product</th>
<th>CarEquipment</th>
<th>Comfort</th>
<th>DrivingAndSafety</th>
<th>Heating</th>
<th>AirConditioning</th>
<th>FrontFogLights</th>
<th>AutomaticHeadLights</th>
<th>AirConditioningFrontAndRear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car1</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Car2</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Car3</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Car4</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Car5</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Car6</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
Hierarchy + Variability

= set of valid configurations

<table>
<thead>
<tr>
<th>Product</th>
<th>AirConditioning</th>
<th>FrontFogLights</th>
<th>AutomaticHeadLights</th>
<th>AirConditioningFrontAndRear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car1</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Car2</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Car3</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Car4</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Car5</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Car6</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
Hierarchy + Variability = set of valid configurations

configuration = set of features selected

{CarEquipment, Comfort, DrivingAndSafety, Healthing, AirConditioning}
Hierarchy + Variability

= set of valid configurations
Hierarchy + Variability = set of valid configurations

Or-group: at least one!
Hierarchy + Variability

= set of valid configurations

\{CarEquipment, Comfort, DrivingAndSafety, Healthing\}

\{Air Conditioning Front And Rear, Front Fog Lights, SA Control\}
\{Air Conditioning Front And Rear, SA Control\}
\{Automatic Head Lights, Air Conditioning, Front Fog Lights\}
\{Air Conditioning Front And Rear, SA Control, Automatic Head Lights, Front Fog Lights\}
\{Front Fog Lights, Air Conditioning\}
\{Automatic Head Lights, Air Conditioning Front And Rear, Front Fog Lights\}
\{Front Fog Lights, Air Conditioning Front And Rear\}
\{SA Control, Air Conditioning\}
Product Comparison Matrix

<table>
<thead>
<tr>
<th>Product</th>
<th>CarEquipment</th>
<th>Comfort</th>
<th>DrivingAndSafety</th>
<th>Heating</th>
<th>AirConditioning</th>
<th>FrontFogLights</th>
<th>AutomaticHeadLights</th>
<th>AirConditioningFrontAndRear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car1</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Car2</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Car3</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Car4</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Car5</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Car6</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
Hierarchy + Variability = set of valid configurations

\[
[f_{m1}] = \{
    \{W, P, R, S, T, A, V\},
    \{W, P, S, T, A\},
    \{W, P, R, T, A\},
    \{W, P, R, U\},
    \{W, P, R, T, V, A\},
    \{W, P, R, S, T, A\},
\}
\]

(Boolean) Feature Models
(Boolean) Feature Models

~ Boolean formula

\[ \phi_{fm_1} = W \quad // \text{root} \]
\[ \land W \iff P \quad // \text{mandatory} \]
\[ // \text{Or-group} \]
\[ \land P \Rightarrow R \lor S \]
\[ \land R \Rightarrow P \land S \Rightarrow P \]
\[ \land V \Rightarrow T \quad // \text{optional} \]
\[ \land A \iff T \quad // \text{mandatory} \]
\[ // \text{Xor-group} \]
\[ \land T \Rightarrow W \]
\[ \land U \Rightarrow W \]
\[ \land \neg T \lor \neg U \]
\[ // \text{constraints} \]
\[ \land V \Rightarrow R \quad // \text{implies} \]
\[ \land \neg U \Rightarrow \neg S \quad // \text{excludes} \]
I want to analyze and play with my specification!
Empty set of configurations

\[ B \implies \neg C \]
\[ B \land C \]
Dead feature

False optional feature

Optional
Xor-Group
Mandatory
Or-Group
Core features

\{\text{CarEquipment, Comfort, DrivingAndSafety, Healthing}\}
Interactive Configuration
## Feature Models and Automated Reasoning

**Benavides et al. survey, 2010**

<table>
<thead>
<tr>
<th>Feature Model</th>
<th>PL</th>
<th>CP</th>
<th>DL</th>
<th>Multi</th>
<th>Others</th>
<th>No support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Void feature model</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>#Products</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Dead features</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Valid product</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>All products</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Explanations</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Refactoring</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Commonality</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Optimization</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Filter</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Valid partial configuration</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Atomic sets</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>False optional features</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Corrective explanations</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Dependency analysis</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ECR</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Generalization</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Core features</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Variability factor</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Arbitrary edit</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Conditional dead features</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Homogeneity</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>LCA</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Multi-step configuration</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Roots features</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Specialization</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Degree of orthogonality</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Redundancies</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Variant features</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Wrong cardinalities</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Feature model notation</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Extended feature model</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Formalization</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

+ Supported  
- No support  
-supported (first reference)  
-supported (first reference)  
B Basic feature model  
C Cardinality-based feature models
Decision problems and complexity

• Validity of a feature model
• Validity of a configuration
• Computation of dead and core features
• Counting of the number of valid configurations
• Equivalence between two feature models
• Satisfiability (SAT) problem
  – NP-complete
How to automate analysis of your feature models?

Binary Decision Diagram (BDD)
SAT solver
Typical implementations

Multifunction printer
- Print
- Scan
- Flex
- Connection
- Laser
- Inkjet
- USB
- Ethernet

Logics

Result

Solvers

Z3
A knowledge compilation map
Adnan Darwiche and Pierre Marquis
Journal of Artificial Intelligence Research Volume 17 Issue 1, July 2002, Pages 229-264
(note: one of the best paper I ever read)

<table>
<thead>
<tr>
<th>Notation</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO</td>
<td>polytime consistency check</td>
</tr>
<tr>
<td>VA</td>
<td>polytime validity check</td>
</tr>
<tr>
<td>CE</td>
<td>polytime clausal entailment check</td>
</tr>
<tr>
<td>IM</td>
<td>polytime implicant check</td>
</tr>
<tr>
<td>EQ</td>
<td>polytime equivalence check</td>
</tr>
<tr>
<td>SE</td>
<td>polytime sentential entailment check</td>
</tr>
<tr>
<td>CT</td>
<td>polytime model counting</td>
</tr>
<tr>
<td>ME</td>
<td>polytime model enumeration</td>
</tr>
</tbody>
</table>

Table 4: Notations for queries.

<table>
<thead>
<tr>
<th>L</th>
<th>CO</th>
<th>VA</th>
<th>CE</th>
<th>IM</th>
<th>EQ</th>
<th>SE</th>
<th>CT</th>
<th>ME</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNF</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>DNNF</td>
<td>√</td>
<td>o</td>
<td>√</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>√</td>
</tr>
<tr>
<td>d-NNF</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>s-NNF</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>f-NNF</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>d-DNNF</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>?</td>
<td>o</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>sd-DNNF</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>?</td>
<td>o</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>BDD</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>FBDD</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>?</td>
<td>o</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>OBDD</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>o</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>OBDD&lt;</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>DNF</td>
<td>√</td>
<td>o</td>
<td>√</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>CNF</td>
<td>o</td>
<td>√</td>
<td>o</td>
<td>√</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>PI</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>o</td>
<td>√</td>
</tr>
<tr>
<td>IP</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>o</td>
<td>√</td>
</tr>
<tr>
<td>MODS</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

Table 5: Subsets of the NNF language and their corresponding polytime queries. √ means “satisfies” and o means “does not satisfy unless P = NP.”
Binary Decision Diagrams  (Bryant 1986)

encoding of a truth table.

<table>
<thead>
<tr>
<th>from</th>
<th>to</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 1</td>
</tr>
</tbody>
</table>
Binary Decision Diagrams

(after reduction)
Binary Decision Diagrams (BDDs)

• Very efficient structure for most of the satisfiability operations
• Polynomial in time for checking satisfiability and determining equivalence between two BDDs
• Graph traversal
• So great?
Binary Decision Diagrams (BDDs): Theoretical Problem

• The size of the BDD is very sensitive to the order of the BDD variables
  – e.g. two equivalent BDDs for the same feature

[Image: Binary Decision Diagrams (BDDs): Theoretical Problem]

Variable Order: C,R,A,B,D

Variable Order: R,D,A,C,B

[Mendonca, slide]
Binary Decision Diagrams (BDDs): Theoretical Problem

- The size of the BDD is very sensitive to the order of the BDD variables
  - e.g. two equivalent BDDs for the same feature

May lead to size explosion

Variable Order: C,R,A,B,D
Variable Order: R,D,A,C,B

[Mendonca, slide]
Binary Decision Diagrams (BDDs): Theoretical Problem

- The size of the BDD is very sensitive to the order of the BDD variables
  - e.g. two equivalent BDDs for the same feature model
Binary Decision Diagrams (BDDs): Practical Problem

- The size of the BDD is very sensitive to the order of the BDD variables. In practice: BDDs cannot be build for feature models with 2000+ features.

[Heuristics needed]

[Variable Order: C,R,A,B,D]

[Variable Order: R,D,A,C,B]

[Mendonca, slide]
How to automate analysis of your feature models?

Let us try with SAT solvers
Satisfiability (SAT) solver

A “SAT solver” is a program that automatically decides whether a propositional logic formula is satisfiable.

- If it is satisfiable, a SAT solver will produce an example of a truth assignment that satisfies the formula.

Basic idea: since all NP-complete problems are mutually reducible:

- Write one really good solver for NP-complete problems (in fact, get lots of people to do it. Hold competitions.)
- Translate your NP-complete problems to that problem.
SAT solver and CNF

• All current fast SAT solvers work on CNF

• Terminology:
  – A literal is a propositional variable or its negation (e.g., p or ¬q).
  – A clause is a disjunction of literals (e.g., (p ∨ ¬q ∨ r)). Since ∨ is associative, we can represent clauses as lists of literals.

• A formula is in conjunctive normal form (CNF) if it is a conjunction of clauses
  – e.g., (p ∨ q ∨ ¬r) ∧ (¬p ∨ s ∨ t ∨ ¬u)
(Boolean) Feature Models

~ Boolean formula

\[ \phi_{fm_1} = W // \text{root} \]
\[ \land W \iff P // \text{mandatory} \]
\[ // \text{Or-group} \]
\[ \land P \Rightarrow R \lor S \]
\[ \land R \Rightarrow P \land S \Rightarrow P \]
\[ \land V \Rightarrow T // \text{optional} \]
\[ \land A \iff T // \text{mandatory} \]
\[ // \text{Xor-group} \]
\[ \land T \Rightarrow W \]
\[ \land U \Rightarrow W \]
\[ \land \neg T \lor \neg U \]
\[ // \text{constraints} \]
\[ \land V \Rightarrow R // \text{implies} \]
\[ \land \neg U \Rightarrow \neg S // \text{excludes} \]
A \land B
C \Rightarrow A \land D \Rightarrow A

\varnothing

\text{FM1bis} = \text{FM} ("foo3.dimacs")
\text{FM1bisbis} = \text{FM} ("foo3.constraints")

\text{fm1} = \text{FM} ("output/fm1.tvl")
\text{root} A \{ 
  \text{group} [3..3] \{ 
    \text{opt} D \{ 
    \},
    B \{
    \},
    \text{opt} C \{ 
    \}
  \}
\}
\text{fm1} : (\text{FEATURE\_MODEL}) A: [D] B [C];

\text{fml> c1 = cores fm1}
\text{fml> s1: (SET) \{B;A\}}
\text{fml> c1bis = cores fm1bis}
\text{fml> s1bis: (SET) \{B\}}
\text{fml> compare fm1 fm1bis}
\text{fml> res7: (STRING) REFACTORING}
\text{fml> compare fm1bis fm1bisbis}
\text{fml> s1bis: (SET) \{B;A\}}
\text{fml> compare fm1bis fm1bisbis}
\text{fml> res8: (STRING) REFACTORING}
\text{fml> s3: (SET) \{B;A\}}
\text{fml> c1 eq c1bisbis}
\text{fml> s4: (BOOLEAN) true}
\text{fml> s5: (BOOLEAN) true}
Consistency

• SAT-Solver
  – SAT(FM)
Core and dead features

- Dead: $\text{SAT}(\text{FM} \land F)$
- Core: $\text{SAT}(\text{FM} \land \neg F)$
Partial configuration

- SAT(FM ^ PK ^ F)
- SAT(FM ^ PK ^ not(F))
Relationship between feature models

- Refactoring
  - Tautology: \( (FM1 \iff FM2) = \neg \text{SAT}(\neg (FM1 \iff FM2)) \)
How to automate analysis of your feature models?

You can encode a feature model as a CSP problem or as an SMT problem.
Formal semantics of a language

– **formal syntax** \((L)\) – clearcut syntactic rules defining all legal diagrams, a.k.a. syntactic domain

– **semantic domain** \((S)\) – a mathematical abstraction of the real-world concepts to be modelled

– **semantic function** \((M: L \rightarrow S)\) – clearcut semantic rules defining the meaning of all legal diagrams

[Harel & Rumpe, IEEE Computer, 2004]
Definition 2 (Feature Diagram) A feature diagram $FD = \langle G, E_{MAND}, G_{XOR}, G_{OR}, I, EX \rangle$ is defined as follows: $G = (\mathcal{F}, E, r)$ is a rooted, labeled tree where $\mathcal{F}$ is a finite set of features, $E \subseteq \mathcal{F} \times \mathcal{F}$ is a finite set of edges and $r \in \mathcal{F}$ is the root feature; $E_{MAND} \subseteq E$ is a set of edges that define mandatory features with their parents; $G_{XOR} \subseteq \mathcal{P}(\mathcal{F}) \times \mathcal{F}$ and $G_{OR} \subseteq \mathcal{P}(\mathcal{F}) \times \mathcal{F}$ define feature groups and are sets of pairs of child features together with their common parent feature; $I$ a set of implies constraints whose form is $A \Rightarrow B$; $EX$ is a set of excludes constraints whose form is $A \Rightarrow \neg B$ ($A \in \mathcal{F}$ and $B \in \mathcal{F}$).

Definition 3 (Feature Model) An FM is a tuple $\langle FD, \psi \rangle$ where $FD$ is a feature diagram and $\psi$ is a propositional formula over the set of features $\mathcal{F}$. 
Definition 1 (Configuration Semantics) A configuration of an FM feature model is defined as a set of selected features. \( fm_1 \) denotes the set of valid configurations of \( fm_1 \) and is a set of sets of features.
Quizz

1) Give two feature models with the same configuration semantics but with different syntax

2) Does it matter?
Feature Model Synthesis Problem

[Czarnecki et al., SPLC’07]
[She et al., ICSE’11]
[Andersen et al., SPLC’12]

\[
A \land B \\
A \Leftrightarrow B \\
C \Rightarrow A \\
D \Rightarrow A
\]

\[
\phi \Rightarrow FM
\]

\[
\text{fm2} = \text{FM} \left( B : A \ [C] \ [D] \ ; \right) \\
\text{fm3} = \text{FM} \left( B : A ; A : [C] \ [D] ; \right) \\
\text{fm4} = \text{FM} \left( A : B \ [D] ; B : [C] ; \right) \\
\text{fm5} = \text{FM} \left( A : B \ [C] ; B : [D] ; \right) \\
\]

\[
b_{12} = \text{compare} \ \text{fm1} \ \text{fm2} \\
b_{13} = \text{compare} \ \text{fm1} \ \text{fm3} \\
b_{14} = \text{compare} \ \text{fm1} \ \text{fm4} \\
b_{15} = \text{compare} \ \text{fm1} \ \text{fm5} \\
\text{assert} \ (b_{12} \ \text{eq} \ \text{REFACTORING})
\]
#1 Reverse Engineering Scenarios

- [Haslinger et al., WCRE’11], [Acher et al., VaMoS’12]

<table>
<thead>
<tr>
<th>P</th>
<th>V</th>
<th>P</th>
<th>D</th>
<th>O</th>
<th>T</th>
<th>M</th>
<th>S</th>
<th>K</th>
<th>Ad</th>
<th>Ae</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>P2</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>P3</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>P4</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>P5</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>P6</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>P7</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>P8</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>P9</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>P10</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>P11</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>P12</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>P13</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>P14</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>P15</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>P16</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

// from product descriptions to feature models
// typically something generated by VariCell (see VaMoS'12 paper or the dedicated web page)

```
// fmR represents the union of configurations/products
// characterized by fm_1, fm_2, ..., fm_16
// merge union fm_*

fmR = \text{ksynthesis} \text{ fmR with hierarchy= VOD : V P D O ; O : K Ad ; D : T M ; T : Ae C ; M : S ;}
```

- Diagram of a product hierarchy with nodes labeled VOD, Play, Record, Display, OS, TV, Mobile, Kernel, Advanced, Aerial, Cable, Smart.
#2 Refactoring

- [Alves et al., GPCE’06], [Thuem et al., ICSE’09]

```c
// refactoring!
fmR2 = ksynthesis fmR with hierarchy= VOD : V P R D O ; O : K Ad ; D : T M ; T : Ae C ; M : S ;
```
Feature Model SemanticS

• As configuration semantics is not sufficient...

• **Ontological** semantics
  – Hierarchy
  – And feature groups
Given a set of configurations $s$, can we always characterize $s$ with a feature diagram $fd$? 

ie $[[fd]] = s$

In other words: is the formalism of feature diagram expressive enough wrt Boolean logic?
Feature Diagram?

s = \{\{A\}, \\
\{A,C,B\}, \\
\{B,A\}, \\
\{C,D,A\}, \\
\{D,A\}, \\
\{A,D,B\}, \\
\{A,C\} \\
\}

fm1 = FM (A : [B] [C] [D] ^
// B, C and D are optional features of A

((B & C) -> !D)

)
Feature Diagram?

<table>
<thead>
<tr>
<th>Identifier</th>
<th>License</th>
<th>Language</th>
<th>Storage</th>
<th>LicenseCostFee</th>
<th>RSS</th>
<th>Unicode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confluence</td>
<td>Commercial</td>
<td>Java</td>
<td>Database</td>
<td>US10</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PBwiki</td>
<td>NoLimit</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MoinMoin</td>
<td>GPL</td>
<td>Python</td>
<td>Files</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DokuWiki</td>
<td>GPL2</td>
<td>PHP</td>
<td>Files</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PmWiki</td>
<td>GPL2</td>
<td>PHP</td>
<td>Files</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DrupalWiki</td>
<td>GPL2</td>
<td>PHP</td>
<td>Database</td>
<td>Different Licences</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TWiki</td>
<td>GPL</td>
<td>Perl</td>
<td>FilesRCS</td>
<td>Community</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MediaWiki</td>
<td>GPL</td>
<td>PHP</td>
<td>Database</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The diagram shows a WikiMatrix with relationships between different licenses, storage, RSS, and languages.
Feature Model (bis)

\[
\begin{align*}
s &= \{\{A\}, \{B\}\} \\
f_d &= ?
\end{align*}
\]
Feature Model: Key Insights

• Semantics
  – Configuration and ontological

• Syntax
  – Feature diagram vs Feature Model
  – Feature diagram not expressively complete

• Feature models are a (syntactical) view of a propositional formula
Feature model synthesis problem

**Input:** $\phi$, a propositional formula representing the dependencies over a set of features $F$.

**Output:** a maximal feature model with a sound configuration semantics

---

From logics to variability model (there and back again) Czarnecki et al. SPLC’07

(end of second part)
Software Variability and Artificial Intelligence

• Very large variability spaces

• **AI#2** Statistical, supervised machine learning to (out of a sample):
  
  – Understand the configuration space
  – Find the best configuration
  – Specialize the configuration space (e.g., by capturing constraints)
  – In a cost-effective way
AI#2 Statistical, supervised machine learning (classification problem)

Paper variants building and measurements

<table>
<thead>
<tr>
<th>LONG_ACK</th>
<th>LONG_AFFILIATION</th>
<th>PARAGRAPH_ACK</th>
<th>PL_FOOTNOTE</th>
<th>VARY_LATEX</th>
<th>bref_size</th>
<th>cserver_size</th>
<th>vspace_bib</th>
<th>nbPages</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0.7</td>
<td>0.9</td>
<td>4.0</td>
<td>4</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0.8</td>
<td>0.6</td>
<td>2.2</td>
<td>4</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0.9</td>
<td>0.6</td>
<td>2.3</td>
<td>4</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>0.7</td>
<td>0.8</td>
<td>1.1</td>
<td>4</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>0.8</td>
<td>0.9</td>
<td>1.8</td>
<td>5</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>0.7</td>
<td>0.8</td>
<td>2.8</td>
<td>5</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>0.7</td>
<td>0.8</td>
<td>2.9</td>
<td>5</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0.9</td>
<td>0.7</td>
<td>4.9</td>
<td>4</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>1.0</td>
<td>0.7</td>
<td>1.7</td>
<td>5</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>1.0</td>
<td>0.6</td>
<td>1.8</td>
<td>5</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>0.7</td>
<td>0.6</td>
<td>2.8</td>
<td>4</td>
</tr>
</tbody>
</table>
Configuration Space
José A. Galindo, Mauricio Alférez, Mathieu Acher, Benoit Baudry, David Benavides: A variability-based testing approach for synthesizing video sequences. ISSTA 2014:
José A. Galindo, Mauricio Alférez, Mathieu Acher, Benoit Baudry, David Benavides: A variability-based testing approach for synthesizing video sequences. ISSTA 2014:
Problem: some video variants are non-acceptable despite specification of numerous constraints

(note: synthesizing a variant takes 30 minutes)
Problem: some video variants are non-acceptable despite specification of numerous constraints
(note: synthesizing a variant takes 30 minutes)

Paul Temple, José Angel Galindo Duarte, Mathieu Acher, and Jean-Marc Jézéquel. Using Machine Learning to Infer Constraints for Product Lines, SPLC’16
Results (training set: 500 video variants; validation set: 4000 variants)

Paul Temple, José Angel Galindo Duarte, Mathieu Acher, and Jean-Marc Jézéquel. Using Machine Learning to Infer Constraints for Product Lines, SPLC’16
Results (training set: 500 video variants; validation set: 4000 variants)

Paul Temple, José Angel Galindo Duarte, Mathieu Acher, and Jean-Marc Jézéquel. Using Machine Learning to Infer Constraints for Product Lines, SPLC’16
Results (training set: 500 video variants; validation set: 4000 variants)

Paul Temple, José Angel Galindo Duarte, Mathieu Acher, and Jean-Marc Jézéquel. Using Machine Learning to Infer Constraints for Product Lines, SPLC’16

Constraints

! (signal_quality.dynamic_noise_level > 0.171472 && signal_quality.compression_artefact_level <= 0.180349) ! (signal_quality.dynamic_noise_level > 0.171472)
Generalization of learning-based specialization

• Configurations have a label/class
  – true/false (video gen) or nbPages={4,5} (VaryLaTeX); without any discussion a classification problem

• However there are scenarios in which the acceptability is defined in terms of performance

• Specialization is a classification problem; we boil down to this problem through a threshold over a quantitative value eg execution time < 1s
(wrap up and exercise)
Humans  Machine

Modeling (SAT/CP)

Knowledge
Soundness
Completeness
Interpretability
Cost
Generalization

Sampling, measuring, learning
Sampling, Measuring, Learning

Learning Software Configuration Spaces: A Systematic Literature Review
Juliana Alves Pereira, Hugo Martin, Mathieu Acher, Jean-Marc Jézéquel, Goetz Botterweck, Anthony Ventresque

Several scenarios

(a) A1: Pure Prediction.
(b) A2: Interpretability.
(c) A3: Optimization.
(d) A4: Dynamic Configuration.
(e) A5: Mining Constraints
(f) A6: Evolution.

Learning Software Configuration Spaces: A Systematic Literature Review
Juliana Alves Pereira, Hugo Martin, Mathieu Acher, Jean-Marc Jézéquel, Goetz Botterweck, Anthony Ventresque
Huge applicability!

### Learning Software Configuration Spaces: A Systematic Literature Review

Juliana Alves Pereira, Hugo Martin, Mathieu Acher, Jean-Marc Jézéquel, Goetz Botterweck, Anthony Ventresque


<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Non-Functional Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thingiverse’s</td>
<td>3D printer</td>
<td>defects</td>
</tr>
<tr>
<td>IBM WebSphere</td>
<td>Application server</td>
<td>throughput</td>
</tr>
<tr>
<td>Clasp</td>
<td>ASP solver</td>
<td>response time</td>
</tr>
<tr>
<td>SNW</td>
<td>Asset management</td>
<td>area and throughput</td>
</tr>
<tr>
<td>Binpacking</td>
<td>Binpacking algorithm</td>
<td>execution time and accuracy</td>
</tr>
<tr>
<td>XGBoost</td>
<td>Boosting algorithms</td>
<td>training time</td>
</tr>
<tr>
<td>SaaS system</td>
<td>Cloud computing</td>
<td>response time</td>
</tr>
<tr>
<td>Clustering</td>
<td>Clustering algorithm</td>
<td>execution time and accuracy</td>
</tr>
<tr>
<td>AJStats</td>
<td>Code analyzer</td>
<td>response time</td>
</tr>
<tr>
<td>SaC</td>
<td>Code analyzer</td>
<td>I/O time, response time</td>
</tr>
<tr>
<td>POLLY</td>
<td>Code optimizer</td>
<td>runtime</td>
</tr>
<tr>
<td>Libssh</td>
<td>Comb Borational model</td>
<td>defects</td>
</tr>
<tr>
<td>Telecom</td>
<td>Communication system</td>
<td>defects</td>
</tr>
<tr>
<td>LLVM</td>
<td>Compiler</td>
<td>memory footprint, performance, response time, code complexity, compilation time</td>
</tr>
<tr>
<td>Compressor SPL</td>
<td>Compression library</td>
<td>compression time, memory usage and compression ratio</td>
</tr>
<tr>
<td>7Z</td>
<td>Compression library</td>
<td>Compression time</td>
</tr>
<tr>
<td>LRZlP</td>
<td>Compression library</td>
<td>compressed size, compression time, compilation time</td>
</tr>
<tr>
<td>RAR</td>
<td>Compression library</td>
<td>code complexity</td>
</tr>
<tr>
<td>XZ</td>
<td>Compression library</td>
<td>compression time</td>
</tr>
<tr>
<td>ZipMe</td>
<td>Compression library</td>
<td>memory footprint, performance, code complexity, time</td>
</tr>
<tr>
<td>Wordpress</td>
<td>Content management</td>
<td>CPU power consumption</td>
</tr>
<tr>
<td>LinkedIn</td>
<td>Data structures</td>
<td>memory footprint, performance, maintainability, binary size</td>
</tr>
<tr>
<td>Curl</td>
<td>Data transfer</td>
<td>code complexity</td>
</tr>
<tr>
<td>Wget</td>
<td>Data transfer</td>
<td>memory footprint, code complexity</td>
</tr>
<tr>
<td>Actian Vector</td>
<td>Database system</td>
<td>memory footprint, code complexity</td>
</tr>
<tr>
<td>Apache Cassandra</td>
<td>Database system</td>
<td>latency</td>
</tr>
<tr>
<td>Berkeley DB</td>
<td>Database system</td>
<td>I/O time, memory footprint, performance, response time, code complexity, maintainability, binary size, performance</td>
</tr>
<tr>
<td>FAME-DBMS</td>
<td>Database system</td>
<td>latency</td>
</tr>
<tr>
<td>MySQL</td>
<td>Database system</td>
<td>memory footprint, performance</td>
</tr>
<tr>
<td>Postgres</td>
<td>Database system</td>
<td>memory footprint, performance</td>
</tr>
<tr>
<td>Prevailer</td>
<td>Database system</td>
<td>memory footprint, performance</td>
</tr>
<tr>
<td>SQLite</td>
<td>Database system</td>
<td>memory footprint, performance</td>
</tr>
<tr>
<td>StockOnline</td>
<td>Database system</td>
<td>response time</td>
</tr>
<tr>
<td>Kafka</td>
<td>Distributed systems</td>
<td>throughput</td>
</tr>
<tr>
<td>DNN</td>
<td>DNNs algorithms</td>
<td>accuracy of predictions</td>
</tr>
<tr>
<td>Curriculum vitae</td>
<td>Document</td>
<td>number of pages</td>
</tr>
<tr>
<td>Paper</td>
<td>Document</td>
<td>number of pages</td>
</tr>
<tr>
<td>RUBiS</td>
<td>E-commerce application</td>
<td>response time</td>
</tr>
<tr>
<td>EMAIL</td>
<td>E-mail client</td>
<td>response time</td>
</tr>
<tr>
<td>MBED TLS</td>
<td>Encryption library</td>
<td>response time</td>
</tr>
<tr>
<td>SAP ERP</td>
<td>Enterprise Application</td>
<td>response time</td>
</tr>
<tr>
<td>noe-CM-log</td>
<td>FPGA</td>
<td>response time</td>
</tr>
<tr>
<td>sort-256</td>
<td>FPGA</td>
<td>response time</td>
</tr>
<tr>
<td>E-Health System</td>
<td>Health</td>
<td>response time</td>
</tr>
<tr>
<td>HIP’A°C</td>
<td>Image processing</td>
<td>response time</td>
</tr>
<tr>
<td>Disparity SPL</td>
<td>Image processing</td>
<td>energy consumption</td>
</tr>
<tr>
<td>PRJob</td>
<td>Instant messenger</td>
<td>memory footprint, performance</td>
</tr>
<tr>
<td>IBM ILOG CPLEX</td>
<td>Integer solver</td>
<td>runtime</td>
</tr>
<tr>
<td>SPECjbb2005</td>
<td>Java Server</td>
<td>response time, throughput</td>
</tr>
<tr>
<td>WEKA</td>
<td>Learning algorithm</td>
<td>accuracy of predictions</td>
</tr>
<tr>
<td>SVD</td>
<td>Linear algebra</td>
<td>execution time and accuracy</td>
</tr>
<tr>
<td>Trimesh</td>
<td>Mesh solver</td>
<td>iterations, response time</td>
</tr>
<tr>
<td>MBENCH</td>
<td>Micro benchmark</td>
<td>time</td>
</tr>
<tr>
<td>ACE+TAO system</td>
<td>Middleware software</td>
<td>defects</td>
</tr>
<tr>
<td>SensorNetwork</td>
<td>Network simulator</td>
<td>memory footprint, performance</td>
</tr>
<tr>
<td>SimulatoR</td>
<td>Network simulator</td>
<td>latency</td>
</tr>
<tr>
<td>NoC</td>
<td>Network-based system</td>
<td>energy and runtime</td>
</tr>
<tr>
<td>Helmholtz 3D</td>
<td>Numerical analysis</td>
<td>execution time and accuracy</td>
</tr>
<tr>
<td>Poisson 2D</td>
<td>Numerical analysis</td>
<td>execution time and accuracy</td>
</tr>
<tr>
<td>Linux kernel</td>
<td>Operating system</td>
<td>memory footprint, performance</td>
</tr>
<tr>
<td>DNN</td>
<td>Optimization algorithm</td>
<td>response time</td>
</tr>
</tbody>
</table>
Software Engineering and Machine Learning

- Automated measurements of thousands of Linux variants
- Learning with a high precision, with a small sample
configuration options: 9K+ for X86_64
100K+ configurations (!!!)
https://github.com/TuxML/ProjetIrma/

<table>
<thead>
<tr>
<th>cid</th>
<th>compilation_date</th>
<th>compilation_time</th>
<th>config_file</th>
<th>stdlog_file</th>
<th>errlog_file</th>
<th>output_file</th>
<th>core_size</th>
<th>dependencies</th>
<th>gcc_version</th>
<th>libc_version</th>
<th>core_used</th>
<th>incremental_mod</th>
<th>tuxx</th>
</tr>
</thead>
<tbody>
<tr>
<td>1464</td>
<td>2018-04-19 15:20:19</td>
<td>204.414</td>
<td>[BLOB - 22,7 Kio]</td>
<td>[BLOB - 33,2 Kio]</td>
<td>[BLOB - 330 Kio]</td>
<td>[BLOB - 3,3 Kio]</td>
<td>36313649</td>
<td>6.3.0-18+deb9u1</td>
<td>2.24-11+deb9u3</td>
<td>16</td>
<td>0</td>
<td>pre-i</td>
<td></td>
</tr>
<tr>
<td>1463</td>
<td>2018-04-19 15:19:23</td>
<td>122.739</td>
<td>[BLOB - 18,6 Kio]</td>
<td>[BLOB - 25,1 Kio]</td>
<td>[BLOB - 205 Kio]</td>
<td>[BLOB - 2,9 Kio]</td>
<td>17455804</td>
<td>6.3.0-18+deb9u1</td>
<td>2.24-11+deb9u3</td>
<td>16</td>
<td>0</td>
<td>pre-i</td>
<td></td>
</tr>
<tr>
<td>1462</td>
<td>2018-04-19 16:18:51</td>
<td>82.1942</td>
<td>[BLOB - 17 Kio]</td>
<td>[BLOB - 18,8 Kio]</td>
<td>[BLOB - 286 Kio]</td>
<td>[BLOB - 3 Kio]</td>
<td>30085248</td>
<td>6.3.0-18+deb9u1</td>
<td>2.24-11+deb9u3</td>
<td>16</td>
<td>0</td>
<td>pre-i</td>
<td></td>
</tr>
<tr>
<td>1461</td>
<td>2018-04-19 15:14:59</td>
<td>108.779</td>
<td>[BLOB - 19,7 Kio]</td>
<td>[BLOB - 19,1 Kio]</td>
<td>[BLOB - 132 Kio]</td>
<td>[BLOB - 3,3 Kio]</td>
<td>24138304</td>
<td>6.3.0-18+deb9u1</td>
<td>2.24-11+deb9u3</td>
<td>16</td>
<td>0</td>
<td>pre-i</td>
<td></td>
</tr>
<tr>
<td>1460</td>
<td>2018-04-19 15:12:37</td>
<td>168.36</td>
<td>[BLOB - 20,1 Kio]</td>
<td>[BLOB - 26,5 Kio]</td>
<td>[BLOB - 2,9 Kio]</td>
<td>[BLOB - 3,3 Kio]</td>
<td>62715560</td>
<td>6.3.0-18+deb9u1</td>
<td>2.24-11+deb9u3</td>
<td>16</td>
<td>0</td>
<td>pre-i</td>
<td></td>
</tr>
<tr>
<td>1459</td>
<td>2018-04-19 15:09:17</td>
<td>204.448</td>
<td>[BLOB - 29,9 Kio]</td>
<td>[BLOB - 30,7 Kio]</td>
<td>[BLOB - 14]</td>
<td>[BLOB - 2,9 Kio]</td>
<td>108303064</td>
<td>6.3.0-2.24-16</td>
<td>16</td>
<td>0</td>
<td>pre-i</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

o1 : {true, false}
o2 : {true, false}
o3 : [0..10]
Enormous configurations space eg Linux has thousands of options, tri-state values \{y, n, m\}
Exercise: The case of Linux

- Sampling
  - translating Kconfig files into SAT: incomplete/unsound
  - Uniform, random sampling Plazar et al. ICST’19
    - Either does not scale
    - Or is not really random

- Measuring
- Learning
- Applicability
- Research questions
Conclusion

• Software variability everywhere for fitting users’ requirements; combinatorial explosion

• **AI#1** Abstraction/languages to formally and efficiently reason about configuration spaces
  – with SAT/CSP/SMT solvers, Eg constrained sampling

• **AI#2** Statistical machine learning (out of a sample):
  – Understand the configuration space, find the best configuration, specialize the configuration space (e.g., by capturing constraints)
  – In a cost-effective way

• Artificial intelligence for fitting software variability

• Human/machines interplay
Software Variability and EJCP

• Empirical Software Engineering
  – We aim to understand real-world variability (data)
  – We aim to develop techniques that are effective on real-world systems

• Constraint Programming
  – SAT/SMP/CP solvers to reason about variability

• Coccinelle and the Linux kernel: a challenging case study for software variability

• Formal verification: many papers on verifying software variants (Thuem et al. ACM Survey 2014)
Possible impacts of software variability on your work

• When performing/reporting scientific experiments
  – (hyper-)parameters? heuristics? default configuration? versions?
  – many independent variables have variability

• When verifying a system
  – does it work on any configuration? on a very specific one? can you adapt your verification for many variants?

• More positively (forget problems!): variability can be a fantastic opportunity to explore and understand a space of possible solutions
Advanced topics
Context and Variability

Performance objectives

Cameras/Sensors

Country (e.g., marketing driven)

Hardware

Lighting Conditions

Contextual Variability

Software Variability

Learning
Contextual Variability Models

Automated Specialization

- Problem: configuring a system is hard
  - combinatorial explosion
  - functional concerns and performance qualities
  - users want to have a maximum of flexibility and perform no configuration error
- Configuration « envelope »
  - Safety (beware of being too permissive)
  - Flexibility (beware of being too restrictive)
- Solution: all option values (and combinations thereof) presented to users should satisfy an “objective”

 variability model (VM)
  o1 : {true, false}
  o2 : {true, false}
  o3 : [0..10]

 Sampling
 Testing
 Learning

 specialized variability model (VM')
  o1 = false
  o2 = true
  o3 : [2..13]

Automated Specialization

x264  --quiet
       --no-progress
       --no-asm
       --rc-lookahead 60
       --ref 9
       -o trailer_480p24.x264
       trailer_2k_480p24.y4m
Configuration space

\[
\begin{align*}
o1 &: \{\text{true, false}\} \\
o2 &: \{\text{true, false}\} \\
o3 &: [0..10]
\end{align*}
\]
Configuration Space
Configuration Space
c10999

x264 --quiet
   --no-mbtree=false
   --no-asm
   --cfr-ratio 18
   --b_bias -50
   -o trailer_480p24.x264
trailer_2k_480p24.y4m

c2

x264 --quiet
   --no-mbtree
   --no-asm
   --cfr-ratio 28
   --b_bias 50
   -o trailer_480p24.x264
trailer_2k_480p24.y4m
I want an execution time < 145s
I want an execution time < 145s
Automated specialization problem:
synthesizing constraints such that
each configuration meets an objective
(you have typically to execute the configuration to know that)

Performance objective: "speed < 145"

SPECIALIZATION

\[ VM' = VM \land cfrRatio > 26.7 \land no_mbtree = false \]
320 features

more variants than estimated atoms in the universe

Impossible to execute and test all configurations

I want an execution time < 145s
I want an execution time $< 145s$

Fig. 2: Number of x264 configurations running under a certain time: X-axis represents a number of configurations; Y-axis represents the execution speed (in seconds) to encode a video benchmark; e.g., about 25994 configurations can encode the video in less than 145.01 seconds.
Sampling, Testing, and Learning
Learning-Based Specialization
Variability Model (VM) → Sampling → Constraints → Machine Learning → Execution (or derivation) and Measurement → Oracle → Objective

Performance objective: "speed < 145"

SPECIALIZATION

VM' = VM
^ cfrRatio > 26.7
^ no_mbtree = false
Problem reduction: a binary classification problem
Learning approach: decision trees (classification trees)
Classification trees and constraints

Why decision trees?
++ Can handle categorical and numerical values
++ Constraints can be extracted into logics
++ Human-readable constraints
Classification trees and constraints

```
1 !(crf_ratio <= 26.7)
2 !( crfRatio > 26.7 & no_mbtree > 0 )
```
Specialization of the configuration set

Can discard lots of non-acceptable configurations (safer)
But can also be too restrictive (losing flexibility)
Specialization of the configuration set

Can identify false positives or false negatives ("missing" flexibility or safety)
Can identify false positives or false negatives ("missing" flexibility or safety)
Evaluation

What is the accuracy of our specialization method for classifying configurations?

What is the precision and recall of our specialization method for classifying configurations?

How safe and flexible are specialized configurable systems when applying our method?

How effective is our learning technique compared to a non-learning technique?
## Evaluation

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>Web Server</td>
<td>C</td>
<td>9/0</td>
<td>192</td>
<td>All</td>
</tr>
<tr>
<td>BerkeleyC</td>
<td>Database</td>
<td>C</td>
<td>18/0</td>
<td>2560</td>
<td>All</td>
</tr>
<tr>
<td>BerkeleyJ</td>
<td>Database</td>
<td>Java</td>
<td>26/0</td>
<td>400</td>
<td>181</td>
</tr>
<tr>
<td>LLVM</td>
<td>Compiler</td>
<td>C++</td>
<td>11/0</td>
<td>1024</td>
<td>All</td>
</tr>
<tr>
<td>SQLite</td>
<td>Database</td>
<td>C</td>
<td>39/0</td>
<td>$10^6$</td>
<td>4553</td>
</tr>
<tr>
<td>Dune</td>
<td>Solver</td>
<td>C++</td>
<td>8/3</td>
<td>2304</td>
<td>All</td>
</tr>
<tr>
<td>HIPA&lt;sup&gt;cc&lt;/sup&gt;</td>
<td>Image Proc.</td>
<td>C++</td>
<td>31/2</td>
<td>13485</td>
<td>All</td>
</tr>
<tr>
<td>HSMGP</td>
<td>Solver</td>
<td>n/a</td>
<td>11/3</td>
<td>3456</td>
<td>All</td>
</tr>
<tr>
<td>JavaGC</td>
<td>Runtime Env.</td>
<td>C++</td>
<td>12/23</td>
<td>$10^{31}$</td>
<td>166k</td>
</tr>
<tr>
<td>x264 (Energy)</td>
<td>Codec</td>
<td>C</td>
<td>8/12</td>
<td>$10^{27}$</td>
<td>69k</td>
</tr>
<tr>
<td>x264 (PSNR)</td>
<td>Codec</td>
<td>C</td>
<td>8/12</td>
<td>$10^{27}$</td>
<td>69k</td>
</tr>
<tr>
<td>x264 (SSIM)</td>
<td>Codec</td>
<td>C</td>
<td>8/12</td>
<td>$10^{27}$</td>
<td>69k</td>
</tr>
<tr>
<td>x264 (Speed)</td>
<td>Codec</td>
<td>C</td>
<td>8/12</td>
<td>$10^{27}$</td>
<td>69k</td>
</tr>
<tr>
<td>x264 (Size)</td>
<td>Codec</td>
<td>C</td>
<td>8/12</td>
<td>$10^{27}$</td>
<td>69k</td>
</tr>
<tr>
<td>x264 (Time)</td>
<td>Codec</td>
<td>C</td>
<td>8/12</td>
<td>$10^{27}$</td>
<td>69k</td>
</tr>
<tr>
<td>x264 (Watt)</td>
<td>Codec</td>
<td>C</td>
<td>8/12</td>
<td>$10^{27}$</td>
<td>69k</td>
</tr>
</tbody>
</table>

**TABLE 1:** Features: number of boolean features / number of numerical features; #$[VM]$: number of valid configurations; Meas.: number of configurations that have been measured.
Independent variables

• Subject systems
• Sampling size
• Performance objective
  – % of non-acceptable configurations
• For each subject system, we compute numerous metrics and perform a sensitivity analysis wrt sampling size and performance objective
Learning-Based Performance Specialization of Configurable Systems

Paul Temple, Mathieu Acher, Jean-Marc Jézéquel, Léo Noel-Baron
Univ Rennes, IRISA
Rennes, France
Emails: firstname.lastname@irisa.fr
José A. Galindo
University of Sevilla
Sevilla, Spain
Email: jagalindo@us.es

paper: https://hal.archives-ouvertes.fr/hal-01467299

Note: we are currently further experimenting with new data and algorithms
Main conclusions

• **High precision and recall** can be obtained with a relative **small number of configurations** with the exception of some "hard" objective values for which the configurable system can be seen as too permissive.

• Our approach can be effective to produce a **safe and flexible system** with a relative small number of configurations

• Even and especially for hard objectives, our specialization method **significantly outperforms a non-learning approach**
--psy-rc <float:float> Strength of psychovisual optimization ["1.0:0.0"]
  #1: RD (requires subopt-6)
  #2: Trellis (requires trellis, experimental)
--no-8x8dct Disable adaptive spatial transform size
-t, --trellis <integer> Trellis RD quantization. [1]
  - 0: disabled
  - 1: enabled only on the final encode of a MB
  - 2: enabled on all mode decisions
--nr <integer> Noise reduction [0]
--cqsmfile <string> Read custom quant matrices from a JM-compatible file

Input/Output:
-o, --output <string> Specify output file
--muxer <string> Specify output container format ["auto"]
  - auto, raw, m4v, flv
--demuxer <string> Specify input container format ["auto"]
  - auto, raw, y4m, avs
--input-fmt <string> Specify input file format (requires lavg support)
--input-csp <string> Specify input colorspace for raw input
--output-csp <string> Specify output colorspace ["I420"]
  - I420, I422, [444], rgb
--input-depth <integer> Specify input bit depth for raw input
--input-range <string> Specify input color range ["auto"]
  - auto, tv, pc
--input-res <int,int> Specify input resolution (width x height)
--index <string> Filename for input index file
--sar <width:height> Specify Sample Aspect Ratio
--framerate <floatfrac> Specify framerate
--seek <integer> First frame to encode
--frames <integer> Maximum number of frames to encode
--level <string> Specify level (as defined by Annex A)
--bluray-compat Enable compatibility hacks for Blu-ray support
--avc-intra-class <integer> Use compatibility hacks for AVC-intra class
  - 50, 100, 200
--stitchable Don’t optimize headers based on video content
  Ensures ability to recombine a segmented encode
Performance Prediction

x264 --no-progress
--no-asm
--rc-lookahead 60
--ref 9
-o trailer_480p24.x264
trailer_2k_480p24.y4m

40 seconds
Performance Prediction

x264 --no-mbtree  
--rc-lookahead 40  
--ref 9  
-o trailer_480p24.x264  
10 seconds
Performance Prediction

x264 ...
-o trailer_480p24.x264 trailer_2k_480p24.y4m

??? seconds
Performance Prediction

x264 --no-mbtree
--rc-lookahead 40
--ref 9
-o trailer_480p24.x264
trailer_2k_480p24.y4m

??? seconds

Regression problem (linear regression, regression tree, random forest, gradient boosting, SVM, etc.)
Guo et al. ASE 2013, Apel et al. ASE’15, Czarnecki et al. SPLC’15, Siegmund et al. FSE’15, Kastner et al. ASE’17, Menzies et al. FSE’17, Batory et al. FSE’17
Input Sensitivity and Transferability of Performance Prediction Models

What if I change the input video?
Can I reuse my performance prediction model?

---

x264 --no-mbtree
--rc-lookahead 40
--ref 9
-o trailer_480p24.x264
trailer_2k_480p24.y4m

---

x264 --no-mbtree
--rc-lookahead 40
--ref 9
-o football.x264
football.y4m

---

55 seconds

?? seconds