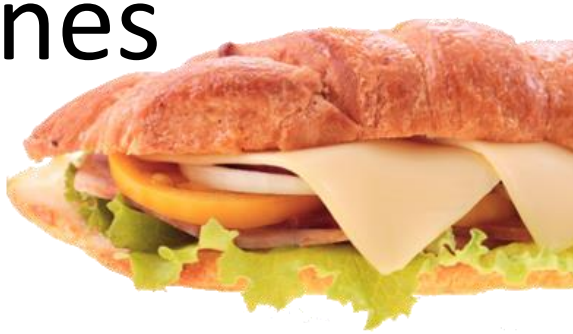
# Software Language Engineering Research @ Marburg

Klaus Ostermann
Philipps-Universität Marburg

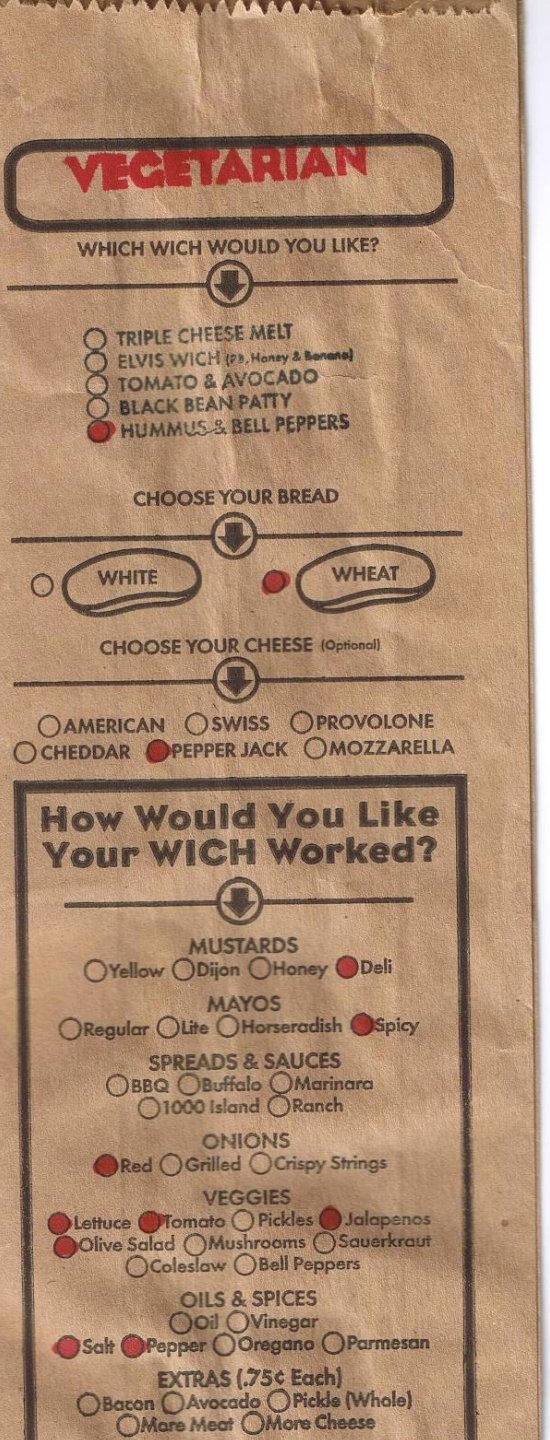
# Overview of the Talk

- Part I: Software Product Line Research
- Part II: DSL Research: SugarJ

# Software Product Lines



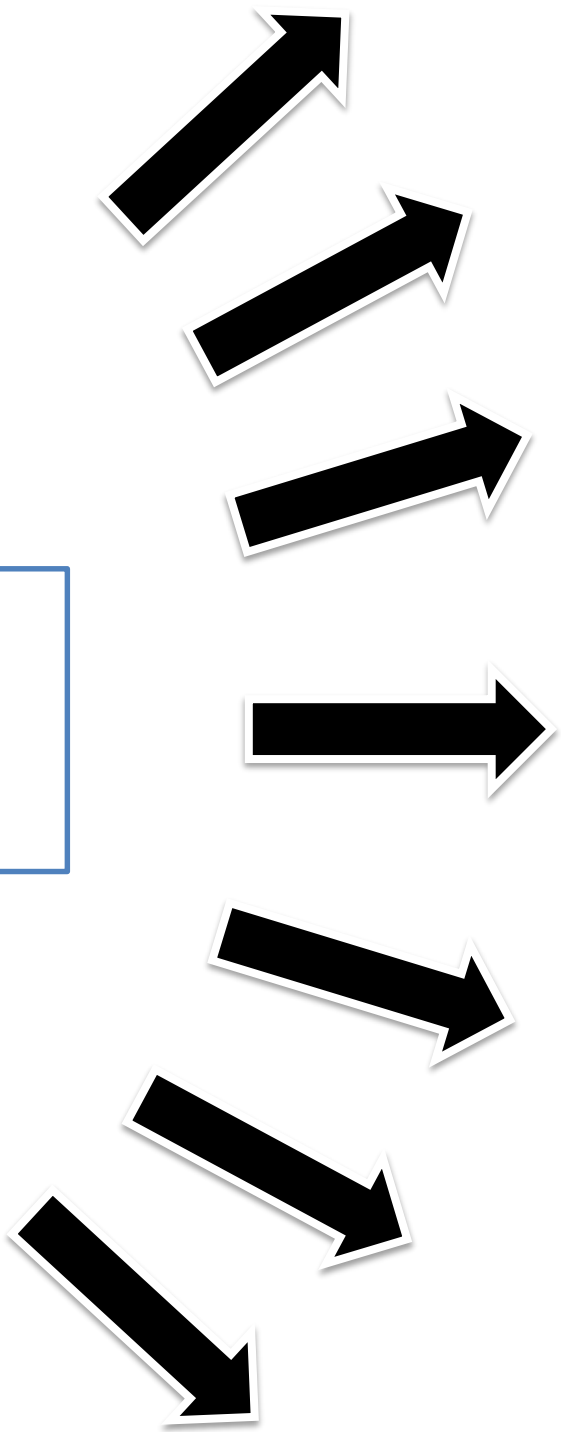
Make domain knowledge reusable by domain-specific abstractions




# Product Line Configuration

Printer Firmware

Linux
Kernel




```
.config - Linux Kernel v2.6.33.3 Configuration
================================================================================
============================ Processor type and features ============================
Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters
are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.
Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
excluded  <M> module  < > module capable

    [ ] Tickless System (Dynamic Ticks)
    [ ] High Resolution Timer Support
    [ ] Symmetric multi-processing support
    [ ] Support for extended (non-PC) x86 platforms
    [ ] Single-depth WCHAN output
    [ ] Paravirtualized guest support  --->
    [ ] Memtest
        Processor family (Generic-x86-64)  --->
        Preemption Model (No Forced Preemption (Server))  --->
    [ ] Reroute for broken boot IRQs (NEW)
    [ ] Machine Check / overheating reporting
    [ ] Dell laptop support
    [ ] /dev/cpu/microcode - microcode support
    [ ] /dev/cpu/*/msr - Model-specific register support
    [ ] /dev/cpu/*/cpuid - CPU information support
        Memory model (Sparse Memory)  --->
    [*] Sparse Memory virtual memmap (NEW)
    [ ] Allow for memory hot-add (NEW)
    [ ] Enable KSM for page merging
    (4096) Low address space to protect from user allocation
    [ ] Check for low memory corruption
    [ ] Reserve low 64K of RAM on AMI/Phoenix BIOSen
    -*- MTRR (Memory Type Range Register) support
    [ ]   MTRR cleanup support
    [ ] Enable seccomp to safely compute untrusted bytecode
    [ ] Enable -fstack-protector buffer overflow detection (EXPERIMENTAL)
        Timer frequency (250 HZ)  --->
    [ ] kexec system call
=====v(+)=====================================================================
================================================================================
                      <Select>    < Exit >    < Help >
```

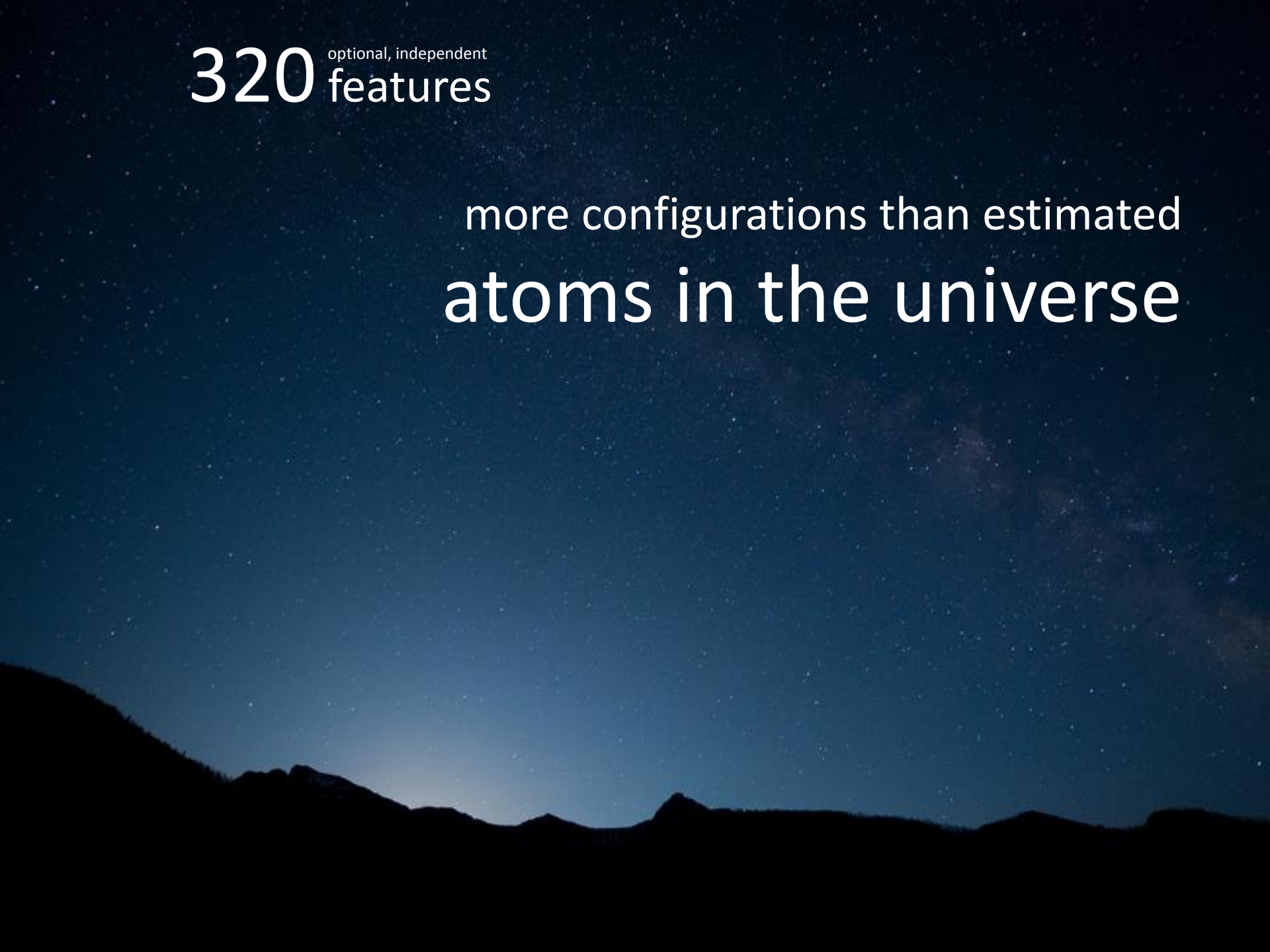
# Software Product Lines in Industry

Boeing
Bosch Group
Cummins, Inc.
Ericsson
General Dynamics
General Motors
Hewlett Packard
Lockheed Martin
Lucent
Nokia
Philips
Siemens
Toshiba
…

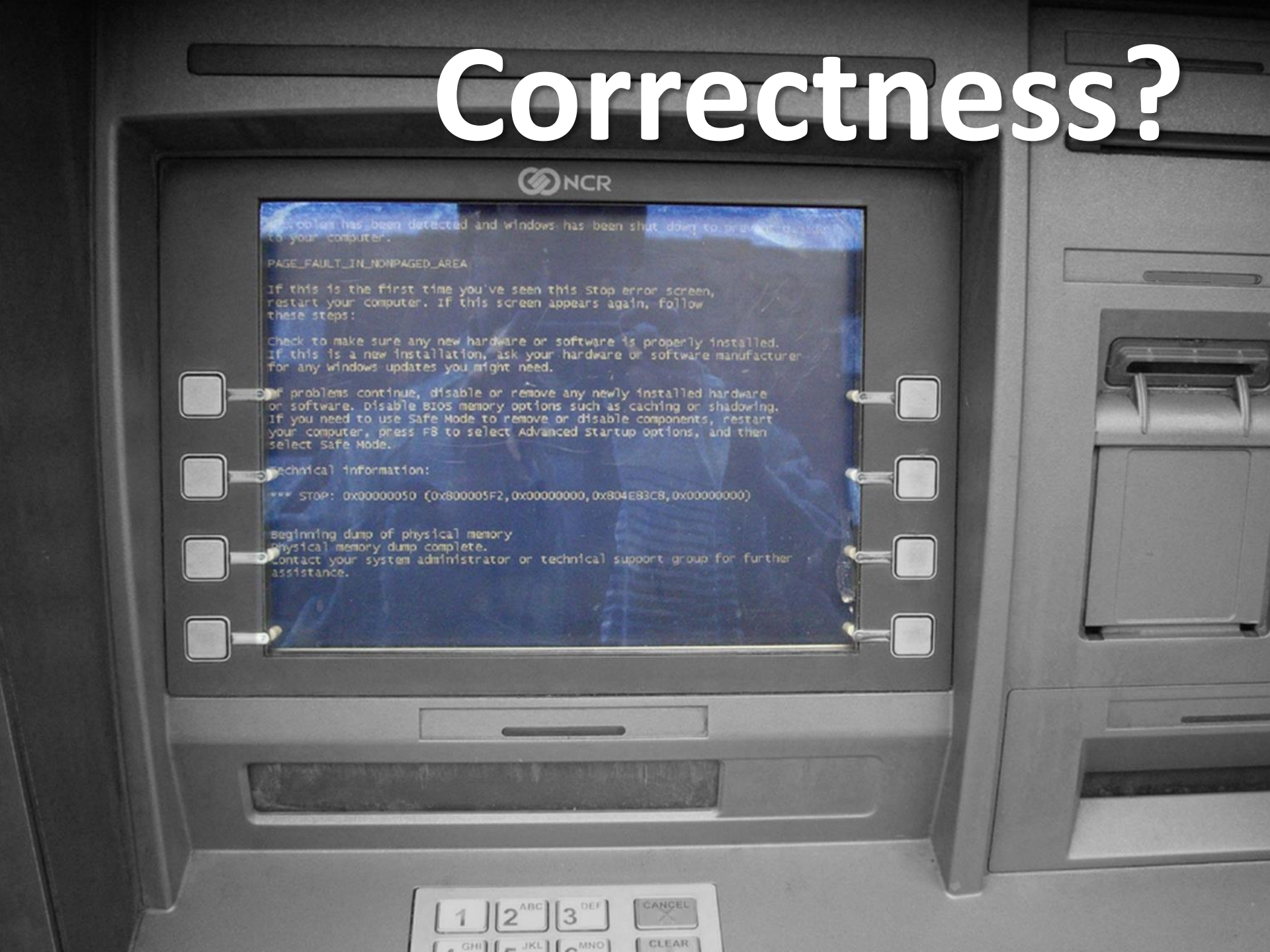
33 optional, independent features

a unique configuration for every

# person on this planet


320 optional, independent features
more configurations than estimated
atoms in the universe

# Correctness?



```
static int _rep_queue_filedone(
   DB_ENV *dbenv,
   REP *rep,
   __rep_fileinfo_args *rfp) {
#ifdef NO_QUEUE
   COMPQUIET(rep, NULL);
   COMPQUIET(rfp, NULL);
   return (__db_no_queue_am(dbenv));
#else
   db_pgno_t first, last;
   u_int32_t flags;
   int empty, ret, t_ret;
#ifdef DIAGNOST
   DB_MSGBUF mb
#endif
   // over 100
}
#endif
```

Excerpt from
Oracle's Berkeley DB

```
#ifdef X
void foo();
#endif

void bar() {
  foo();
}
```

# Conditional Compilation

# Objections / Criticism

Designed in the 70th and hardly evolved since

"#ifdef considered harmful"

"#ifdef hell"

"maintenance becomes a 'hit or miss' process"

"is difficult to determine if the code being viewed is actually compiled into the system"

"incomprehensible source texts"

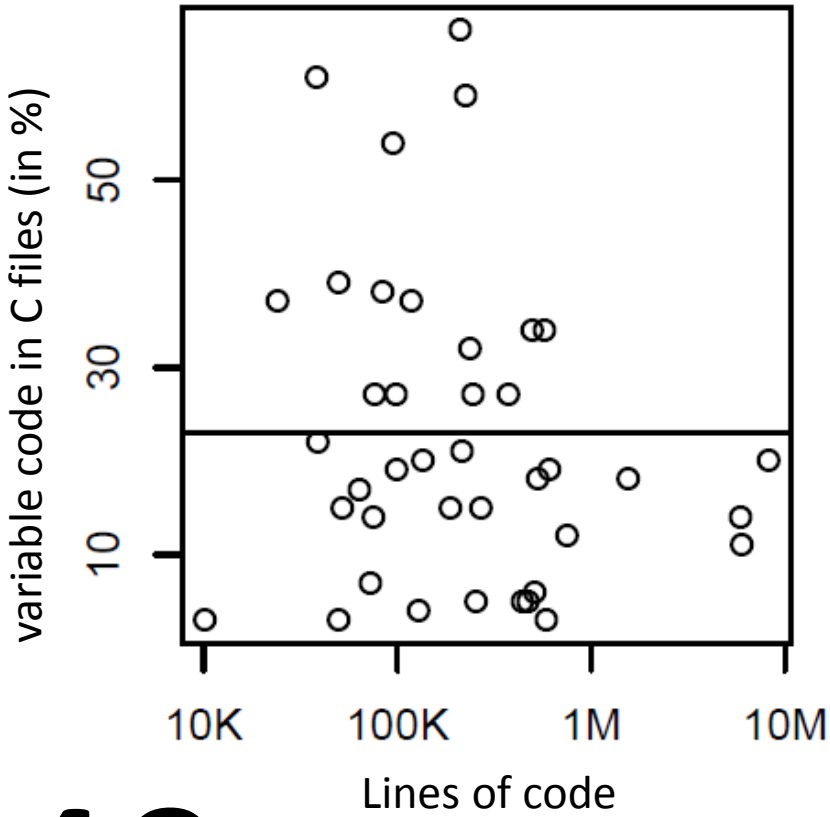
"CPP makes maintenance difficult"

"programming errors are easy to make and difficult to detect"

"source code rapidly becomes a maze"

"preprocessor diagnostics are poor"

# Practical relevance of CPP-based SPLs




apache, berkely db,
cherokee, clamav, dia,
emacs, freebsd, gcc,
ghostscript, gimp, glibc,
gnumeric, gnuplot, irssi,
libxml, lighttpd, linux, lynx,
minix, mplayer, mpsolve,
openldap, opensolaris,
openvpn, parrot, php, pidgin,
postgresql, privoxy, python,
sendmail, sqlite, subversion,
sylpheed, tcl, vim, xfig,
xine-lib, xorg-server, xterm

**40 Open-Source C Projects**

# Handling Complexity


## Inherent Complexity

```
#include <stdio.h>

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

main() {
  printf(msg);
}
```


## SAT Problem

# Variability-Aware Analysis

**Parser**

Type System

Static Analysis

Bug Finding

Testing

Model Checking

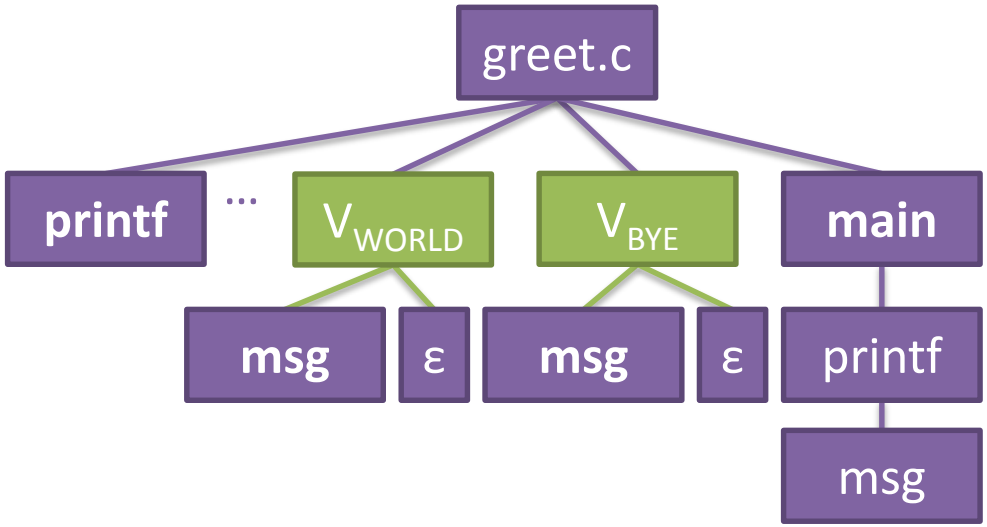
Theorem Proving

…

# Variability-Aware Parsing

[OOPSLA'11]

```
#include <stdio.h>

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

main() {
  printf(msg);
}
```




AST with Variability Information

# Parsing C without Preprocessing

Macro expansion needed for parsing

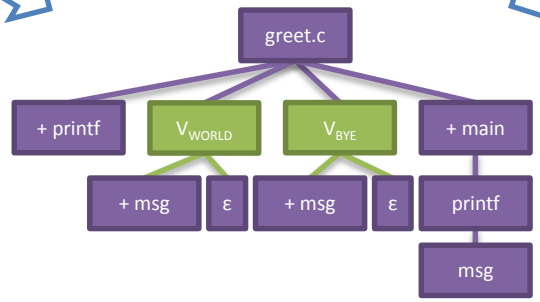
```
#define P(msg) \
    printf(msg);

main() {
  P("Hello\n")
  P("World\n")
}
```

Undisciplined annotations

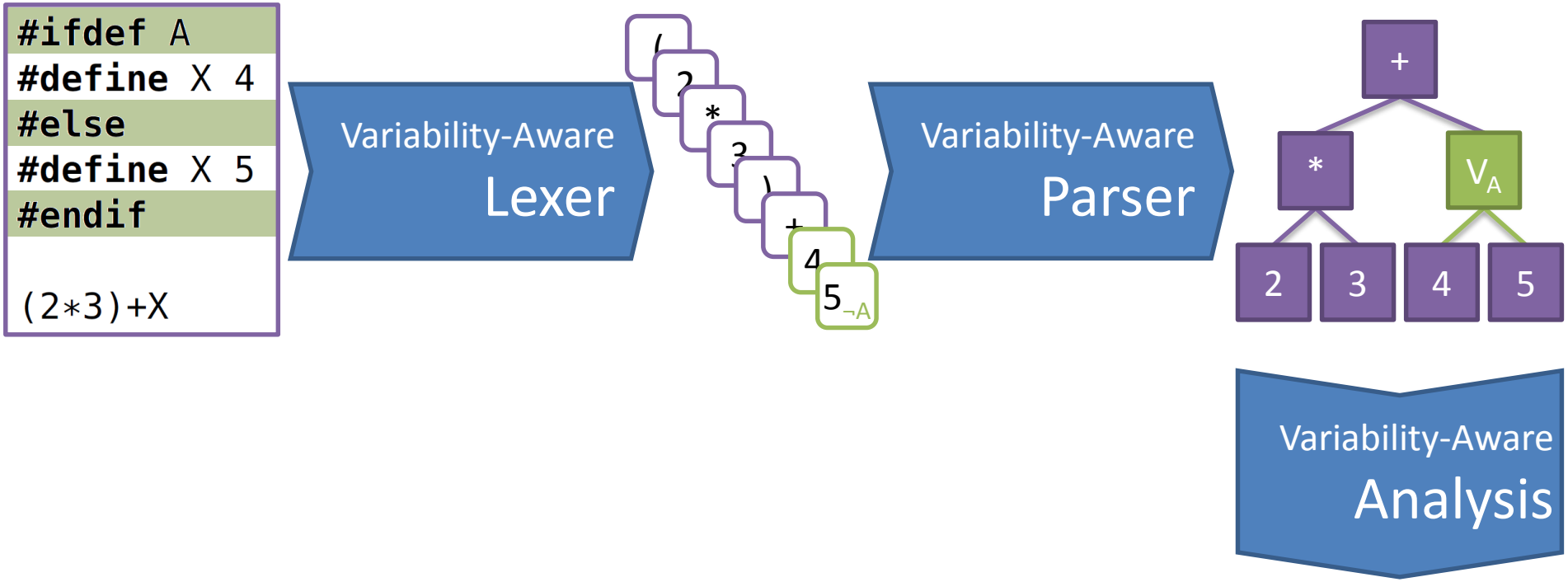
```
if (!initialized)
#ifdef DYNAMIC
  if (enabled)
#endif
    init();
```

Alternative macros

```
#ifdef BIGINT
#define SIZE 64
#else
#define SIZE 32
#endif

allocate(SIZE);
```

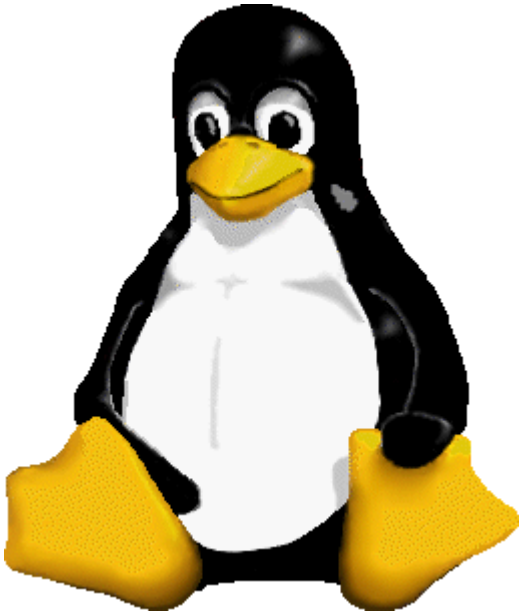


# Variability-Aware Parsing [OOPSLA'11]



Real-world C code
C preprocessor
10000 features
6 million lines of GnuC code

Parsed all variants successfully
Reasonably efficient
Found parse errors



Our main test case

# Variability-Aware Analysis

Parser ✓
Type System
Module System
Static Analysis
Bug Finding
Testing
IDE Support
Refactoring Tools
Model Checking
Theorem Proving
…

# Variability-Aware Modules [OOPSLA'12]

## Module interfaces reflect variability!

```
import select: String -> Table
       update: String -> Table if !READONLY
       read: Tid -> Rec

variability XQUERY=>INDEX;

export storeXML: Xml -> Tid if !READONLY
       query: String -> Xml if XQUERY
```

# Variability-Aware Modules [OOPSLA'12]

- Formalized module system
- Proved correctness

**Additional notation:**

| | |
|---|---|
| $f \in F$ | configuration options |
| $c \in C = 2^F$ | configurations |
| $v \in V = 2^C$ | variability models |
| $\Gamma \in C \to X \to T$ | variable contexts |
| $\Delta \in C \to X \to E \times T$ | variable definitions |
| $m = (v, \Gamma, \Delta) \in M^v$ | module |

**Auxiliary functions:**

$$Sig : (C \to X \to E \times T) \to (C \to X \to T)$$
$$Sig(\Delta)(c)(x) = t \quad \text{where } \Delta(c)(x) = (e, t)$$

**Module typing:**

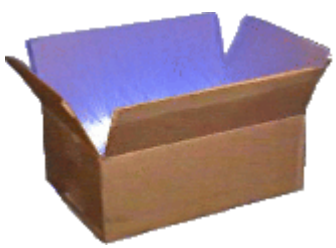
$$\frac{v \subseteq dom(\Gamma) \qquad v \subseteq dom(\Delta) \qquad \forall c \in v.\ (\Gamma(c), \Delta(c))\ \texttt{OK} \qquad v \neq \emptyset}{(v, \Gamma, \Delta)\ \texttt{OK}}$$

$$\begin{aligned}
conflictpresence(\Gamma_1, \Gamma_2) &= \{c \in dom(\Gamma_1) \cap dom(\Gamma_2) \mid dom(\Gamma_1(c)) \cap dom(\Gamma_2(c)) \neq \emptyset\} \\
conflicttype(\Gamma_1, \Gamma_2) &= \{c \in dom(\Gamma_1) \cap dom(\Gamma_2) \mid \exists x \in dom(\Gamma_1(c)) \cap dom(\Gamma_2(c)).\ \Gamma_1(c)(x) \neq \Gamma_2(c)(x)\} \\
conflict(\Gamma_1, \Delta_1, \Gamma_2, \Delta_2) &= \bigcup \left\{ \begin{array}{l} conflictpresence(Sig(\Delta_1), Sig(\Delta_2)), conflicttype(\Gamma_1, \Gamma_2), \\ conflicttype(\Gamma_1, Sig(\Delta_2)), conflicttype(Sig(\Delta_1), \Gamma_2) \end{array} \right\}
\end{aligned}$$

**Module compatibility and composition:**

$$\frac{v' = \bigcup_{x \neq y} conflict(\Gamma_x, \Delta_x, \Gamma_y, \Delta_y) \qquad v = v_1 \cap \ldots \cap v_n \qquad v \backslash v' \neq \emptyset}{\div\{(v_1, \Gamma_1, \Delta_1), \ldots, (v_n, \Gamma_n, \Delta_n)\}}$$

$$\frac{\begin{array}{c} v' = v_1 \cap v_2 \setminus conflict(\Gamma_1, \Delta_1, \Gamma_2, \Delta_2) \\ \Gamma'(c) = \Gamma_1(c) \cup \Gamma_2(c) \setminus (sig(\Delta_1(c)) \cup sig(\Delta_2(c))) \\ \Delta'(c) = \Delta_1(c) \cup \Delta_2(c) \end{array}}{(v_1, \Gamma_1, \Delta_1) \bullet (v_2, \Gamma_2, \Delta_2) = (v', \Gamma', \Delta')}$$



- Implemented module system for C and CPP
- Implemented interface inference
- Type-checked BusyBox Product Line
- Found real bugs
- Next up: Type-checking all variants of Linux

# Variability-Aware Analysis

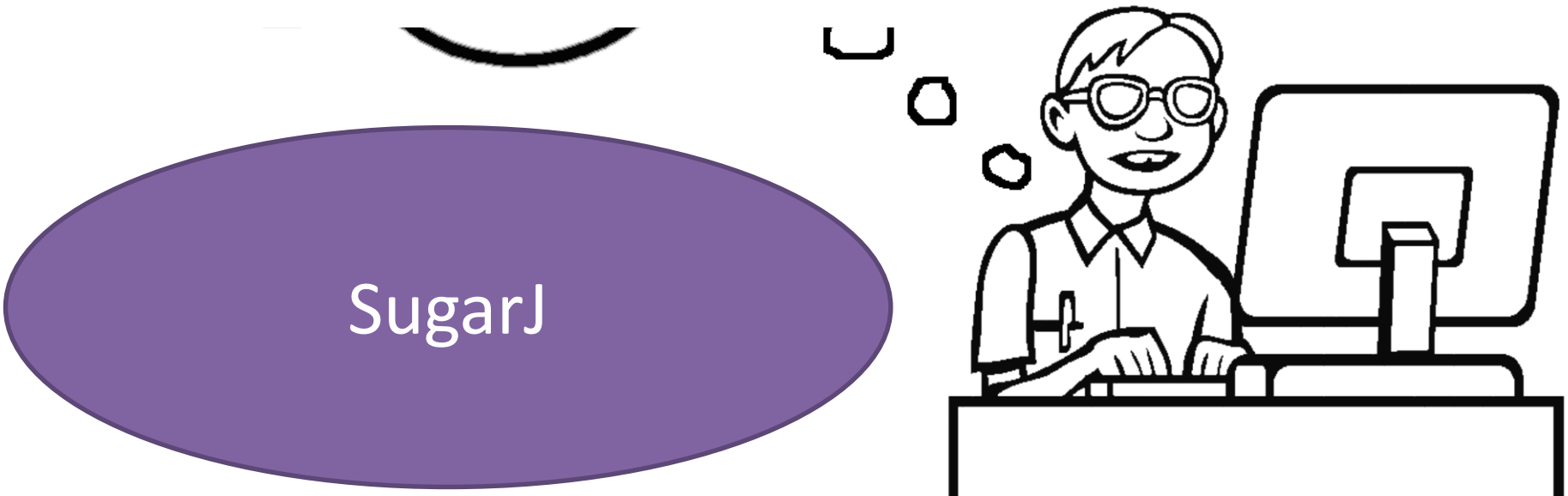
- Parser ✓
- Type System ✓
- Module System ✓
- Static Analysis
- Bug Finding
- Testing
- IDE Support
- Refactoring Tools
- Model Checking
- Theorem Proving
- …

# Overview of the Talk

- Part I: Software Product Line Research
- Part II: DSL Research: SugarJ

## Goals

- domain-specific syntax
- language composition
- extensible language-definition mechanism




[OOPSLA‘11,GPCE‘11,SLE‘12,HASKELL‘12,SLE‘13,...]



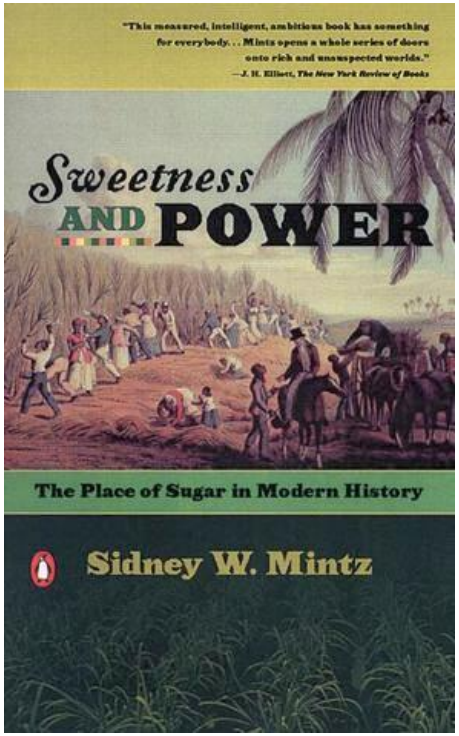


```
import Pair
import Reg

public clas
  private (St
    ("/Users/s
}
```

```
import Pairs;

public class Test {
  private (String, Integer) p = ("12", 34);
}
```

# Data serialization with XML

task: serialize information about books using XML




serialize

```
<book title="Sweetness and Power">
  <author name="Sidney W. Mintz" />
  <editions>
    <edition year="1985"
             publisher="Viking Press" />
    <edition year="1986"
             publisher="Penguin Books" />
  </editions>
</book>
```

# Example: XML serialization

## in Java using SAX

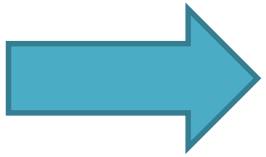
```
public void appendBook(ContentHandler ch) {
  String title = "Sweetness and Power";
  ch.startDocument();
  AttributesImpl bookAttrs = new AttributesImpl();
  bookAttrs.addAttribute("", "title", "title", "CDATA", title);
  ch.startElement("", "book", "book", bookAttrs);
  AttributesImpl authorAttrs = new AttributesImpl();
  authorAttrs.addAttribute("", "name", "name", "CDATA", "Sidney W. Mintz”);
  ch.startElement("", "author", "author", authorAttrs);
  ch.endElement("", "author", "author");
  ch.startElement("", "editions", "editions", new AttributesImpl());
  AttributesImpl edition1Attrs = new AttributesImpl();
  edition1Attrs.addAttribute("", "year", "year", "CDATA", "1985");
  edition1Attrs.addAttribute("", "publisher", "publisher", "CDATA", "Viking");
  ch.startElement("", "edition", "edition", edition1Attrs);
  ch.endElement("", "edition", "edition");
  ch.endElement("", "editions", "editions");
  ch.endElement("", "book", "book");
  ch.endDocument();
}
```

# XML in SugarJ

```
import XML;

public void appendBook(ContentHandler ch) {
  String title = "Sweetness and Power";

  ch.<book title="{title}">
      <author name="Sidney W. Mintz" />
      <editions>
        <edition year="1985" publisher="Viking Press" />
        <edition year="1986" publisher="Penguin Books" />
      </editions>
    </book>;
}
```

# Sugar libraries


```
ch.<book title="Sweetness and P
    <author name="Sidney W. Mi
    <editions>
      <edition year="1985" pub
      <edition year="1986" pub
    </editions>
  </book>
```



```
ch.startDocument();
AttributesImpl bookAttrs = new AttributesImpl();
bookAttrs.addAttribute("", "title", "title", "CDATA", "Sweetn
ch.startElement("", "book", "book", bookAttrs);
AttributesImpl authorAttrs = new AttributesImpl();
authorAttrs.addAttribute("", "name", "name", "CDATA", "S
ch.startElement("", "author", "author", authorAttrs);
ch.endElement("", "author", "author");
ch.startElement("", "editions", "editions", new AttributesIm
AttributesImpl edition1Attrs = new AttributesImpl();
edition1Attrs.addAttribute("", "year", "year", "CDATA", "19
edition1Attrs.addAttribute("", "publisher", "publisher", "CD
```

```
public sugar Pairs {

  context-free syntax
    "(" JavaExpr "," JavaExpr ")" -> JavaExpr


  rules
    pair-desugaring
      |[ (~e1, ~e2) ]|

  desugarings
    pair-desugaring
}
```

← SDF

```
import Pairs;

public class Test {
  private (String, Integer) p = ("12", 34);
}
```

← Stratego

```
private (String, Integer) p = ("12", 34);
```

Desugar

```
private Pair<String, Integer> p = new Pair("12", 34);
```

libraries are self-applicable

# Self-applicability

language embeddings can build on other language embeddings


```
{
  book : {
    title : "Sweetness"
    author : {
      name : "Sidney"
    }
    editions : { … }
  }
}
```

desugar

```
<book
  title="Sweetness">
  <author
    name="Sidney" />
  <editions> …
  </editions>
</book>
```

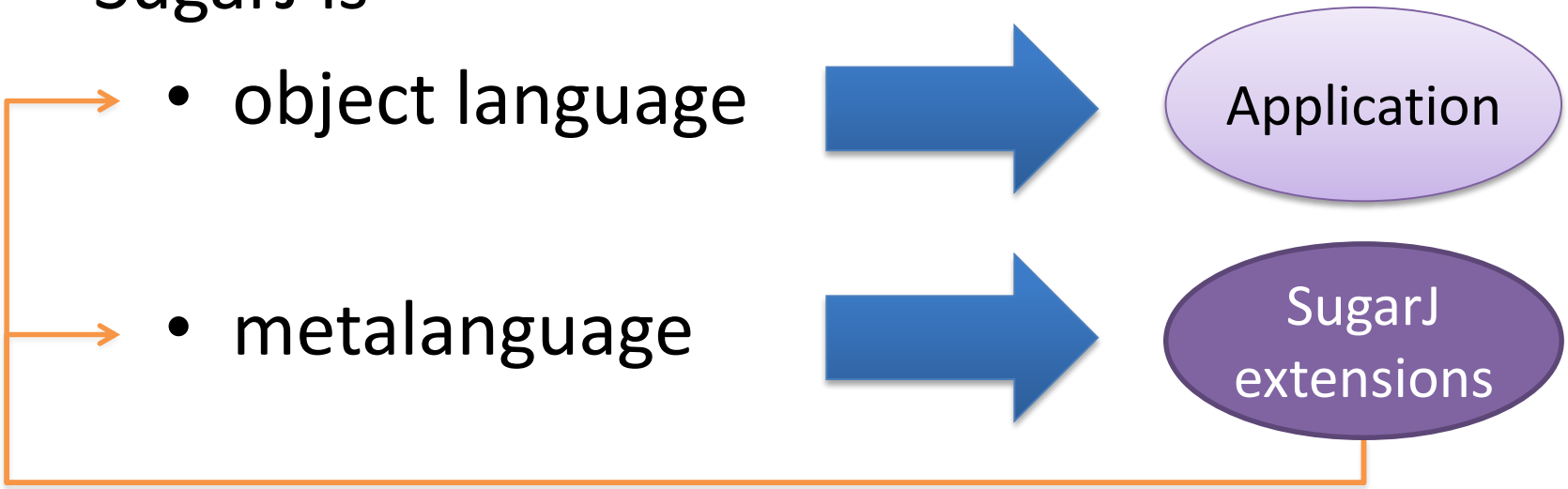
desugar

```
ch.startDocument();
AttributesImpl bookAttrs = new AttributesImpl();
bookAttrs.addAttribute("", "title", "title", "CDATA", "Swe
ch.startElement("", "book", "book", bookAttrs);
AttributesImpl authorAttrs = new AttributesImpl();
authorAttrs.addAttribute("", "name", "name", "CDATA",
ch.startElement("", "author", "author", authorAttrs);
ch.endElement("", "author", "author");
ch.startElement("", "editions", "editions", new Attributes
AttributesImpl edition1Attrs = new AttributesImpl();
edition1Attrs.addAttribute("", "year", "year", "CDATA", "
edition1Attrs.addAttribute("", "publisher", "publisher", "
ch.startElement("", "edition", "edition", edition1Attrs);
ch.endElement("", "edition", "edition");
ch.endElement("", "editions", "editions");
ch.endElement("", "book", "book");
ch.endDocument();
```

# Metalevels and SugarJ

SugarJ is

- object language
- metalanguage




libraries can affect both

# Metalanguage extensions

- alternative syntax for grammars
- meta-DSL: XML Schema

ANTLR

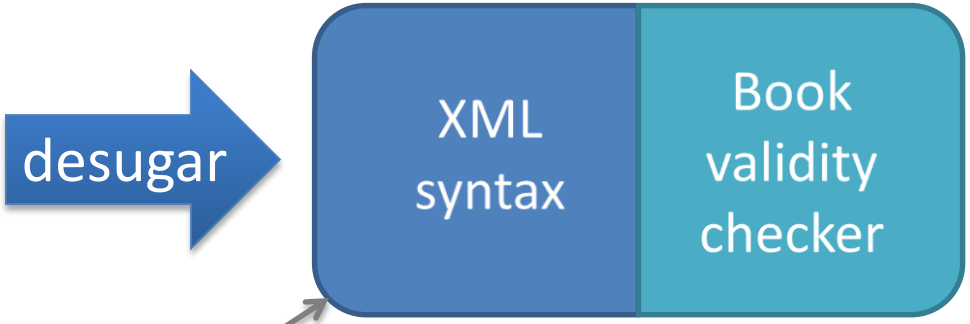
```
prog   :   stat+ ;
stat   :       expr NEWLIN
       |       ID '=' expr
       |       NEWLINE                    ;
```

desugar

```
rules
  pair-desugaring :
    PExpr(e1, e2) ->
    NewInstance(
      None(),
      ClassOrInterfaceType(TypeName(Id("Pair")), None()),
      [e1, e2],
      None())
```

# XML Schema

```
<xsd:schema targetNamespace="lib">
  <xsd:element name="book" type="B
  <xsd:complexType name="Book">
    <xsd:choice maxOccurs="unbound
      <xsd:element name="author" t
      <xsd:element name="editions"
    </xsd:choice>
    <xsd:attribute name="title" ty
  </xsd:complexType>
</xsd:schema>
```




```
import BookSchema;

ch.<book title="{title}">
     <author name="Sidney W. Mintz" />
     <editions>
       <edition year="1985" publisher="Viking Press" />
       <edit    year="1986" publisher="Penguin Books" />
     </editions>
   </book>;
```

!

!

# Editor Libraries [GPCE '11]

er.sugj – Eclipse – /Users/seba/tmp/ecli...

BookSchema.sugj | *BookHandler.sugj

```
import xml.Sugar;
import xml.Editor;
import xml.schema.BookSchema;

public class BookHandler {
  public void appendBook(ContentHandler ch) throws SAXException {
    String title = "Sweetness and Power";
    @Validate
    ch.<{lib}book title="{new String(title)}">
        <{lib}author name="Sidney W. Mintz" />
        <{lib}editions>
          <{lib}edition year="1985" publisher="Viking Press" />
          <{lib}edit    year="1986" publisher="Penguin Books" />
        </{lib}editions>

      <{lib}author
      <{lib}book
      <{lib}edition
      <{lib}editions
```

Outli

- BookHandler
  - appendBook
    - book
      - author
      - editions
  - isPublished
  - getLanguage

Problems | Search

1 error, 1 warnin

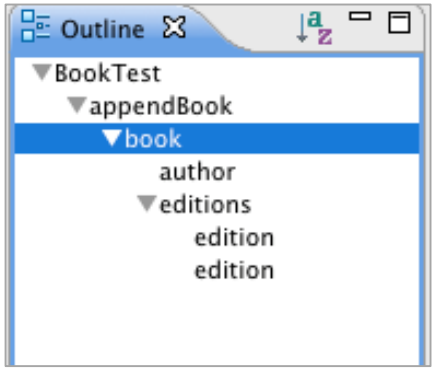
| Description | Resource | Location |
|---|---|---|
| Errors (1 item) | | |
| expected element edition of namespace lib | BookHandler.sugj | line 18 |
| Warnings (1 item) | | |
| skipping validation of quoted attribute value | BookHandler.sugj | line 14 |

Writable | Smart Insert

# XML editor libraries

```
package xml;

import editor.Colors;
import editor.Editor;
import xml.XmlSyntax;

public editor services Editor {
  colorer
    ElemName : blue        (recursive)
    AttrName : darkorange  (recursive)
    AttValue : darkred     (recursive)
    CharData : black       (recursive)

  folding
    Element

  outliner
    Element
}
```

```
ch.<book title="{new String(title
    <author name="Sidney W. Mint
    <editions>
      <edition year="1985" publi
      <edition year="1986" publi
    </editions>
  </book>;
```



# Turning Eclipse into a powerful LaTeX editor [GPCE'11]

Java – strategoxt-sugar-papers/editorlibs/Document.sugj – Eclipse – /Users/seba/tmp/...

Transform

*Document.sugj

```
The generated editor services may well be of auxilia
from method names to the documentation of these meth
editor service can query to display documentation of
In general, Sugarclipse utilizes the transformation
Stratego~\cite{} for auxiliary editor services, and
brings the gene
```

AotoYT09
AxelssonHL08
BachrachP01
BatoryLS98
BlumeA99
BrabrandS02
Bracha04
BravenboerDV10
BravenboerV04
BravenboerV08
CalderKMR03
Cardelli1994

```
\begin{figure}
\begin{lstsugar
import xml.sche

public xmlschem
  <xsd:schema t
    <xsd:elemen

    <xsd:complexType name="Book">
      <xsd:choice maxOccurs="unbounded">
        <xsd:element name="author" type="Person" />
        <xsd:element name="editions" type="Editions"
```

Outline

- Document
  - Introduction
  - An overview of Sugarclipse
  - Editor libraries
    - Domain-specific editor c
    - Staged editor libraries
    - Self-applicability
  - Editor composition
  - Technical realization
    - Architecture
    - Incremental parsing
    - Dynamic loading of edito
  - Case studies
  - Discussion
  - Related work
  - Conclusion and future work

Writable | Smart Insert