

Modeling and managing variability of software intensive systems

Overview and Principles

Mathieu Acher

Maître de Conférences

mathieu.acher@irisa.fr

Material

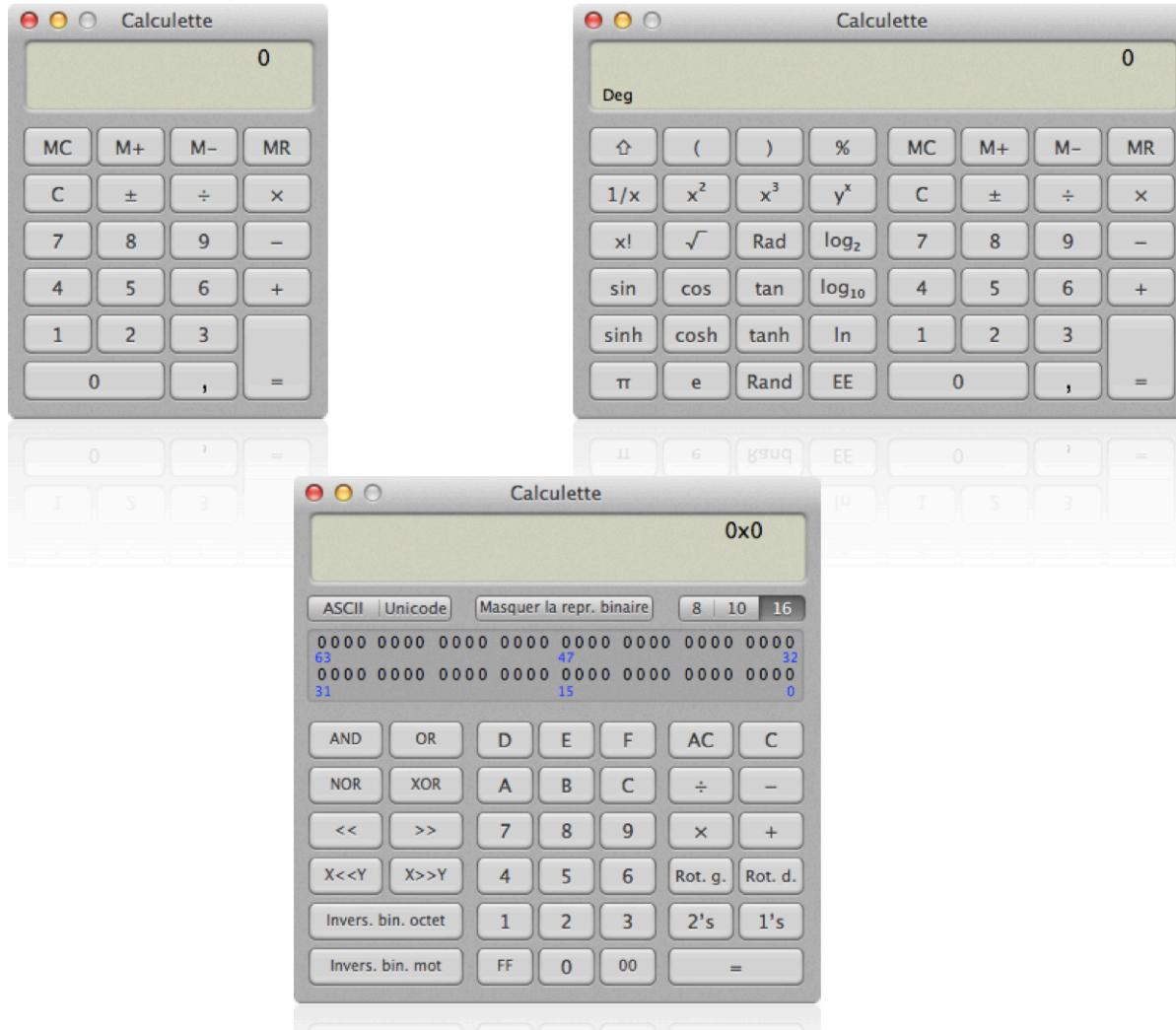
<http://mathieuacher.com/teaching/SPL/>

« A set of programs is considered to constitute a **family**, whenever it is worthwhile to study programs from the set by **first studying the common properties** of the set and then determining the **special properties** of the individual family members »



aka Variability

David L. Parnas — “On the design and development of program families” in Transactions on Software Engineering, SE-2(1):1–9, 1976³



**Starter****Home Premium Upgrade****Professional Upgrade****Ultimate Upgrade**

\$119.99*

[Buy](#)

\$199.99*

[Buy](#)

\$219.99*

[Buy](#)

Communication

Bluetooth support	✓	✓	✓	✓
Join a homegroup	✓	✓	✓	✓
Internet Explorer 8	✓	✓	✓	✓
View Available Networks	✓	✓	✓	✓
Windows Connect Now (WCN)	✓	✓	✓	✓
Create a homegroup		✓	✓	✓
Location and other sensors support		✓	✓	✓
Support for joining domains			✓	✓

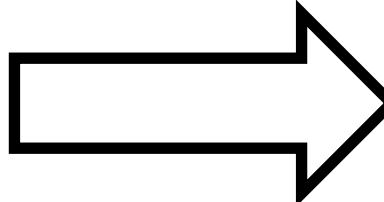
Entertainment

DirectX 11	✓	✓	✓	✓
Gadgets	✓	✓	✓	✓
Games Explorer	✓	✓	✓	✓
Play To	✓	✓	✓	✓
Windows Media Player 12	✓	✓	✓	✓
Create and play DVDs		✓	✓	✓
Internet TV		✓	✓	✓





Software-intensive systems

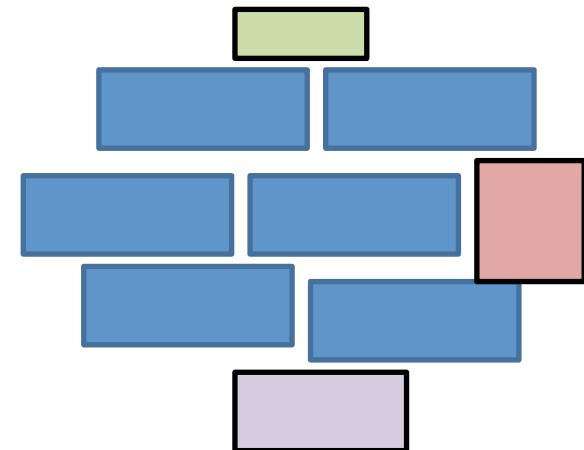


come in many variants

Software Product Line Engineering

Factoring out **commonalities**

for **Reuse** [Krueger et al., 1992] [Jacobson et al., 1997]



Managing **variabilities**

for Software **Mass Customization** [Bass et al., 1998] [Krueger et al., 2001], [Pohl et al., 2005]



Variability

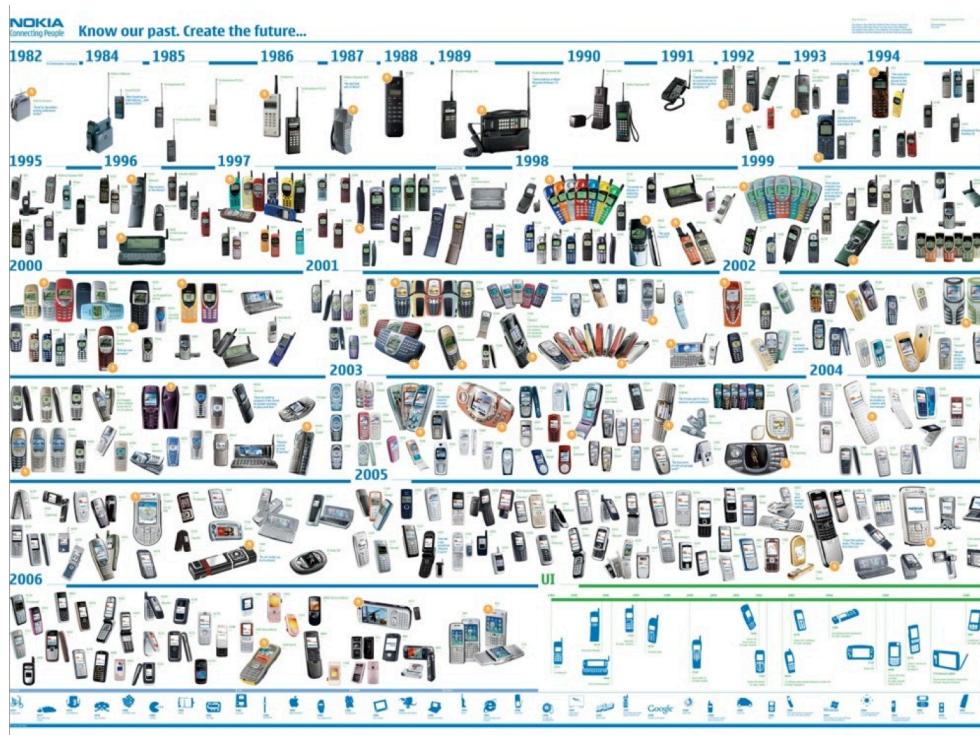
“the ability of a system to be efficiently extended, changed, customized or configured for use in a particular context”

Mikael Svahnberg, Jilles van Gurp, and Jan Bosch (2005)



Variability in time vs in space

- **Variability in Time (releases)**
 - the existence of different **versions** of an artifact that are valid at different times
- **Variability in Space (variants)**
 - the existence of an artifact in different **shapes** at the same time



Benefits

Improve product reliability

Improve usability

Improve consistency across products...



Benefits

Reduce production costs



Reduce certification costs



Shorten time-to-market



Hall of Fame

splc.net/fame.html



BOSCH

Invented for life



PHILIPS



NOKIA
Connecting People

CelsiusTech

ERICSSON



Lucent Technologies
Bell Labs Innovations





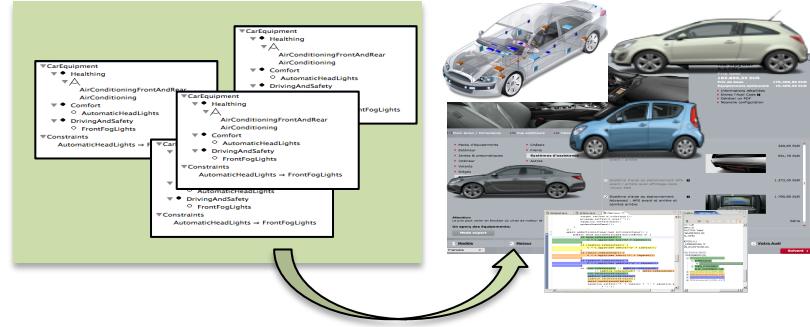
Printer Firmware

- Production cost reduced by 75%
- Development time reduced by 33%
- Reported defects reduced by 96%

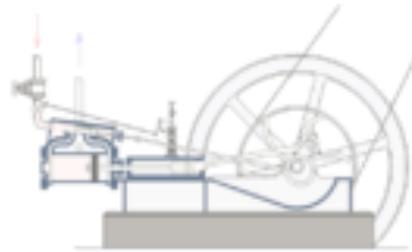


Objectives

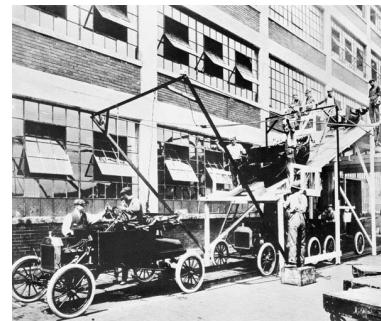
- Variability and Software Product Line Engineering
 - Overview
 - Challenges
- The (possible) role of Model-Driven Engineering
- Overview of techniques



A Bit of History: Industrial Revolution



1698
Thomas Savery



1901
Henry Ford



1980s

Nowaday: Product Lines Everywhere



Product Lines of Cars



This image may contain optional equipment. [360°](#)

Agila, Club
1.2i 16v, 5 Speed
Blaze Red, Melt / Elba Charcoal
Total € 15,684.00

[Exterior](#) | [Interior](#) [Side](#) | [Front](#) | [Rear](#) [360°](#)

[1. Trims/Series](#) | [2. Engine/Transmission](#) | [3. Colour & Style](#) | [4. Options](#) | [5. Summary](#) [Next Step](#)

Choose Your Options

<input type="checkbox"/> CD 30	Standard	- MP3 CD player with MP3 format, stereo radio, steering wheel mounted audio controls
<input checked="" type="checkbox"/> Air conditioning	€ 923.00	
<input checked="" type="checkbox"/> Electronic Stability Programme (ESP)	€ 411.00	
<input checked="" type="checkbox"/> Emergency tyre inflation kit in lieu of space-saver spare wheel and tyre	Standard	

[Audio/Comms/Nav](#) | [Heating/Ventilation](#) | [Mechanical](#) | [Safety/Security](#) | [A-Z](#)

[Next Step: Summary](#)

Pricing Details

Club	€ 14,350.00
1.2i 16v, 5 Speed	
Blaze Red	€ 0.00
Melt / Elba Charcoal	€ 0.00
15-inch steel wheels with 185/60 R 15 tyres and flush wheel covers	€ 0.00
Options (2)	
You selected:	
<input checked="" type="checkbox"/> Air conditioning	€ 923.00
<input checked="" type="checkbox"/> Electronic Stability Programme (ESP)	€ 411.00
Total	€ 15,684.00

Legend

- Selected Option
- Selectable Option
- Option contained in an option pack
- Option contained in an option pack or standard equipment which has been replaced by another option
- Option that is only selectable together with another option. Please click for details

Willkommen bei selve - the shoe individualizer

http://www.selve.net/index_js.html

KOLLEKTION FUSSTYP MYSELVE INFO HOME selve

MODELLE
LOOKBOOK

SELVE-ID
PASSWORD
>>ANMELDEN

selve Kollektion -> Style: casuals -> Modell: Opal

modell-details >> hier clicken

>>SELVE SCHUHREGAL Inhalt:0

>>SHOPPING BAG Inhalt:0

A. Erstes Oberleder
 Veloursleder Sand
 Veloursleder Bordeaux
 Veloursleder Cognac
 Veloursleder Sand
 Putzenleder
 Beige

B. Absatz
 Hufeisen Braun

C. Sohle
 Gummisoche

>>ÄNDERN
 >>ZURÜCKLEGEN

Müsli individuell online mixen! Bio-Müsli. - Mozilla Firefox

File Edit View History Bookmarks Tools Help

m http://www.mymuesli.com/muesli/index.php?vw=mixer&ec=step1&mnid=1&mnpt=1&type=t0 softwareproduktlinien ABP S

Müsli individuell online mixen! Bio-M... +

mymuesli custom-mixed cereals

muesli mixer blog fragen about us

Müslibasis Basis verfeinern Früchte Nüsse & Kerne Extras

Früchte

Köstliche Bio-Trockenfrüchte, müsligerecht aufbereitet. Du kannst eine Frucht auch mehrmals auswählen, um deren Anteil zu steigern.

Ananas
lecker, exotisch und wunderbar | 0.65€ (30g)
[mehr Infos](#)

Apfelstücke
Ohne Worte weil Klassiker | 0.45€ (25g)
[mehr Infos](#)

Aprikosen

hoch ▲ ▼ runter

Apfelstücke
Buchweizenflocken
C'Mohn, baby!

Nährwerte pro 100g ▲
575g nur 4,70€
entspricht 8,17€/kg
inkl. MWSt., zzgl. Versandkosten

fertig gemixt?
weiter

Done en-US



Der Dell Online-Shop: Stellen Sie Ihr eigenes System zusammen - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

Getting Started Latest Headlines

<http://configure2.euro.dell.com/dellstore/config.aspx?c=de&cs=dedhs1&kc=3058j=de&oc=W06390xp&s=dhs&sbc=pr>

Bestellen Sie online oder wählen Sie 0800 533 55 40 03(gebührenfrei)

DELL Produkte Service Support Einkaufsunterstützung Suche

Dell empfiehlt Windows Vista™ Home Premium.

Sie befinden sich hier: Deutschland > PRIVATANWENDER

1 Meinen Dell konfigurieren **2 Zubehör auswählen** **3 Elektronik** **4 Software & Service** **5 Bestätigen & zum Warenkorb hinzufügen**

Als Symbol anzeigen

ECC DDR2-SDRAM-Speicher mit 4,0 GB und 667 MHz (2 x 2,0 GB DIMM) [plus 0,19,99 € oder zu 0 €/Monat]

Grafikkarte

128 MB nVidia NVS285 DVI/VGA-Grafikkarte

Auswahlhilfe

- 256 MB ATI Fire GL V7200-Grafikkarte [plus 416,50 € oder 13 €/Monat¹]
- 128 MB nVidia Quadro FX550-Grafikkarte [plus 69,02 € oder 2 €/Monat¹]
- 256 MB nVidia Quadro FX3450-Grafikkarte [plus 547,40 € oder 17 €/Monat¹]
- 128 MB nVidia NVS285 DVI/VGA-Grafikkarte [Im Preis enthalten]
- Grafikkarte PCIe x16 (DVI/VGA) Matrox QID LP PCIe, 128 MB, DVI- oder VGA-Grafikkarte für 4 Monitore [plus 630,70 € oder 20 €/Monat¹]
- 128 MB ATI Fire GL V3400-Grafikkarte [plus 44,03 € oder 1 €/Monat¹]

Festplatte

80 GB Serial ATA-II-Festplatte (7.200 U/min) mit NCQ

Auswahlhilfe

- 160 GB Serial ATA-II-Festplatte (7.200 U/min) mit NCQ [plus 16,66 €]
- 80 GB Serial ATA-II-Festplatte (7.200 U/min) mit NCQ [Im Preis enthalten]

Sicher Einkaufen mit Trusted Shops und Geld-zurück-Garantie.

Dell Precision™ 390 Essential (W06390xp)

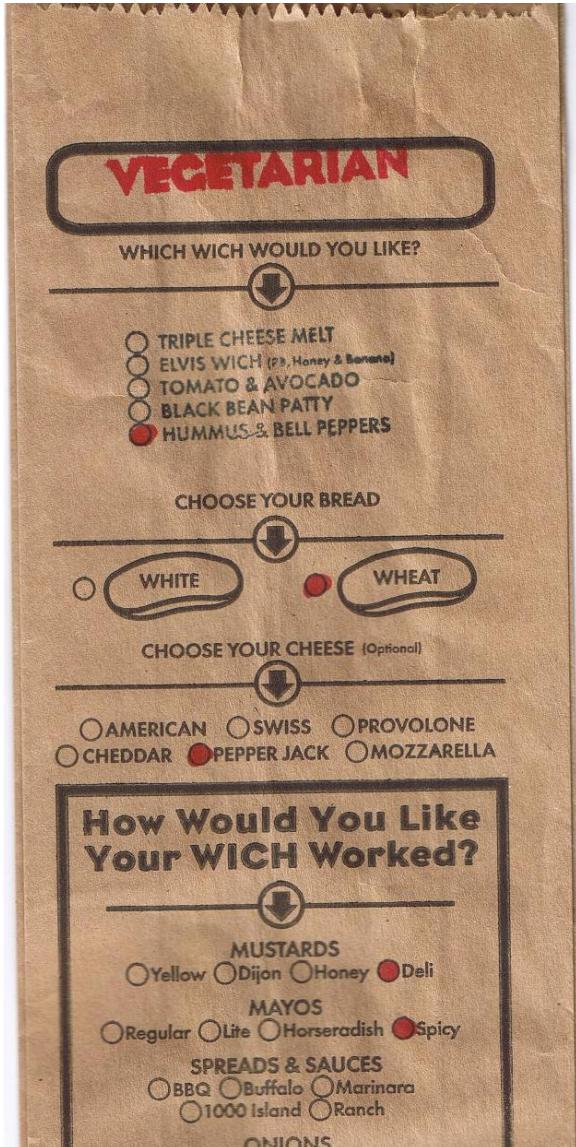
inkl. MwSt., zzgl. 19,04 € Versand
Ermäßiger Sonderpreis
913,92 € Es gelten keine zusätzlichen Preismarkierungen. Das Angebot gilt für maximal 5 Systeme

Finanzierung ab **30 €/mtl.**². Jetzt finanzieren - erst ab Januar 2008 zahlen! Weitere Informationen zur Ratenfinanzierung

Für einen noch umfassenderen Schutz Ihres Systems beinhaltet der oben erwähnte Preis ein Upgrade Service Paket. Um auf den beworbenen Preis zu kommen, entmarkieren Sie die Kategorie "Business Support".

Transferring data from i.dell.com...

Food? Product lines!







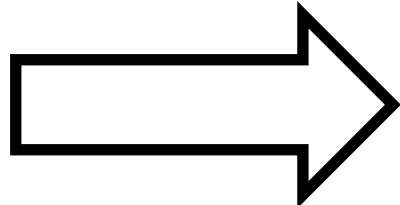
Mass production

**What about
software?**

**Product lines of
software intensive systems**

Software intensive systems

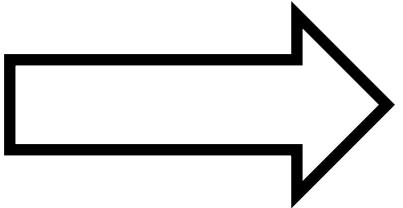
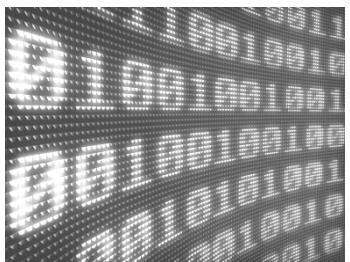
are declined in many variants



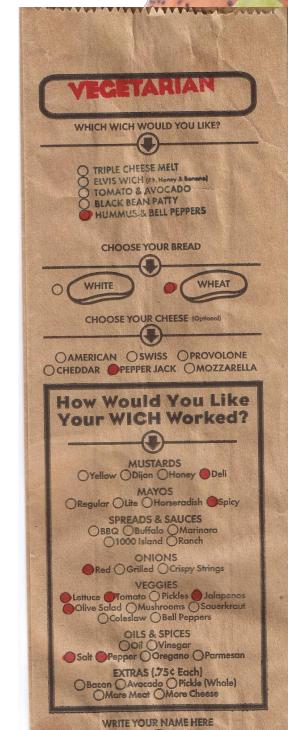


Software intensive systems

are declined in many variants



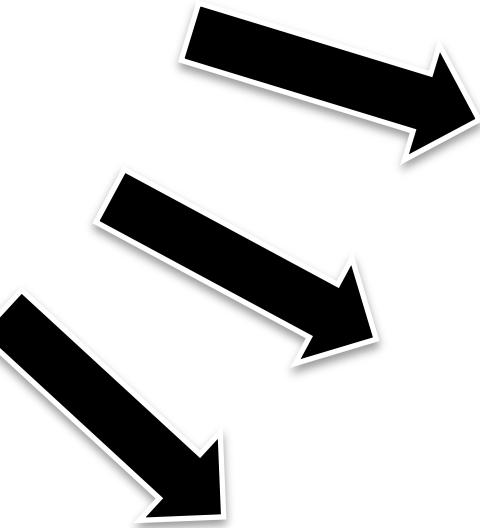
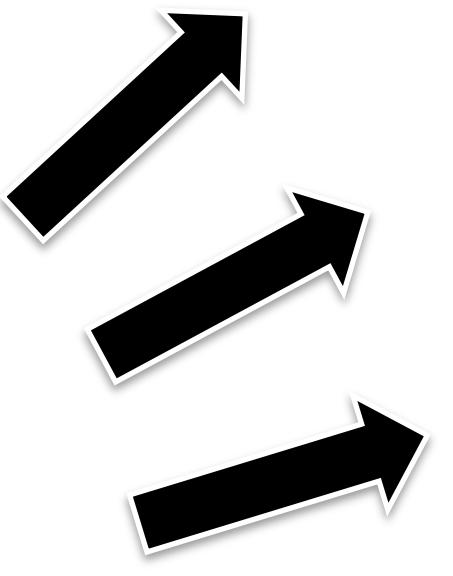
Software Product Lines



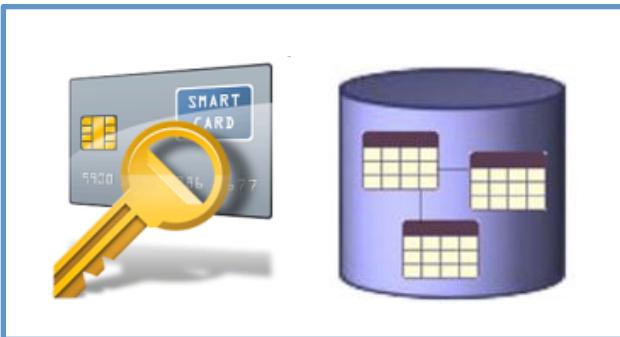
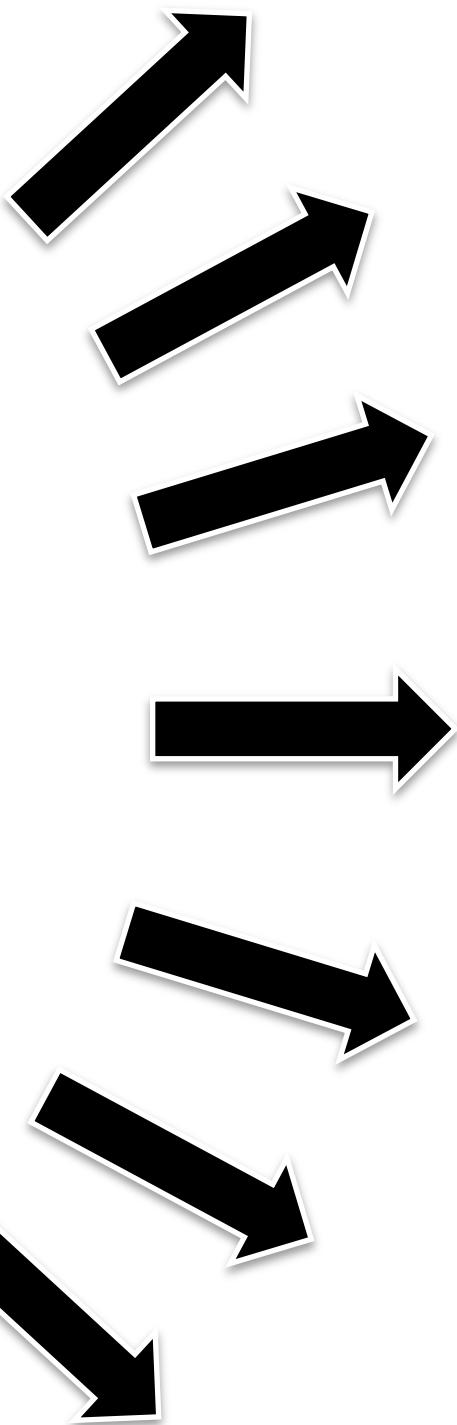
01011011
11011110
00110110
11001101
10001111
10100110
10001010
10101011
00001110
11010101
11011100
01100100
01010101
11010110
10101010



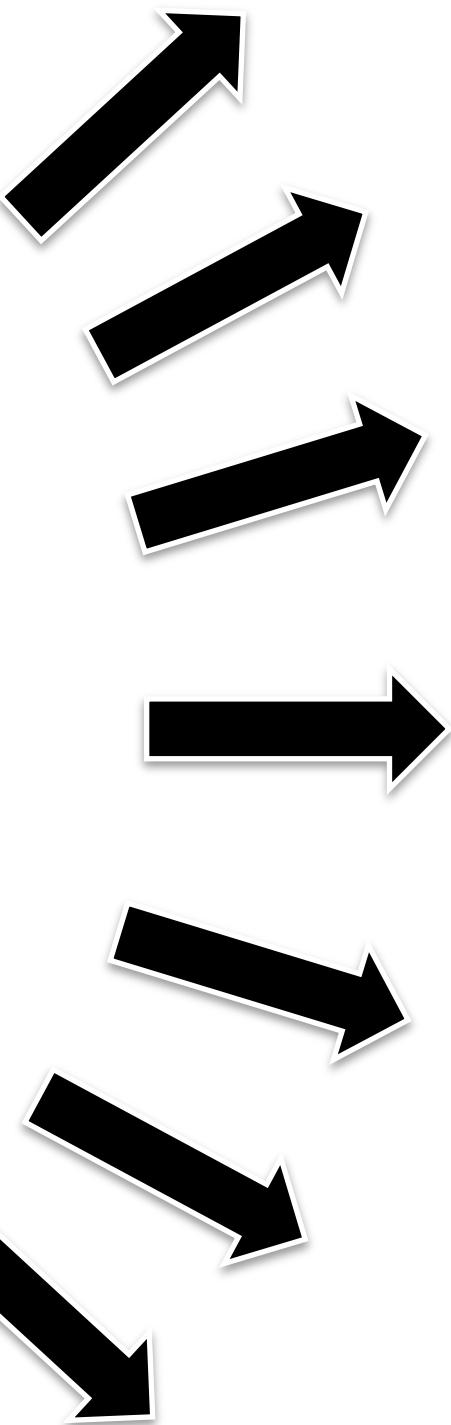
Car



Database Engine



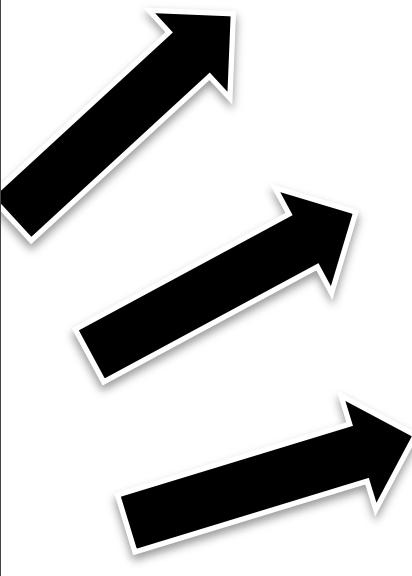
Printer Firmware



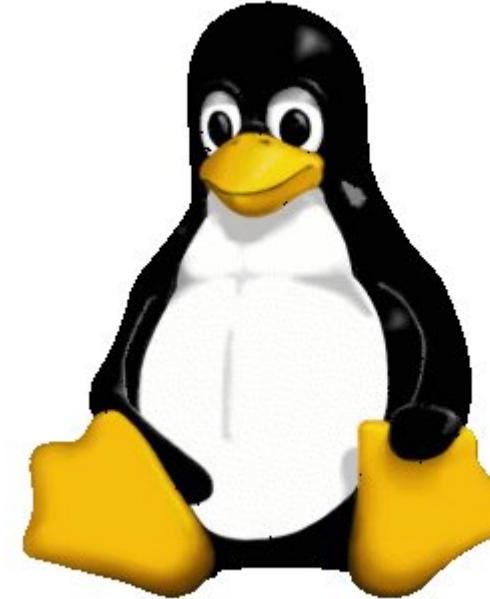
```
1 karmo 2 karmo
Line Feed CR Encoding iso-8859-1 generic .config - Linux Kernel v2.6.33.3 Configuration
Processor type and features
Arrow keys navigate the menu. <Enter> selects submenus -->. Highlighted letters
are hotkeys. Pressing <> includes, <> excludes, <M> modularizes features.
Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded [<> module] < > module capable
[*] Tickless System (Dynamic Ticks)
[ ] High Resolution Timer Support
[ ] SYSENTER/SYSEXIT handling support
[ ] Support for extended (non-PC) x86 platforms
[ ] Single-depth ICHAN output
[ ] Paravirtualized guest support ...
[ ] Memtest
[ ] Processor family (Generic-x86-64) --->
[ ] Preemption Model (No Forced Preemption (Server)) --->
[ ] Renote 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 64M 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 >
```

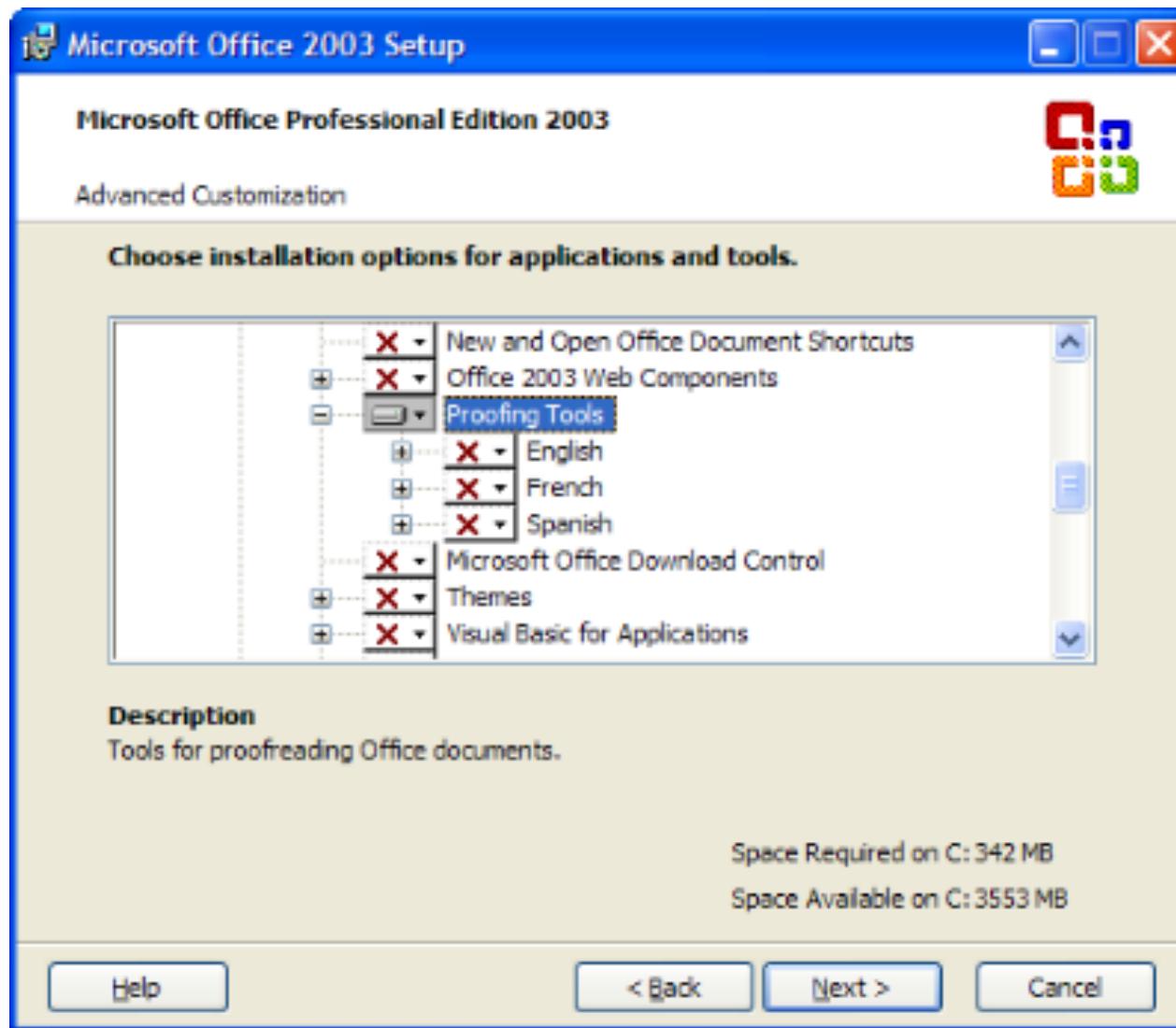
Linux Kernel



Linux-Kernel



Features in Microsoft Office



The development of a **family of software systems**

is much more challenging than the
development of
a single software system

A large, intricate 3D white maze is set against a light gray background. The maze consists of many interconnected paths and dead ends, creating a complex network of levels and corners. It occupies the entire frame, from the top left to the bottom right.

Variability = Complexity

33 features



a unique variant for every
person on this planet

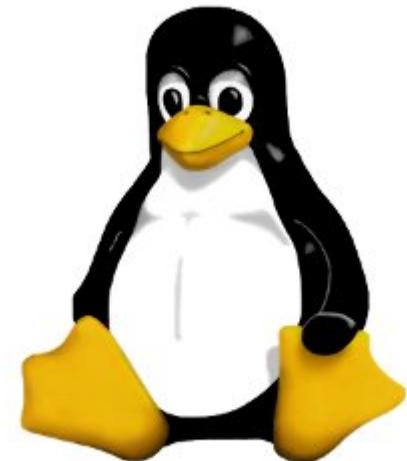
320^{optional, independent}
features

more variants than estimated
atoms in the universe



2000 features

10000
features

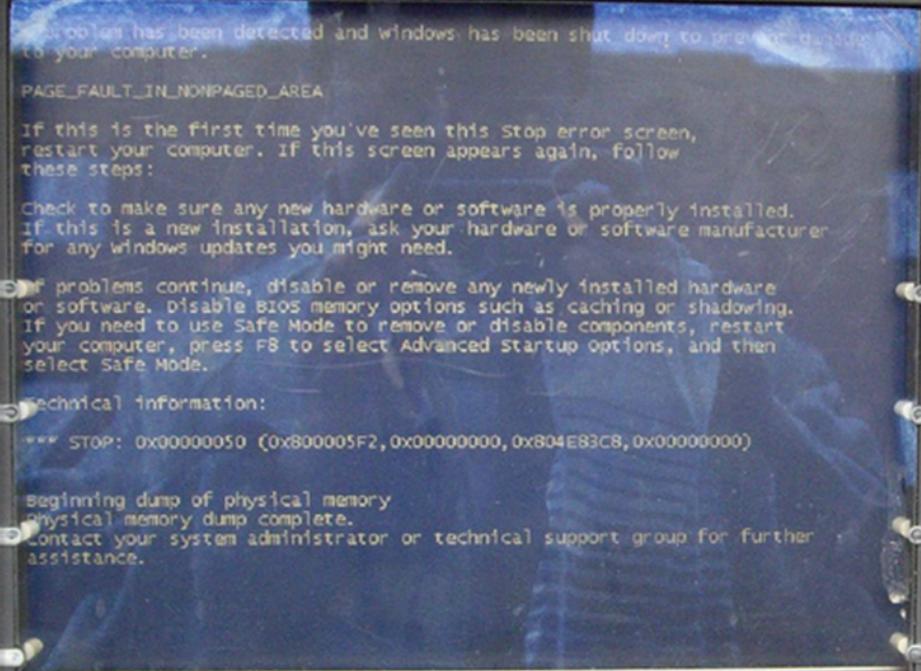


Automation?

Avoid solving the same problem!

2, 3...n times

Correctness





Maintenance?
Comprehension?

Checking Products



2000 Features
100 Printers
30 New Printers per Year

Printer
Firmware



Linux
Kernel

Checking Products



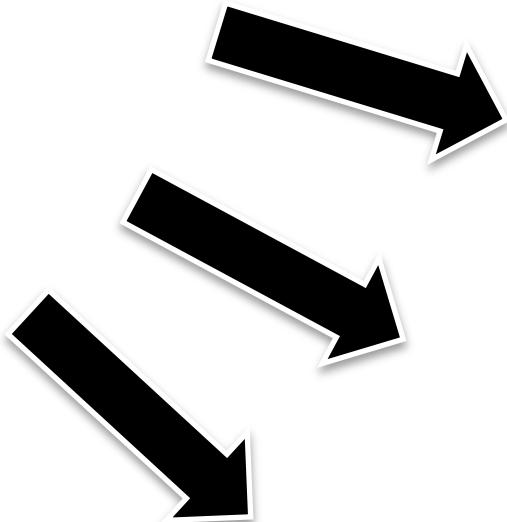
8000 Features
? Products



Checking Product Line

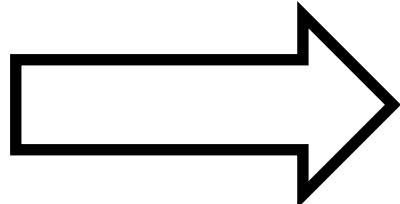
Implementation with 10000 Features
#ifdef, Frameworks, FOP, AOP, ...

Linux
Kernel



Software product line engineering

= modeling and managing variability



The development of a
family of software systems

differs from the development of
a **single** software system

**THANKS CAPTAIN
OBVIOUS**



« The development of a
family of software systems
differs from the development of
a **single** software system »

Reuse

Commonality

Customization

Variability

Automation

A photograph of a car assembly line. In the foreground, a worker wearing a white shirt and red overalls is working on the interior of a silver car. The car's front door is open. Behind the worker, several other cars are lined up on the assembly line. The background shows the industrial interior of a factory with various equipment and a digital display showing the number "042 066 002".

Assembly Line and Mass Customization



**Reuse
and
Mass Customization**



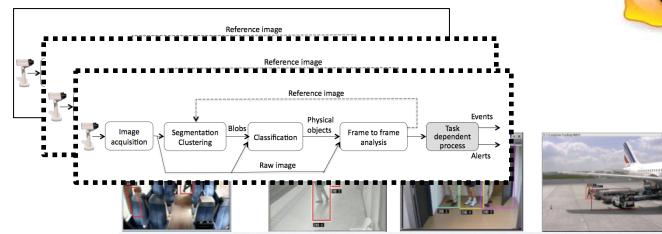
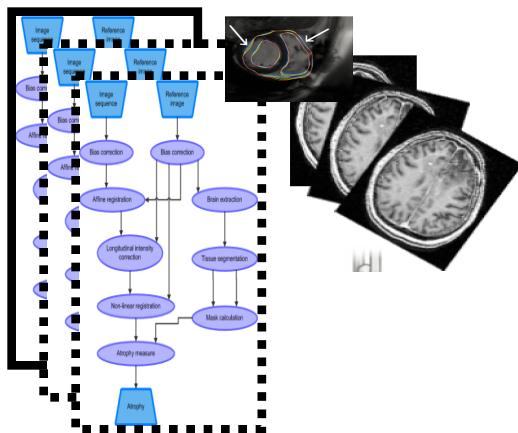
Starting from scratch?

A wide-angle photograph of a massive aircraft assembly hangar. In the center, a large aircraft is being assembled with its fuselage and wings visible. To the left, several rows of desks with computer monitors are arranged, likely for engineering or design work. The hangar is filled with industrial equipment, scaffolding, and workers moving throughout the space.

You cannot start from scratch

“a set of software- intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way” [Clements et al., 2001]

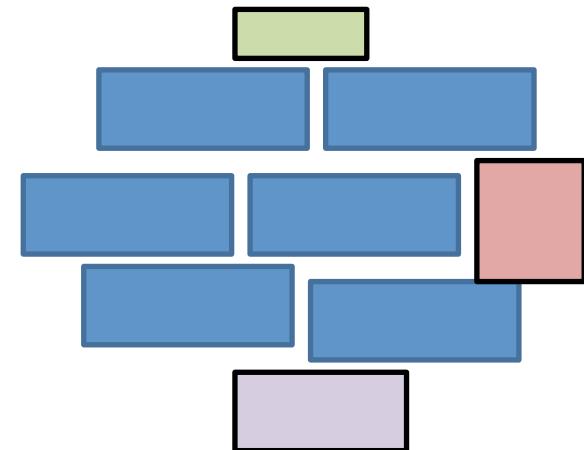
Software Product Lines



Software Product Line Engineering

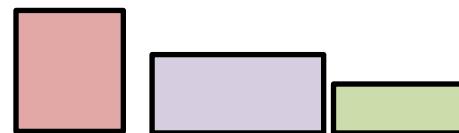
Factoring out **commonalities**

for **Reuse** [Krueger et al., 1992] [Jacobson et al., 1997]

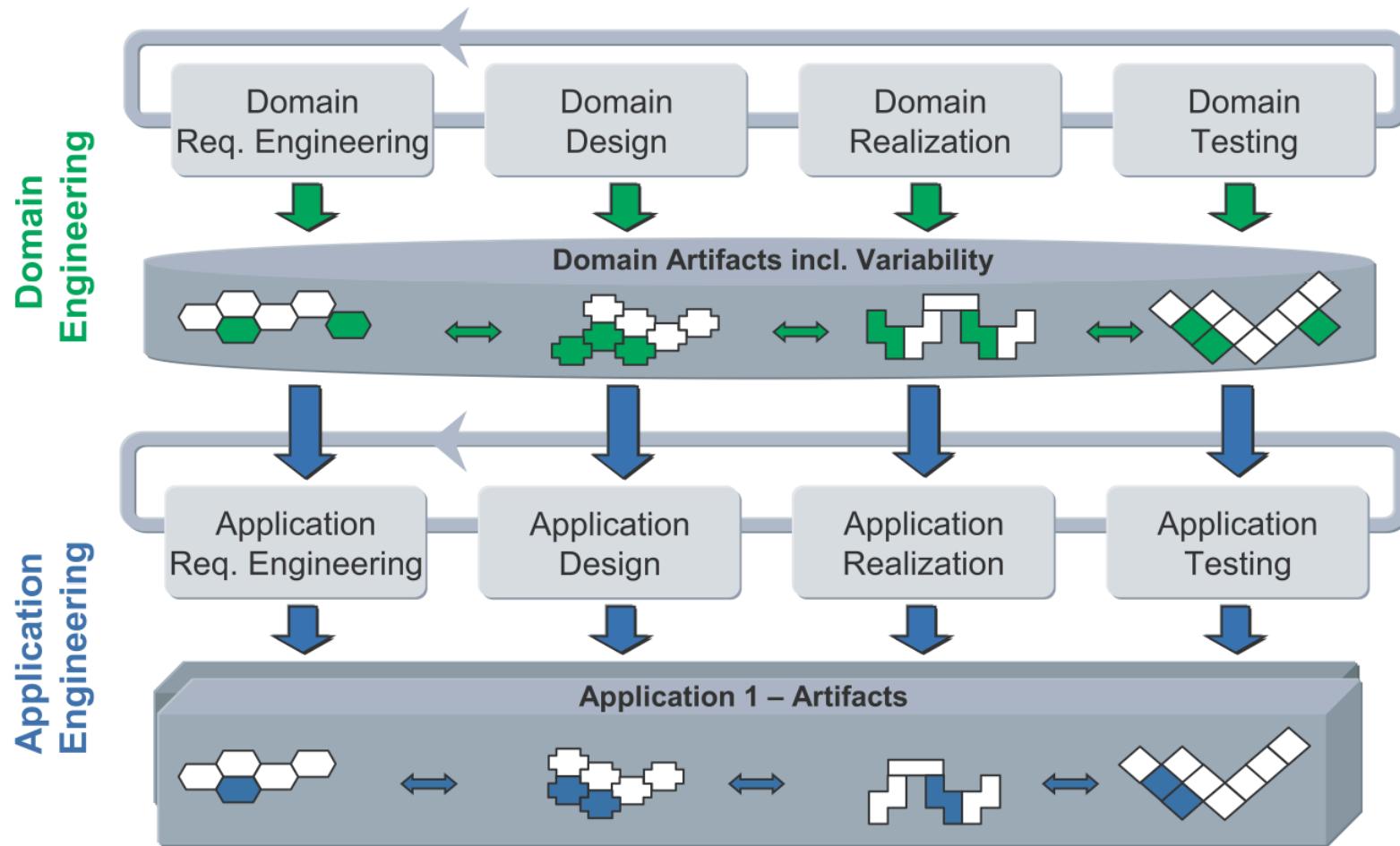


Managing **variabilities**

for Software **Mass Customization** [Bass et al., 1998] [Krueger et al., 2001], [Pohl et al., 2005]



Software Product-Line Engineering



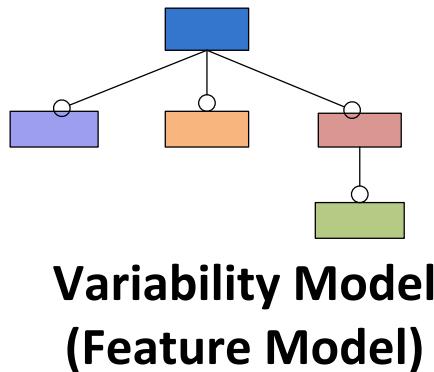


“Reuse-in-the-large works best in families of related systems, and thus is domain dependent.” [Glass, 2001]

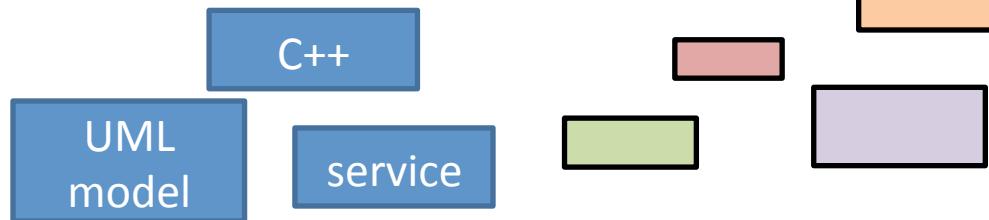
Domain engineering

Domain Analysis (problem)

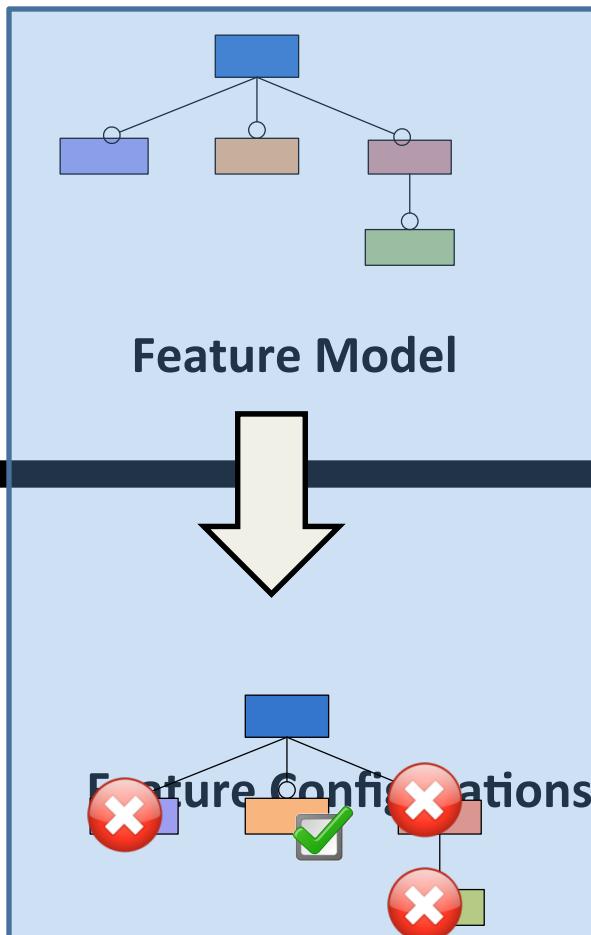
- elicitate requirements and scope the line
- variability modeling: determine commonalities and variabilities usually in terms of features



Domain Implementation (solution)

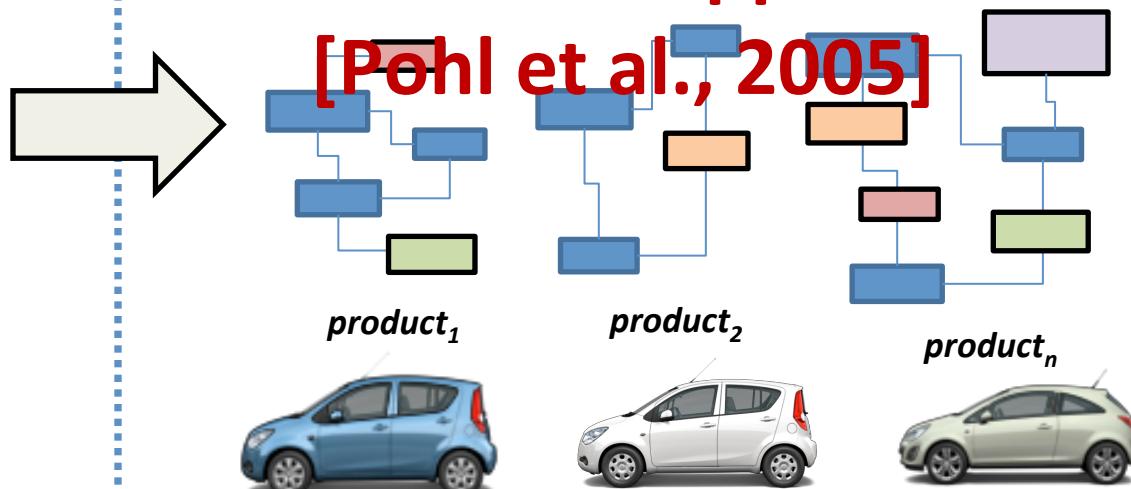


Domain engineering (development for reuse)



“central to the software product line paradigm is the modeling and management of variability, that is, the commonalities and differences in the applications”

[Pohl et al., 2005]



Application engineering (development with reuse)

What is new?

Family vs single systems

Focus on reuse

Domain engineering

Factoring out commonality

Managing variability

« variability »

Is it really new?

Parameter

```
Administrator: C:\Windows\system32\cmd.exe
C:\Users\kaestner.INFORMATIK.000>dir /?
Displays a list of files and subdirectories in a directory.

DIR [drive:][path][filename] [/A[[:l]attributes] [/B] [/C] [/D] [/L] [/N]
  [/O[[:l]sortorder]] [/P] [/Q] [/R] [/S] [/T[[:l]timefield]] [/W] [/X] [/4]

[drive:][path][filename]
      Specifies drive, directory, and/or files to list.

/A          Displays files with specified attributes.
attributes   D  Directories                  R  Read-only files
              H  Hidden files                A  Files ready for archiving
              S  System files                I  Not content indexed files
              L  Reparse Points             -  Prefix meaning not
/B          Uses bare format (no heading information or summary).
/C          Display the thousand separator in file sizes. This is the
            default. Use /-C to disable display of separator.
/D          Same as wide but files are list sorted by column.
/L          Uses lowercase.
/N          New long list format where filenames are on the far right.
/O          List by files in sorted order.
sortorder    N  By name (alphabetic)        S  By size (smallest first)
              E  By extension (alphabetic)  D  By date/time (oldest first)
              G  Group directories first   -  Prefix to reverse order
/P          Pauses after each screenful of information.
```

Parameter -i in grep

```
1 int match_icase;
2
3 int main (int argc, char **argv)
4 {
5     [...]
6     while ((opt = get_nondigit_option (argc, argv, &default_color))
7         switch (opt)
8         {
9             [...]
10            case 'i':
11                match_icase = 1;
12                break;
13            }
14        }
15
16
17 static const char *
18 print_line_middle (const char *beg, const char *lim,
19                     const char *line_color, const char *match_color)
20 {
21     [...]
22     if (match_icase)
23     {
24         ibeg = buf = (char *) xmalloc(i);
25         while (--i >= 0)
26             buf[i] = tolower(beg[i]);
27     }
}
```

Global configuration

```
class Config {  
    public static boolean isLogging = false;  
    public static boolean isWindows = false;  
    public static boolean isLinux = true;  
}  
class Main {  
    public void foo() {  
        if (isLogging)  
            log(„running foo()“);  
        if (isWindows)  
            callWindowsMethod();  
        else if (isLinux)  
            callLinuxMethod();  
        else  
            throw RuntimeException();  
    }  
}
```

Configuration

httpd.conf -- win32 Apache Building a Web Server, for Windows

```
Listen 80
ServerRoot "/www/Apache2"
DocumentRoot "/www/webroot"
```

```
ServerName localhost:80
ServerAdmin admin@localhost
```

```
ServerSignature On
ServerTokens Full
```

```
DefaultType text/plain
AddDefaultCharset ISO-8859-1
```

```
UseCanonicalName Off
```

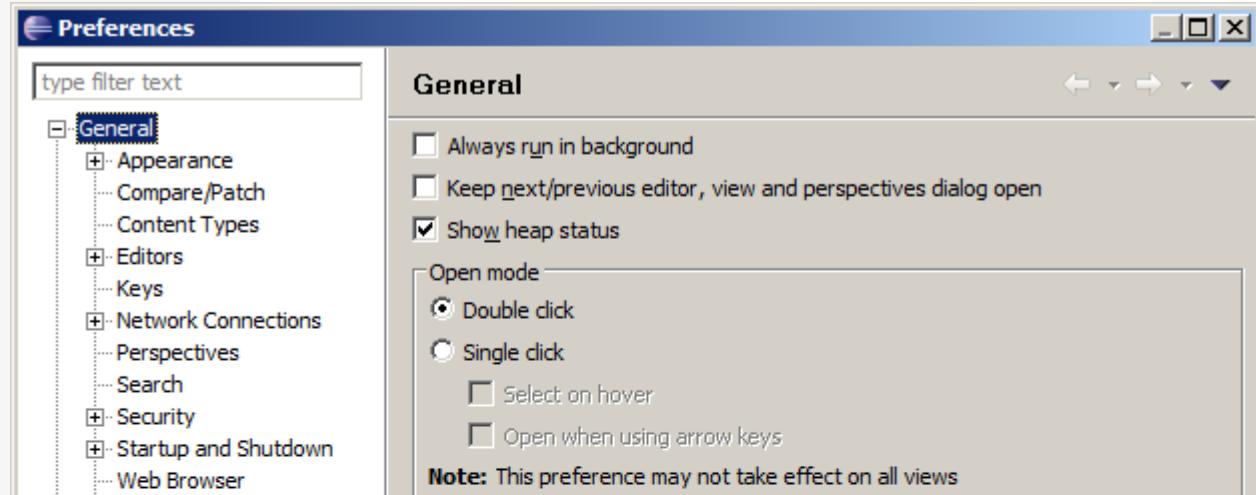
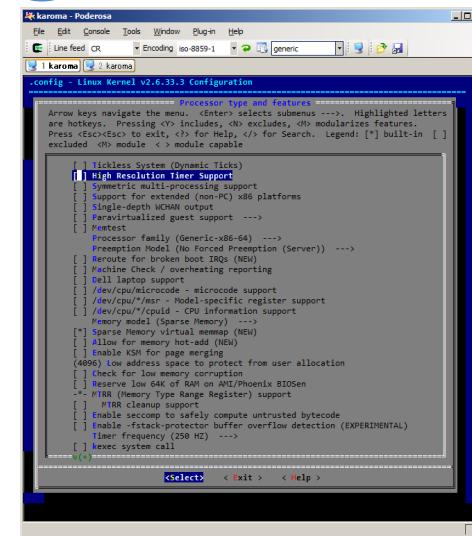
```
HostnameLookups Off
```

```
ErrorLog logs/error.log
LogLevel error
```

```
PidFile logs/httpd.pid
```

```
Timeout 300
```

```
KeepAlive On
MaxKeepAliveRequests 100
```



Conditional compilation

#ifdef (Berkeley DB)

```
static int __rep_queue_filedone(dbenv, rep, rfp)
    DB_ENV *dbenv;
    REP *rep;
    __rep_fileinfo_args *rfp; {
#ifndef HAVE_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;
#endif
#ifdef DIAGNOSTIC
    DB_MSGBUF mb;
#endif
    // over 100 lines of additional code
}
#endif
```

Intentional Code Cloning

~ Copy & Paste

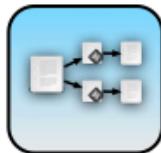
Code Cloning (example, Linux driver)

cyberstorm.c

```
....  
static void dma_dump_state(struct NCR_ESP *esp)  
{  
    ESPLOG("esp%d: dma -- cond_reg<%02x>\n",  
           esp->esp_id, ((struct cyber_dma_registers *)  
                           (esp->dregs))->cond_reg);  
    ESPLOG("intreq:<%04x>, intena:<%04x>\n",  
           custom.intreq, custom.intenar));  
}  
  
static void dma_init_read(struct NCR_ESP *esp, __u32 addr, int  
length)  
{  
    struct cyber_dma_registers *dregs =  
        (struct cyber_dma_registers *) esp->dregs;  
  
    cache_clear(addr, length);  
  
    addr &= ~(1);  
    dregs->dma_addr0 = (addr >> 24) & 0xff;  
    dregs->dma_addr1 = (addr >> 16) & 0xff;  
    dregs->dma_addr2 = (addr >> 8) & 0xff;  
    dregs->dma_addr3 = (addr      ) & 0xff;  
    ctrl_data &= ~(CYBER_DMA_WRITE);  
}.....
```

cyberstormll.c

```
....  
static void dma_dump_state(struct NCR_ESP *esp)  
{  
    ESPLOG("esp%d: dma -- cond_reg<%02x>\n",  
           esp->esp_id, ((struct cyberll_dma_registers *)  
                           (esp->dregs))->cond_reg));  
    ESPLOG("intreq:<%04x>, intena:<%04x>\n",  
           custom.intreq, custom.intenar));  
}  
  
static void dma_init_read(struct NCR_ESP *esp, __u32 addr, int  
length)  
{  
    struct cyberll_dma_registers *dregs =  
        (struct cyberll_dma_registers *) esp->dregs;  
  
    cache_clear(addr, length);  
  
    addr &= ~(1);  
    dregs->dma_addr0 = (addr >> 24) & 0xff;  
    dregs->dma_addr1 = (addr >> 16) & 0xff;  
    dregs->dma_addr2 = (addr >> 8) & 0xff;  
    dregs->dma_addr3 = (addr      ) & 0xff;  
}  
.....
```

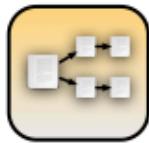


Replicate & Specialize

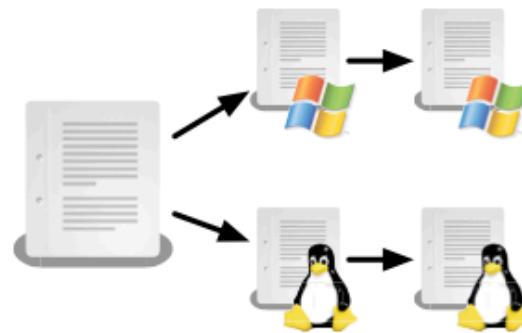


Clone to reuse and adapt existing solutions

- + Less effort needed
- Long-term cost outweighs short-term benefit
- ~ Cost of refactoring rises over time

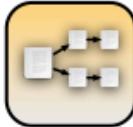


Platform Variations

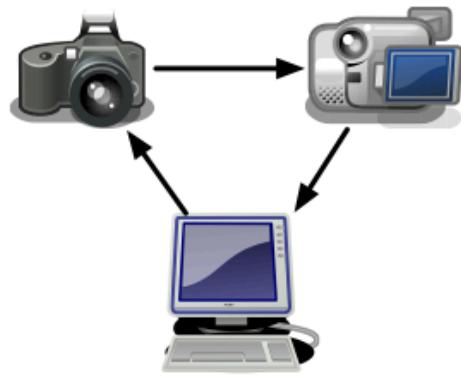


**Clone existing code and fix
low level platform interaction**

- + Avoid complexity of virtualization layer
- Hard to propagate bug fixes
- ~ Ensure consistent behavior of all clones



Hardware Variations



Clone existing driver

- + No risk of changing existing driver
- Code growth
- ~ Dead code can creep into system

Inheritance (OOP)

Base Class encapsulate commonalities

Derive classes specialize peculiarities

Generic Programming

C++ template

```
template <typename T>
T max(T x, T y)
{
    return x < y ? y : x;
}
```

Generics in Java

```
public interface List<E> {
    void add(E x);
    Iterator<E> iterator();
}
public interface Iterator<E> {
    E next();
    boolean hasNext();
}
```

Design Patterns

Template Method

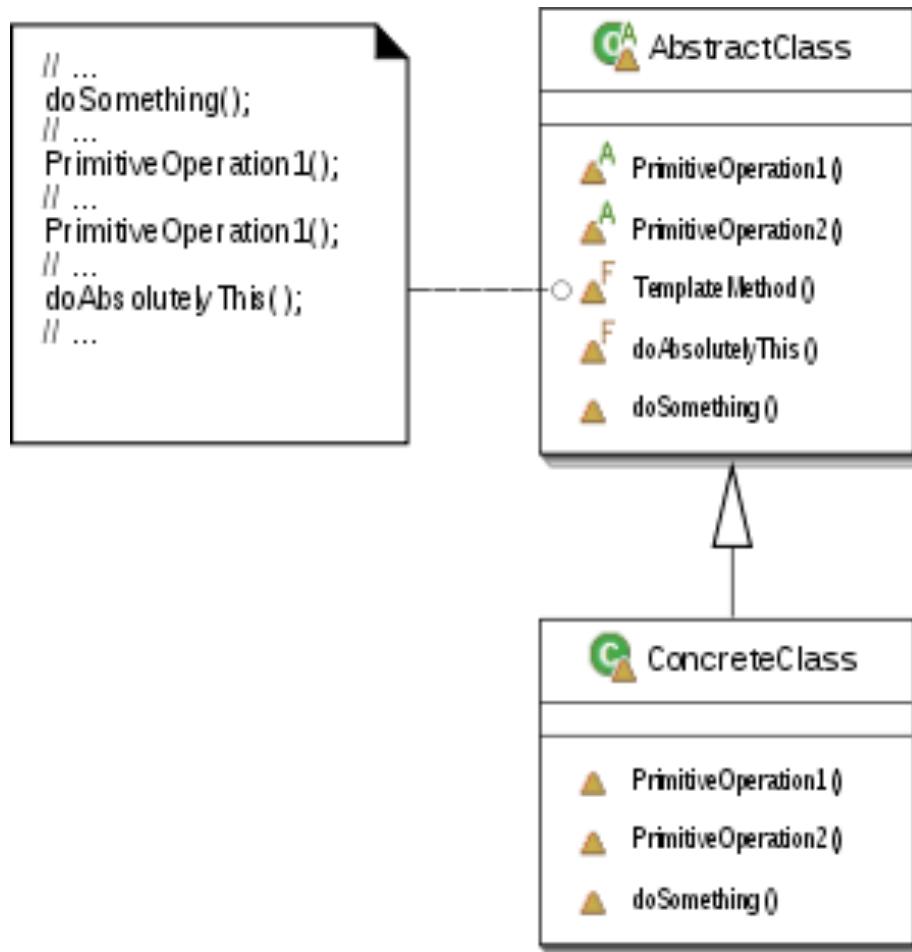
Factory

Strategy

Decorator

....

Template Method



API Framework

Plugin-based systems

httpd.conf -- win32 Apache

Building a Web Server, for Windows

```
Listen 80
ServerRoot "/www/Apache2"
DocumentRoot "/www/webroot"

ServerName localhost:80
ServerAdmin admin@localhost

ServerSignature On
ServerTokens Full

DefaultType text/plain
AddDefaultCharset ISO-8859-1

UseCanonicalName Off

HostnameLookups Off

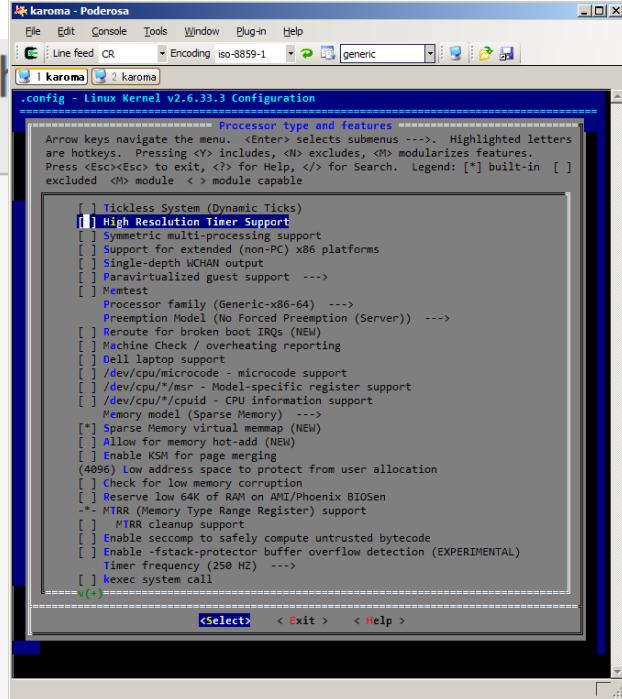
ErrorLog logs/error.log
LogLevel error

PidFile logs/httpd.pid

Timeout 300

KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 15

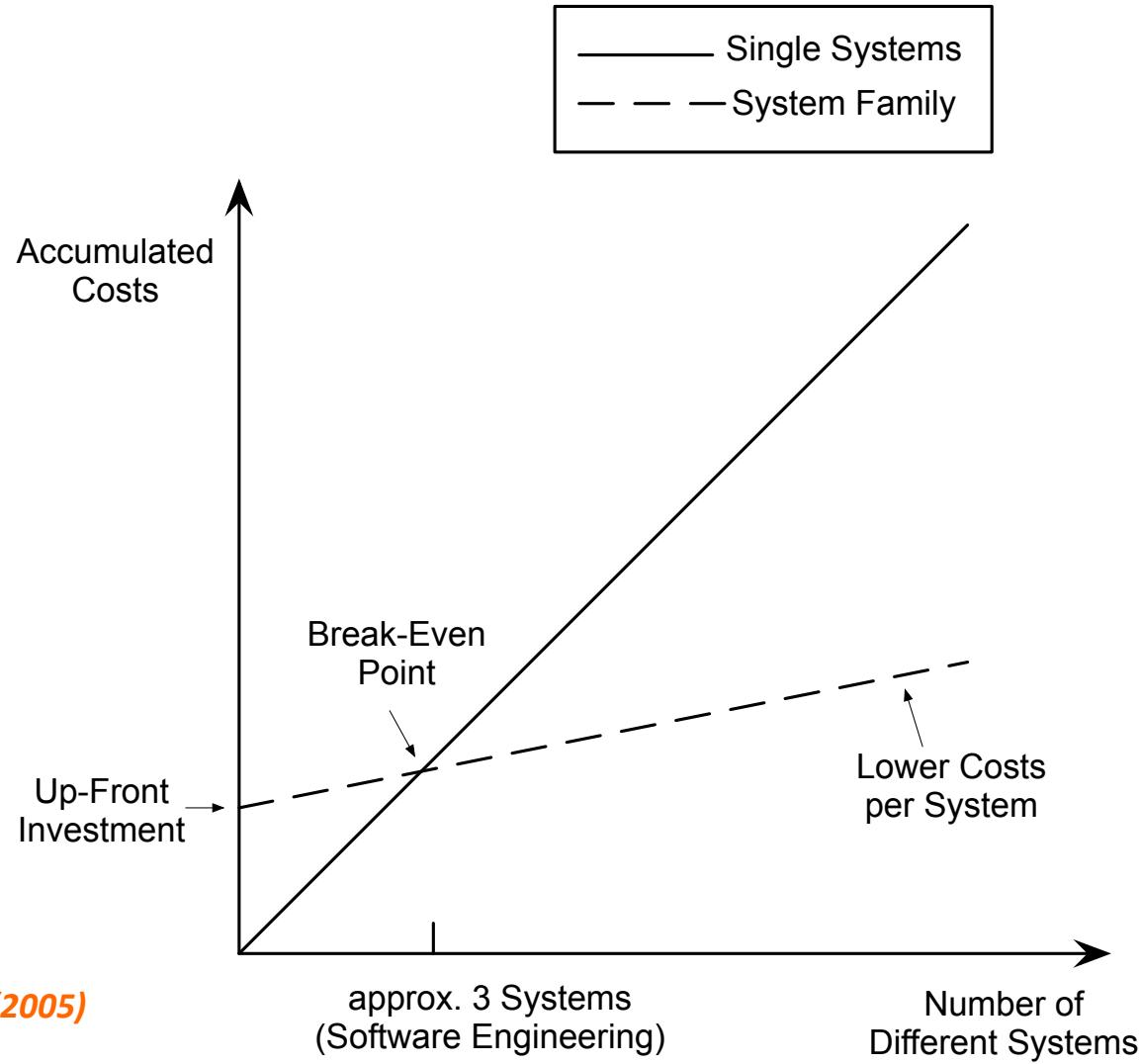
<IfModule mpm_winnt.c>
    ThreadsPerChild 250
    MaxRequestsPerChild 0
</IfModule>
```



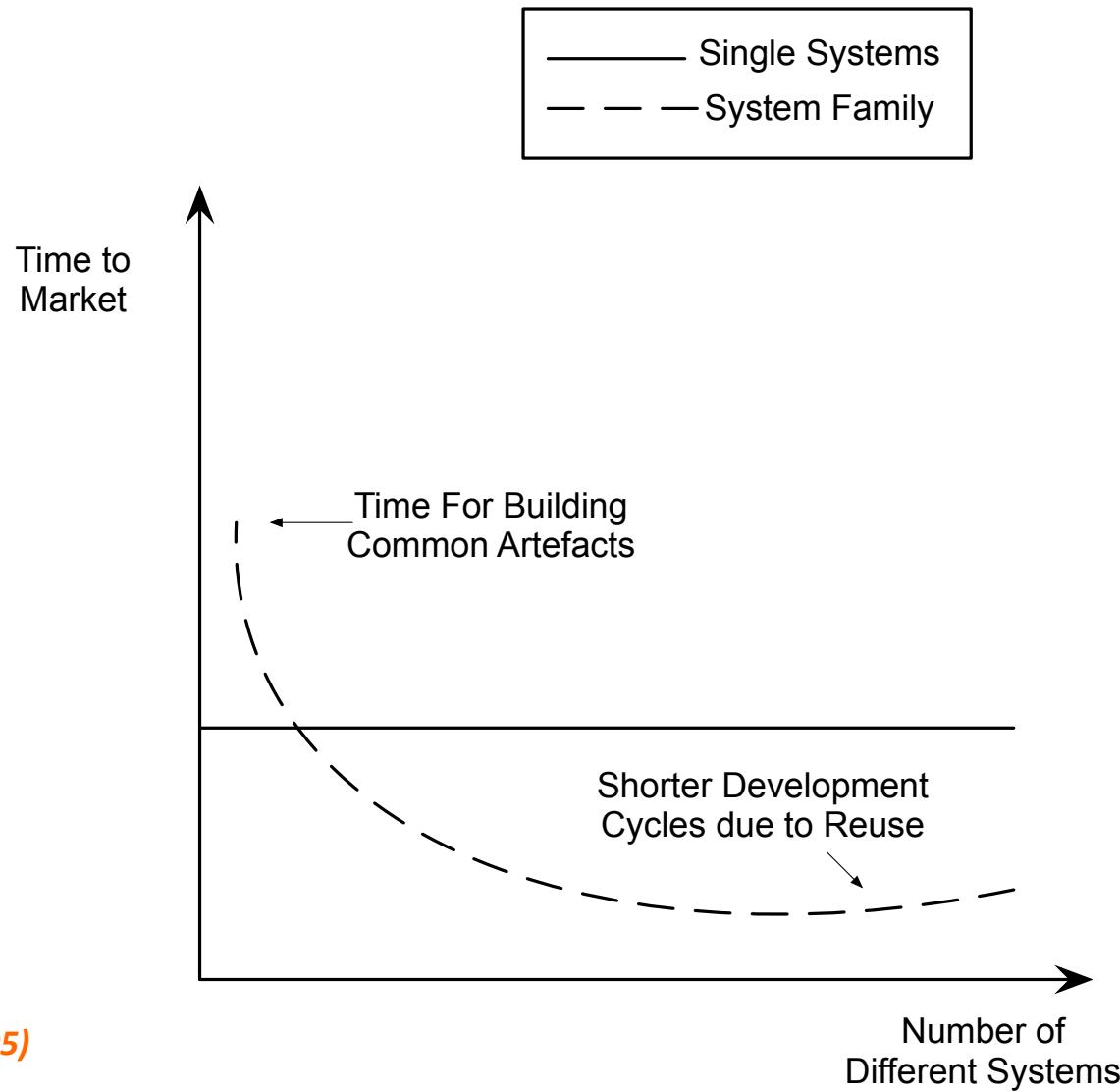
A screenshot of the Renault Vans website. The header includes the Renault logo and 'RENAULT VANS'. Below it is a navigation menu with links like CARS, VANS, ELECTRIC VEHICLES, RENAULT BUSINESS, USED CARS, OWNER SERVICES, ABOUT RENAULT, and RENAULT SHOP. The main content area shows a 'NEW KANGOO VAN RANGE' section. It features three tabs: 01 Preferences, 02 Version, and 03 Equipment & options. The 03 tab is active, showing options like Central storage console & armrest between seats (£50.00), Electric door mirrors (£0.00), and ESC (Electronic Stability Control) with traction and understeer control (£200.00). To the right is an image of a white Renault Kangoo van.

A screenshot of the Eclipse IDE. The title bar says 'Preferences'. A search bar at the top left says 'type filter text'. The left sidebar shows a tree view of preferences categories: General, Appearance, Compare/Patch, Content Types, Editors, Keys, Network Connections, Perspectives, Search, Security, Startup and Shutdown, Web Browser, Workspace, Ant, Help, IMP, Install/Update, Java, and LPG. The 'General' category is expanded, showing options like Always run in background, Keep next/previous editor, view and perspectives dialog open, and Show heap status (which is checked). Below these are Open mode options (Double click is selected) and a note: 'Note: This preference is shared with the Java editor'. The central workspace shows code snippets for Notepad.java, Actions.java, and Main.java. The right side features the 'ASTView' panel, which displays the abstract syntax tree (AST) for the selected code, showing nodes for statements, expressions, and other programming constructs.

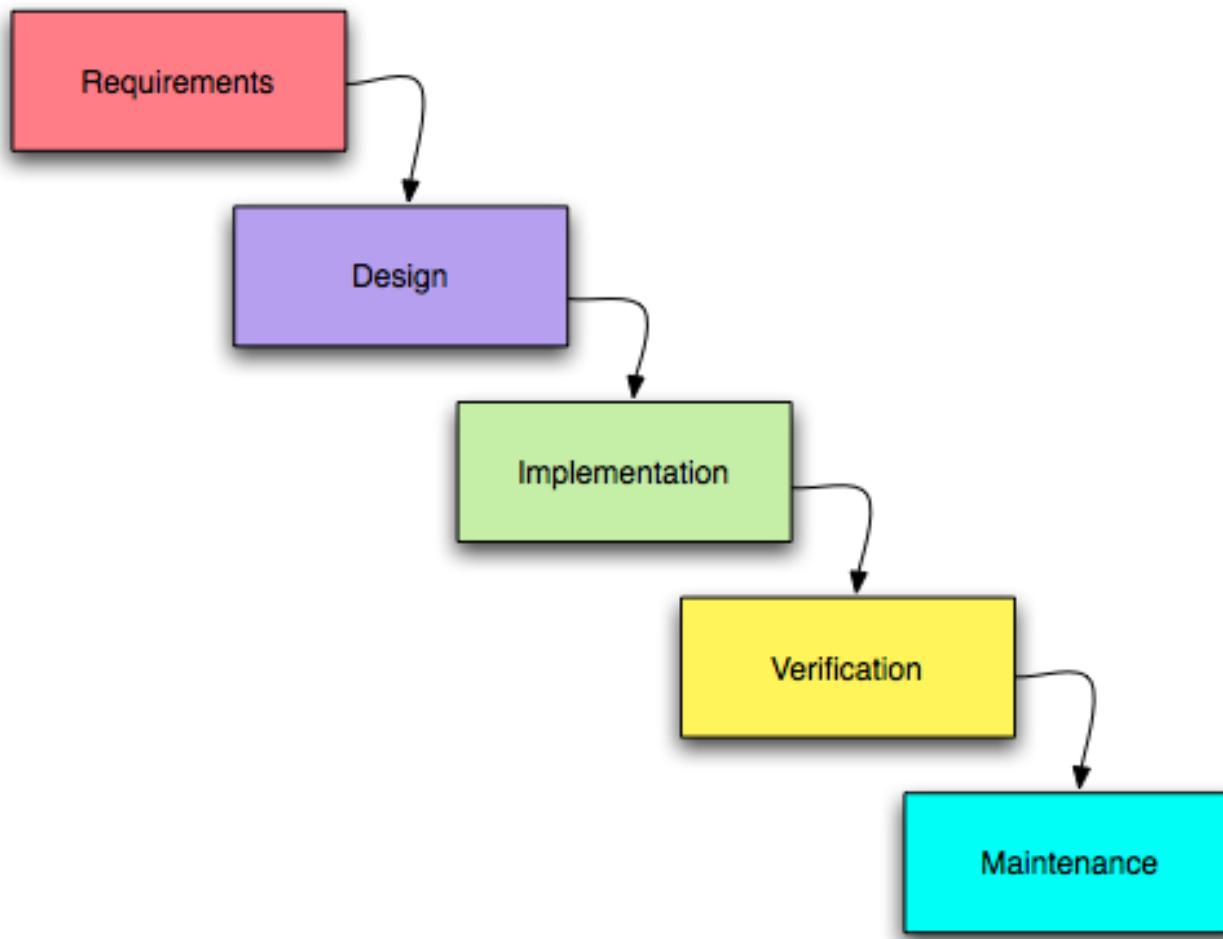
Promises of Software Product Line Engineering



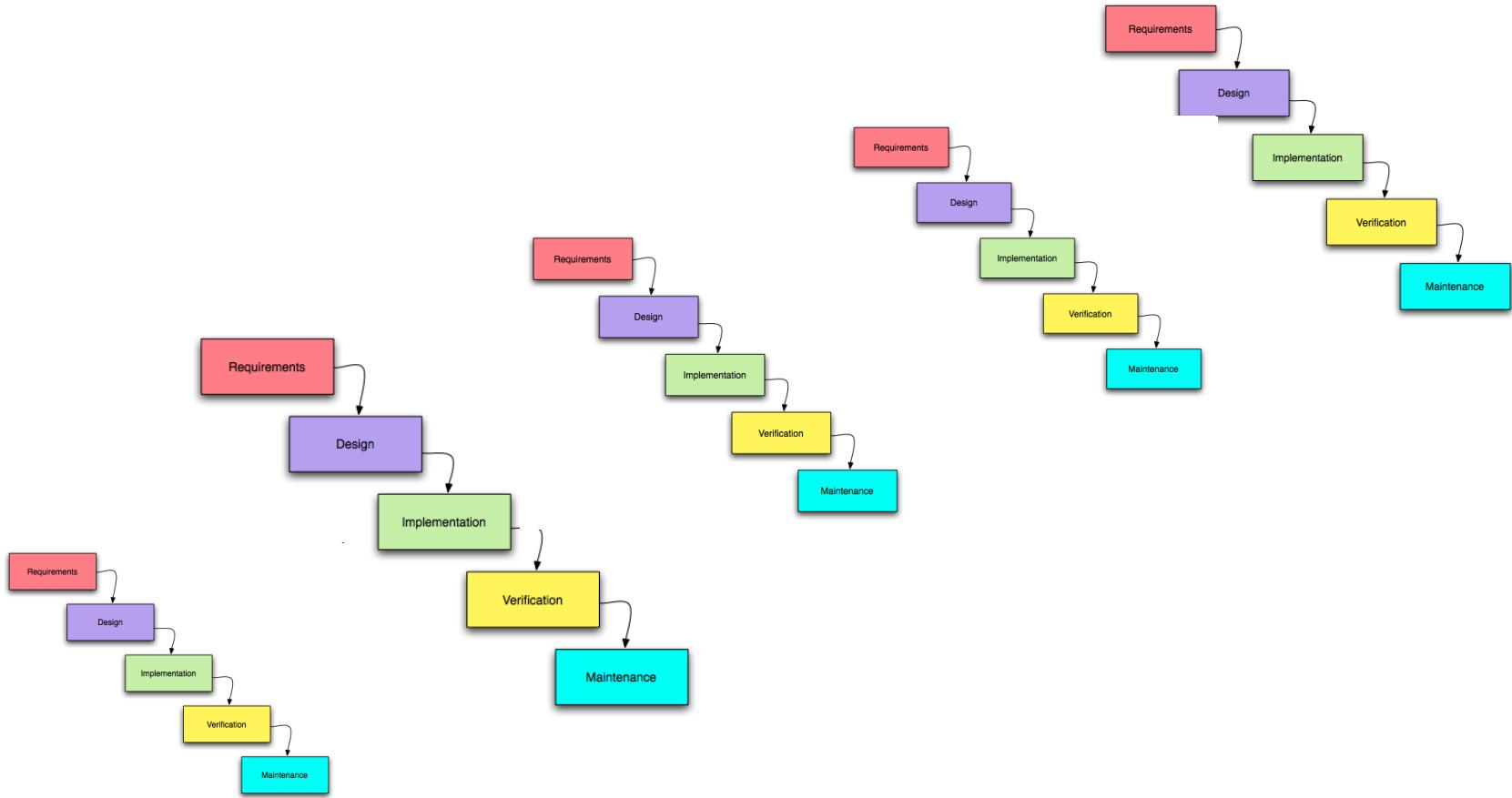
Promises of Software Product Line Engineering



Single Software Development



Software Product Line Development?



Time and Effort: not scalable!

We need an engineering
process specific to
software product lines

Observation: “Reuse-in-the-large works best in families of related systems, and thus is domain dependent.” [Glass, 2001]

Domain Engineering

[...] is the activity of collecting, organizing, and storing past experience in building systems [...] in a particular domain in the form of reusable assets [...], as well as providing an adequate means for reusing these assets (i.e., retrieval, qualification, dissemination, adaptation, assembly, and so on) when building new systems.

K. Czarnecki and U. Eisenecker

Domain Engineering



Product Line Engineering

The conventional software engineering
concentrates on satisfying the
requirements for a **single** system

Domain Engineering concentrates on
providing **reusable** solutions for
families of systems.

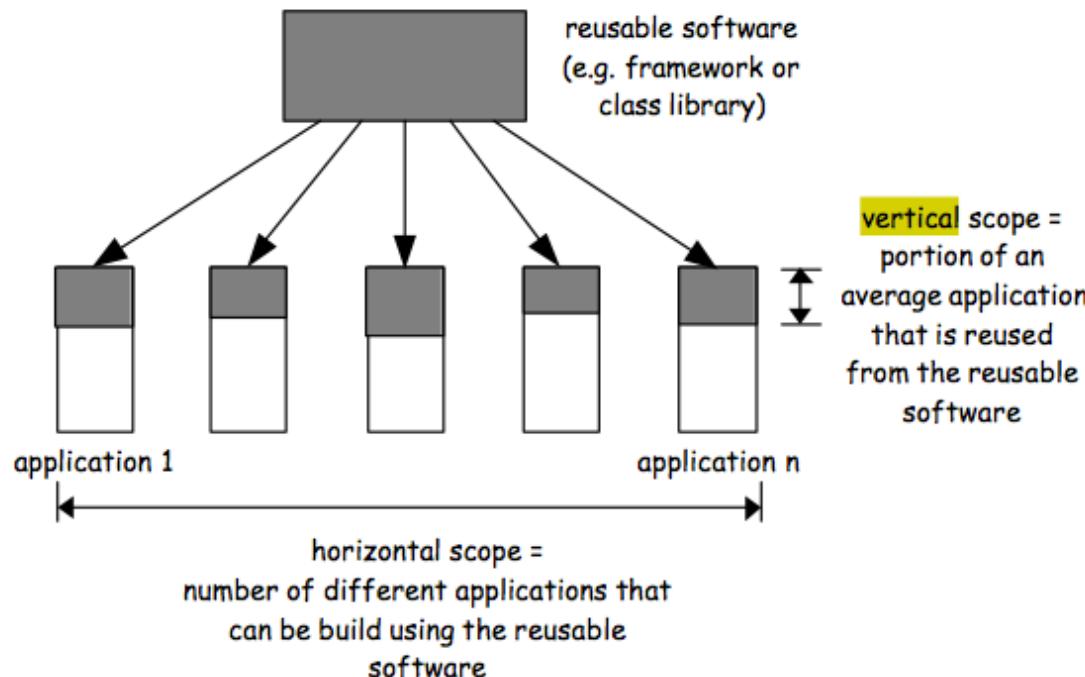
Domains of systems vs subsystems

– vertical domains

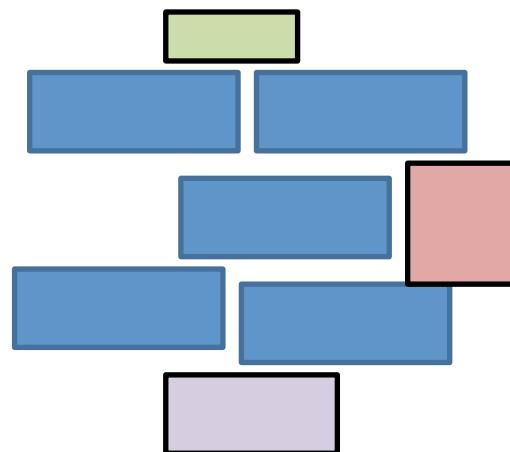
- e.g. domain of medical record systems, domain of portfolio management systems, etc.

– horizontal domains

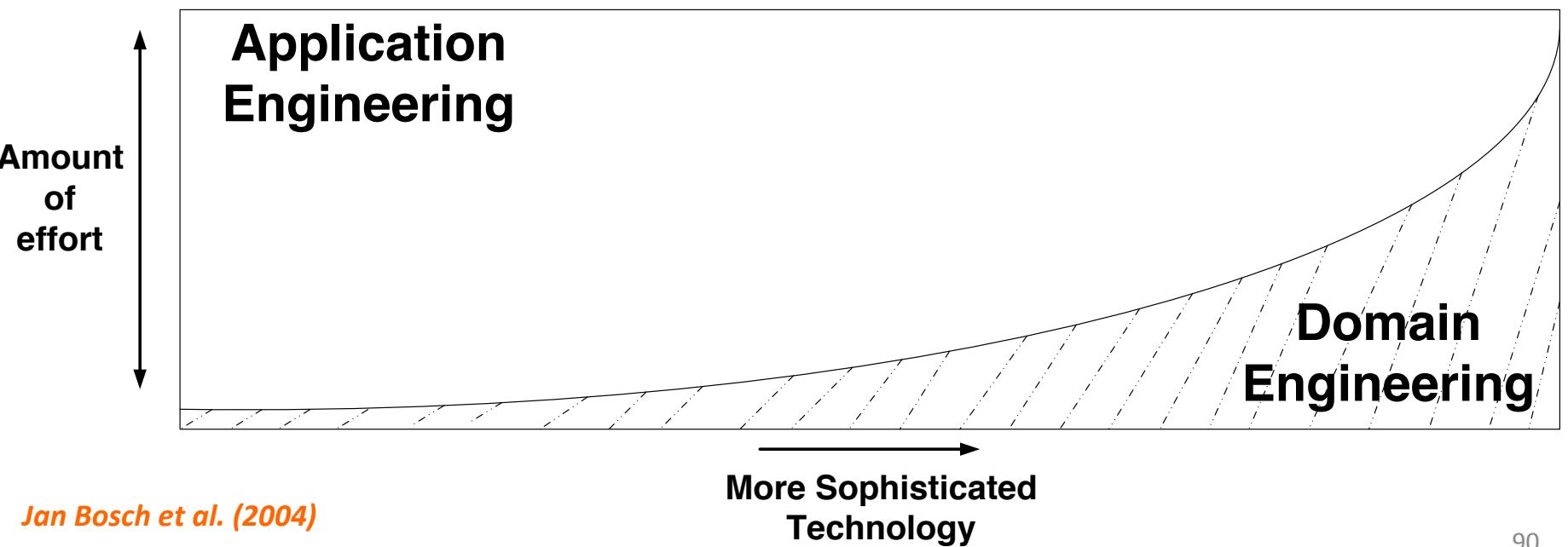
- e.g. database systems, numerical code libraries, financial components library, etc.



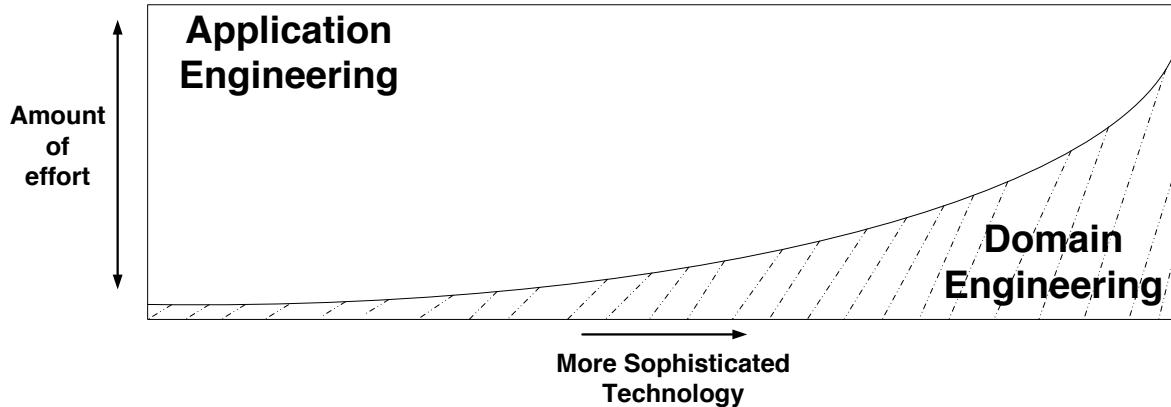
Key idea: building a reusable platform during domain engineering



Key idea: « the investments required to develop the reusable artifacts during **domain engineering**, are outweighed by the benefits of deriving the individual products during **application engineering** »



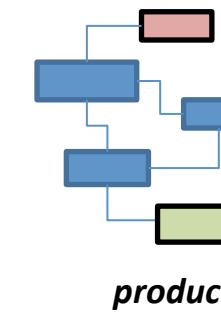
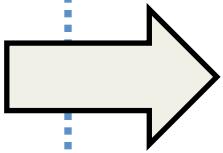
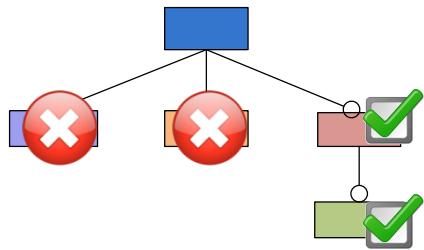
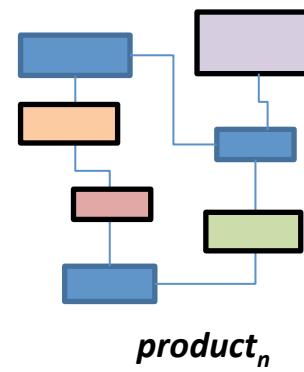
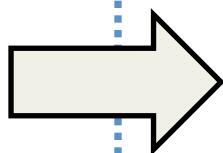
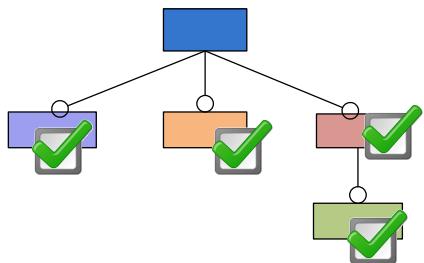
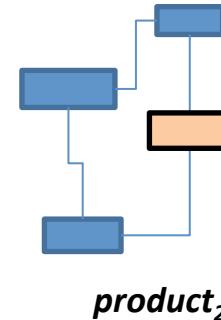
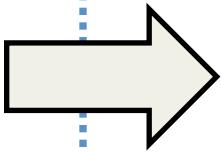
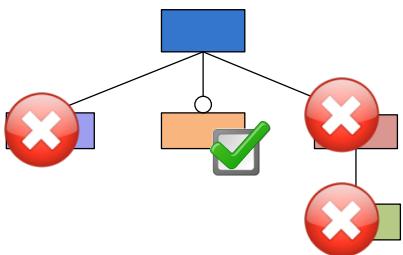
The dream for an SPL practitioner



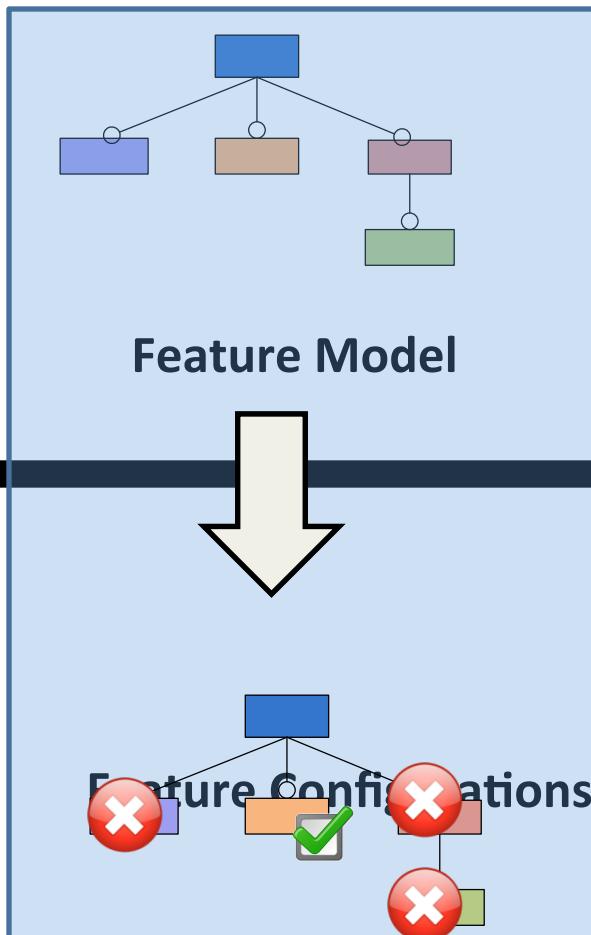
99% domain engineering, 1% application engineering

- specifies what you want (click, click, click) a customized product is automatically built for you
- Iterate the process for n products

Specific requirements

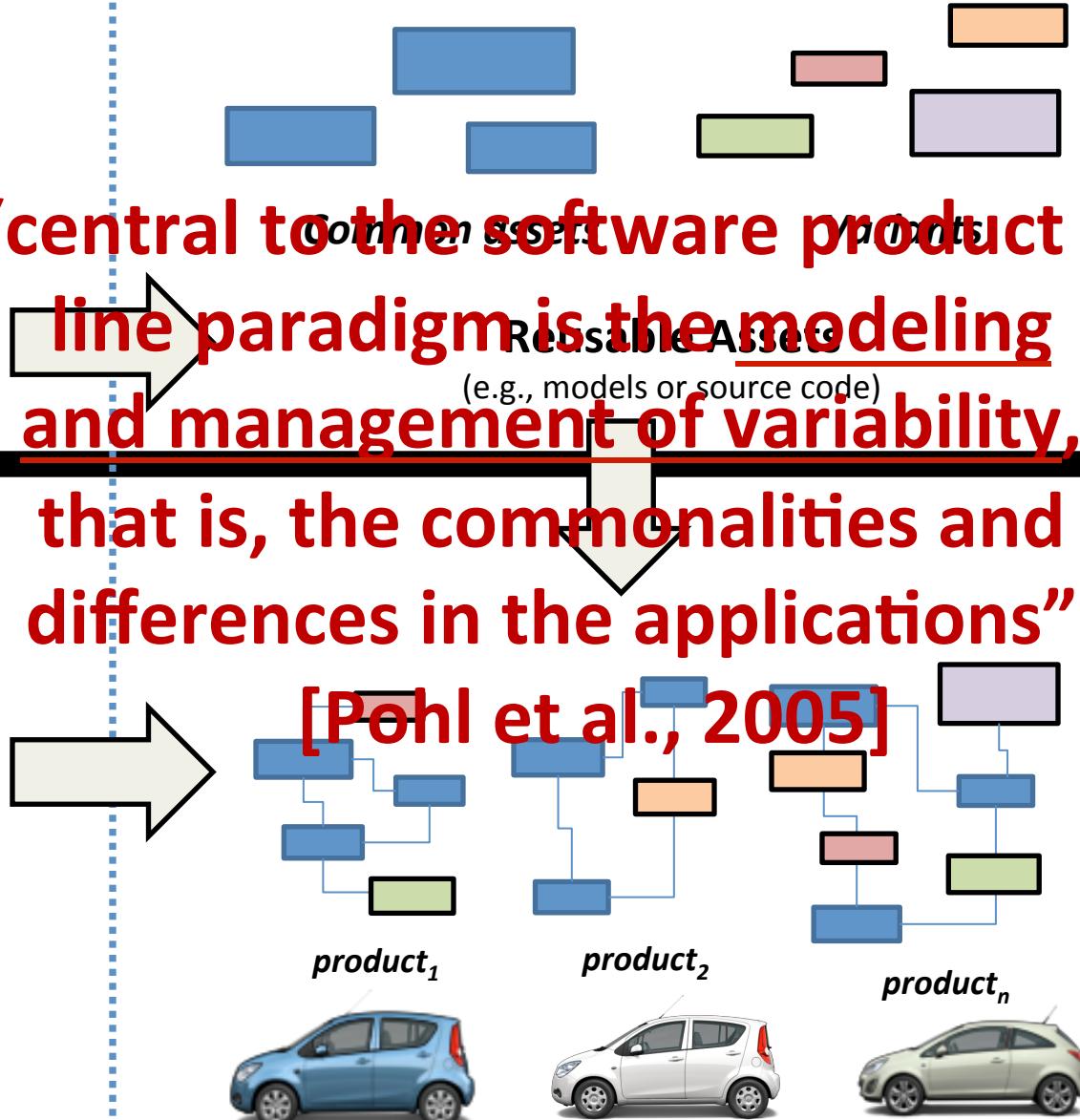


Domain engineering (development for reuse)



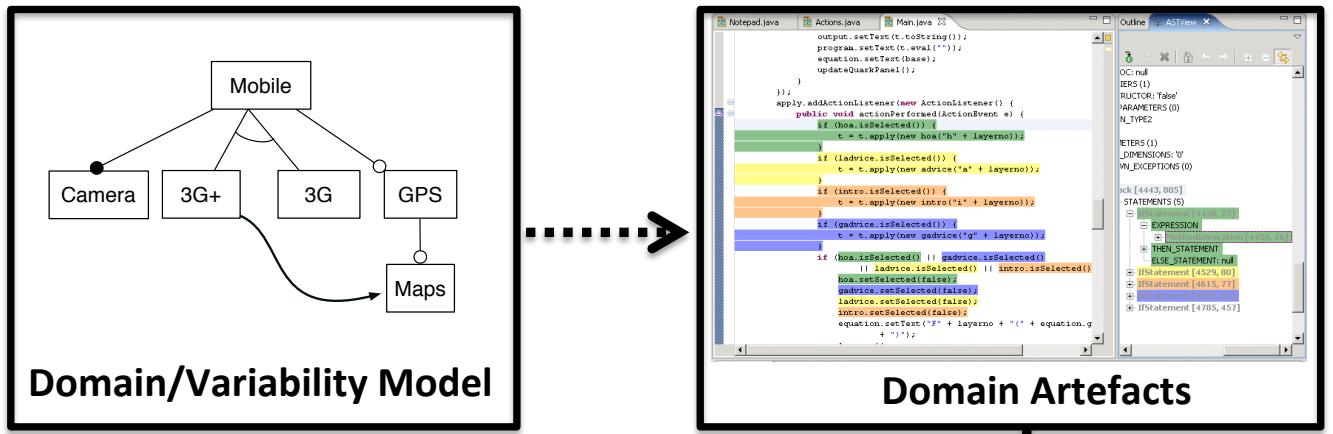
“central to the software product line paradigm is the modeling and management of variability, that is, the commonalities and differences in the applications”

[Pohl et al., 2005]

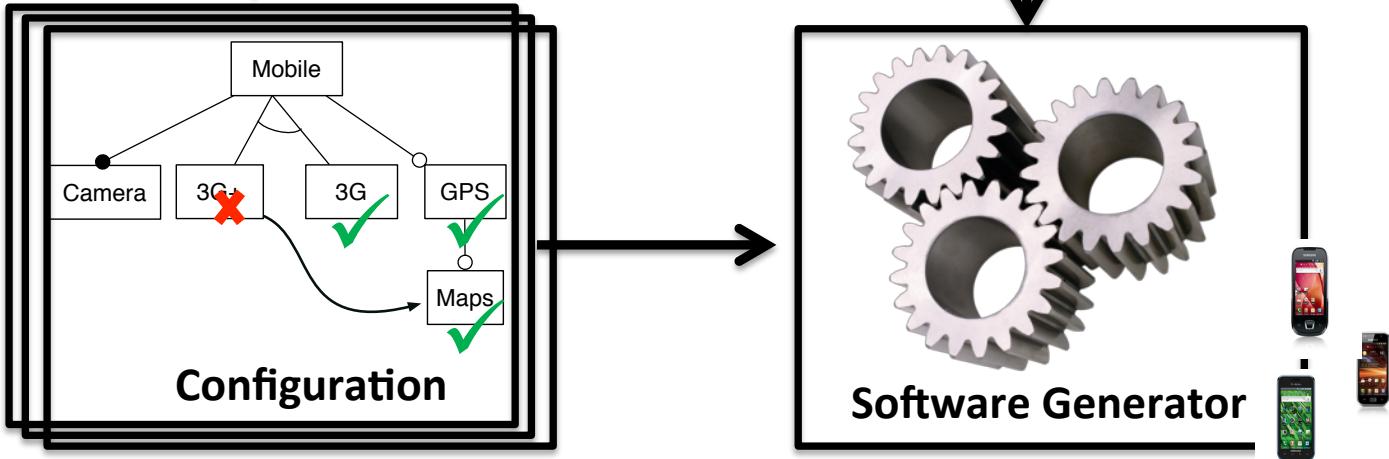


Application engineering (development with reuse)

Domain Engineering



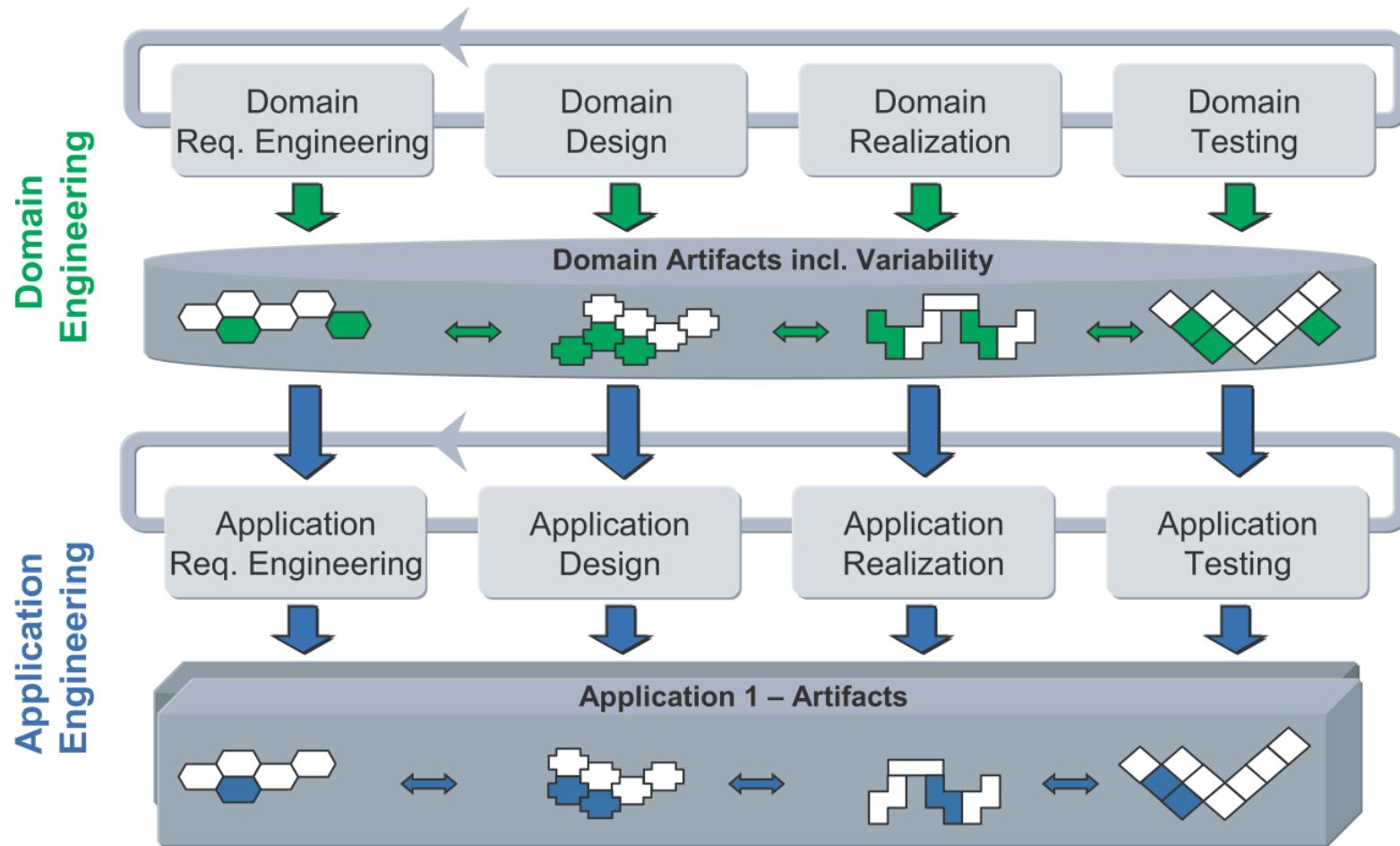
Application Engineering



« the investments required to develop the reusable artifacts during **domain engineering**, are outweighed by the benefits of deriving the individual products during **application engineering** »

Activities related to domain engineering and application engineering

Software Product-Line Engineering



Domain Analysis

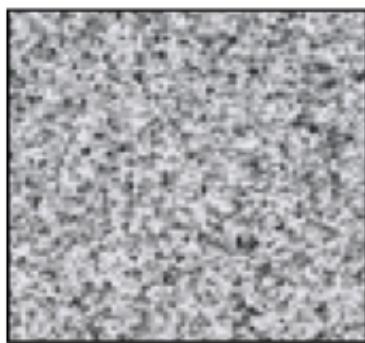
- Collect relevant domain information
 - domain experts (interviews, workshops)
 - system handbooks, textbooks, prototyping, experiments,
 - already known requirements on future systems
 - Creative activity
- Domain Definition
 - examples of systems in a domain,
 - counterexamples (i.e. systems outside the domain),
 - generic rules of inclusion or exclusion (e.g. “Any system having the capability X belongs to the domain.”).
- Domain vocabulary
- Domain concepts
- and integrate it into a coherent *domain model*
 - more or less formal

Czarnecki and
Eisenecker (2000)

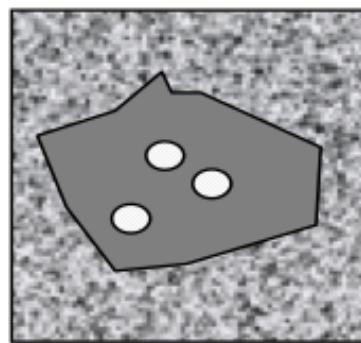
Domain Model

- Ontology, ER, UML, Feature Model
- Analysis of similarity
 - Analyze similarities between entities, activities, events, relationships, structures, etc.
- Analysis of variations
 - Analyze variations between entities, activities, events, relationships, structures, etc.
- Clustering
- Abstraction
- Classification
- Generalization
- Vocabulary construction

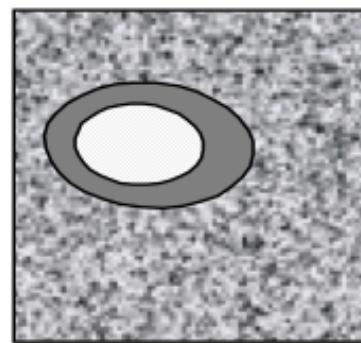
Scope



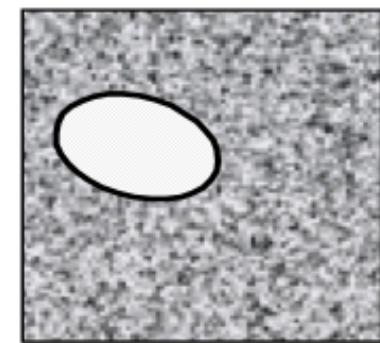
a.



b.



c.



d.

A photograph of an old, green-painted pickup truck that has been left to decay in a field. The truck is heavily rusted, particularly on the body and the front fenders. The driver's side door is open, revealing the interior frame and some remaining mechanical components. The truck is positioned in front of a dense thicket of green bushes and tall grass. In the bottom left corner, there are some wooden planks and metal debris, suggesting a construction or demolition site nearby.

Unused flexibility

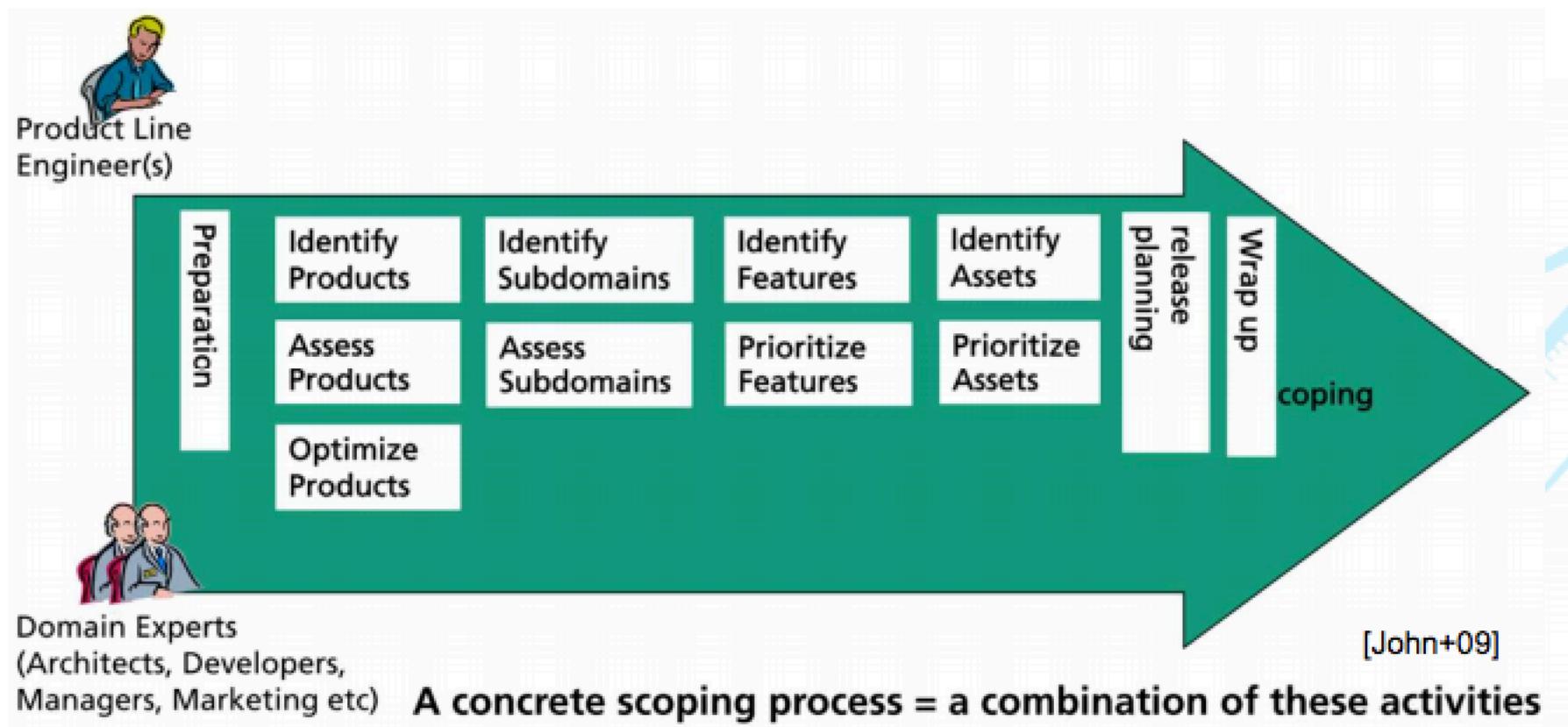
A photograph of a white police sedan engulfed in intense orange and yellow flames. The car is positioned in the foreground, angled towards the right. Behind it, a large building facade is visible, featuring signs for "GUESS", "DRUM SHOP", and "KIRK'S". To the right of the burning car, a crowd of people stands behind a metal fence, some holding cameras. The scene is set at night or in low light, with the fire being the primary light source.

DRUM SHOP

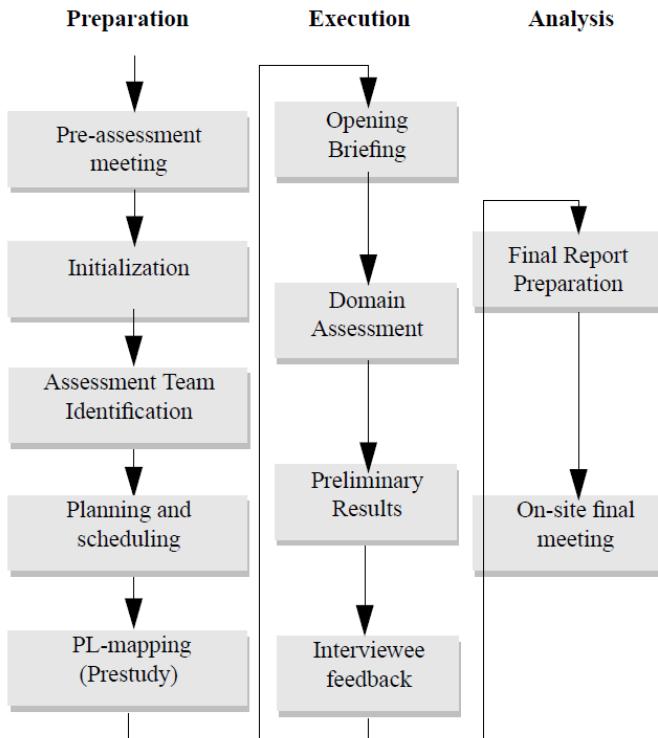
JAS
STORE

Illegal Variant

Scoping Activities



Domain/Product Line Scoping



Schmid 2002

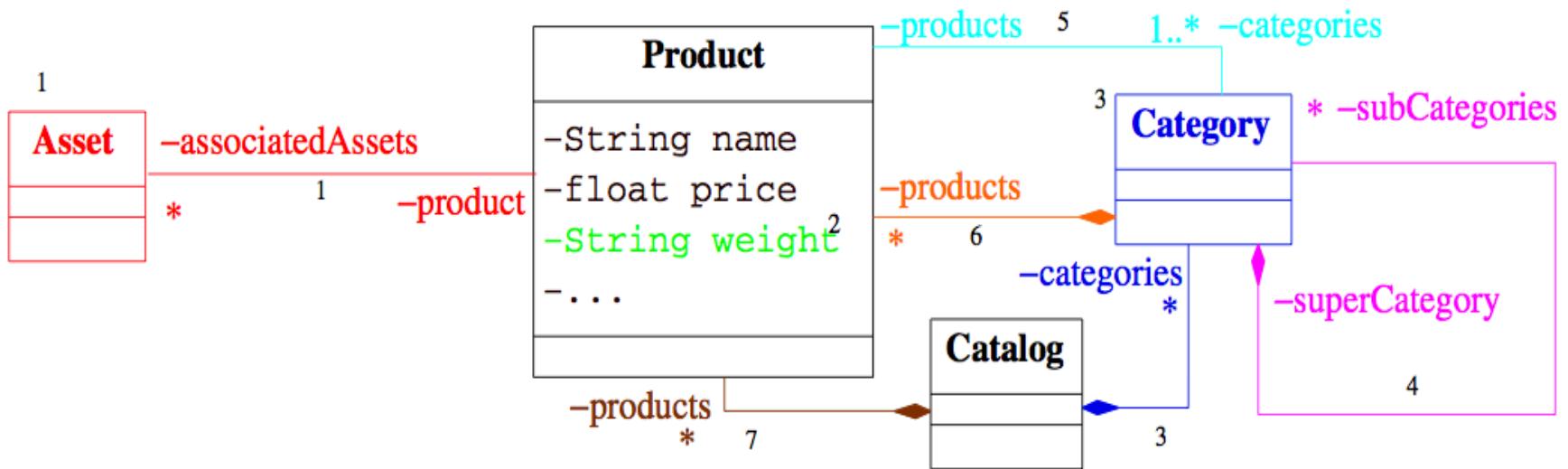
		exist.	planned		potent.
			P1	P2	
Domain 1	Sub-Domain 1.1	Feature 1.1.1	X	X	X
	Sub-Domain 1.1	Feature 1.1.2	—	X	X
	Sub-Domain 1.1	Feature 1.1.3	X	X	—

	Sub-Domain 1.n	Feature 1.n.1	X	—	X
Domain 2	Sub-Domain 2.1	Feature 2.1.1	—	X	X
	Sub-Domain 2.1
	Sub-Domain 2.1
	Sub-Domain 2.1	Feature m.1.1	—	X	—
...

Domain Design

Presence conditions:

true		MultiLevel		4
AssociatedAssets		MultipleClassification		5
PhysicalGoods		Categories & !MultipleClassification		6
Categories		MultipleClassification !Categories		7



Domain Implementation

- Reusable assets
 - e.g. reusable services
- Domain-specific languages
- Generators
- Reuse infrastructure
- Many implementation techniques

Domain-specific language (DSL)

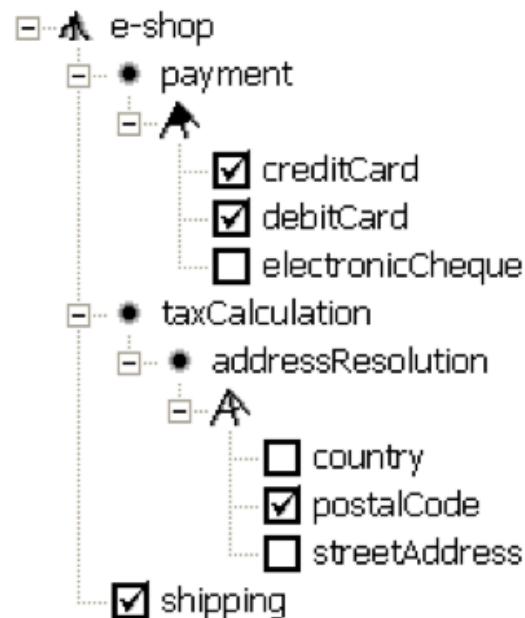
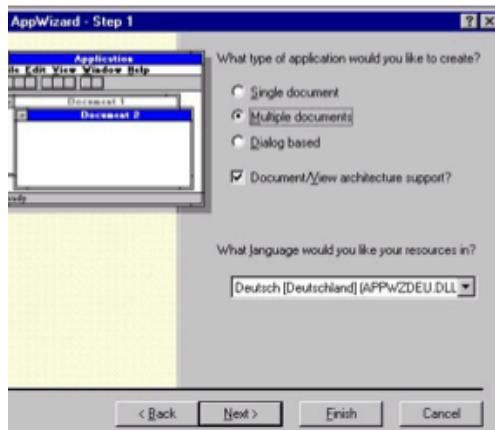
- Domain-specific **abstractions**
 - a DSL provides pre-defined abstractions to directly represent concepts from the domain.
- Domain-specific **concrete syntax**
 - a DSL offers a natural notation for a given domain and avoids syntactic clutter that often results when using a general-purpose language.
- Domain-**specific** error checking
- Domain-**specific** optimizations
- Domain-**specific** tool support

DSLs

- textual languages (stand-alone or embedded in a general-purpose programming language),
- visual, diagrammatic languages
- form-based languages

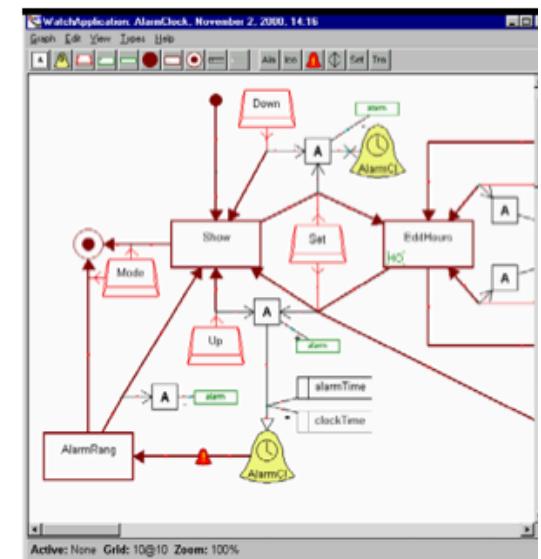
Routine configuration

Creative construction



Wizard

Feature-based configuration



Graph-like language

Dummy Feature Model

```
feature runtimeCalibration : false
feature bumper : true
feature sonar : false
feature debugOutput : true
```

```
{sonar} task sonartask cyclic prio = 2 every = 100 {
    int s = ecrobot_get_sonar_sensor(SENSOR_PORT_T::NXT_PORT_S2);
    sonarHistory[sonarIndex] = s;
    sonarIndex = sonarIndex + 1;
    if ( sonarIndex == 10 ) {
        sonarIndex = 0;
    }
    int ss = 0;
    for ( int i = 0; i < 10; i = i + 1; ) {
        ss = ss + sonarHistory[i];
    }
    currentSonar = ss / 10;
    { debugOutput } { debugInt(2, "sonar:", currentSonar); }
}
```

doc This is the cyclic task that is called every 1ms to do the actual control of the task run cyclic prio = 2 every = 2 {

```
stateswitch linefollower
state running
{bumper} int bump = ecrobot_get_touch_sensor(SENSOR_PORT_T::NXT_PORT_S3);
{bumper} if ( bump == 1 ) {
    {debugOutput} { debugString(3, "bump:", "BUMP!"); }
    event linefollower:bumped
    terminate;
}
```

```
{sonar} if ( currentSonar < 150 ) {
    event linefollower:blocked
    terminate;
}
```

```
int light = ecrobot_get_light_sensor(SENSOR_PORT_T::NXT_PORT_S1);
if ( light < ( WHITE + BLACK ) / 2 ) {
    updateMotorSettings(SLOW, FAST);
} else {
    updateMotorSettings(FAST, SLOW);
}
```

```
{debugOutput} { debugInt(4, "light:", light); }
```

```
{sonar} state paused
updateMotorSettings(0, 0);
if ( currentSonar < 255 ) {
    event linefollower:unblocked
}
{bumper} state crash
updateMotorSettings(0, 0);
```

```
default
<noop>;
```

Voelter (SPLC'11)

Configuring Models and Code

Preprocessor for Java code (Munge)

```
class Example {  
    void main() {  
        System.out.println("immer");  
        /*if[DEBUG]*/  
        System.out.println("debug info");  
        /*end[DEBUG]*/  
    }  
}
```

java Munge ~~-DDEBUG -DFEATURE2~~ Example.java

↑
configuration option

Kastner's slide

```

class Graph {
    Vector nv = new Vector(); Vector ev = new Vector();
    Edge add(Node n, Node m) {
        Edge e = new Edge(n, m);
        nv.add(n); nv.add(m); ev.add(e);
        e.weight = new Weight();
        return e;
    }
    Edge add(Node n, Node m, Weight w)
        Edge e = new Edge(n, m);
        nv.add(n); nv.add(m); ev.add(e);
        e.weight = w; return e;
    }
    void print() {
        for(int i = 0; i < ev.size(); i++) {
            ((Edge)ev.get(i)).print();
        }
    }
}

```

```

class Color {
    static void setDisplayColor(Color c) { ... }
}

```

```

class Weight { void print() { ... } }

```

```

class Node {
    int id = 0;
    Color color = new Color();
    void print() {
        Color.setDisplayColor(color);
        System.out.print(id);
    }
}

```

```

class Edge {
    Node a, b;
    Color color = new Color();
    Weight weight = new Weight();
    Edge(Node _a, Node _b) { a = _a; b = _b; }
    void print() {
        Color.setDisplayColor(color);
        a.print(); b.print();
        weight.print();
    }
}

```

Kastner's slide

Mapping: an example

```

class Graph {
    Vector nv = new Vector(); Vector ev = new Vector();
    Edge add(Node n, Node m) {
        Edge e = new Edge(n, m);
        nv.add(n); nv.add(m); ev.add(e);
        e.weight = new Weight();
        return e;
    }
    Edge add(Node n, Node m, Weight w) {
        Edge e = new Edge(n, m);
        nv.add(n); nv.add(m); ev.add(e);
        e.weight = w; return e;
    }
    void print() {
        for(int i = 0; i < ev.size(); i++) {
            ((Edge)ev.get(i)).print();
        }
    }
}

```

```

class Node {
    int id = 0;
    Color color = new Color();
    void print() {
        Color.setDisplayColor(color);
        System.out.print(id);
    }
}

```

```

class Edge {
    Node a, b;
    Color color = new Color();
    Weight weight = new Weight();
    Edge(Node _a, Node _b) { a = _a; b = _b; }
    void print() {
        Color.setDisplayColor(color);
        a.print(); b.print();
        weight.print();
    }
}

```

```

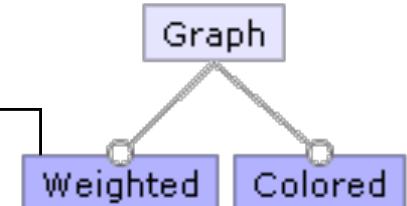
class Color {
    static void setDisplayColor(Color c) { ... }
}

```

```

class Weight { void print() { ... } }

```



```

class Graph {
    Vector nv = new Vector(); Vector ev = new Vector();
    Edge add(Node n, Node m) {
        Edge e = new Edge(n, m);
        nv.add(n); nv.add(m); ev.add(e);
        /*if[WEIGHT]*/
        e.weight = new Weight();
        /*end[WEIGHT]*/
        return e;
    }
    /*if[WEIGHT]*/
    Edge add(Node n, Node m, Weight w)
        Edge e = new Edge(n, m);
        nv.add(n); nv.add(m); ev.add(e);
        e.weight = w; return e;
    }
    /*end[WEIGHT]*/
    void print() {
        for(int i = 0; i < ev.size(); i++) {
            ((Edge)ev.get(i)).print();
        }
    }
}

/*if[WEIGHT]*/
class Weight { void print() { ... } }
/*end[WEIGHT]*/

```

```

class Edge {
    Node a, b;
    /*if[COLOR]*/
    Color color = new Color();
    /*end[COLOR]*/
    /*if[WEIGHT]*/
    Weight weight;
    /*end[WEIGHT]*/
    Edge(Node _a, Node _b) { a = _a; b = _b; }
    void print() {
        /*if[COLOR]*/
        Color.setDisplayColor(color);
        /*end[COLOR]*/
        a.print(); b.print();
        /*if[WEIGHT]*/
        weight.print();
        /*end[WEIGHT]*/
    }
}

/*if[COLOR]*/
class Color {
    static void setDisplayColor(Color c) { ... }
}
/*end[COLOR]*/

```

```

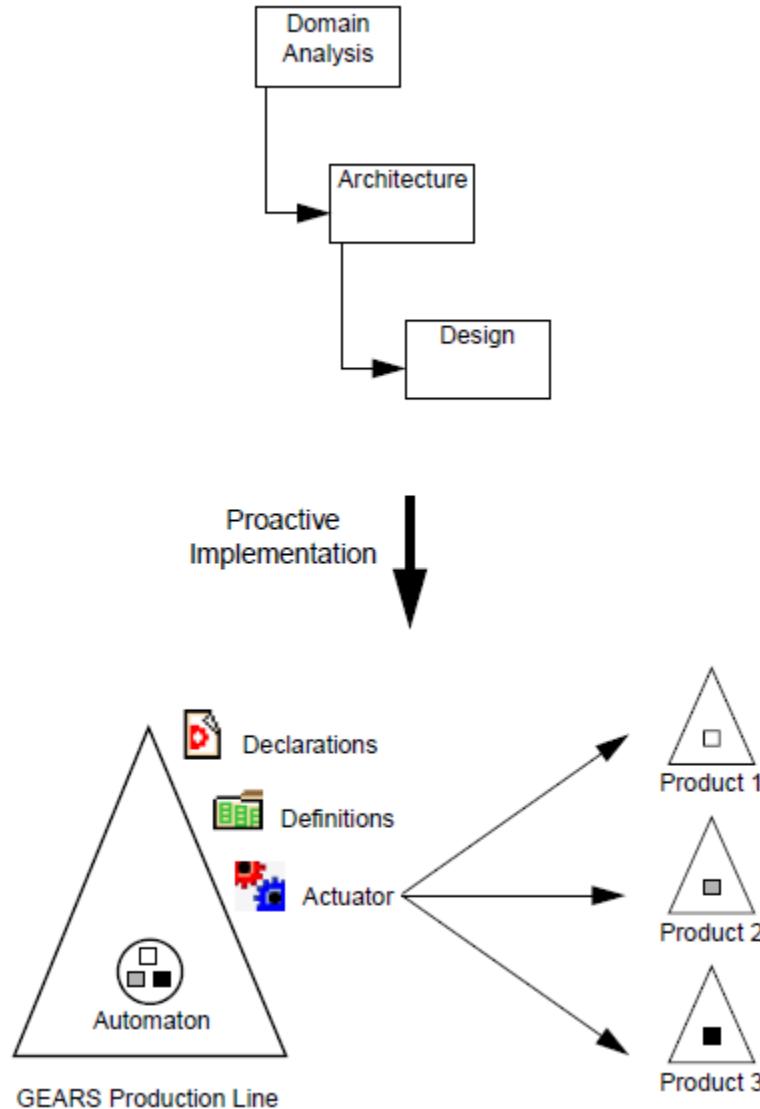
class Node {
    int id = 0;
    /*if[COLOR]*/

```

Adoption and Strategies

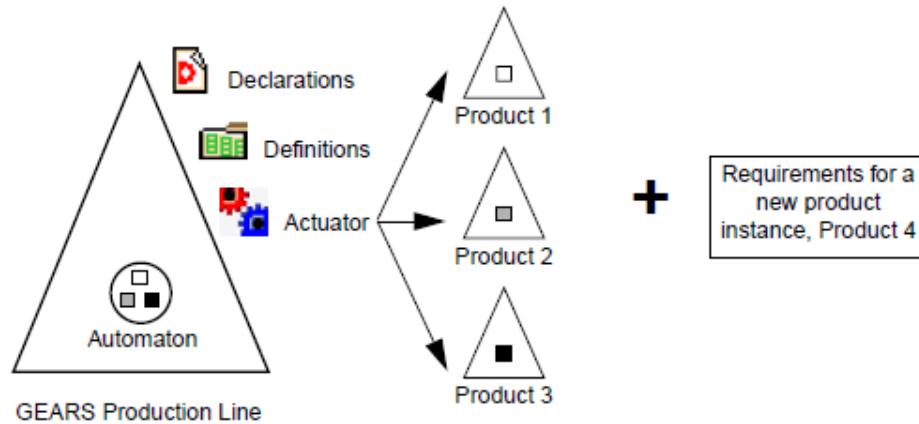
- **Proactive**
- **Reactive**
- **Extractive**

Proactive



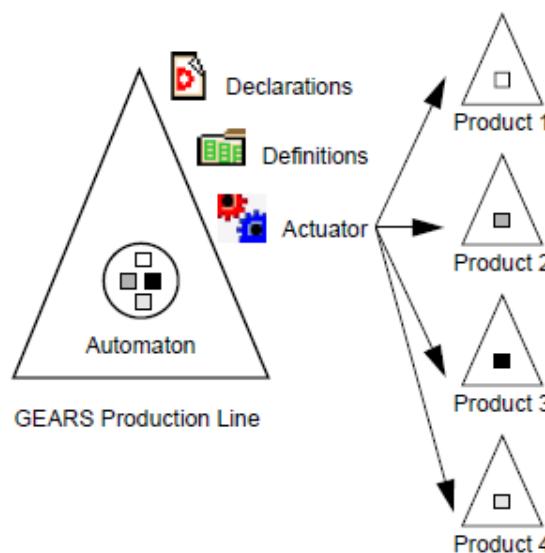
[Krueger 2002]

Reactive



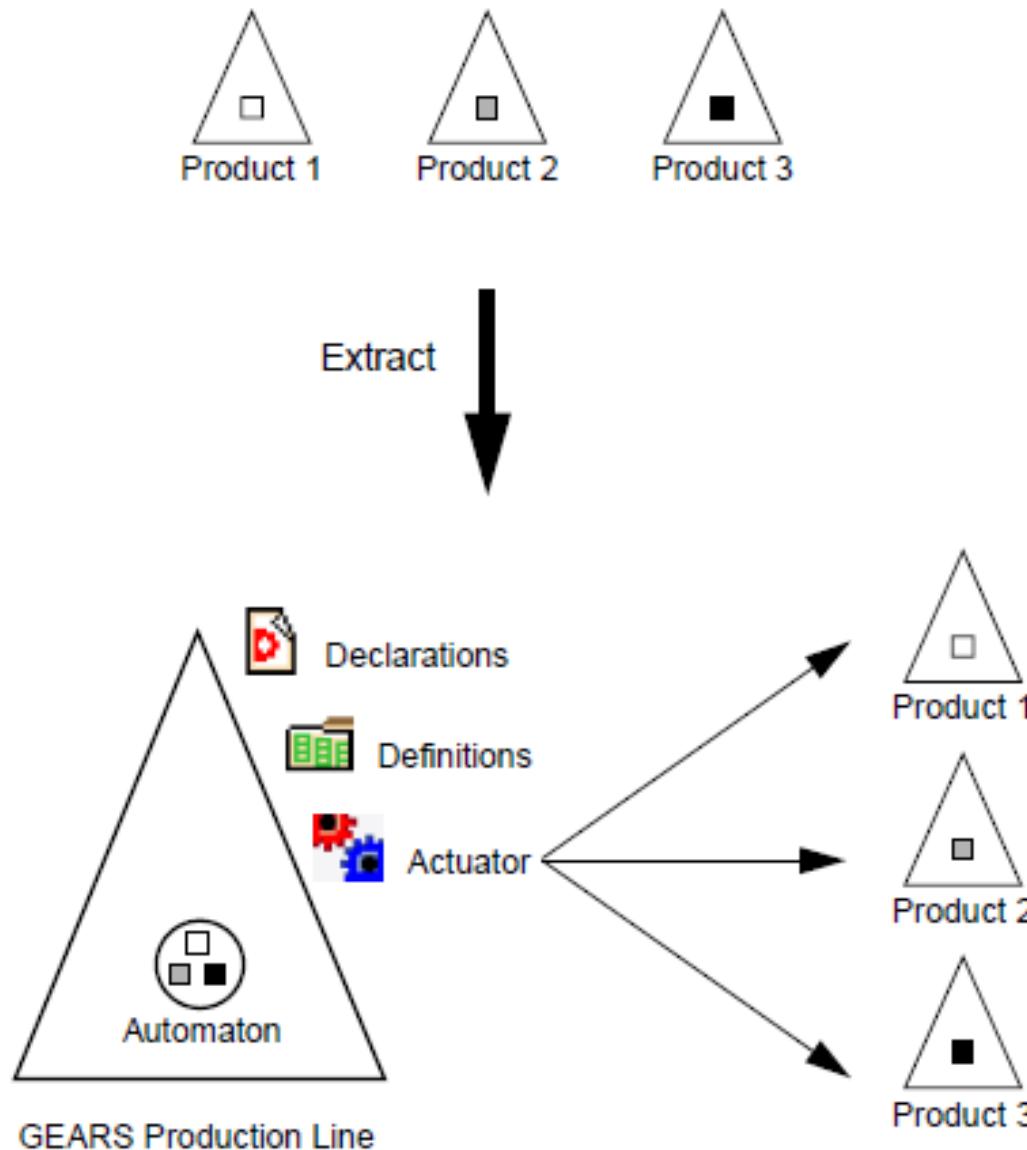
React

Iterate



[Krueger 2002]

Extractive

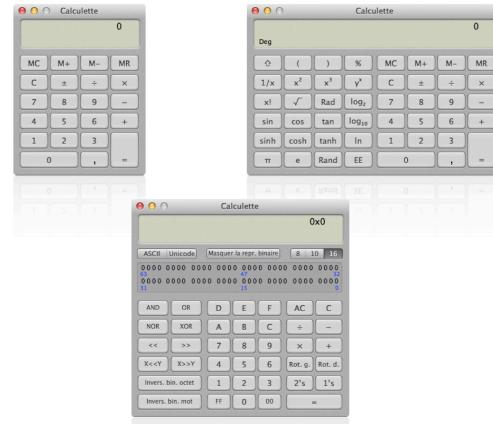
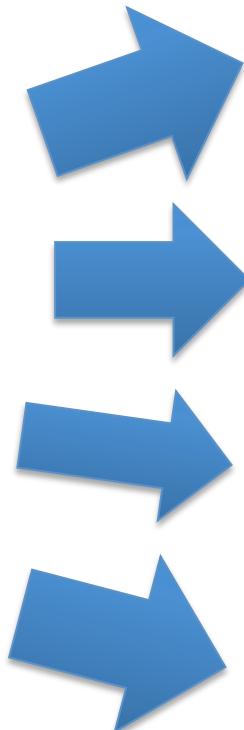


Software Product Line Engineering

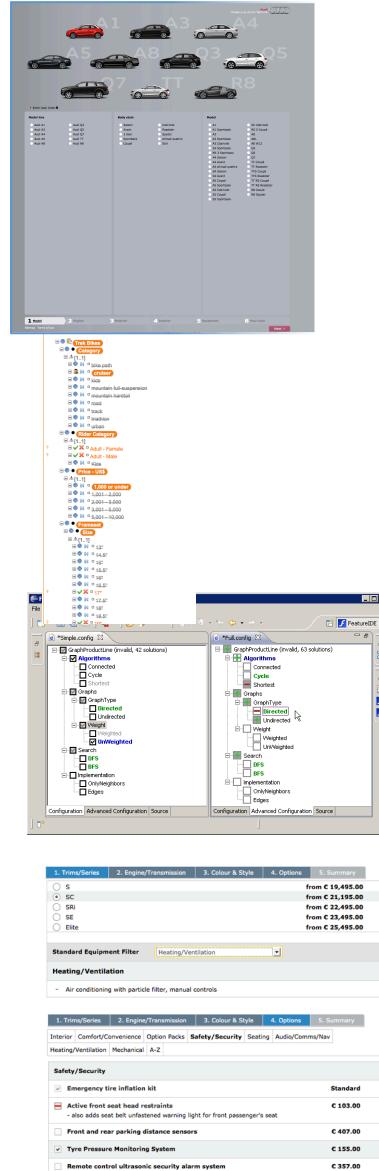
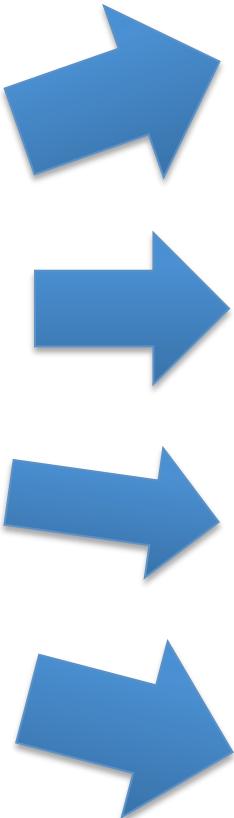
Model-based perspective

Modeling and implementing system families such that a desired system can be automatically generated from a specification written in one or more textual or graphical domain-specific languages.

Models
And
Languages

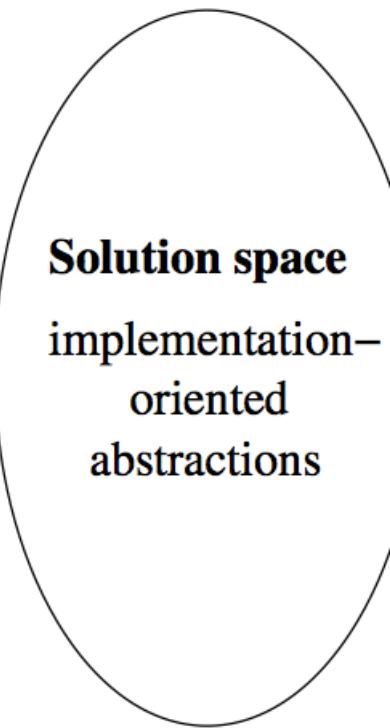
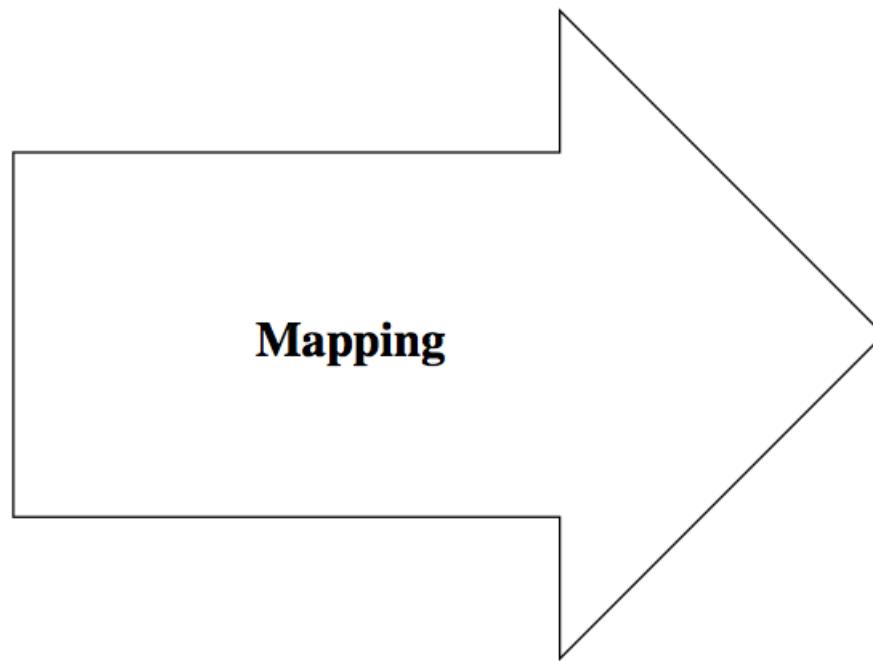
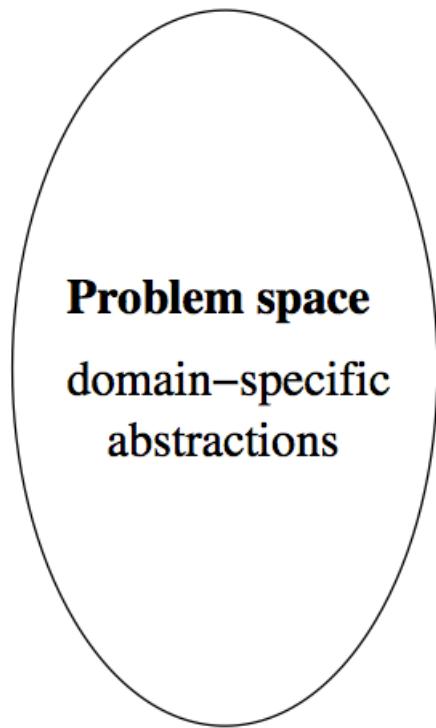


Models And Languages

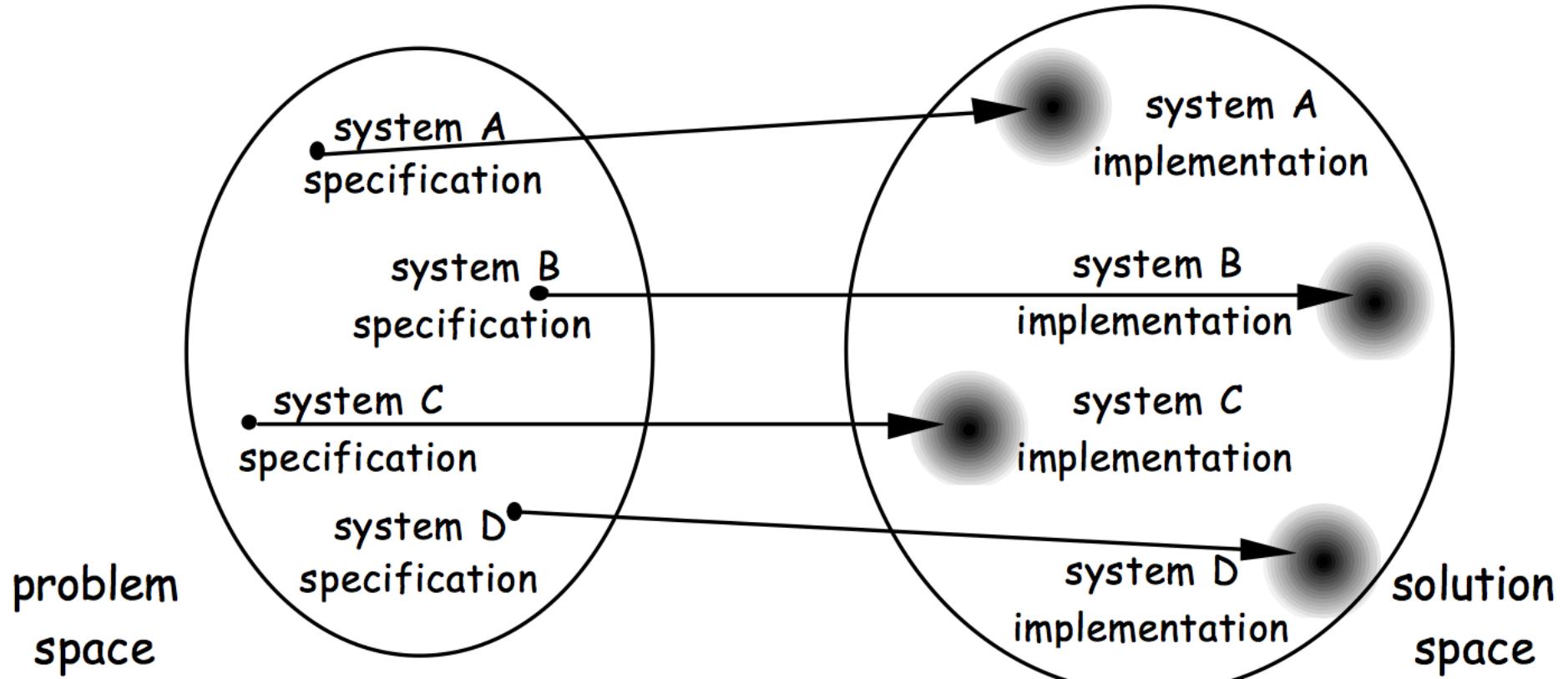


Generative approach

- Programming the generation of programs
 - Very old practice
 - Metaprogramming: generative language and target language are the same
 - Reflection capabilities
- Generalization of this idea:
 - from a specification written in one or more textual or graphical domain-specific languages
 - you generate customized variants

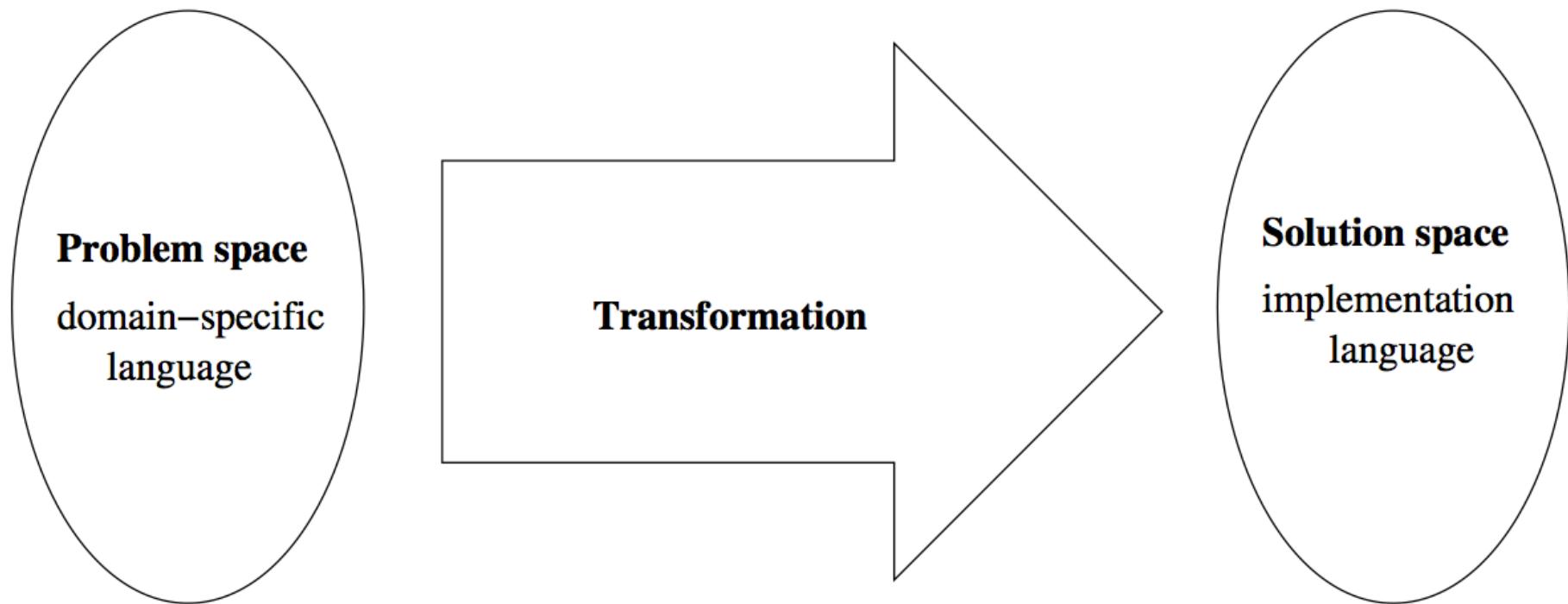


[Czarnecki and Eisenecker 2000]

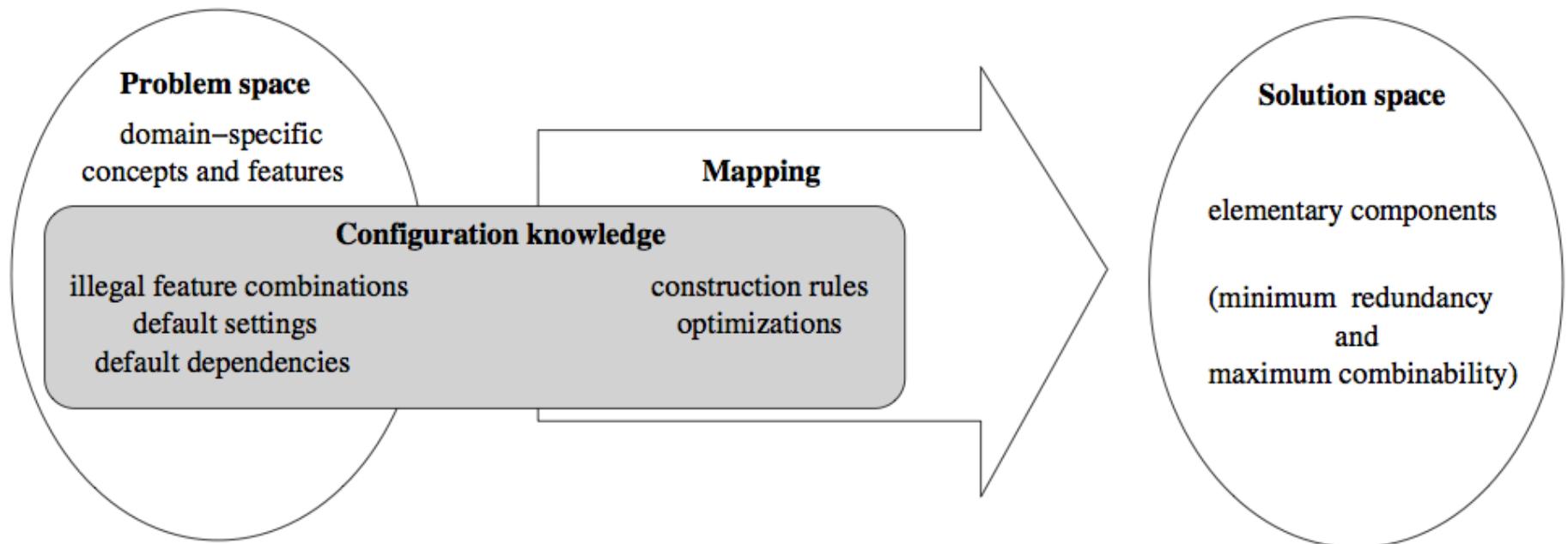


[Czarnecki, PhD thesis]

S



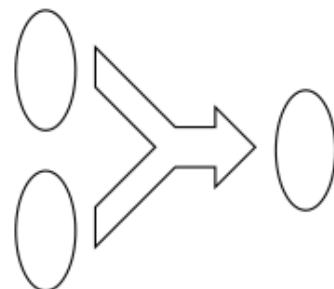
[Czarnecki and Eisenecker 2000]



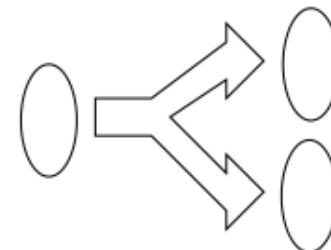
[Czarnecki and Eisenecker 2000]



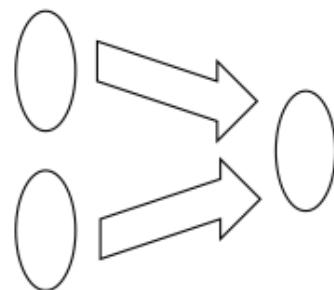
a. Chaining of mappings



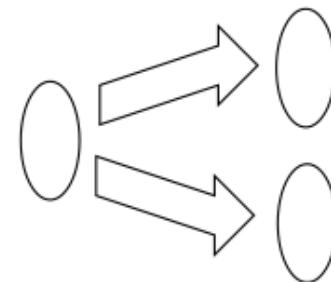
b. Multiple problem spaces

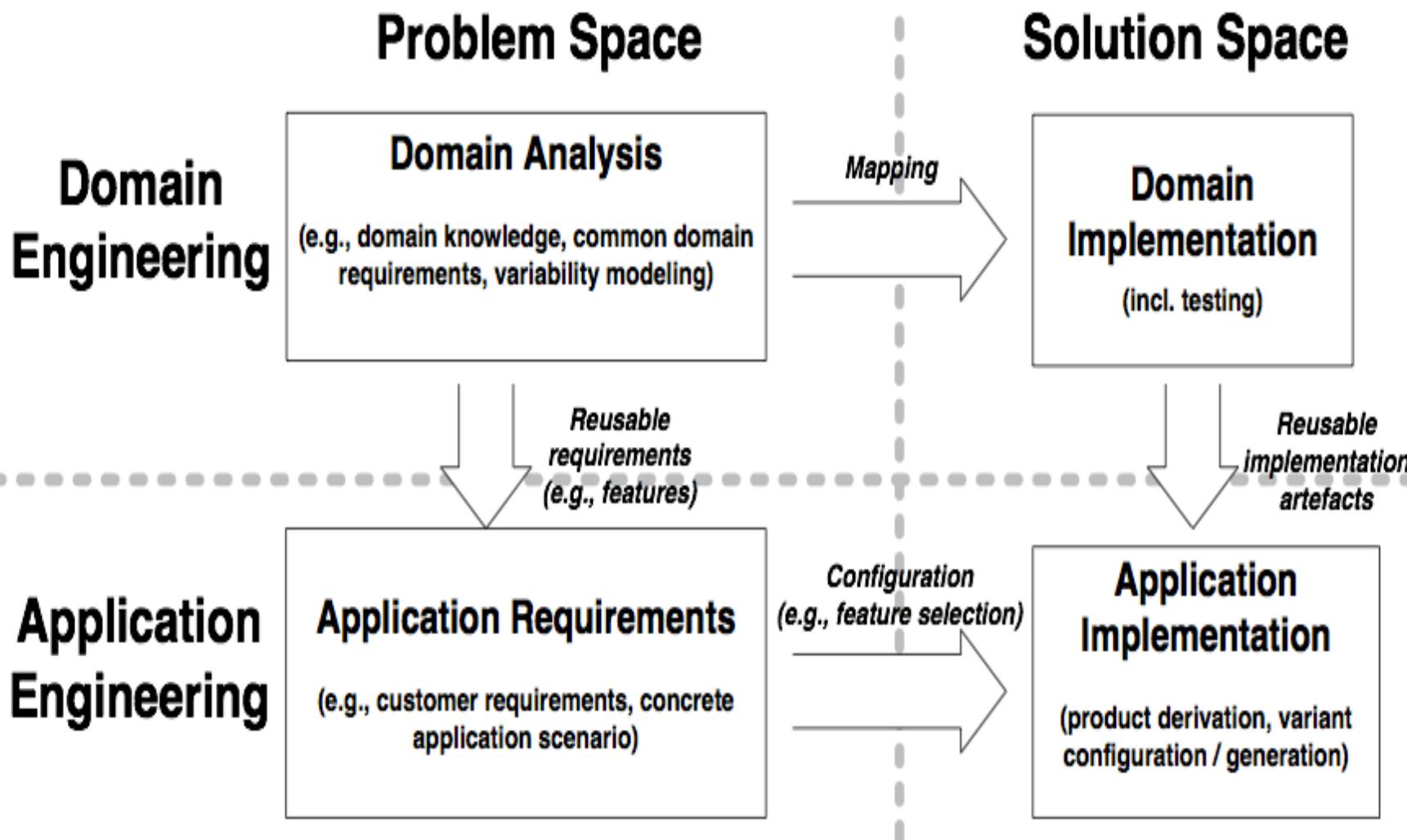


c. Multiple solution spaces

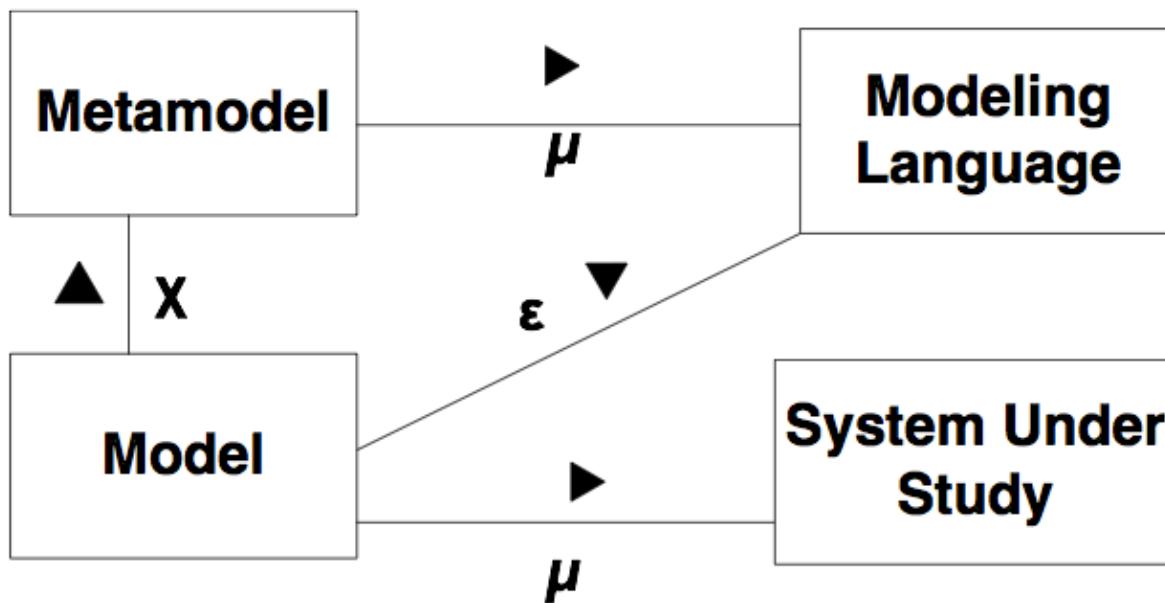


d. Alternative problem spaces e. Alternative solution spaces





MDE



X : ConformantTo

μ : RepresentationOf

ϵ : ElementOf

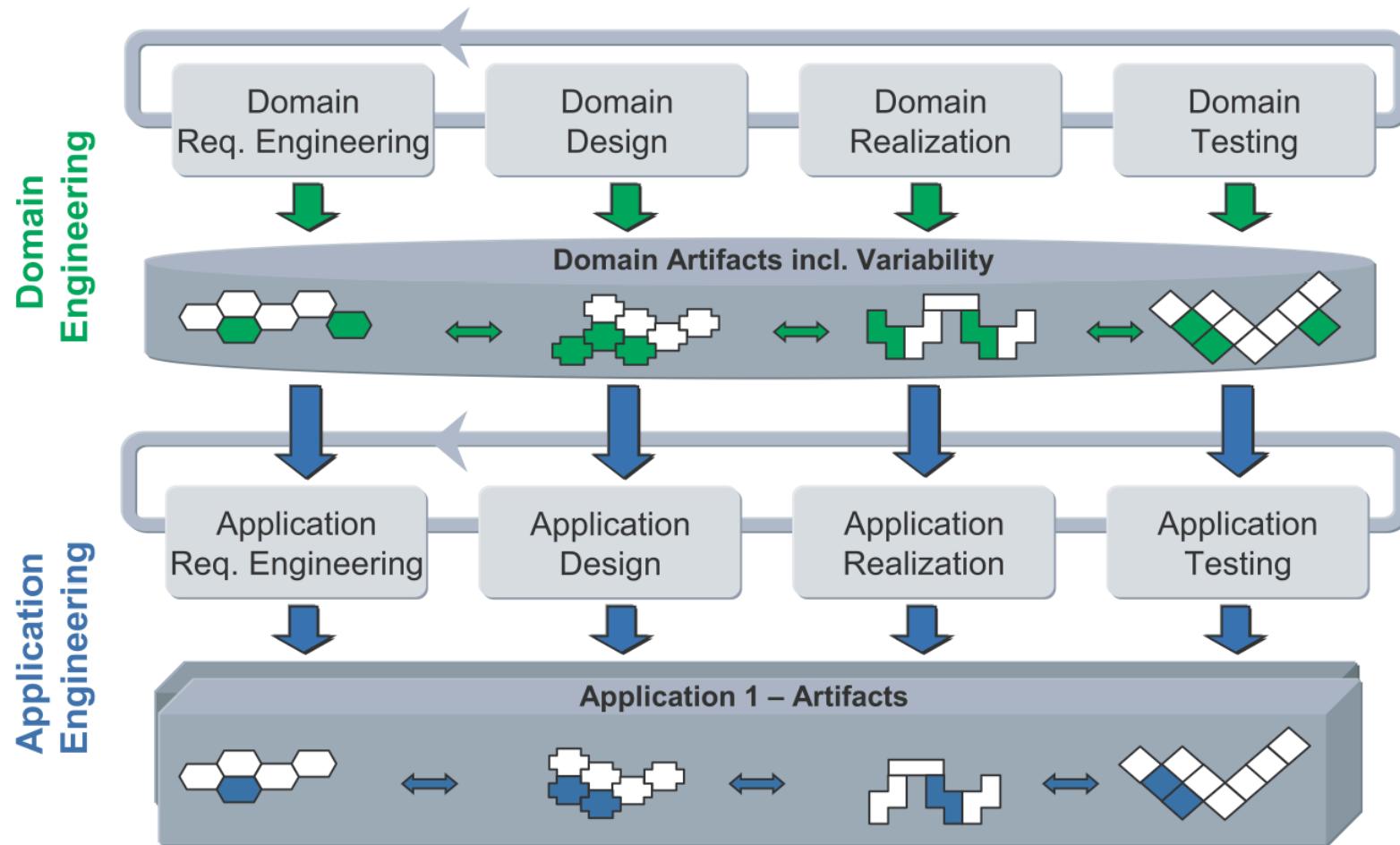
Model in a nutshell

- In essence, a model is an **abstraction** of some aspect of a system under study.
- Some details are hidden or removed to **simplify** and focus attention.
- A model is an abstraction since **general** concepts can be formulated by abstracting common properties of instances or by extracting common features from specific examples.

Promises of Model-driven Engineering

- Reducing the gap between the problem space and the solution space
- Raising the level of abstraction
- Avoiding accidental complexity
- **Abstraction & Transformations**

Software Product-Line Engineering



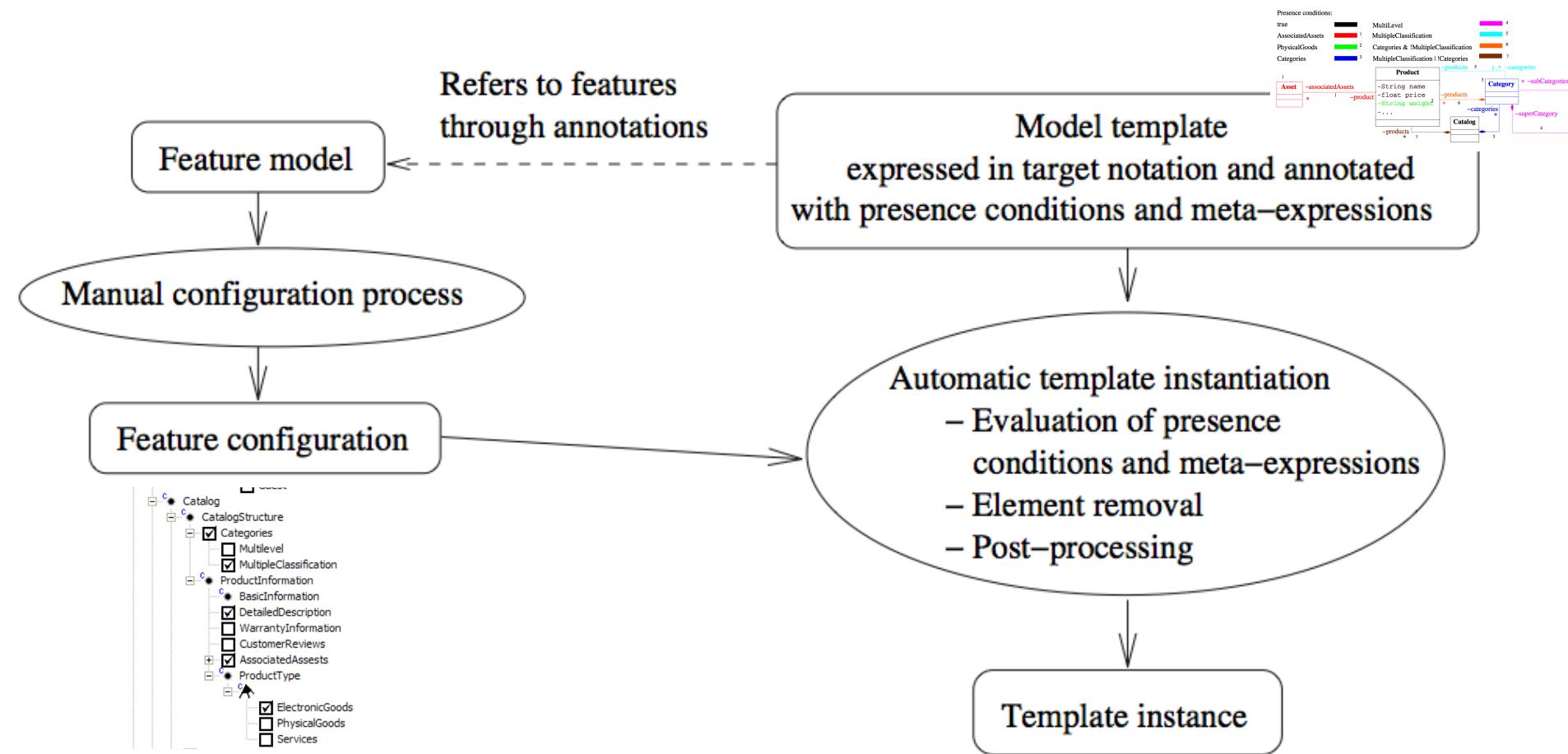
Model-driven Engineering in the SPL framework

- Domain Engineering
 - Domain Models
 - Level of abstraction
 - Domain-specific modeling languages
 - (visual or textual) syntax, precise semantics
 - analyzed (verification)
 - Traceability between the artefacts
- Application Engineering
 - Model transformations (automation)
- Reduce the gap

Feature-based

Model Templates

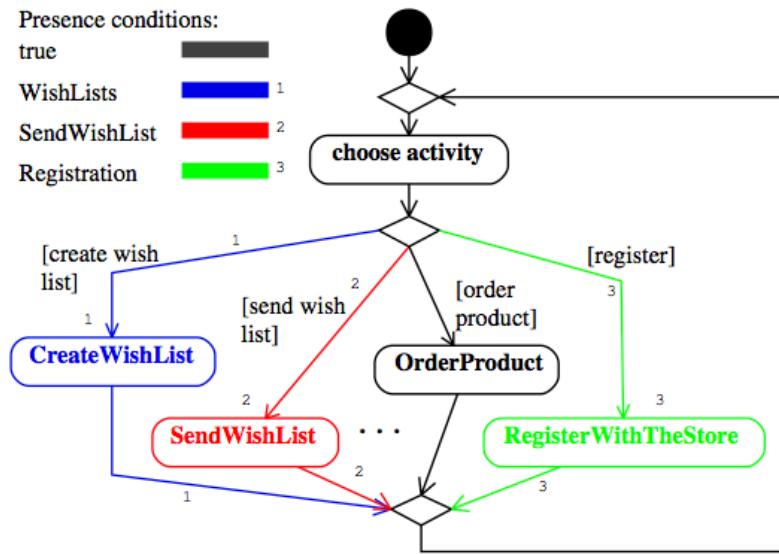
Realizing variability: derivation of models



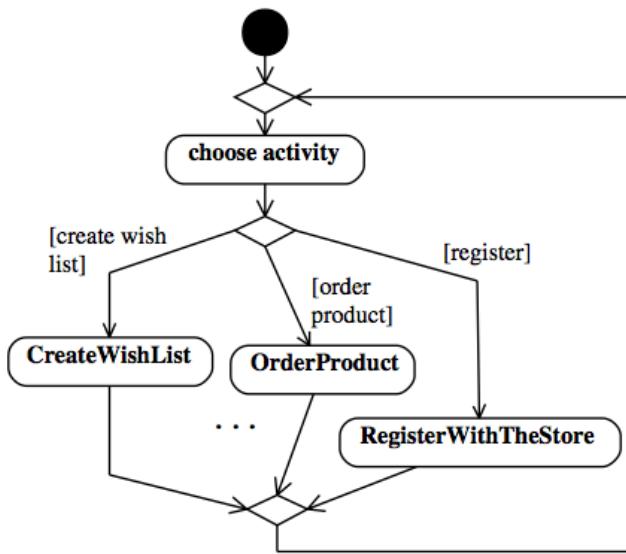
Example

Presence conditions:

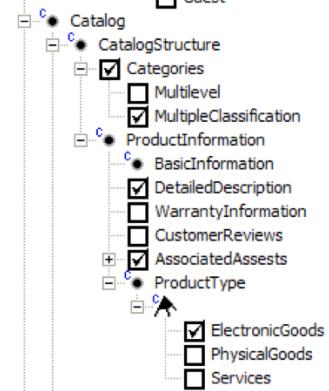
- true
- WishLists
- SendWishList
- Registration



(a) Storefront template

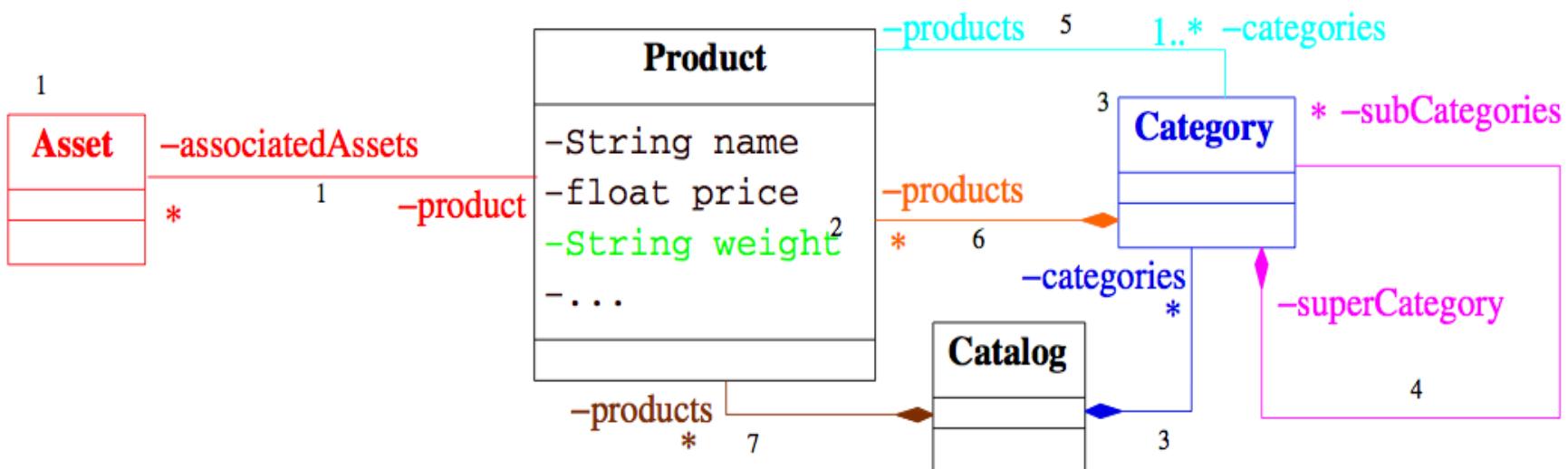


(b) Storefront instance



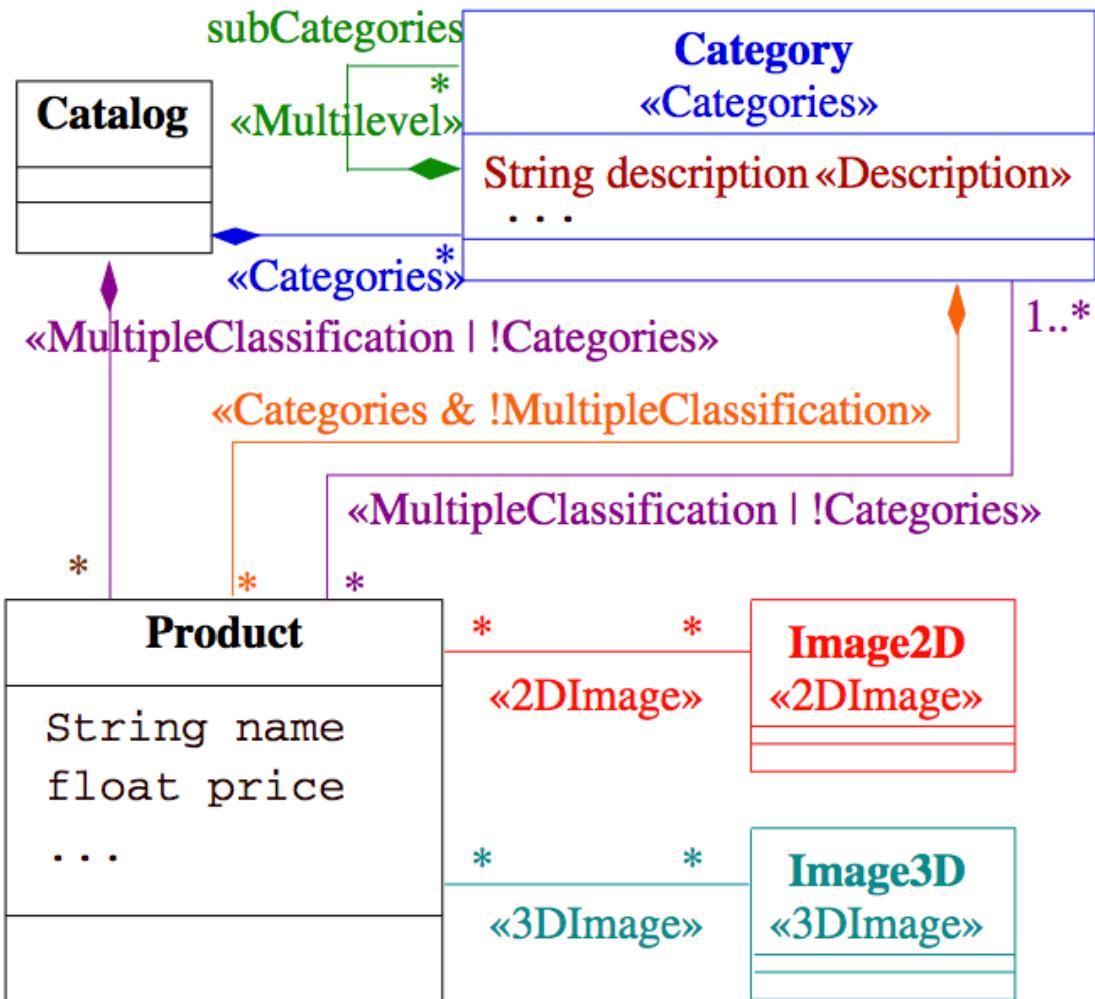
Presence conditions:

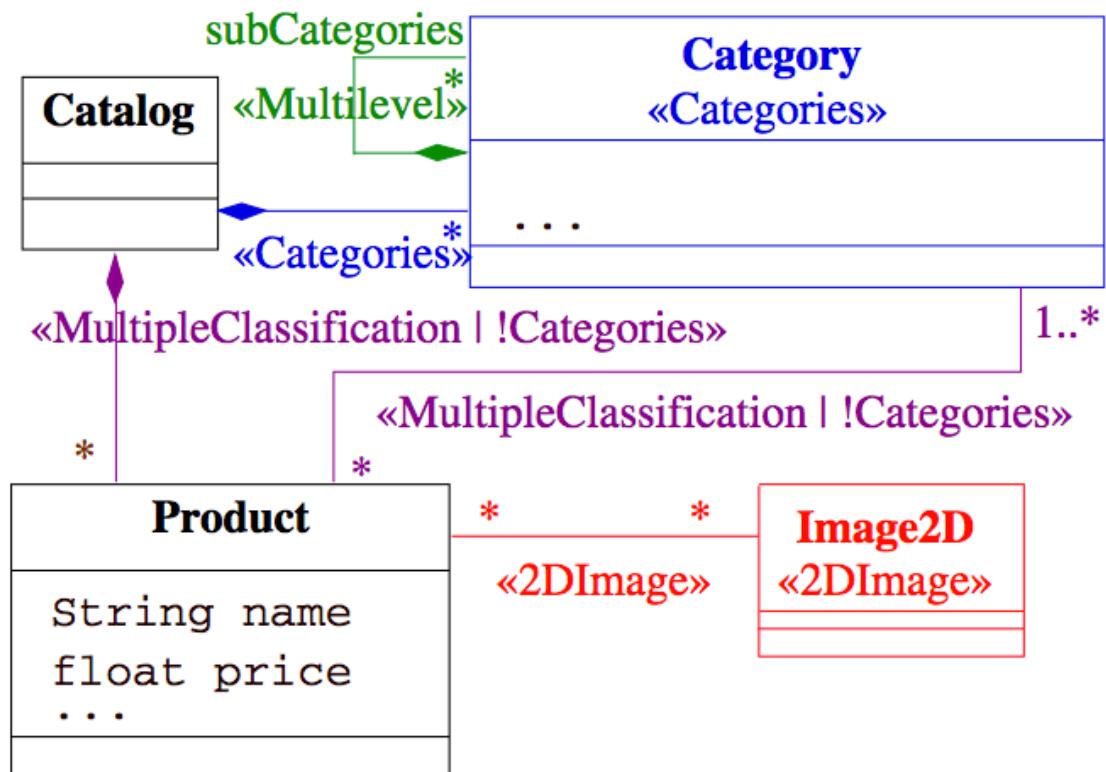
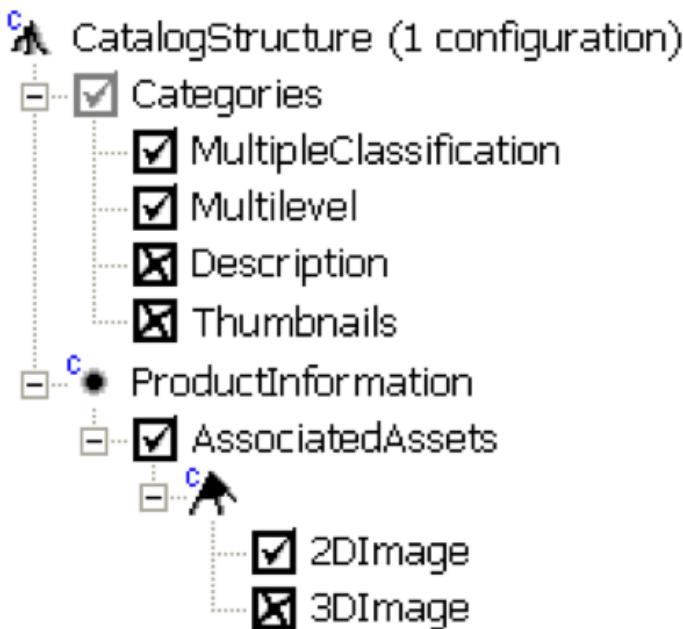
true		MultiLevel		4
AssociatedAssets		MultipleClassification		5
PhysicalGoods		Categories & !MultipleClassification		6
Categories		MultipleClassification !Categories		7



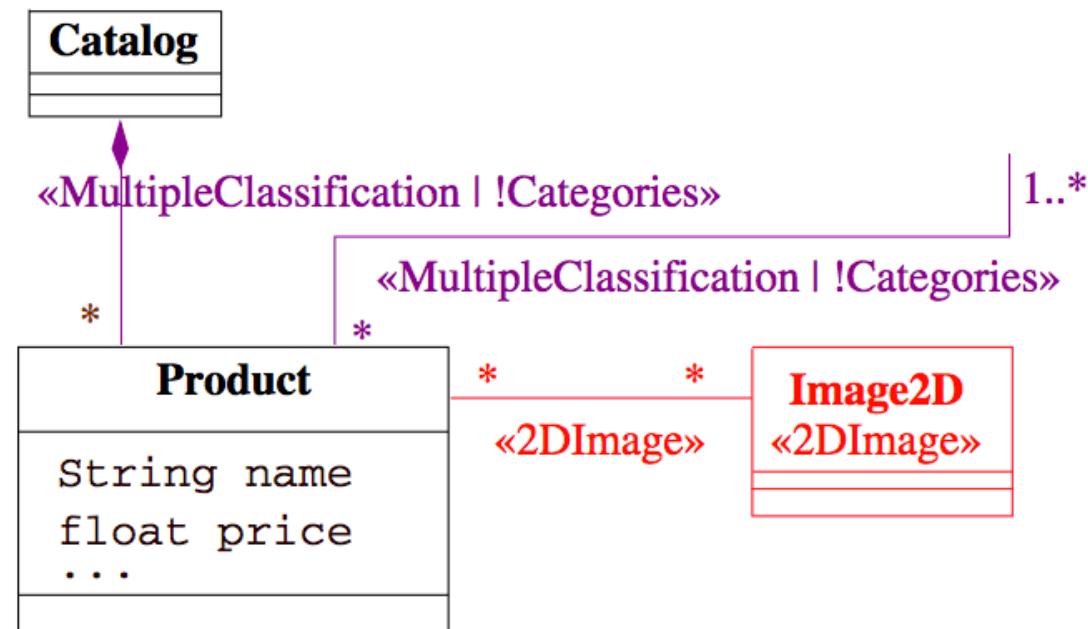
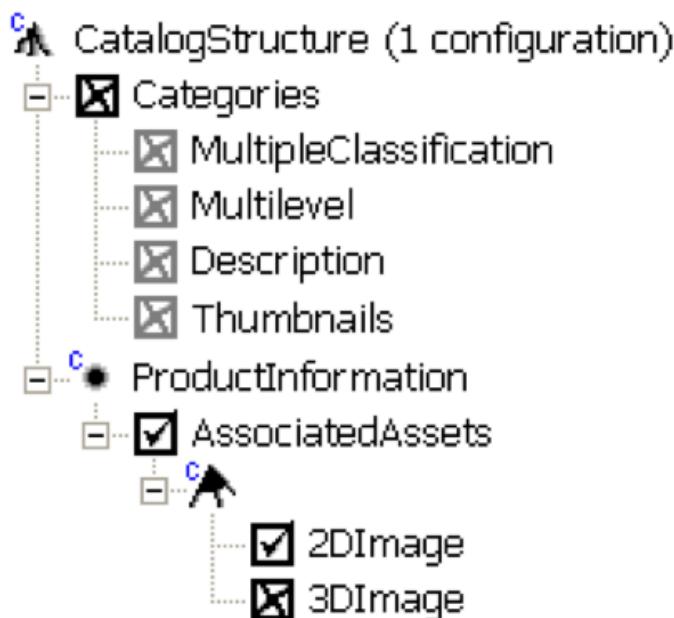
▲ CatalogStructure (52 configurations)

- ● Categories
 - ○ MultipleClassification
 - ○ Multilevel
 - ○ Description
 - ○ Thumbnails
- ● ProductInformation
 - ○ AssociatedAssets
 - □ 2DImage ←
 - □ 3DImage





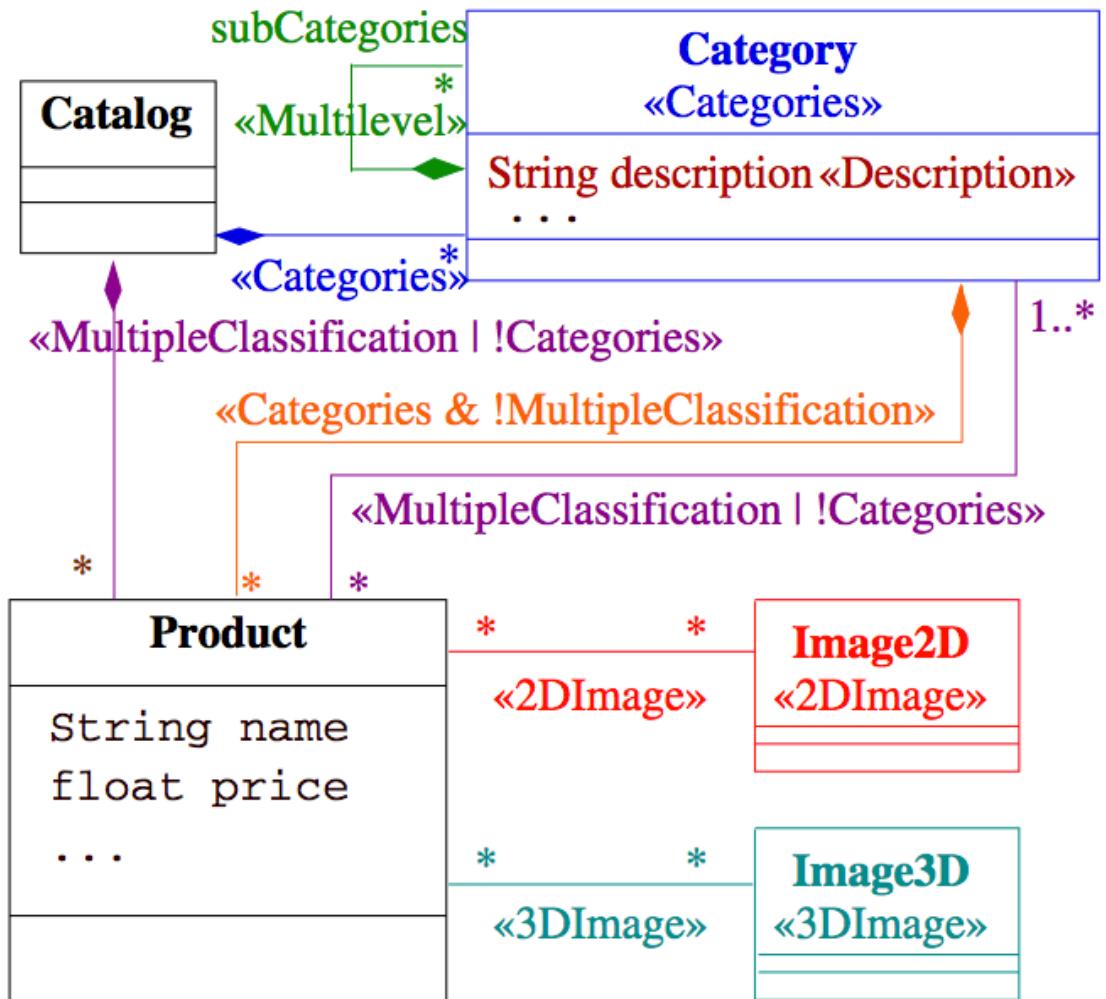
Ooops



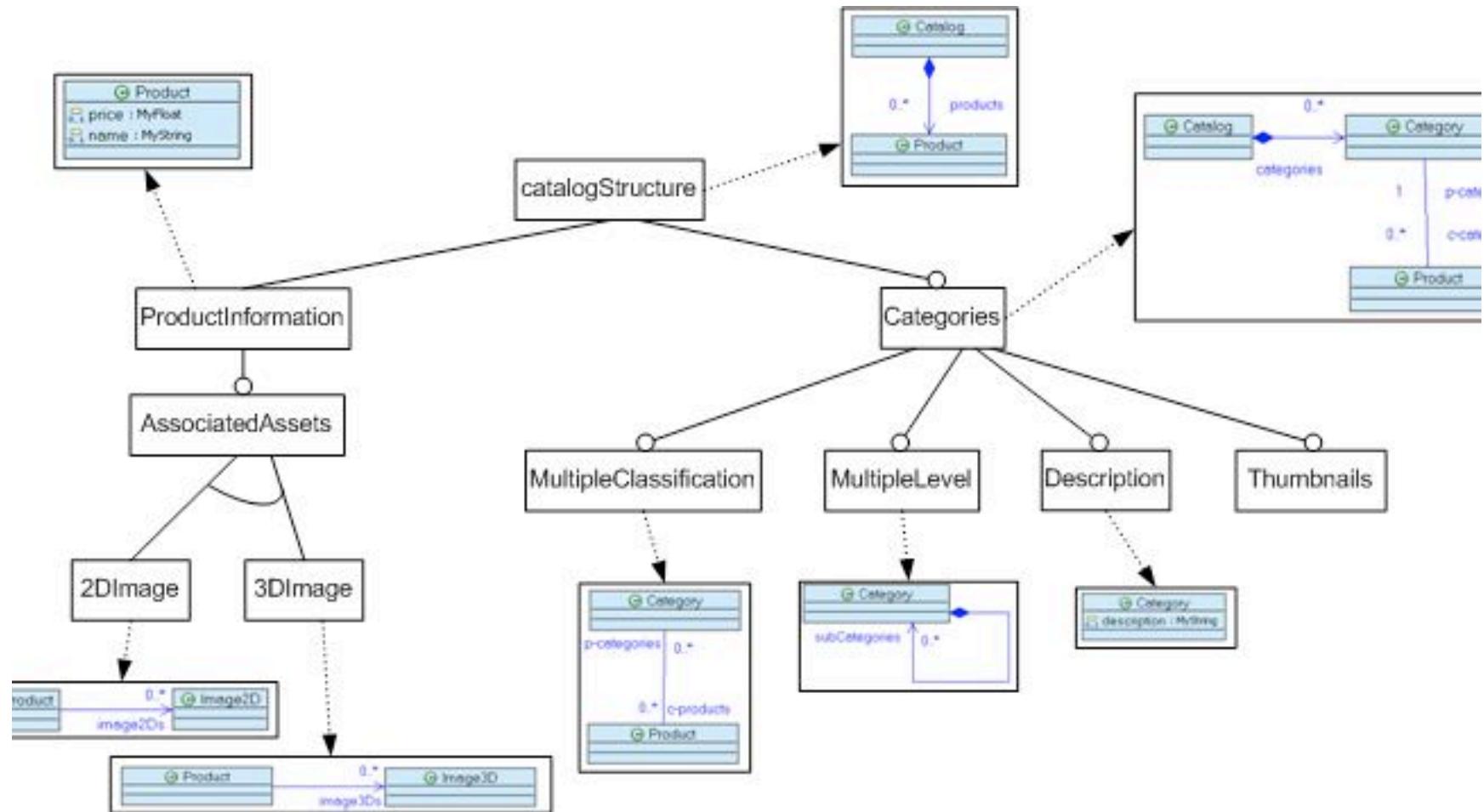
Safe composition? No!

CatalogStructure (52 configurations)

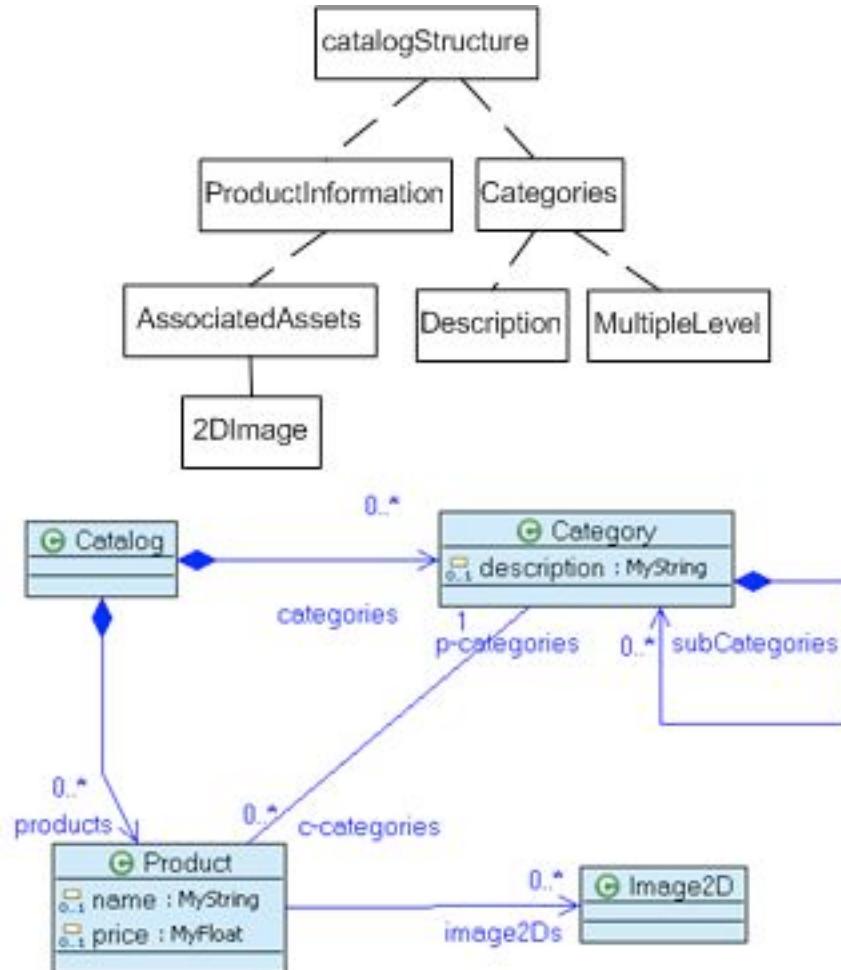
- Categories
 - MultipleClassification
 - Multilevel
 - Description
 - Thumbnails
- ProductInformation
 - AssociatedAssets
 - 2DImage
 - 3DImage



Another approach



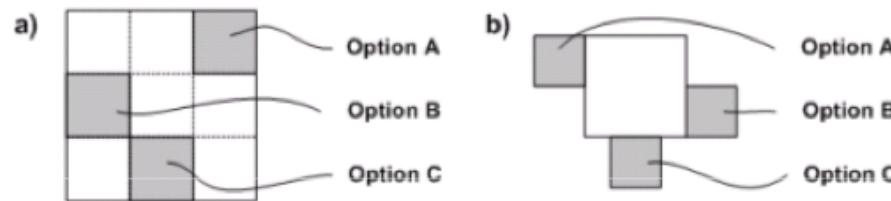
Composition



Realizing variability

- Negative Variability (pruning, annotative)
 - takes optional parts away from an „overall whole“
- Positive Variability (merging, compositional)
 - adds optional parts to a minimal core

Negative vs. Positive Variability



- We need both!

Variability Management

Common features

print – connect with computer...

Variable features

fax – scan – USB port...

Product-specific features

serial port



Variability modeling is...

- ... about identifying
 - common features of concepts
 - variable features of concepts
 - and their dependencies
- and documenting them in a coherent model, called a **feature model**
- Feature modeling is a core activity of important domain-engineering methods

Feature and Psychology

- The Human mind processes lots of information
- Important function of human information processing: reducing the amount of information
- • Describing concepts by features, e.g. a bird has wings, a bill, and can fly
- • Classifying a perceptual unit as an instance of a concept, e.g. Aunt Martha's bird
- Acquisition and evolution of concepts will not be pursued further in this course

Modeling variability (and commonality)

- **Features** are a convenient way to think about commonalities and differences between family members
 - e.g. "MP3 player" – some phones have it, some don't
- Features are usually **coarse-grained** abstractions
 - Abstract from detailed requirements
 - e.g. "when in shuffle mode, the player selects the next track using a random function"
 - Abstract from design/implementation details
 - e.g. "MP3_Player is a concrete subclass of Media_Player"
- An **old but still vague** notion

What's a **feature**? (survey excerpt)

- *[Kang et al.]* “a prominent or distinctive **user-visible aspect, quality or characteristic** of a software system or systems”
- *[Kang et al.]* “distinctively identifiable **functional abstractions** that must be implemented, tested, delivered, and maintained”
- *[Eisenecker and Czarnecki]*. “**anything users** or client programs might want to **control about a concept**”
- *[Bosch et al.]* “A **logical unit of behaviour** specified by a set of **functional and non-functional requirements.**”
- *[Chen et al.]* “a **product characteristic** from user or customer views, which essentially consists of a cohesive **set of individual requirements**”
- *[Batory]* “an elaboration or augmentation of an entity(s) that introduces a **new service, capability** or relationship”

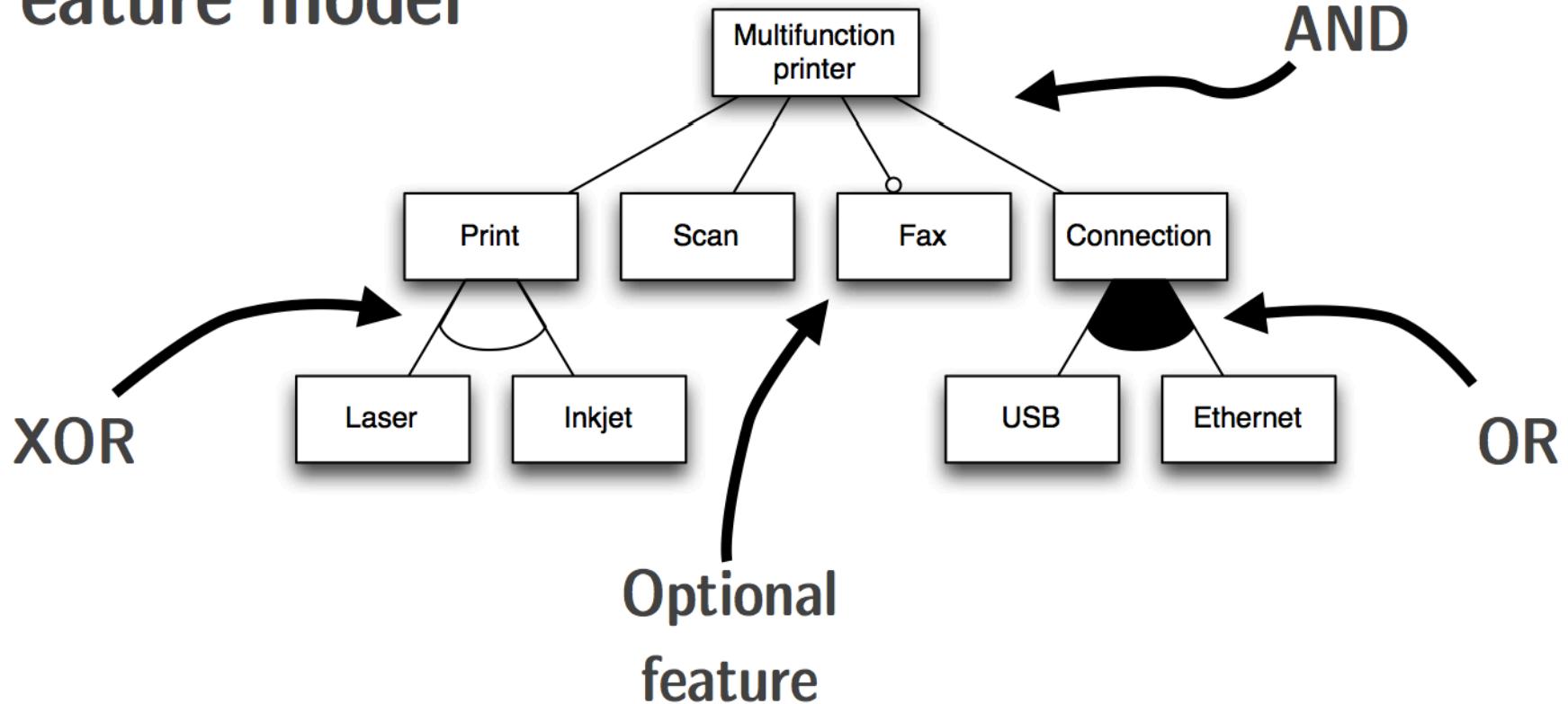
What's in a feature?

- Confusion as to what a feature generally represents (i.e., blackbox view)
- Redefine “Feature” as a 3-tuple of requirements, domain assumptions and specifications
 - Requirements (R)
 - The purpose of the system (optative): ‘*Notify the police of the presence of burglars*’
 - Domain assumptions (W)
 - The given behaviour of the environment (indicative): ‘*Burglars cause movement when they break in*’ and ‘*There is a movement sensor monitoring the house, connected to the system*’
 - Specifications (S)
 - What the system should do to satisfy R given W holds (optative): ‘*Alert the police if the sensor captures movement*’

$$\begin{array}{ccc} \text{Specification} & & \text{Requirement} \\ S, W \models R & & \\ & \text{Domain} & \\ & \text{Assumption} & \end{array}$$

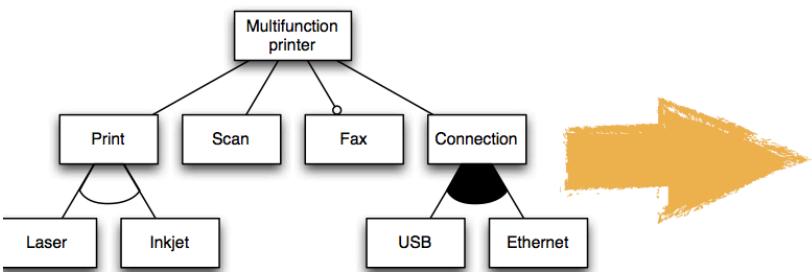
Variability Modelling

Feature model



Feature Models Semantics

d



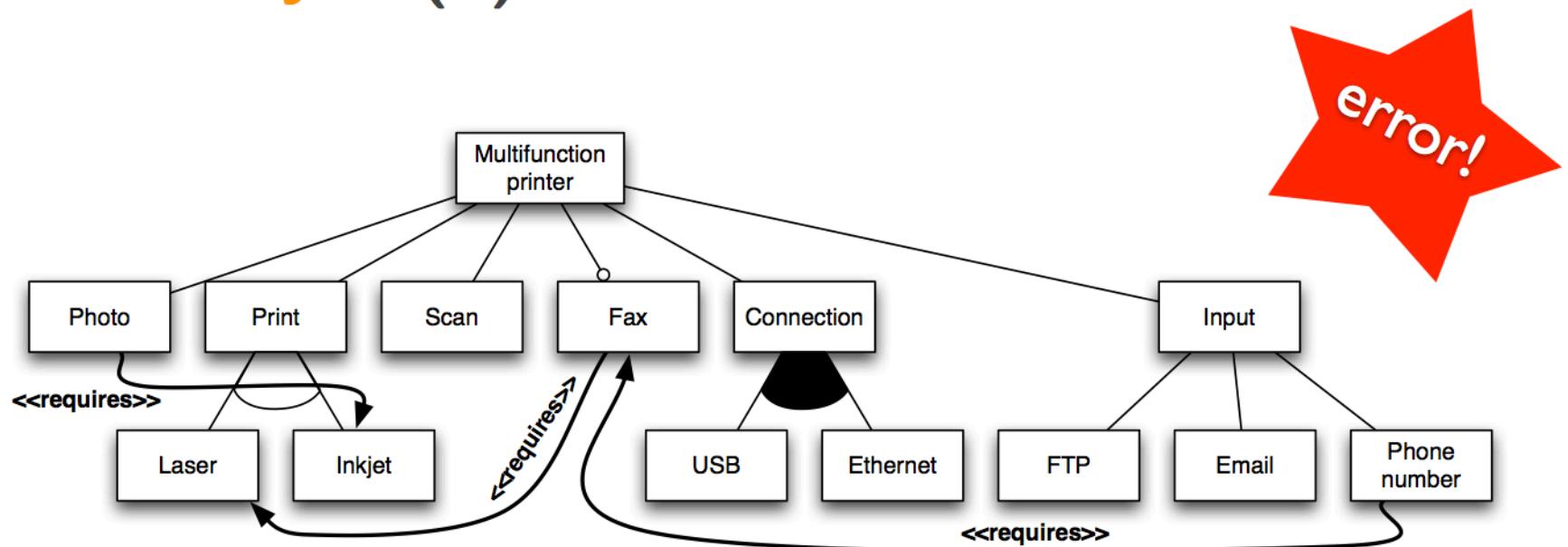
$M(d)$

{ {MP, Print, Laser, Scan, Fax, Connection, USB, Ethernet},
 {MP, Print, Laser, Scan, Fax, Connection, Ethernet},
 {MP, Print, Laser, Scan, Fax, Connection, USB},
 {MP, Print, Laser, Scan, Connection, USB, Ethernet},
 {MP, Print, Laser, Scan, Connection, Ethernet},
 {MP, Print, Laser, Scan, Connection, USB},
 {MP, Print, Inkjet, Scan, Fax, Connection, USB, Ethernet},
 {MP, Print, Inkjet, Scan, Fax, Connection, Ethernet},
 {MP, Print, Inkjet, Scan, Fax, Connection, USB},
 {MP, Print, Inkjet, Scan, Connection, USB, Ethernet},
 {MP, Print, Inkjet, Scan, Connection, Ethernet},
 {MP, Print, Inkjet, Scan, Connection, USB} }

$$\#M(d) = O(2^{\#F})$$

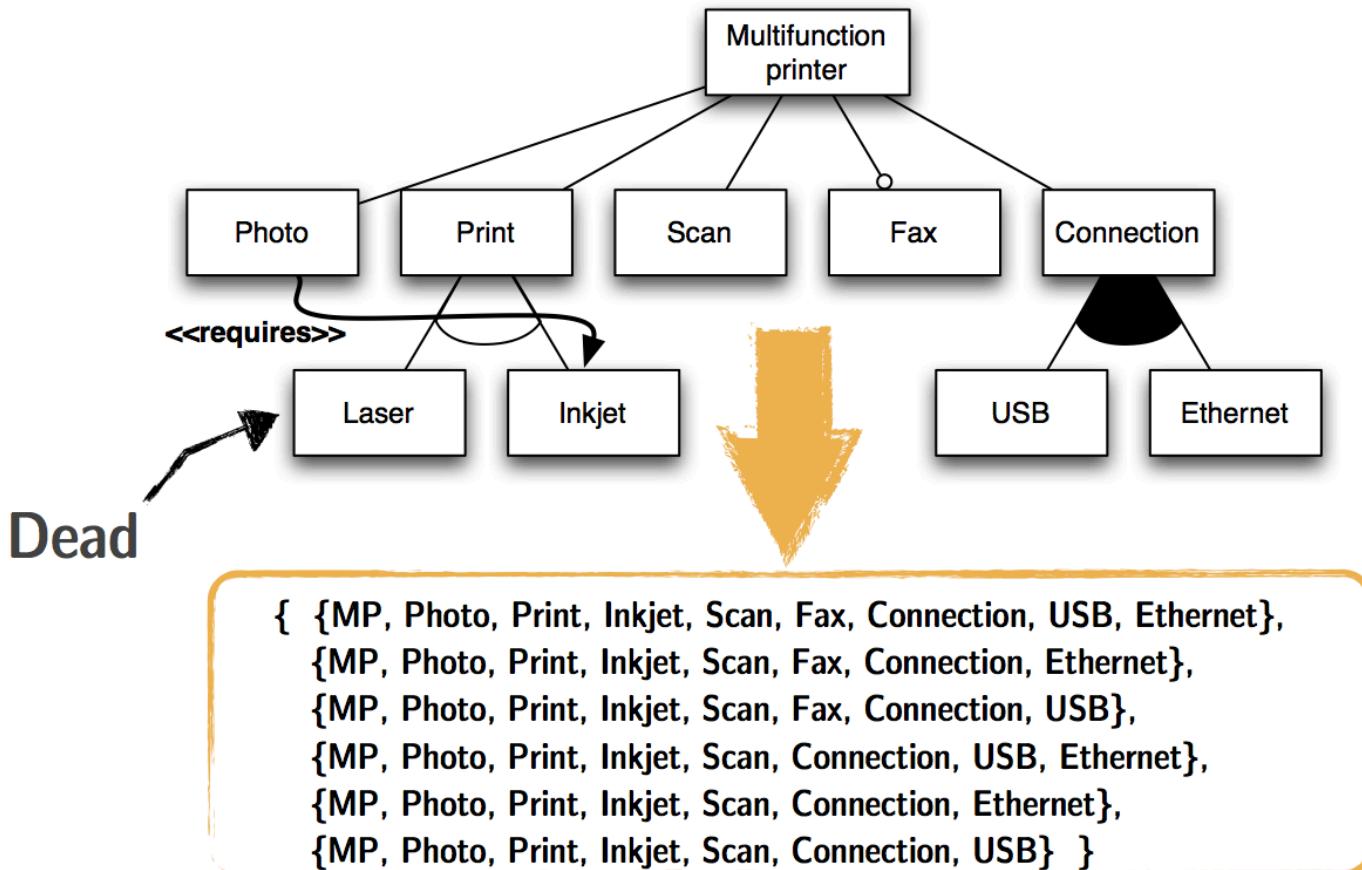
SPL Automations

Satisfiability $M(d) \neq \emptyset$?



SPL Automations

Dead features $F \setminus \cup M(d)$?



SPL Automations

Satisfiability $M(d) \neq \emptyset$?

Dead features $F \setminus \cup M(d)$?

Product checking $\{f_1, \dots, f_n\} \in M(d)$?

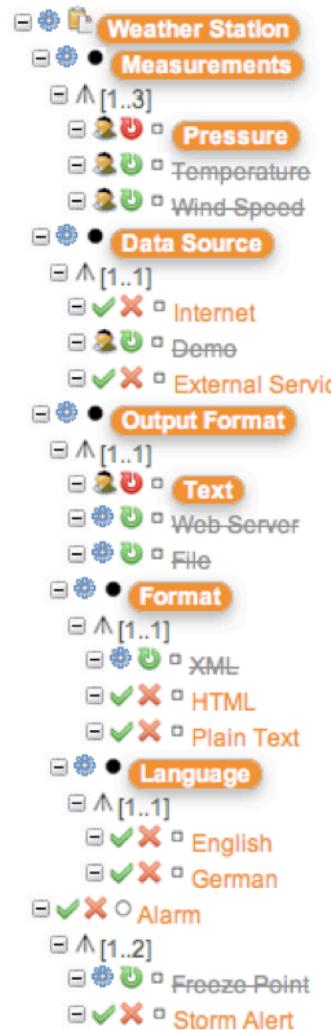
Product counting $\#M(d)$?

Equivalence $M(d_1) = M(d_2)$?

...

Interactive Configuration

Weather Station (23 features)



Configuration Steps [reset]					
65 %					
Step	Decision	#Decisions (cumulative)	#Propagations (at step)	#SAT checks (at step)	SAT time (at step)
1	✓ Weather Station	6 (26.1%)	5	9	1 ms
2	✓ Text	10 (43.5%)	3	6	0 ms
3	✗ Demo	11 (47.8%)	0	2	0 ms
4	✗ Wind Speed	12 (52.2%)	0	2	0 ms
5	✓ Pressure	13 (56.5%)	0	3	1 ms
6	✗ Temperature	15 (65.2%)	1	3	0 ms

↳ Auto-completion: [Less Features](#) | [More Features](#)

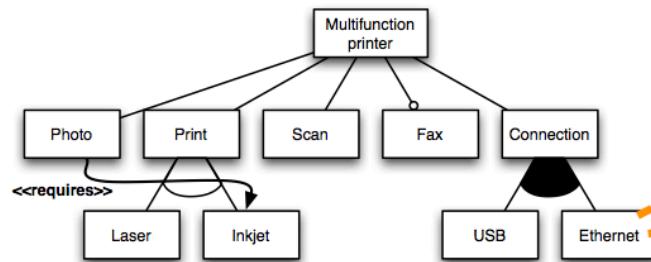
S.P.L.O.T.

Software Product Lines Online Tools

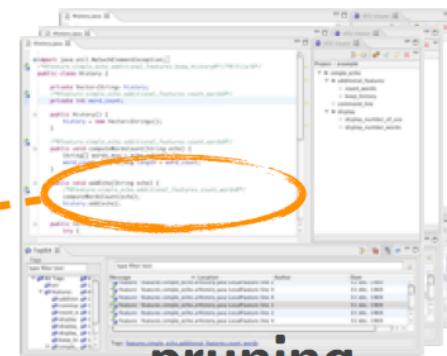
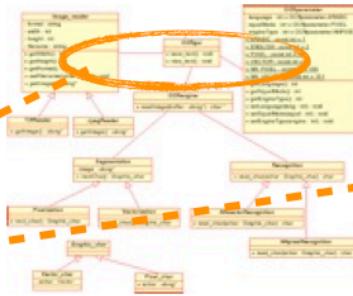
www.splot-research.org

Product Derivation

feature model



variable model and code assets



pruning,

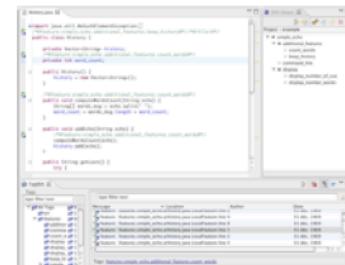
composition,
weaving,
transformation

configuration

{ MP, Photo, Print, Inkjet, Scan,
Fax, Connection, USB, Ethernet }

product spec

product



Summary

- **Software product line engineering**
 - Mass customization
 - Family of software intensive systems
 - Systematic reuse
 - Domain engineering
 - Variability management
 - **Variability** everywhere
 - Applied and applicable to many industries and domains
 - **Modeling and implementing** variability: an overview
- 
- 