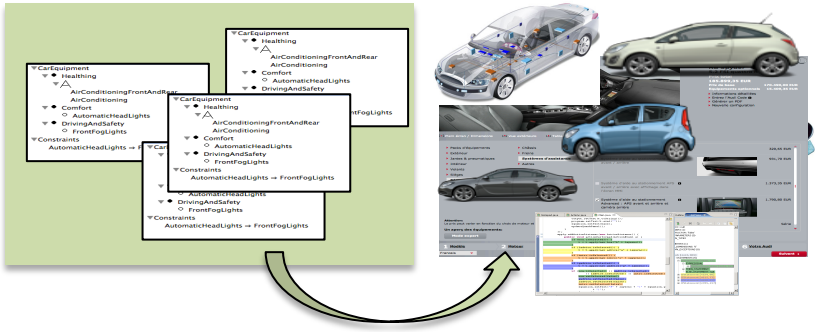


Modeling and managing variability of software intensive systems

Overview and Principles

Mathieu Acher

Maître de Conférences

mathieu.acher@irisa.fr

UNIVERSITÉ DE RENNES 1

istic Informatique Électronique

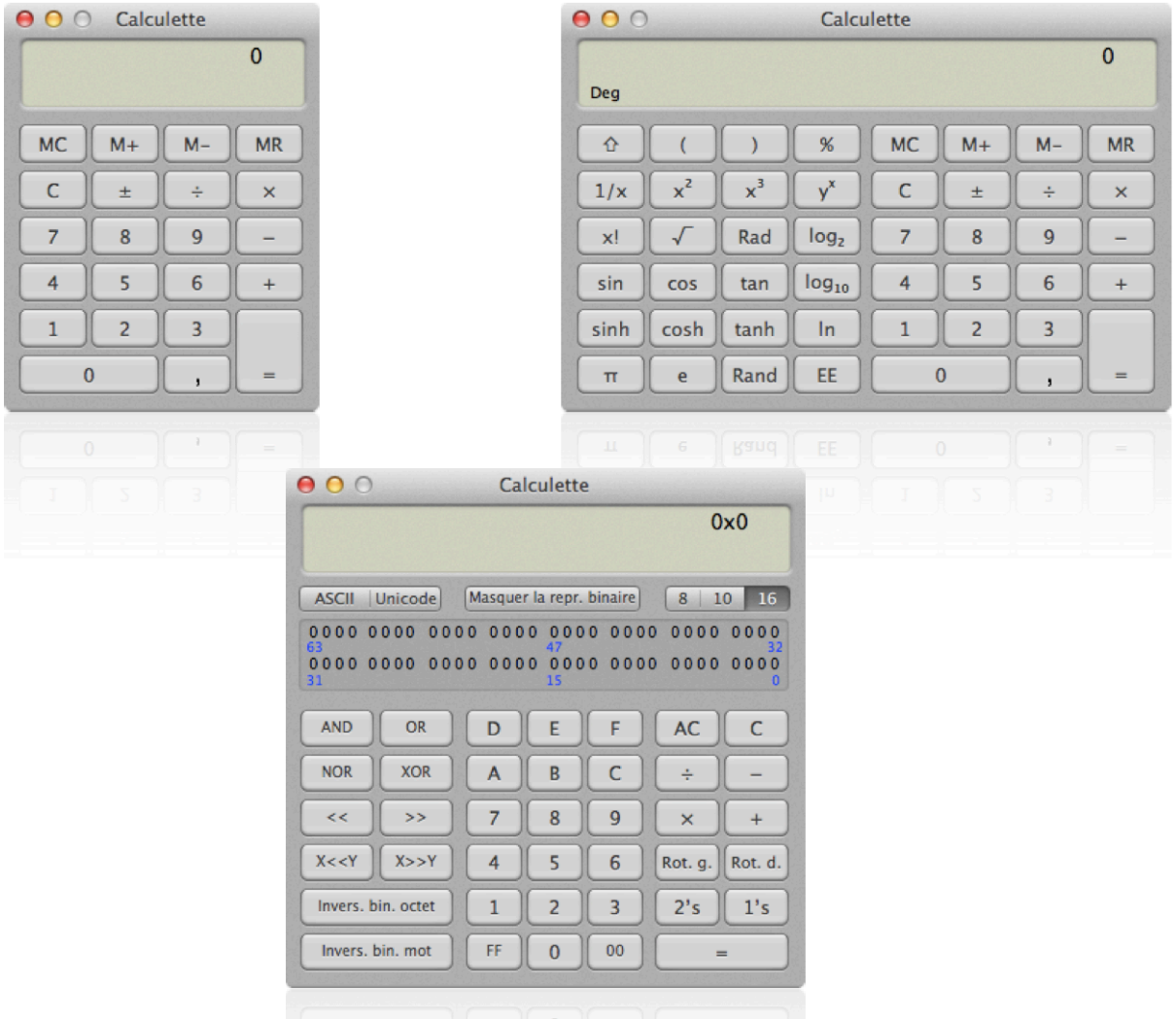
Material

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**

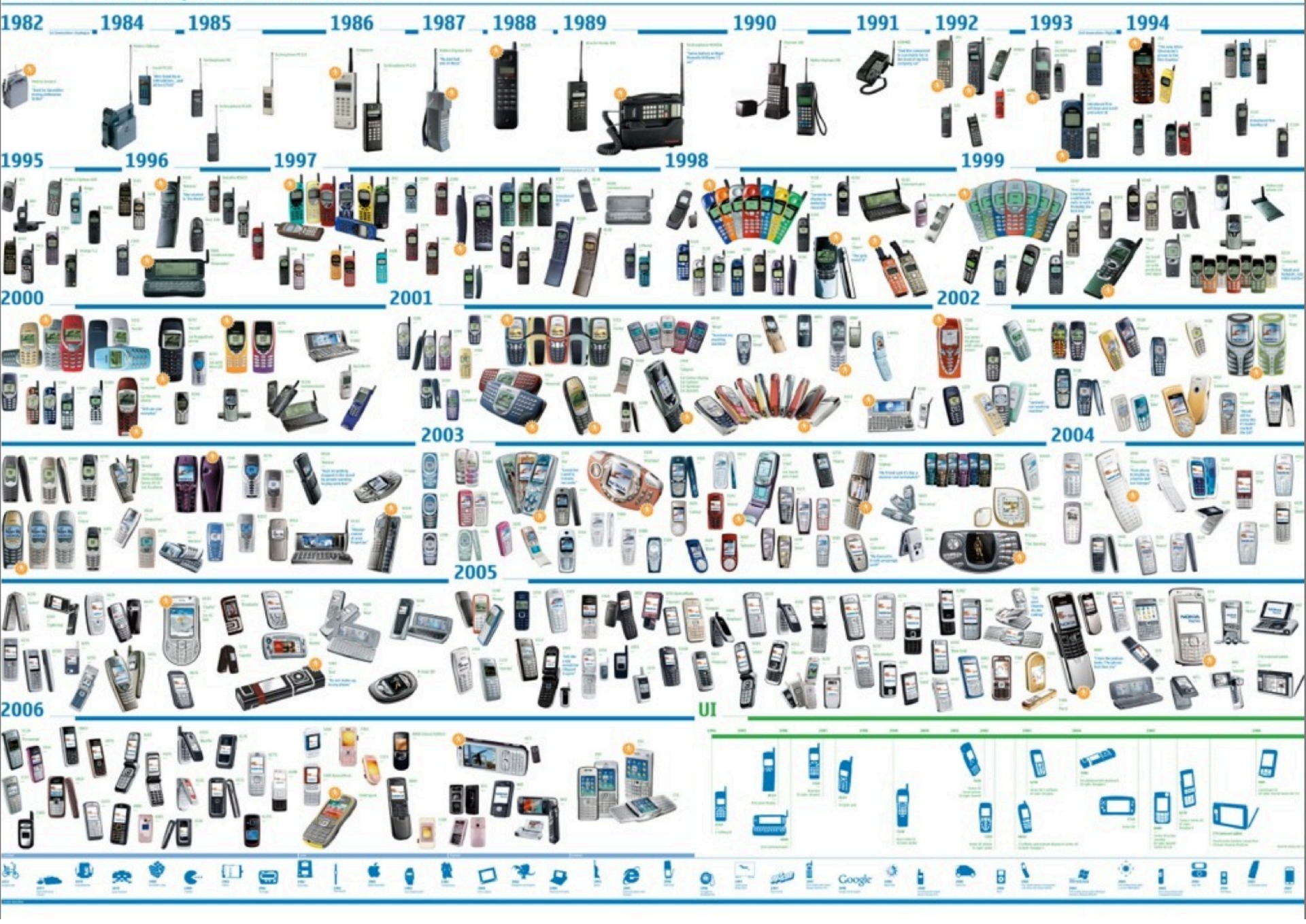
David L. Parnas — "On the design and development of program families" in Transactions on Software Engineering, SE-2(1):1–9, 1976


Calculette
0
MC M+ M− MR
C ± ÷ ×
7 8 9 −
4 5 6 +
1 2 3 =
0 ,
Calculette
0
Deg
⇧ ( ) % MC M+ M− MR
1/x x² x³ yˣ C ± ÷ ×
x! √ Rad log₂ 7 8 9 −
sin cos tan log₁₀ 4 5 6 +
sinh cosh tanh ln 1 2 3 =
π e Rand EE 0 ,
Calculette
0x0
ASCII Unicode
Masquer la repr. binaire
8 10 16
0000 0000 0000 0000 0000 0000 0000 0000
63 47 32
0000 0000 0000 0000 0000 0000 0000 0000
31 15 0
AND OR D E F AC C
NOR XOR A B C ÷ −
<< >> 7 8 9 × +
X<<Y X>>Y 4 5 6 Rot. g. Rot. d.
Invers. bin. octet 1 2 3 2's 1's
Invers. bin. mot FF 0 00 =

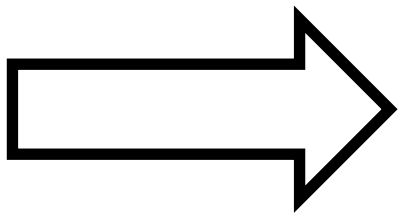
	 **Starter**	 **Home Premium Upgrade** $119.99* Buy	 **Professional Upgrade** $199.99* Buy	 **Ultimate Upgrade** $219.99* Buy
Communication				
Bluetooth support	✓	✓	✓	✓
Join a homegroup	✓	✓	✓	✓
Internet Explorer 8	✓	✓	✓	✓
View Available Networks	✓	✓	✓	✓
Windows Connect Now (WCN)	✓	✓	✓	✓
Create a homegroup		✓	✓	✓
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		✓	✓	✓

NOKIA
Connecting People

Know our past. Create the future...



Software-intensive systems

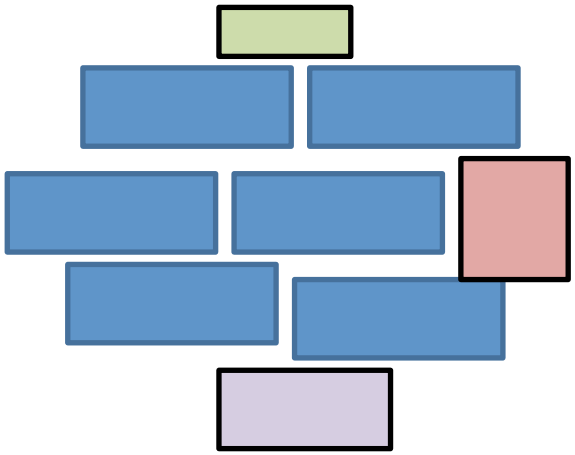


come in many **variants**

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]

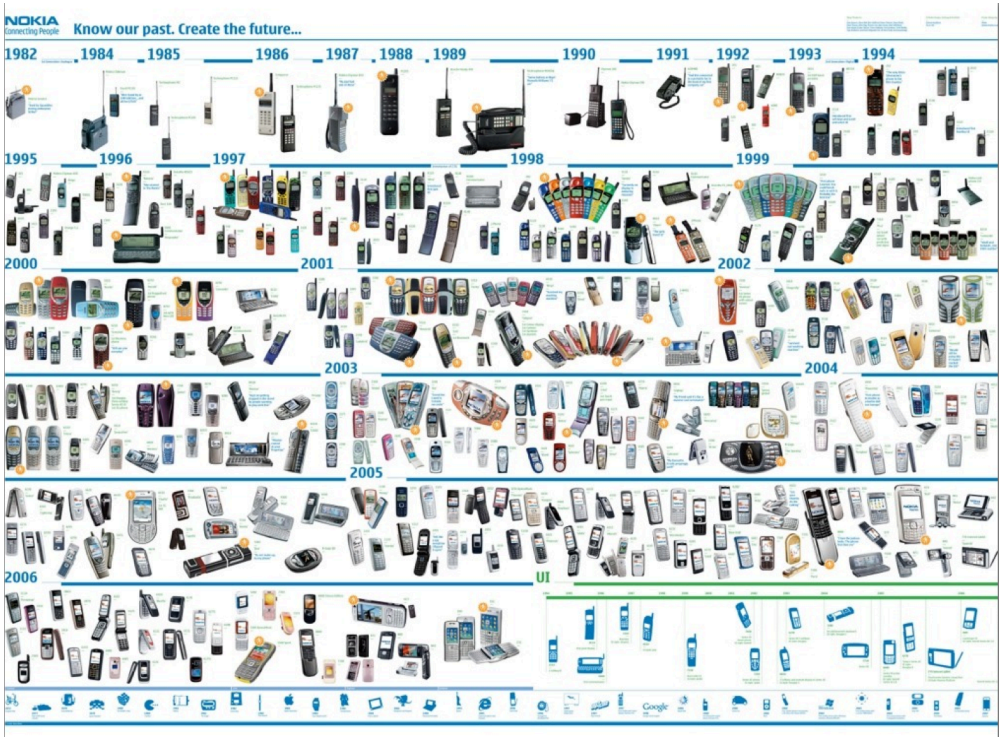
Variability

"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**

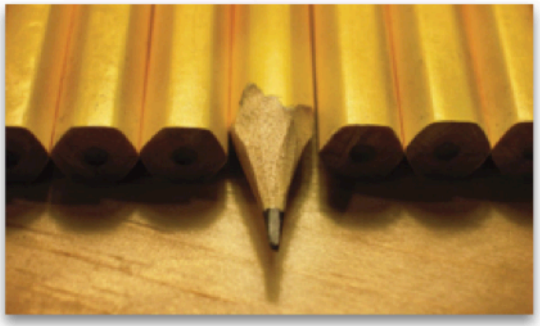


Benefits

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

ERICSSON

NOKIA
Connecting People

CelsiusTech

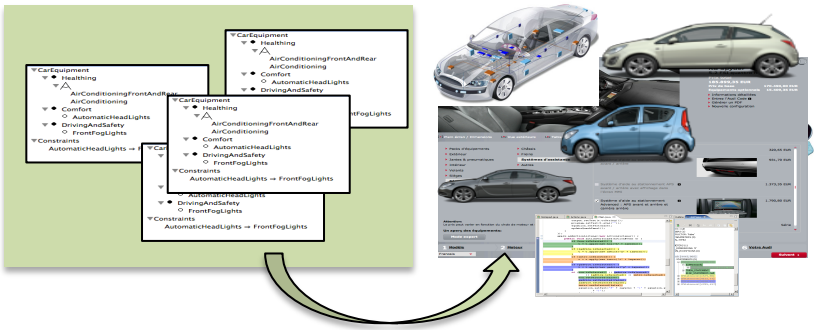
Lucent Technologies
Bell Labs Innovations

Printer Firmware

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

Objectives

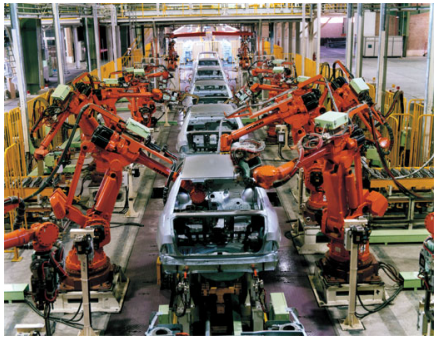
- Variability and Software Product Line Engineering
  - Overview
  - Challenges



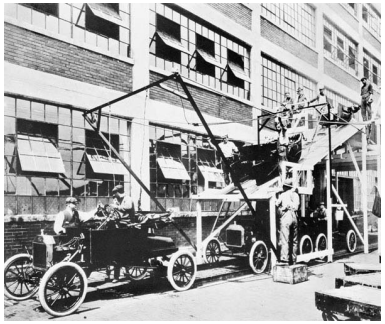
- The (possible) role of Model-Driven Engineering

- Overview of techniques

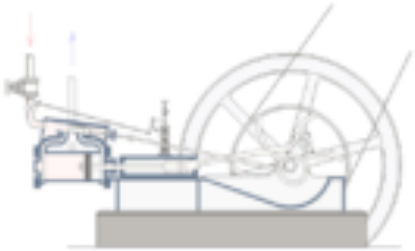
A Bit of History: Industrial Revolution



1980s



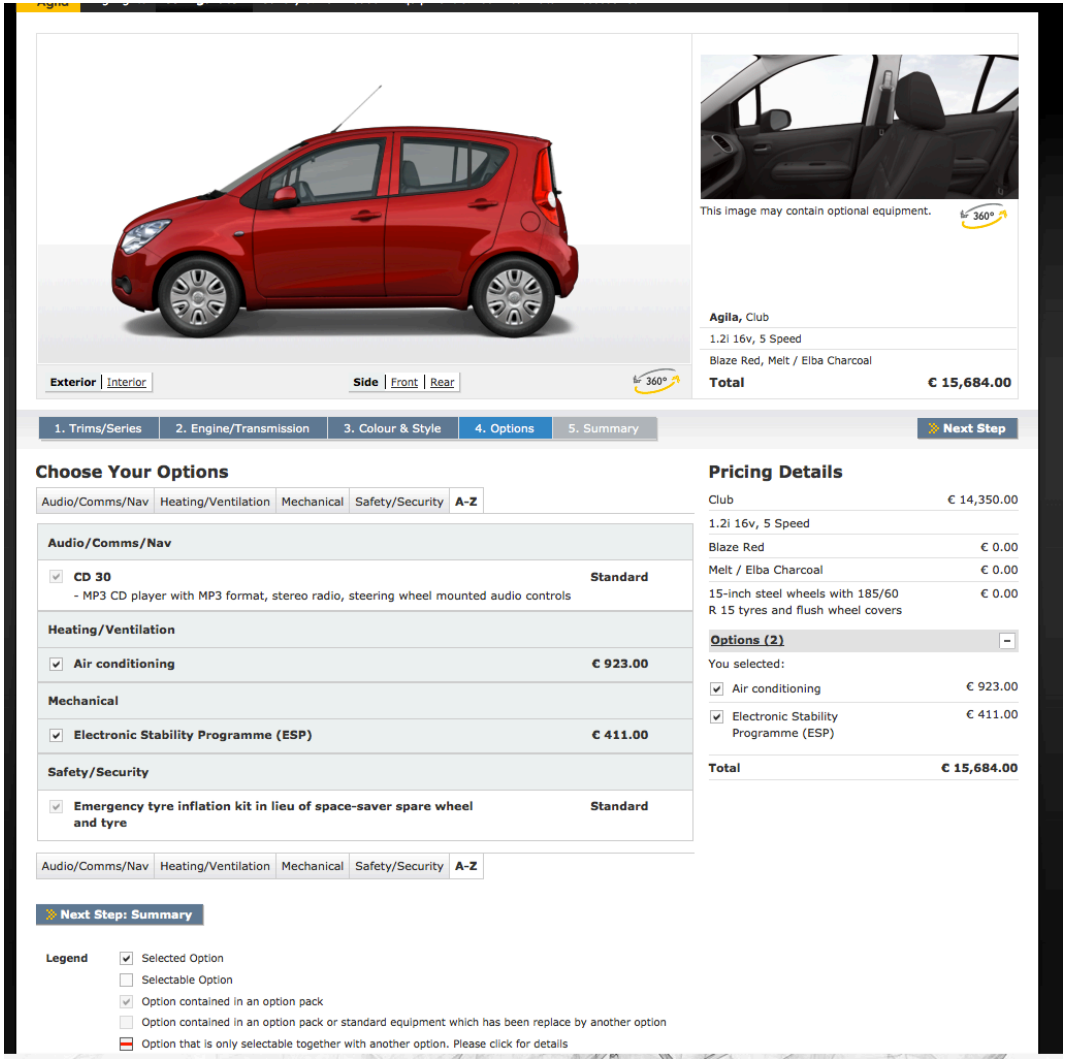
1901
Henry Ford



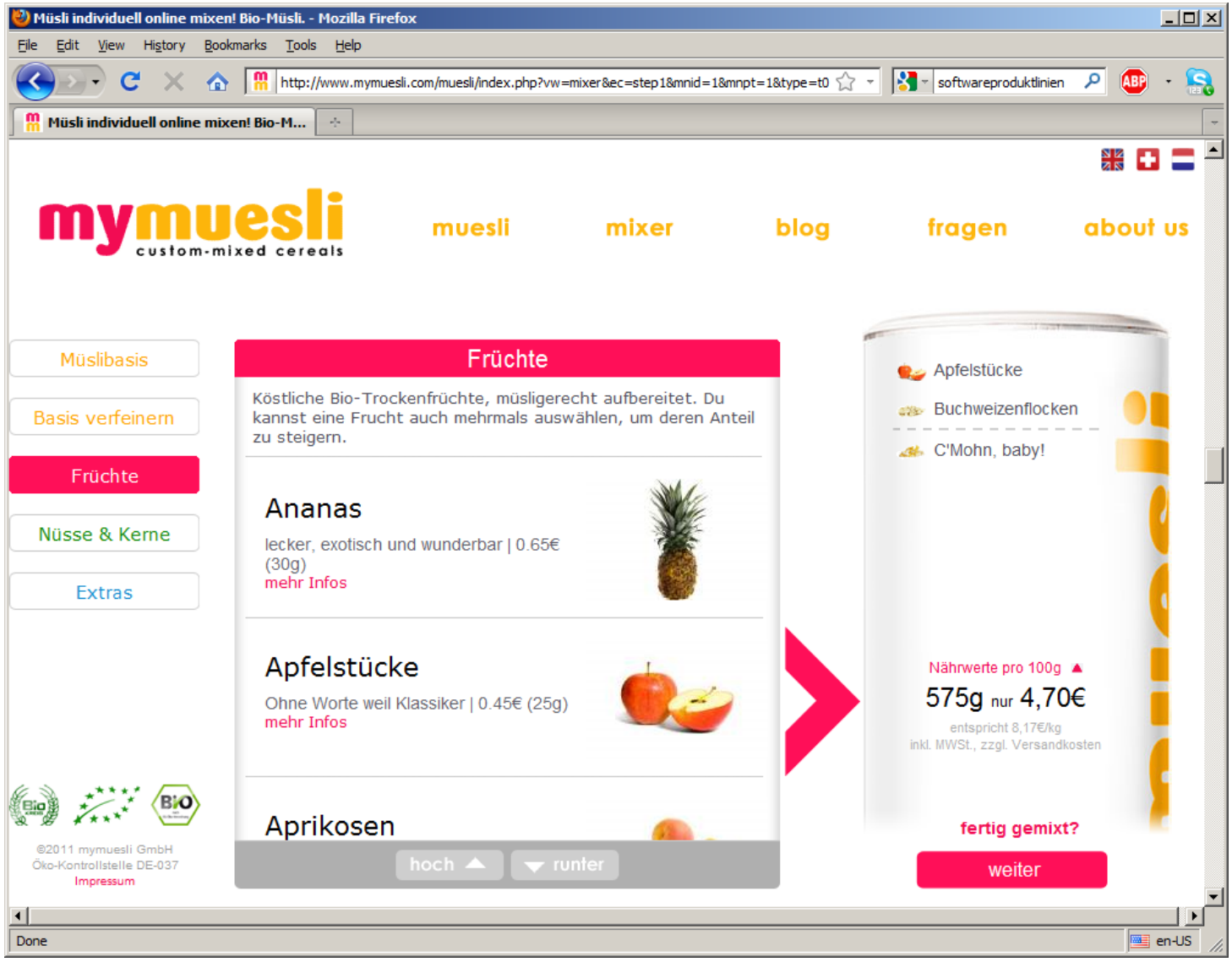
1698
Thomas Savery

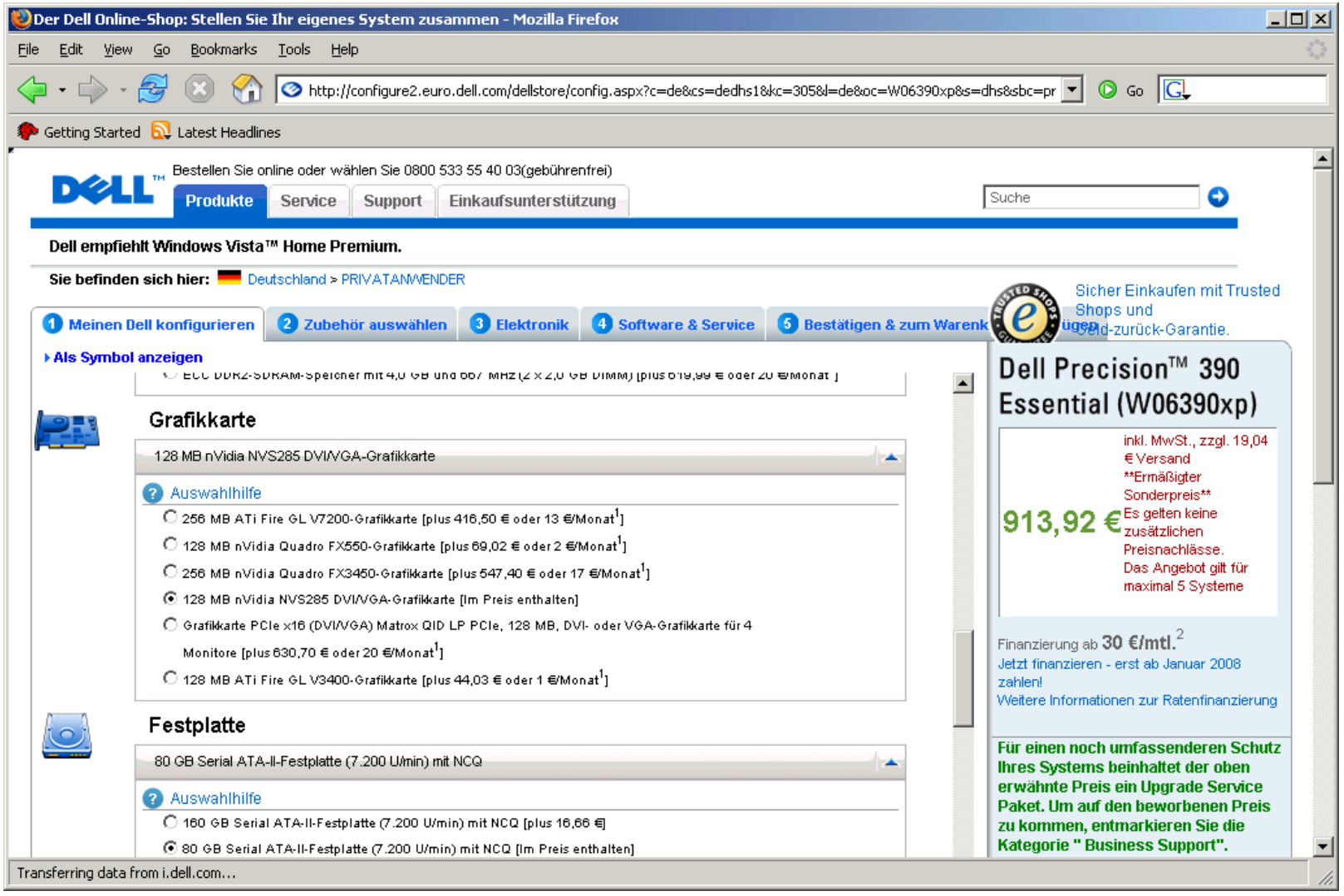
Nowaday: Product Lines Everywhere

Product Lines of Cars

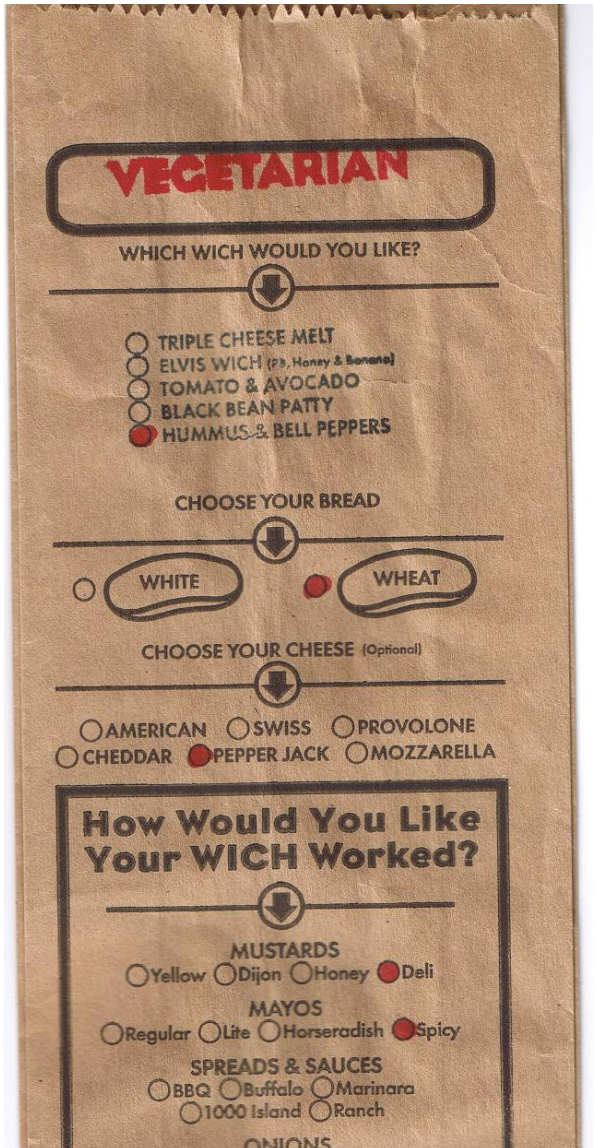



Willkommen bei selve - the shoe individualizer
http://www.selve.net/index_js.html
KOLLEKTION
FUSSTYP
MYSELVE
INFO
HOME
MODELLE
LOOKBOOK
SELVE ID
PASSWORT
>>ANMELDEN
selve
selve Kollektion -> Style: casuals -> Modell: Opal
modell-details
>>hier clicken
>>SELVE SCHUHREGAL
inhalt: 0
>>SHOPPING BAG
inhalt : 0
A. Erstes Oberleder
Velours leder Sand
Velours leder Bordeaux
Velours leder Cognac
Velours leder Sand
B.
C. Futterleder
Beige
D. Absatz
Hufeisen Braun
E. Sohle
Gummisohle
>>ÄNDERN
>>ZURÜCKLEGEN


Müsli individuell online mixen! Bio-Müsli. - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.mymuesli.com/muesli/index.php?vw=mixer&ec=step1&mnid=1&mnpt=1&type=t0
softwareproduktlinien
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
©2011 mymuesli GmbH
Öko-Kontrollstelle DE-037
Impressum
Done
en-US


Der Dell Online-Shop: Stellen Sie Ihr eigenes System zusammen - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://configure2.euro.dell.com/dellstore/config.aspx?c=de&cs=dedhs1&kc=305&l=de&oc=W06390xp&s=dhs&sbc=pr
Go
Getting Started
Latest Headlines
Bestellen Sie online oder wählen Sie 0800 533 55 40 03(gebührenfrei)
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 Warenk
Sicher Einkaufen mit Trusted Shops und Geld-zurück-Garantie.
Als Symbol anzeigen
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]
Dell Precision™ 390 Essential (W06390xp)
913,92 €
inkl. MwSt., zzgl. 19,04 € Versand
Ermäßigter Sonderpreis
Es gelten keine zusätzlichen Preisnachlässe.
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...

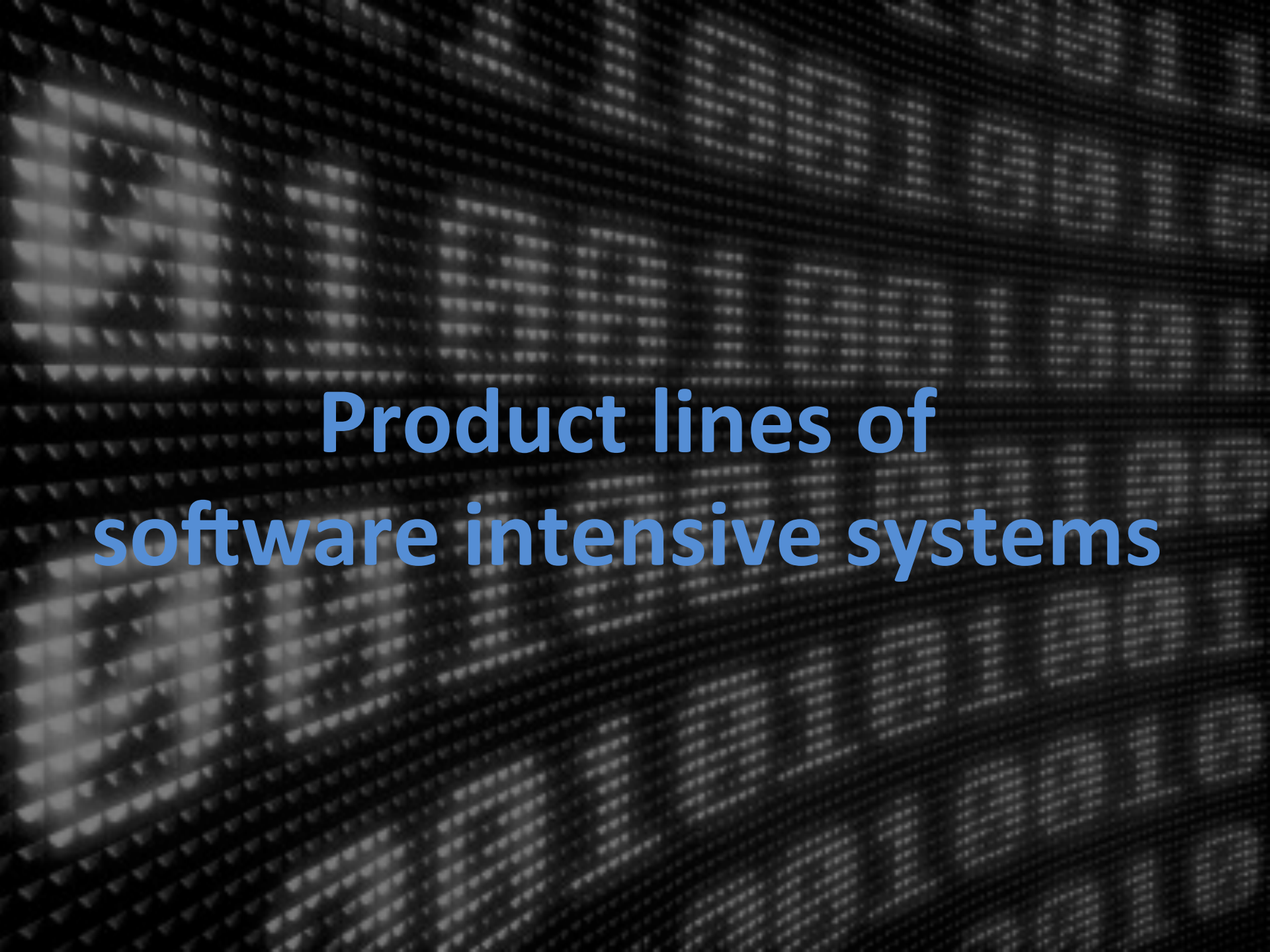
Food? Product lines!




VEGETARIAN
How Would You Like Your WICH Worked?
VEGETARIAN
How Would You Like Your WICH Worked?

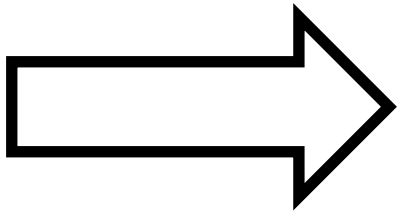
Mass production

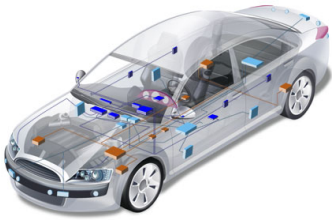

What about
software?


Product lines of
software intensive systems

Software intensive systems

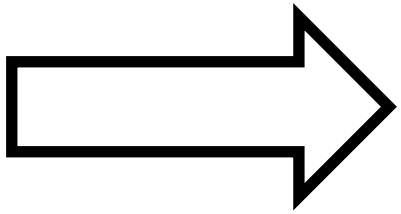
are declined in many **variants**





Software intensive systems

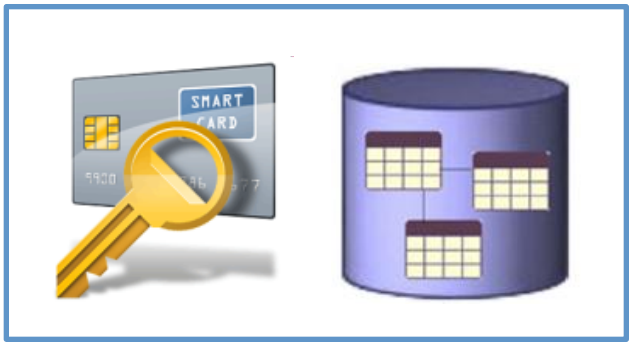
are declined in many **variants**



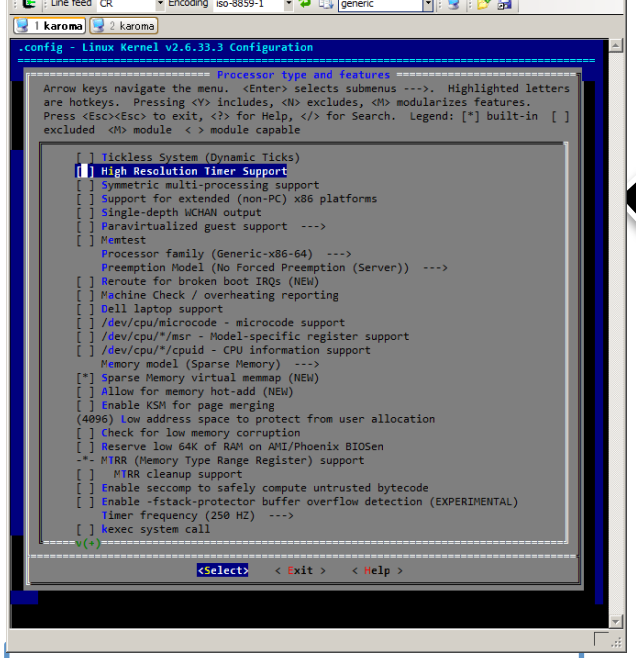
Software Product Lines

Car

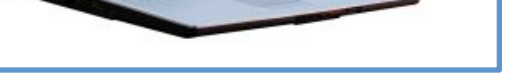
Database Engine



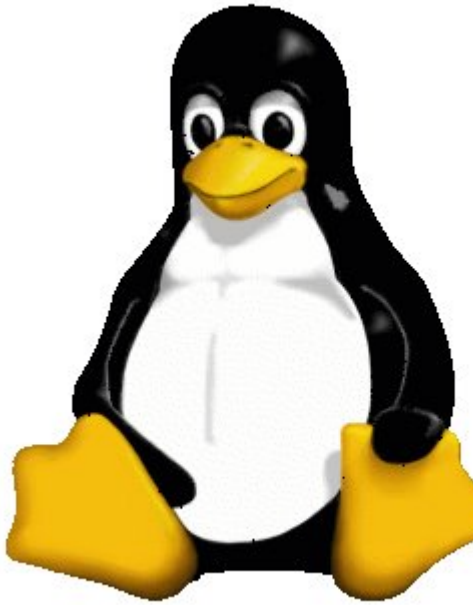
Printer Firmware



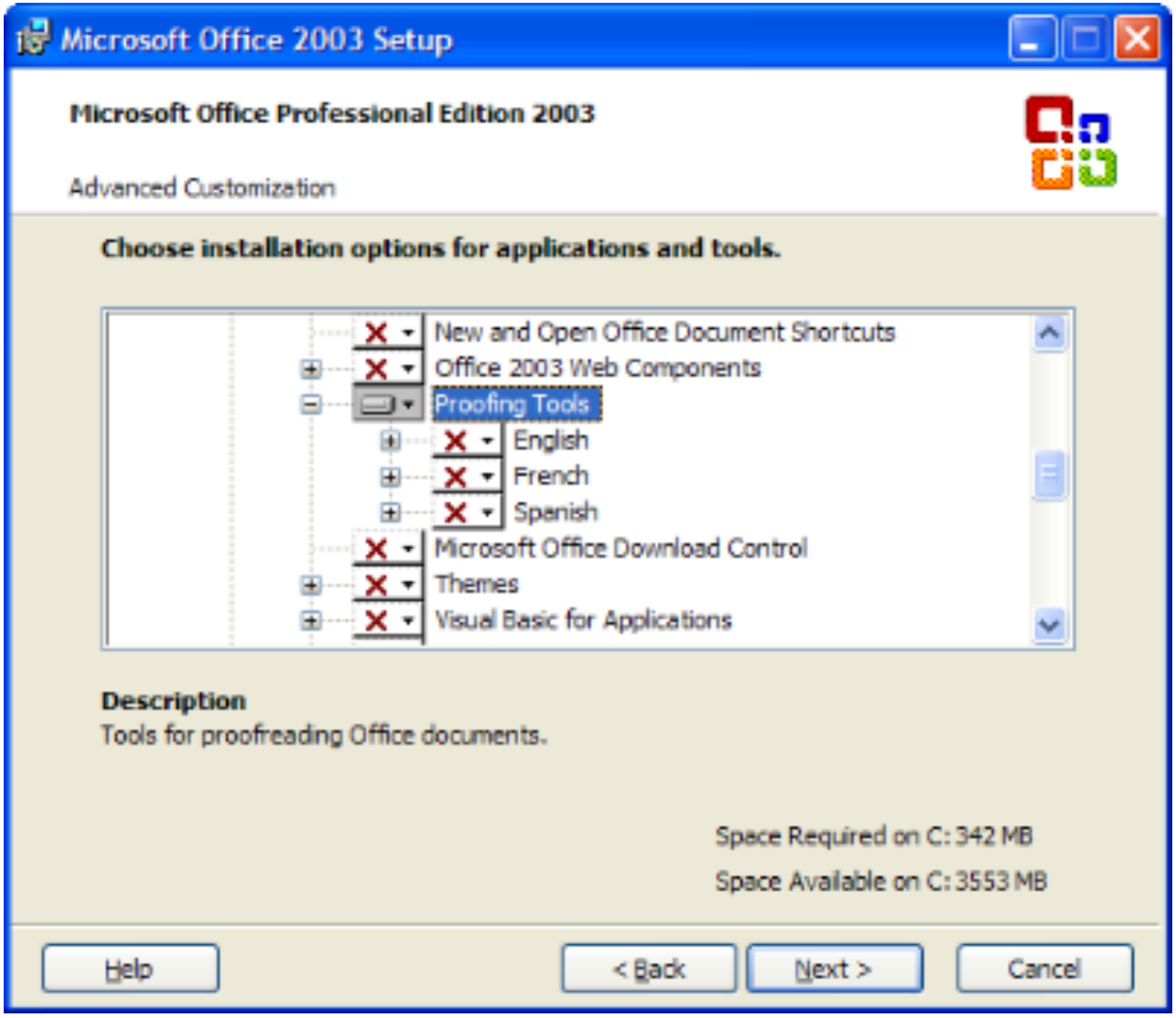
Linux Kernel



Linux-Kernel



Features in Microsoft Office



The development of a

family of software systems

is much more challenging than the development of

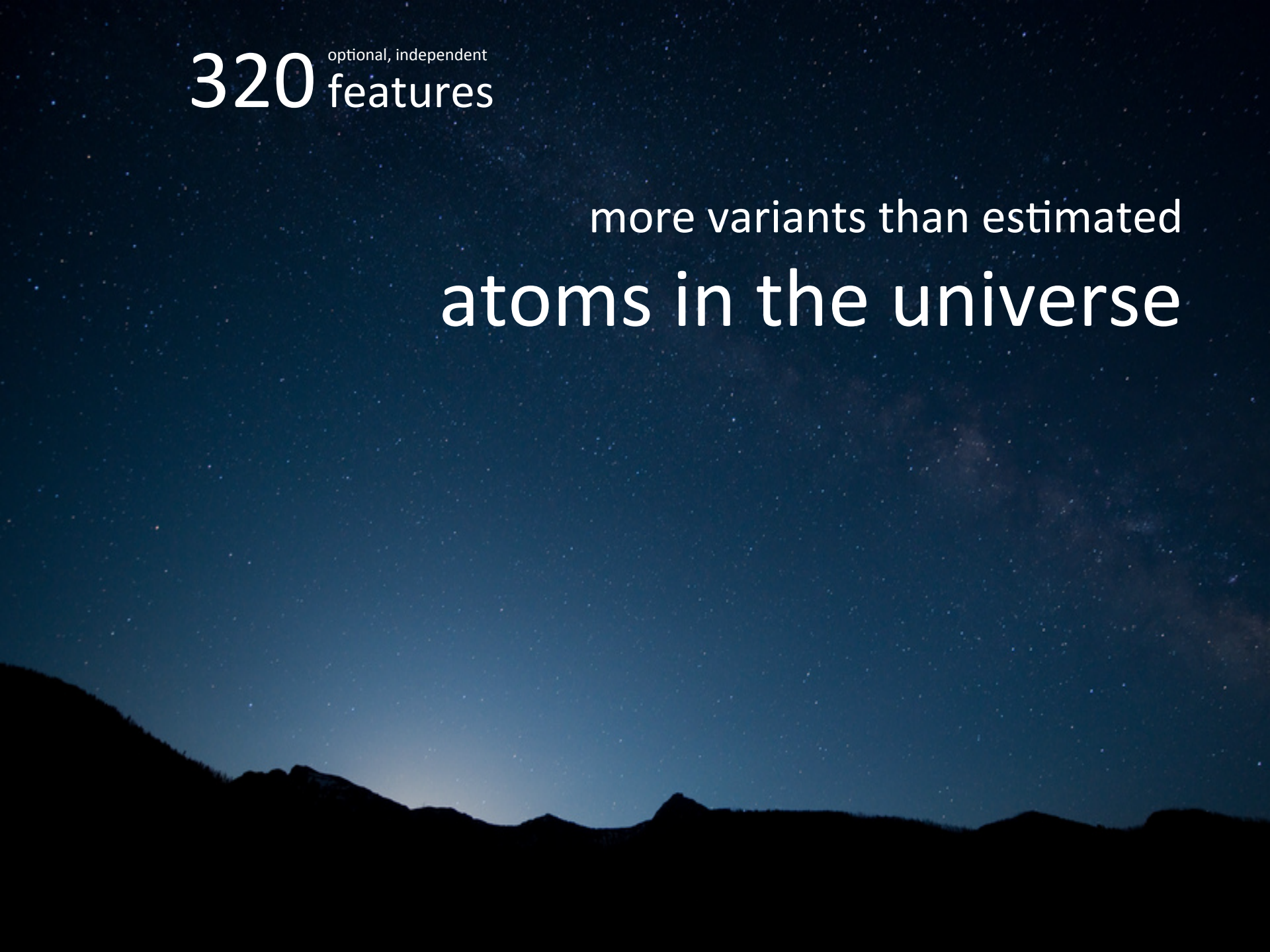
a single software system


Variability = Complexity

33 optional, independent features

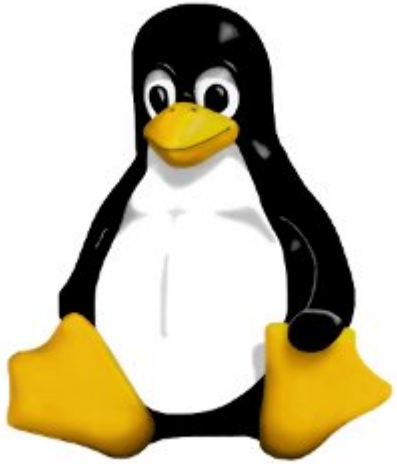
a unique variant for every

person on this planet

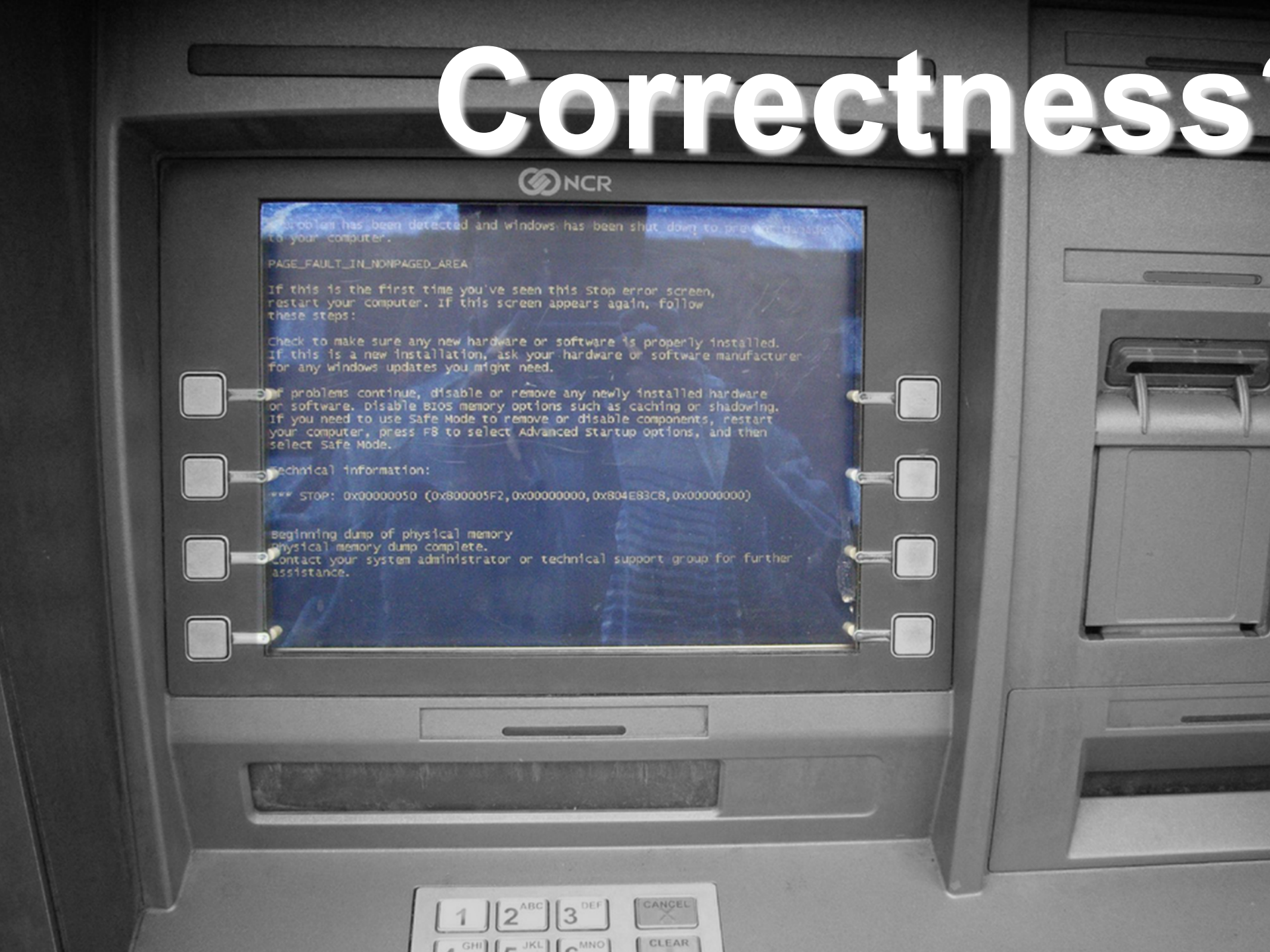

320 optional, independent features
more variants than estimated
atoms in the universe

2000 features

10000 features




Automation?
Avoid solving the same problem!
2, 3…n times


Correctness
NCR
PAGE_FAULT_IN_NONPAGED_AREA
If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:
Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any windows updates you might need.
If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup options, and then
select Safe Mode.
Technical information:
*** STOP: 0x00000050 (0x800005F2,0x00000000,0x804E83C8,0x00000000)
Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.
1
2 ABC
3 DEF
CANCEL
CLEAR

Maintenance?
Comprehension?

Checking **Products**

2000 Features
100 Printers
30 New Printers per Year

Printer Firmware

Checking **Products**

8000 Features

? Products

Linux Kernel

Checking **Product Line**

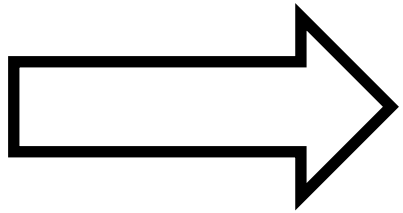
Implementation with 10000 Features

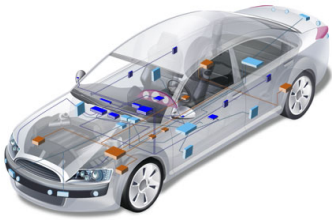
#ifdef, Frameworks, FOP, AOP, …

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

Customization

Automation

Commonality

Variability

Assembly Line and Mass Customization

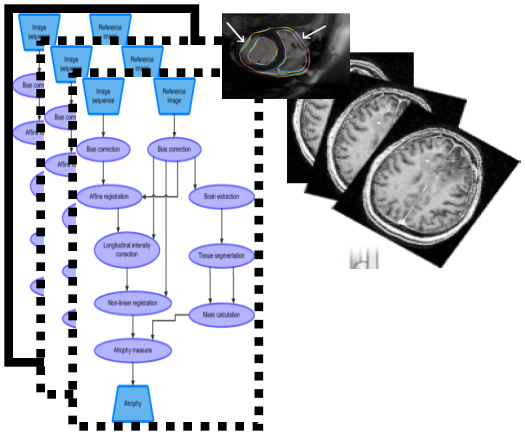
Reuse and Mass Customization

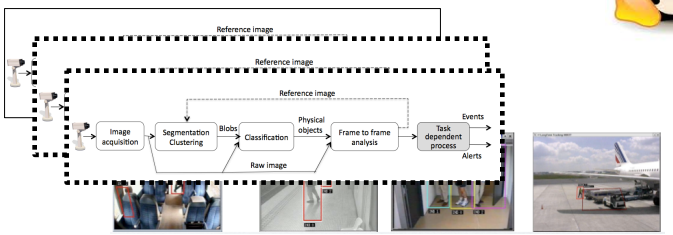

Starting from scratch?


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

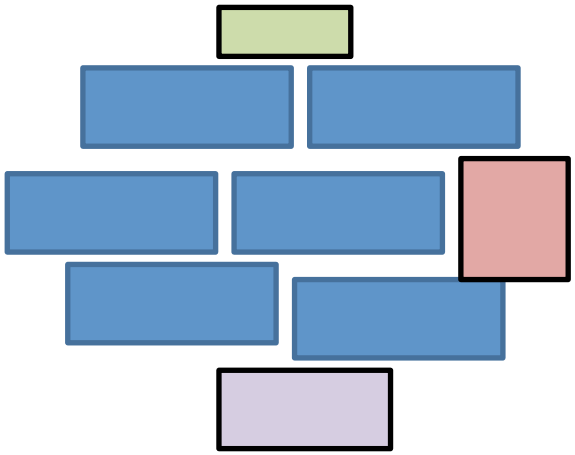




Software Product Line Engineering

Factoring out **commonalities**

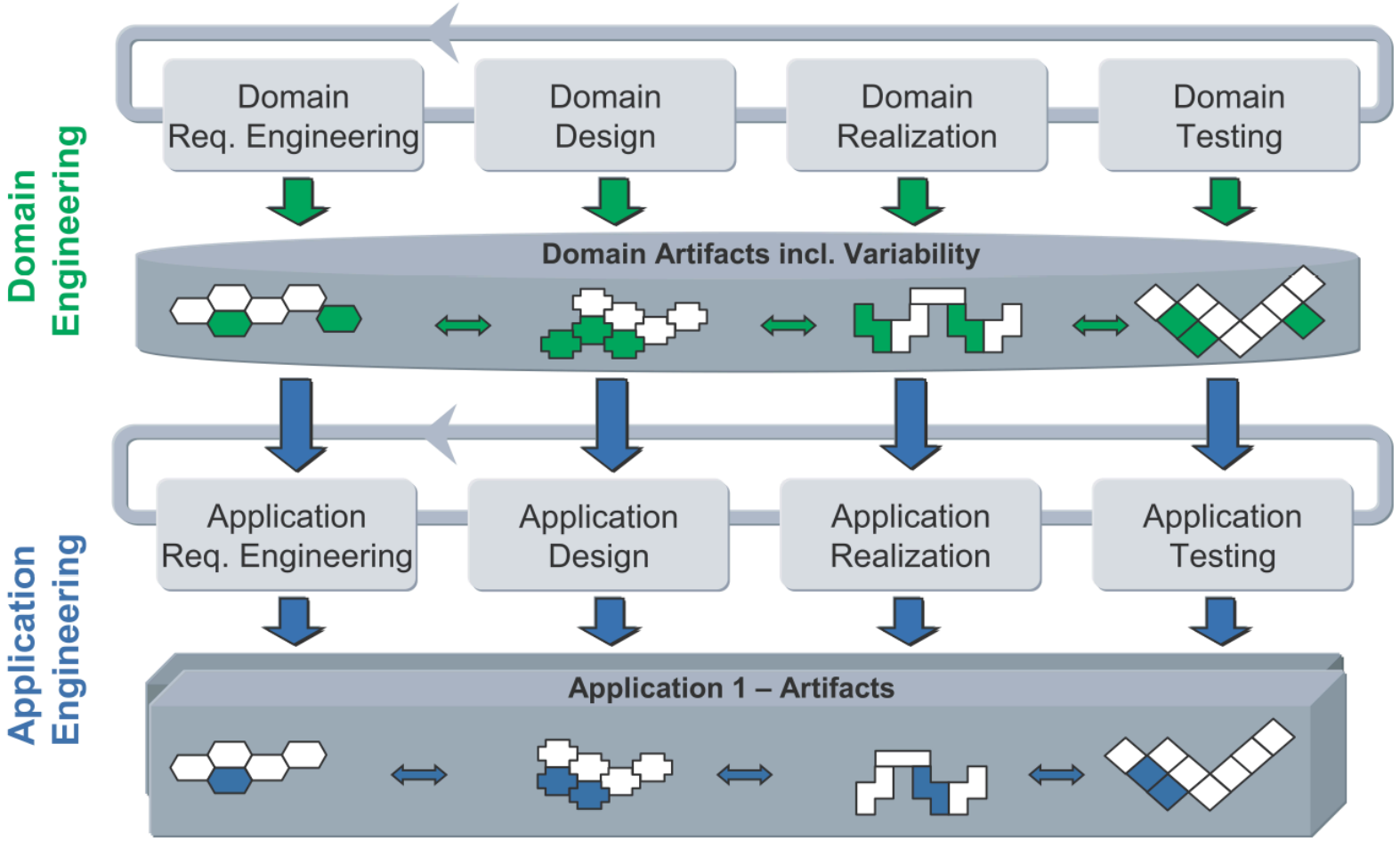
for **Reuse** [Krueger et al., 1992] [Jacobson et al., 1997]

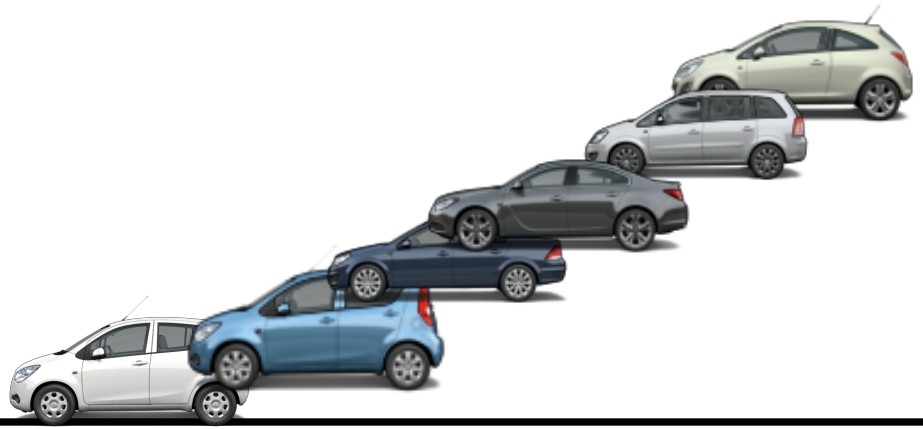


Managing **variabilities**

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

Software Product-Line Engineering





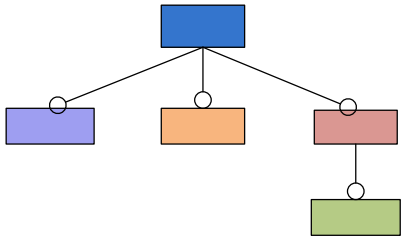
"Reuse-in-the-large works best in families of related systems, and thus is domain dependent." [Glass, 2001]

Domain engineering

Domain Analysis

(problem)

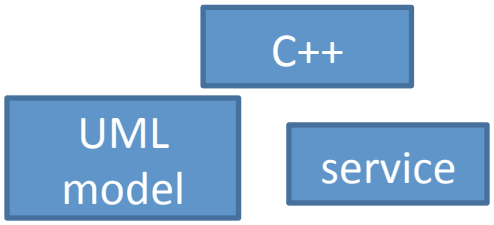
- elicitate requirements and scope the line
- variability modeling: determine commonalities and variabilities usually in terms of features



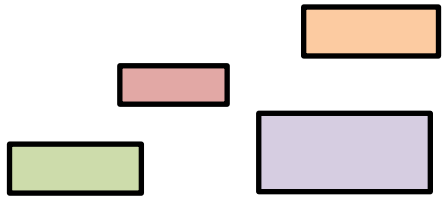
Variability Model (Feature Model)

Domain Implementation

(solution)



Common assets

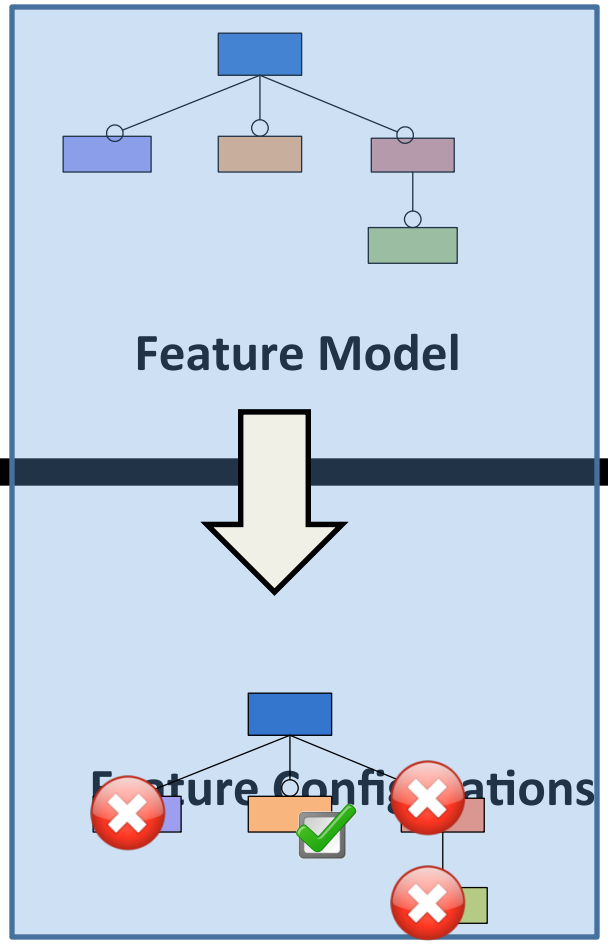


Variants

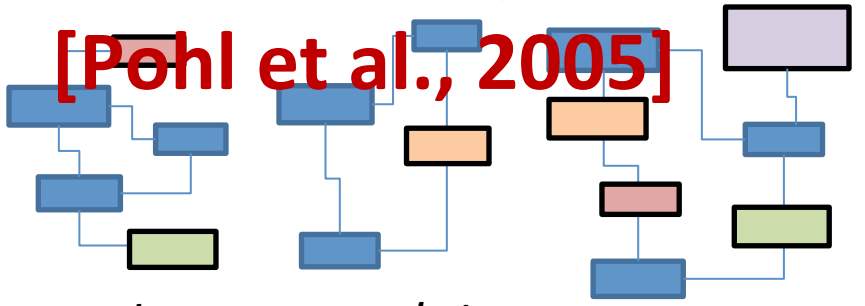
Reusable Assets

(e.g., models or source code)

Domain engineering (development for reuse)



Reusable Assets
(e.g., models or source code)



"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

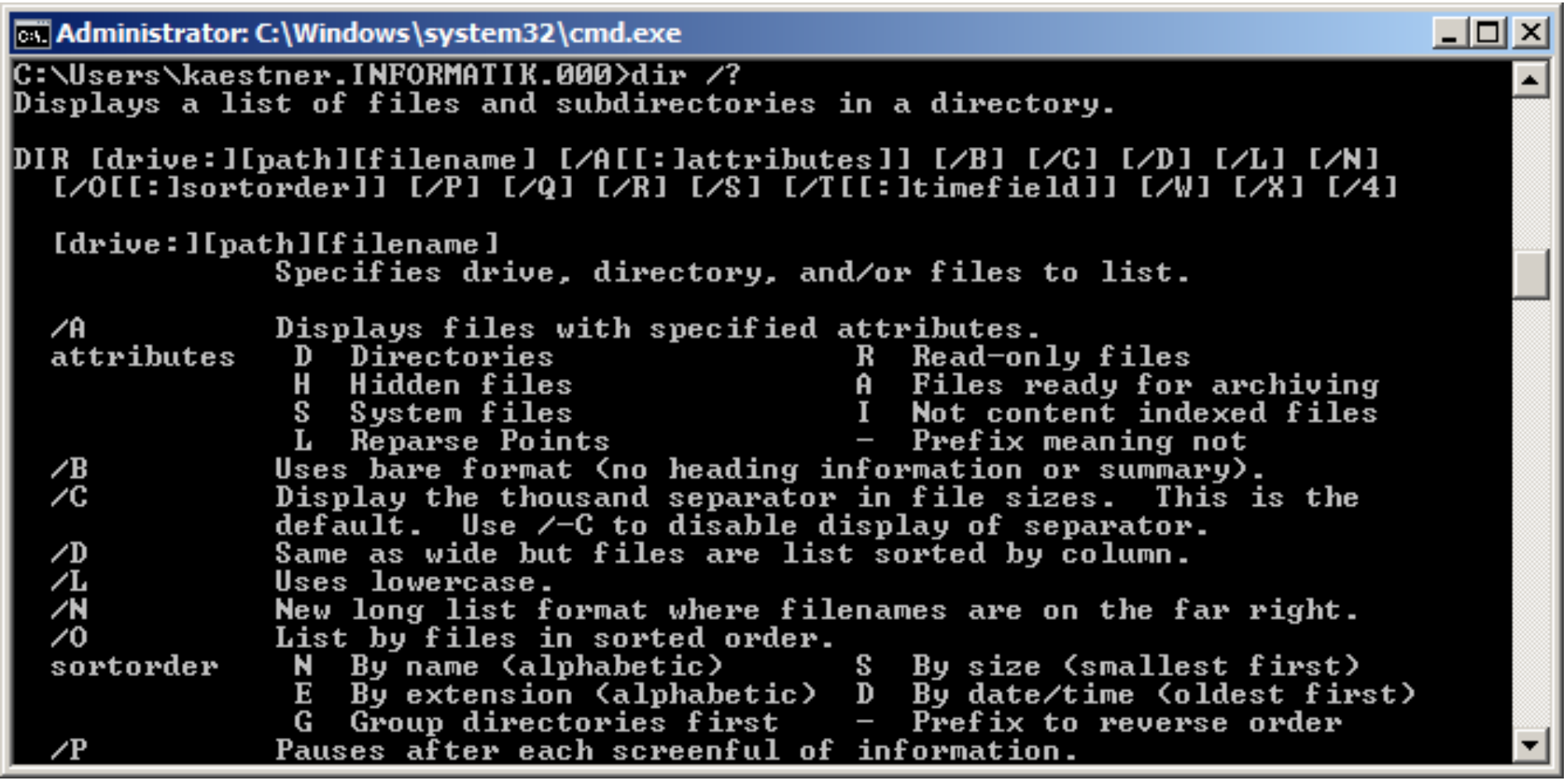
Factoring out commonality

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[[:]attributes]] [/B] [/C] [/D] [/L] [/N]
[/O[[:]sortorder]] [/P] [/Q] [/R] [/S] [/T[[:]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_c
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     }
```

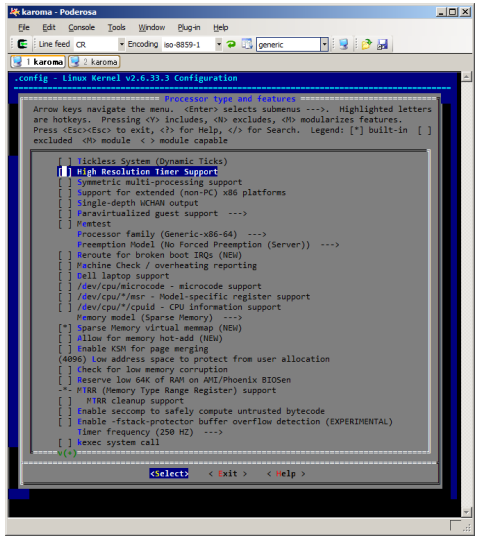
Global configuration

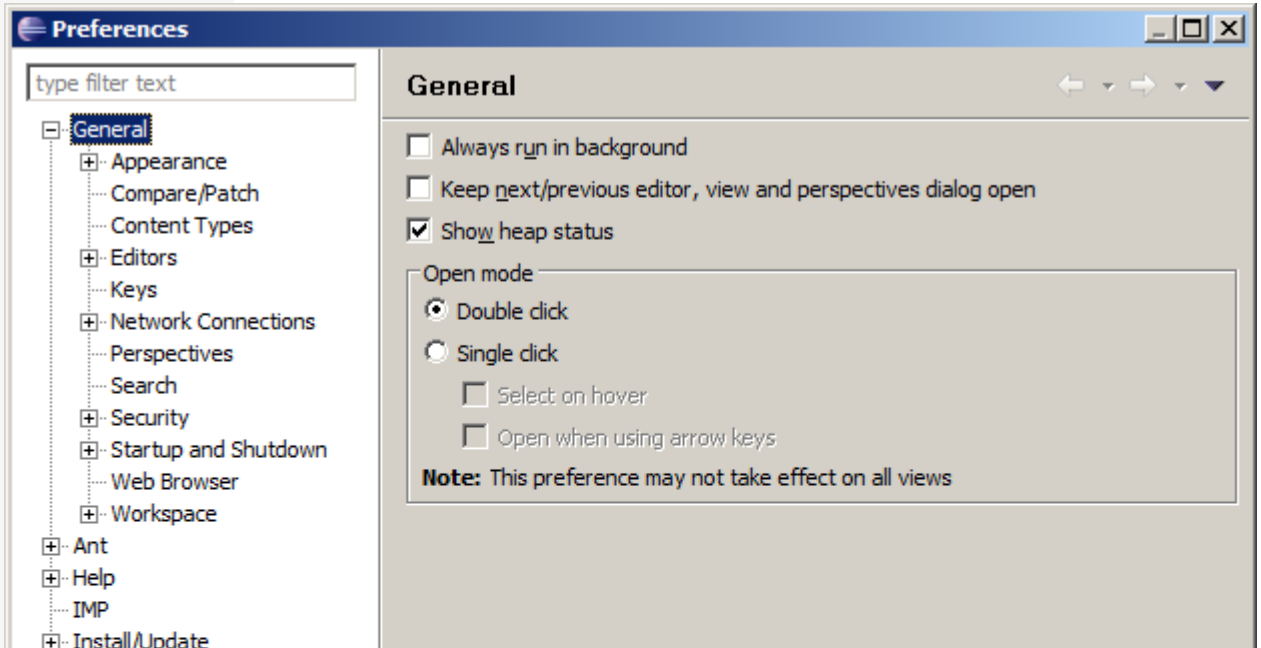
```
class Config {
  public static boolean isLogging = false;
  public static boolean isWindows = false;
  public static boolean isLinux = true;
}
class Main {
  public void foo() {
    if (isLogging)
      log(„running foo()“);
    if (isWindows)
      callWindowsMethod();
    else if (isLinux)
      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;
#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.intreqr, 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);
.......
```

cyberstormII.c

```
....
static void dma_dump_state(struct NCR_ESP *esp)
{
        ESPLOG(("esp%d: dma -- cond_reg<%02x>\n",
                esp->esp_id, ((struct cyberII_dma_registers *)
                                        (esp->dregs))->cond_reg));
        ESPLOG(("intreq:<%04x>, intena:<%04x>\n",
                custom.intreqr, custom.intenar));
}

static void dma_init_read(struct NCR_ESP *esp, __u32 addr, int
        length)
{
        struct cyberII_dma_registers *dregs =
                (struct cyberII_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

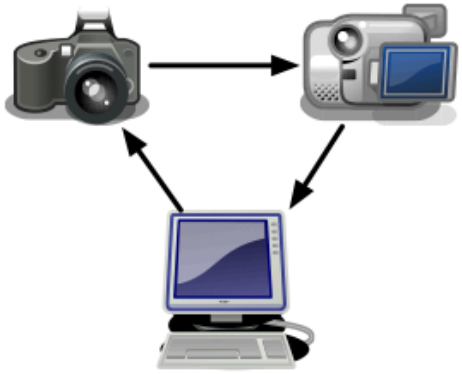
Clone existing code and fix low level platform interaction

+ 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

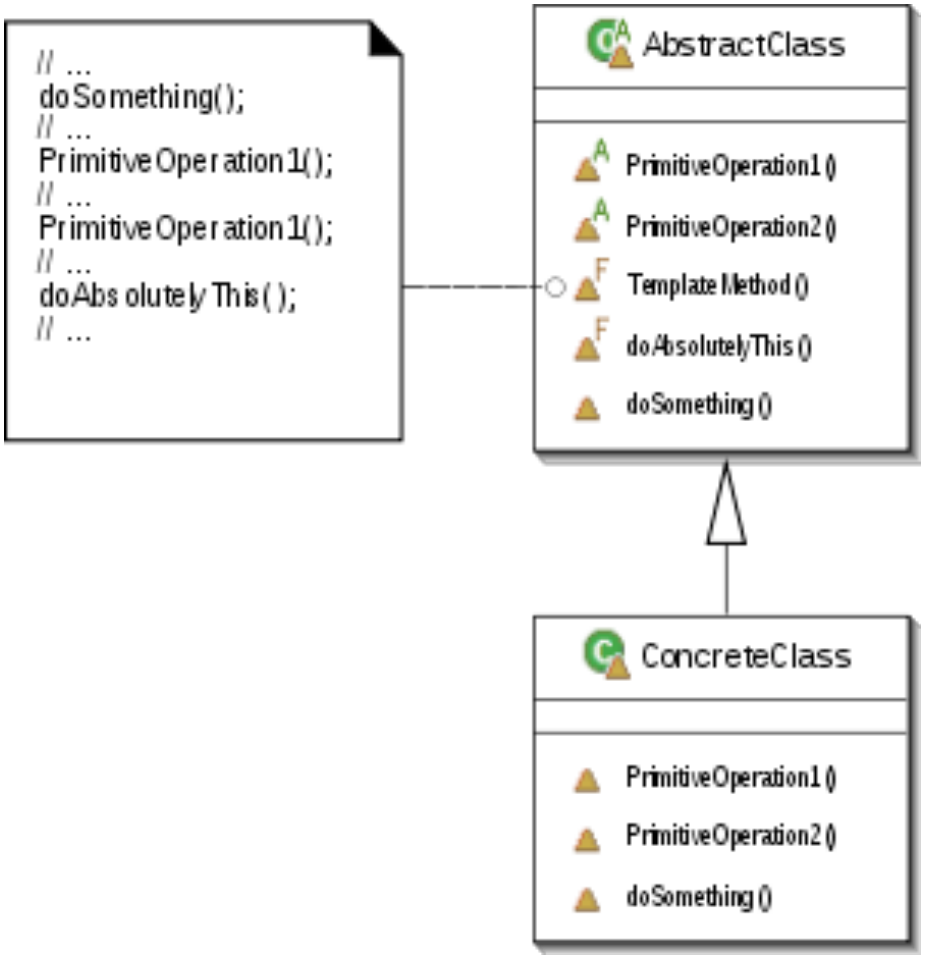
Factory

Strategy

Decorator

....

Template Method



API

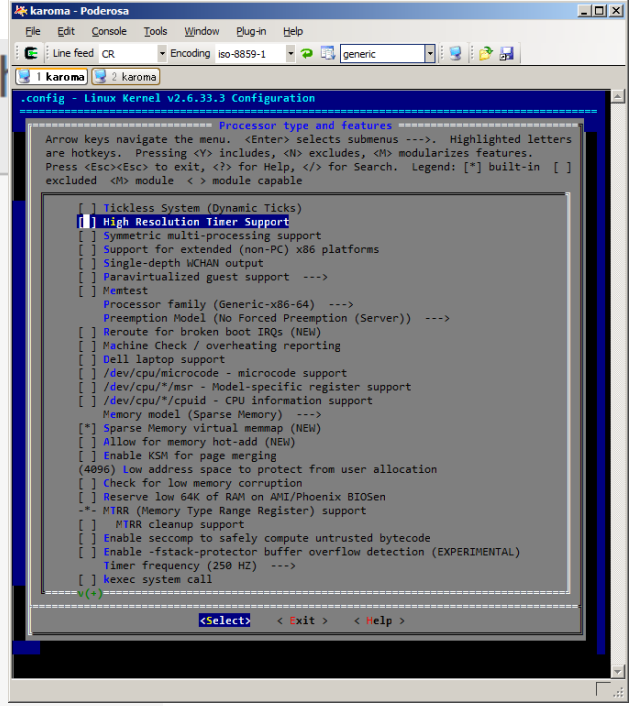
Framework

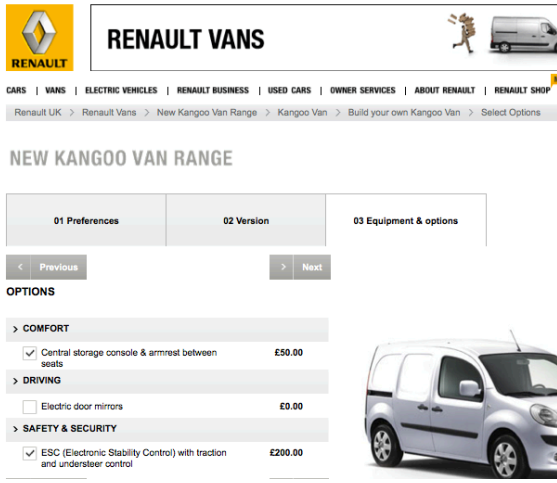
Plugin-based systems

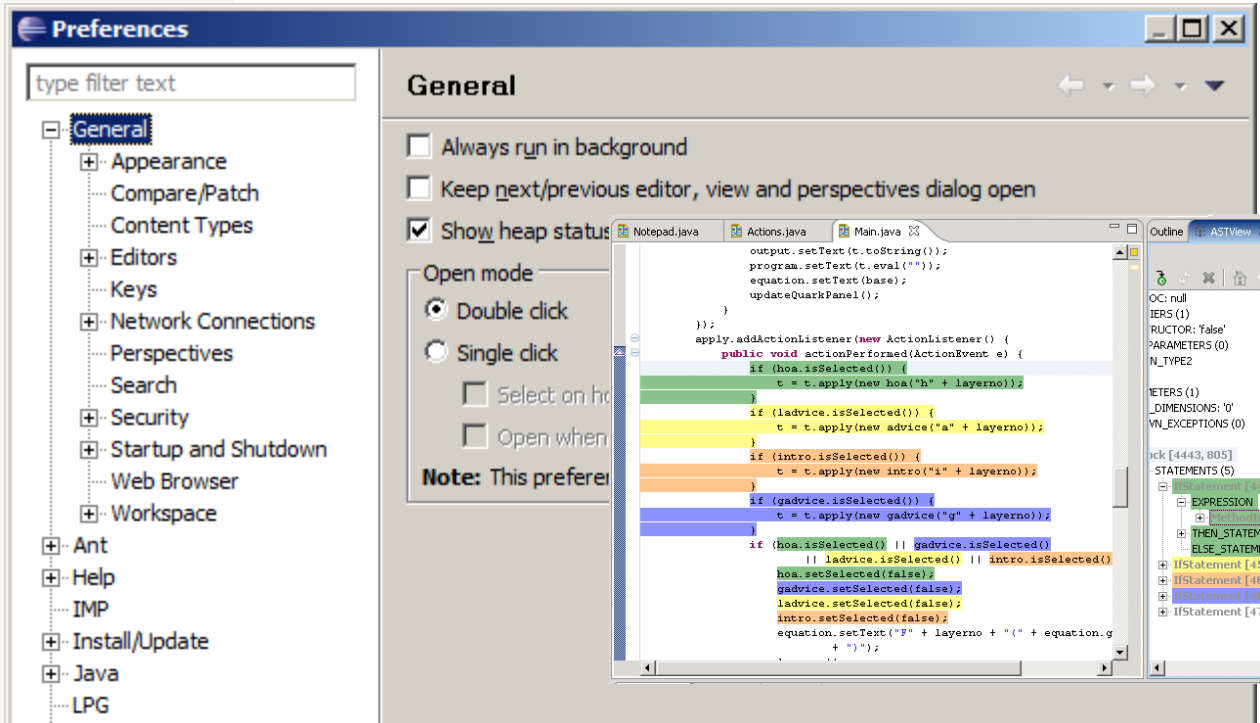
httpd.conf -- win32 Apach

Building a Web Server, for Windows

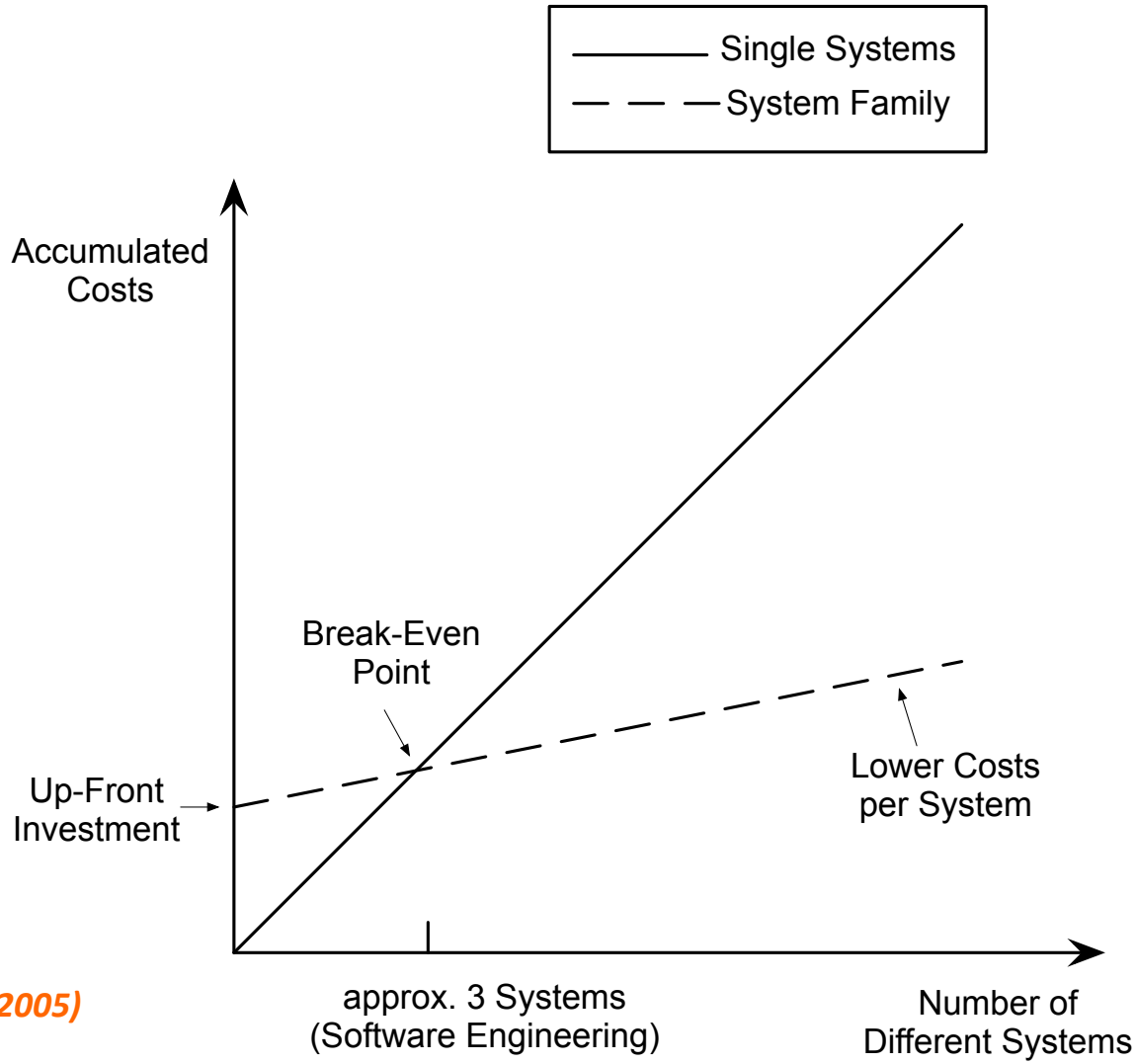
```
Listen 80
ServerRoot "/www/Apache2"
DocumentRoot "/www/webroot"

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>
```





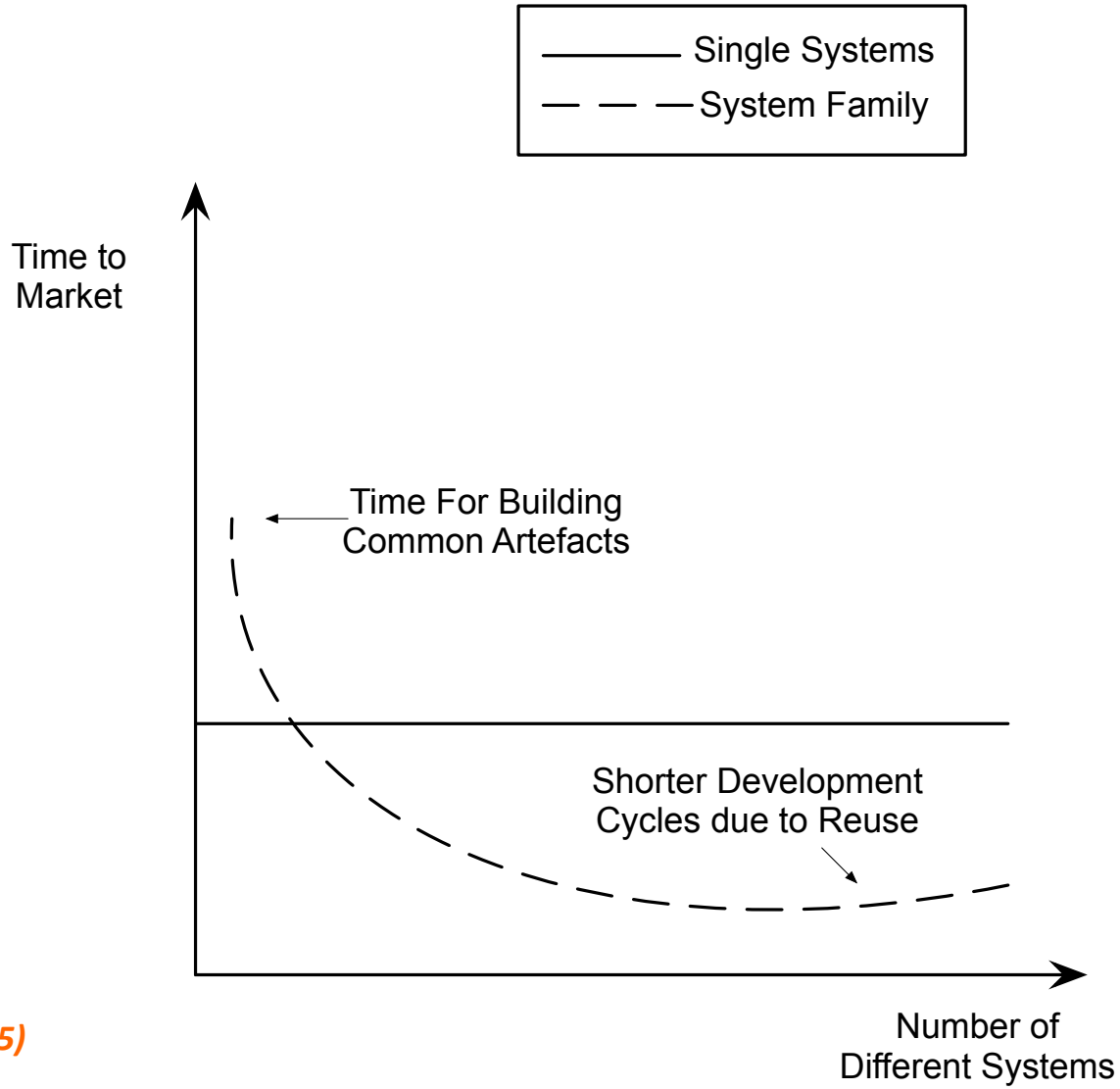


Promises of Software Product Line Engineering



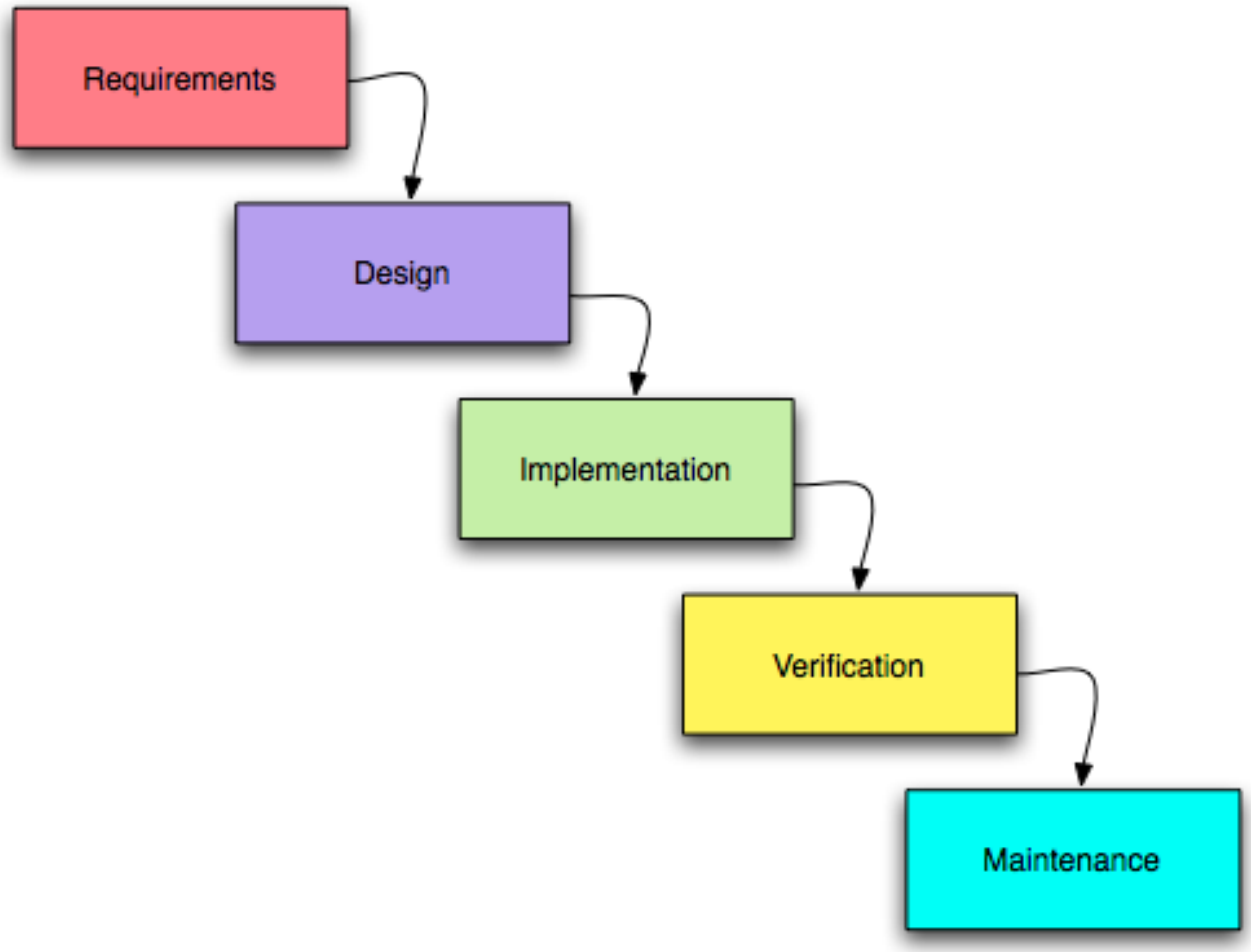
Klaus Pohl (2005)

Promises of Software Product Line Engineering

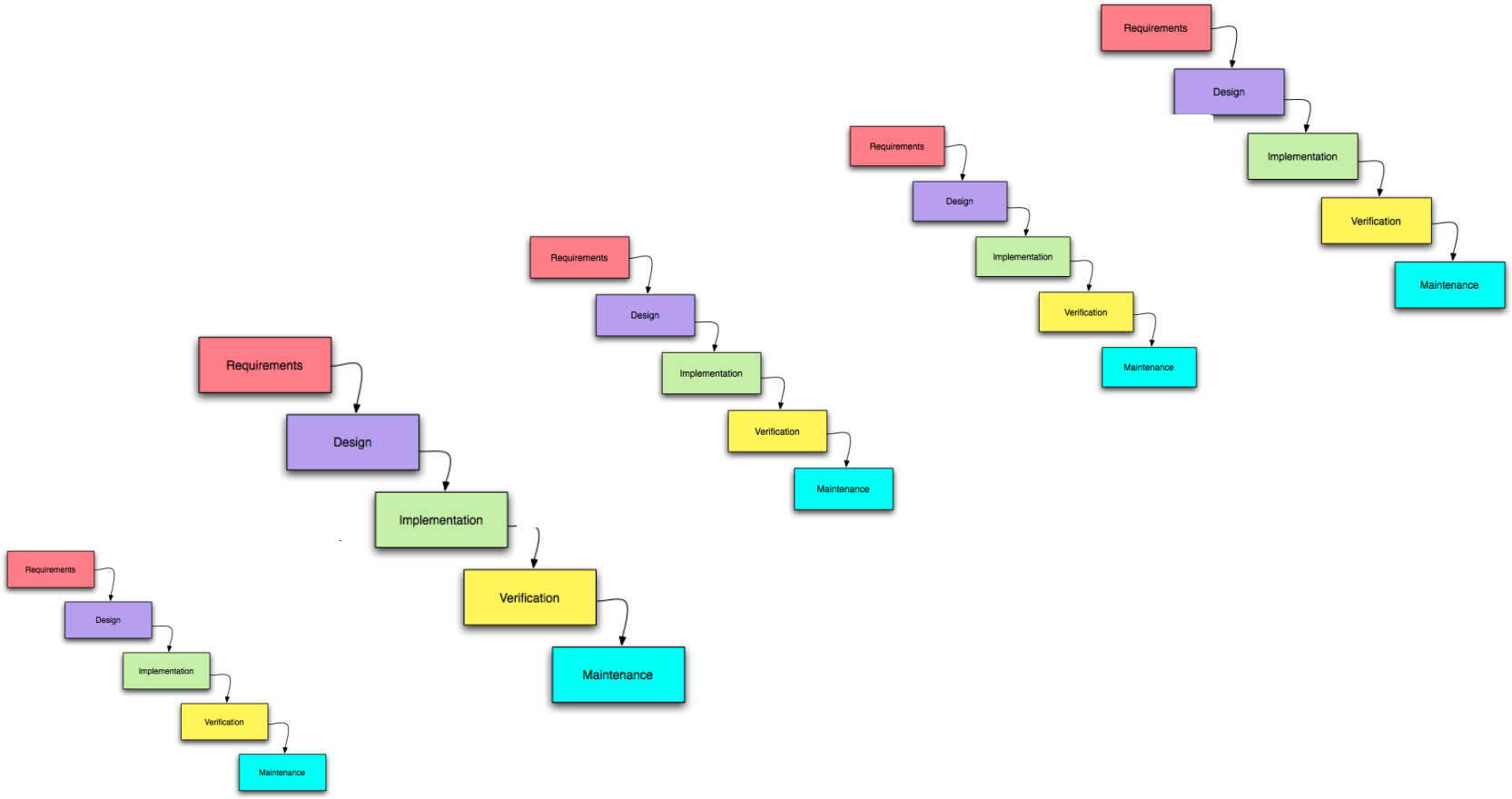


Klaus Pohl (2005)

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

~=

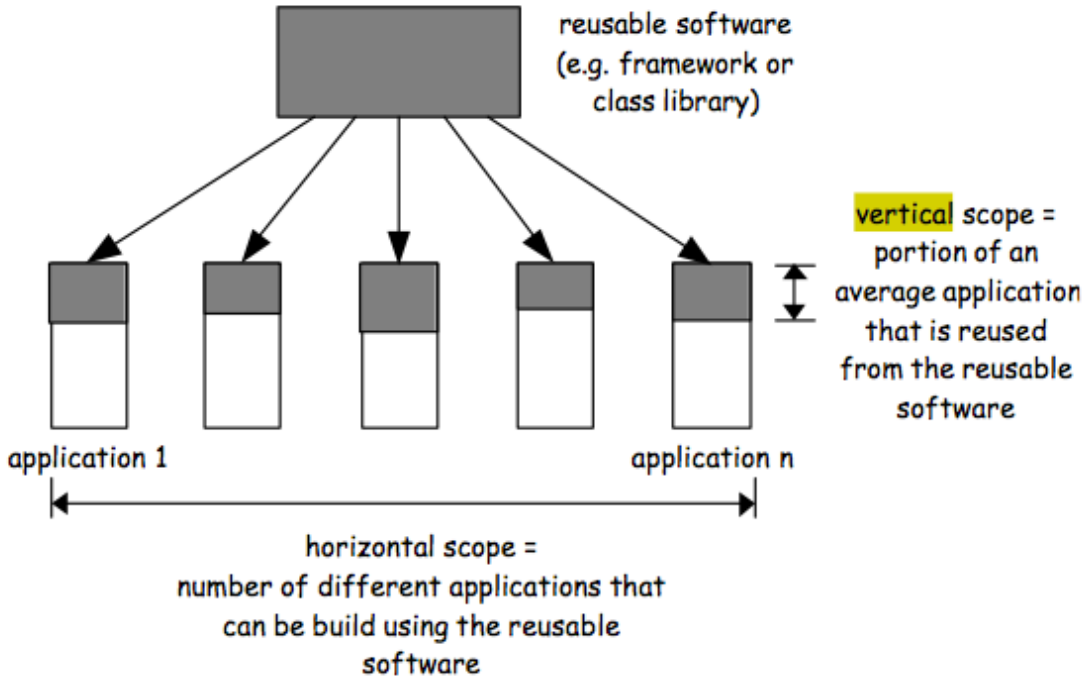
Product Line Engineering

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

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

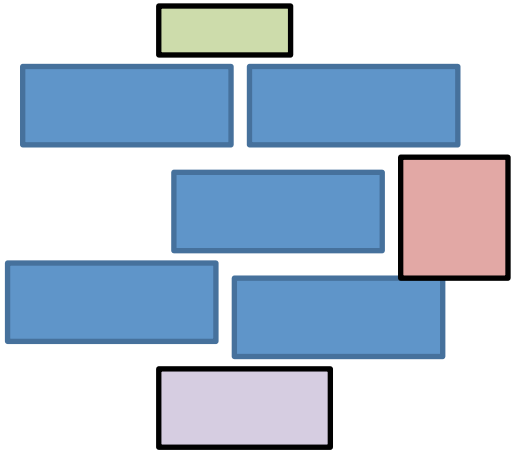
Domains of systems vs subsystems

- **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.

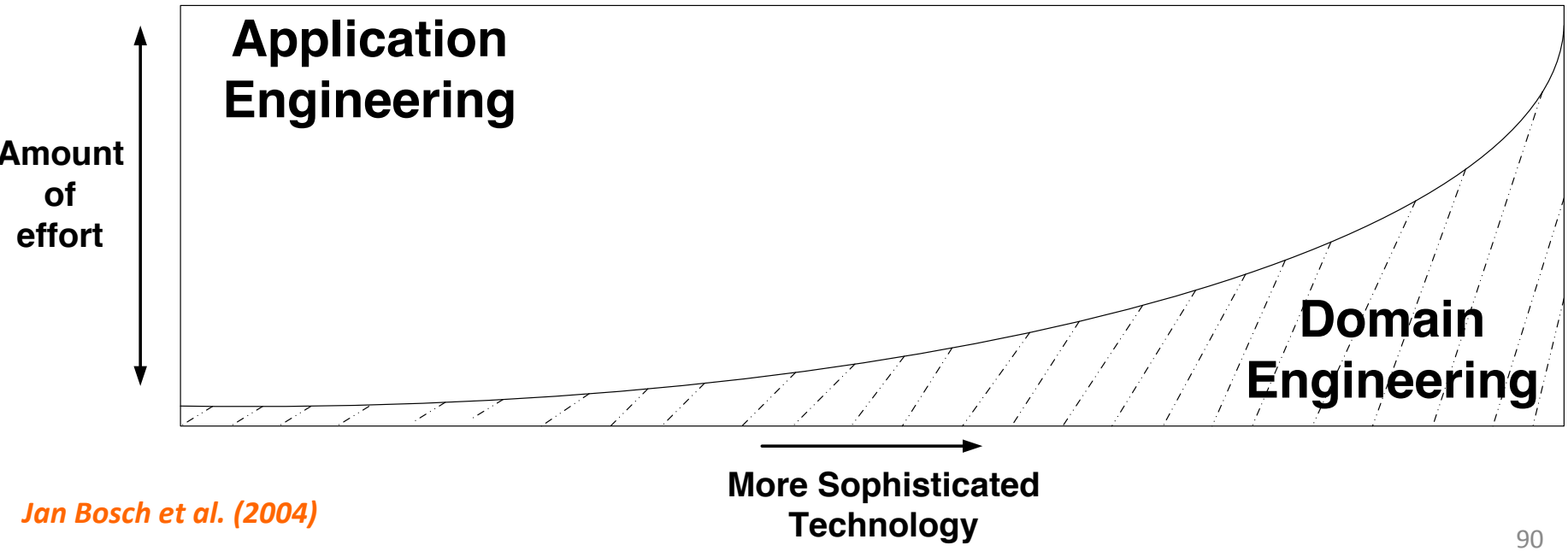


Czarnecki (PhD thesis)

Key idea: building a reusable platform during domain engineering

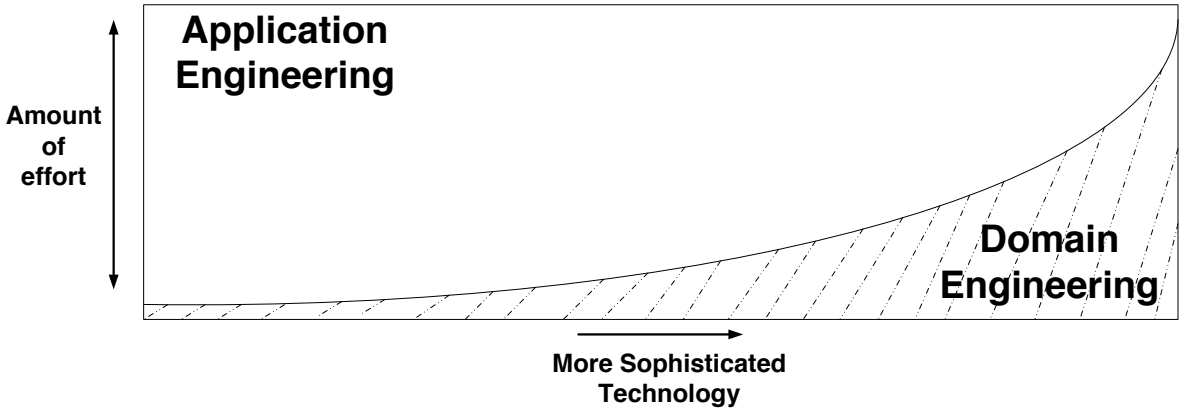


Key idea: *« the investments required to develop the reusable artifacts during **domain engineering**, are outweighed by the benefits of deriving the individual products during **application engineering** »*



Jan Bosch et al. (2004)

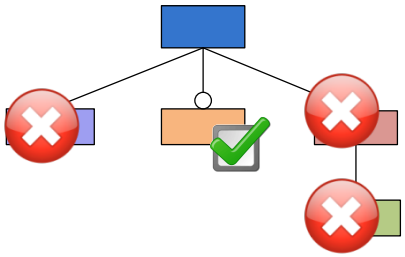
The dream for an SPL practitioner

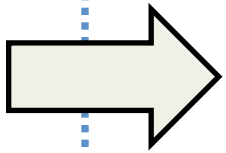


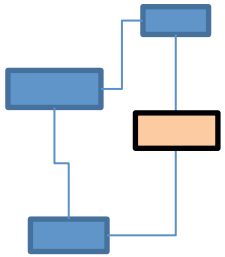
99% domain engineering, 1% application engineering

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

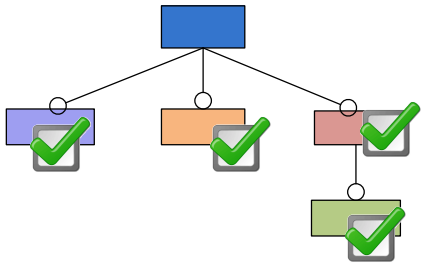
Specific requirements

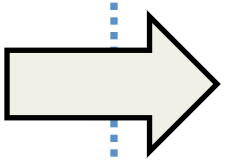


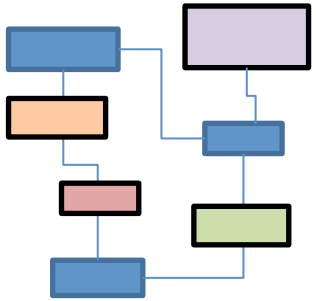




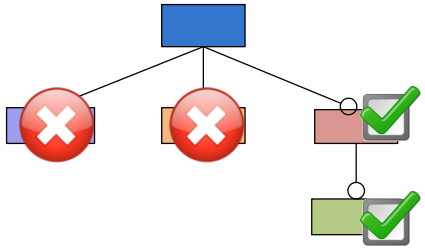
$product_2$

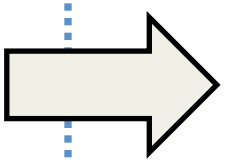


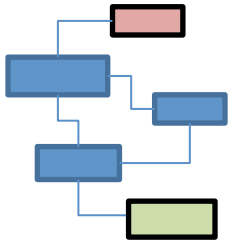




$product_n$

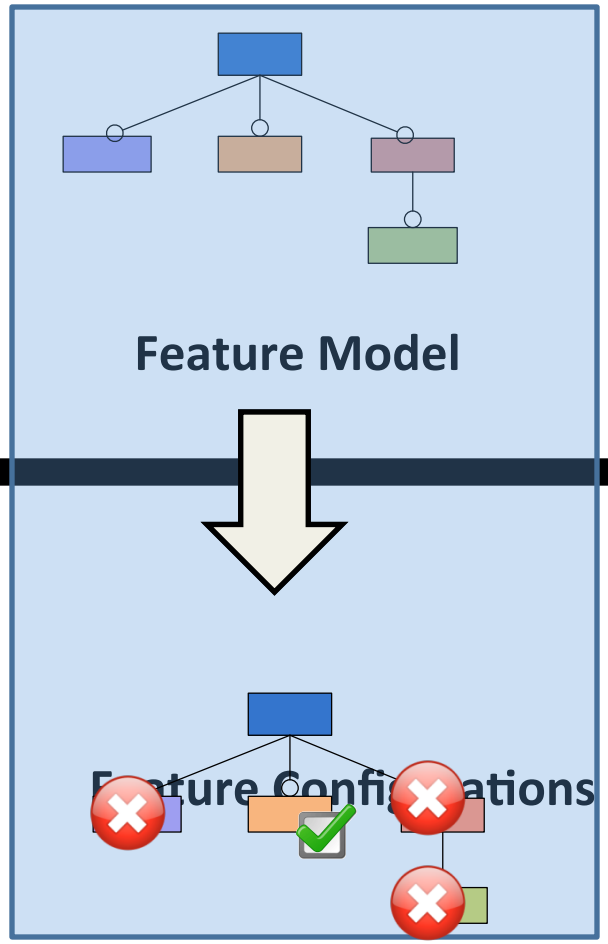






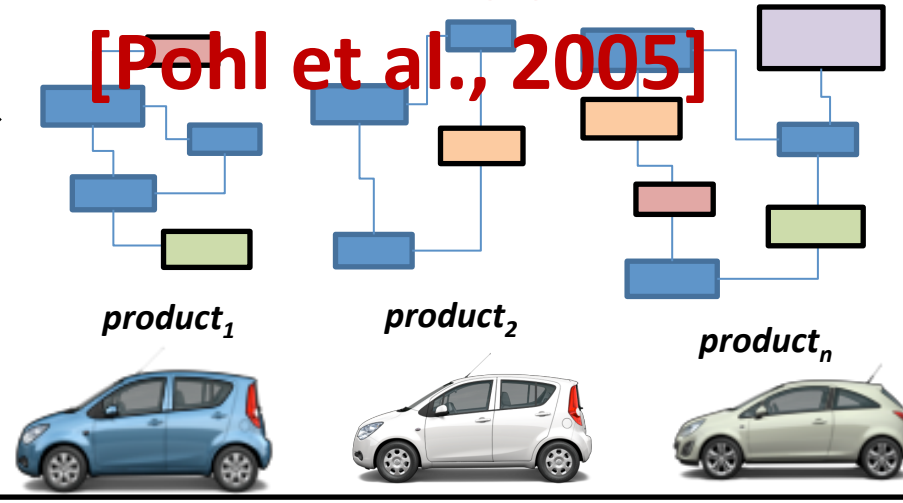
$product_1$

Domain engineering (development for reuse)



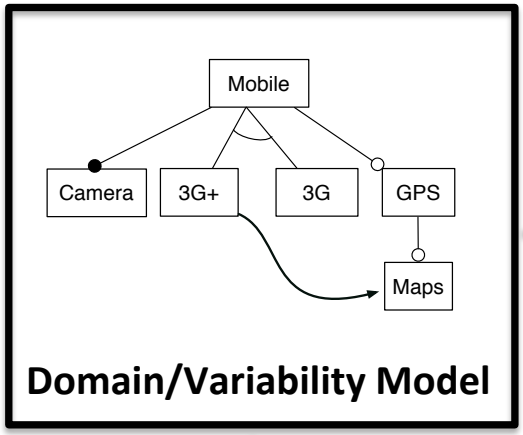
Reusable Assets
(e.g., models or source code)

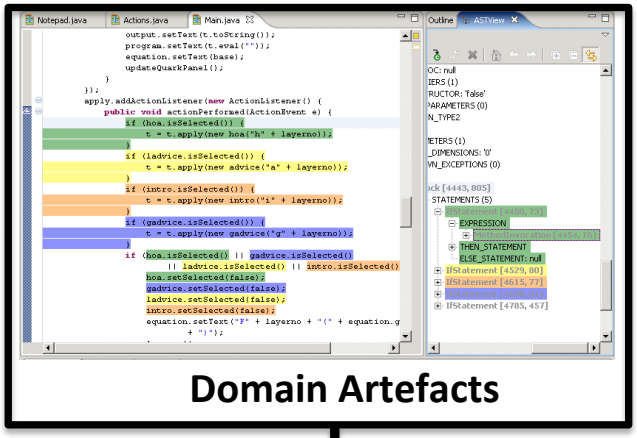
"central to the software product line paradigm is the modeling and management of variability, that is, the commonalities and differences in the applications" [Pohl et al., 2005]



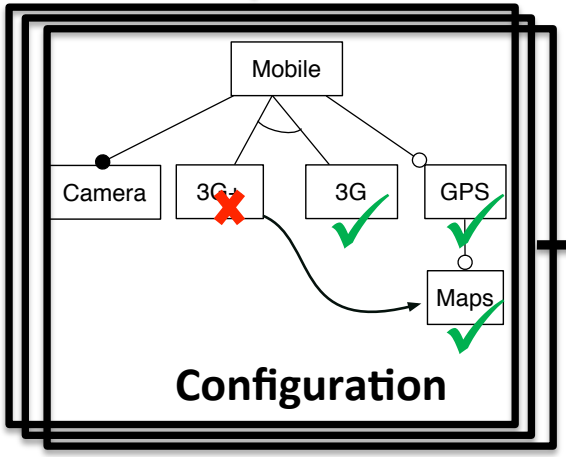
Application engineering (development with reuse)

Domain Engineering





Application Engineering

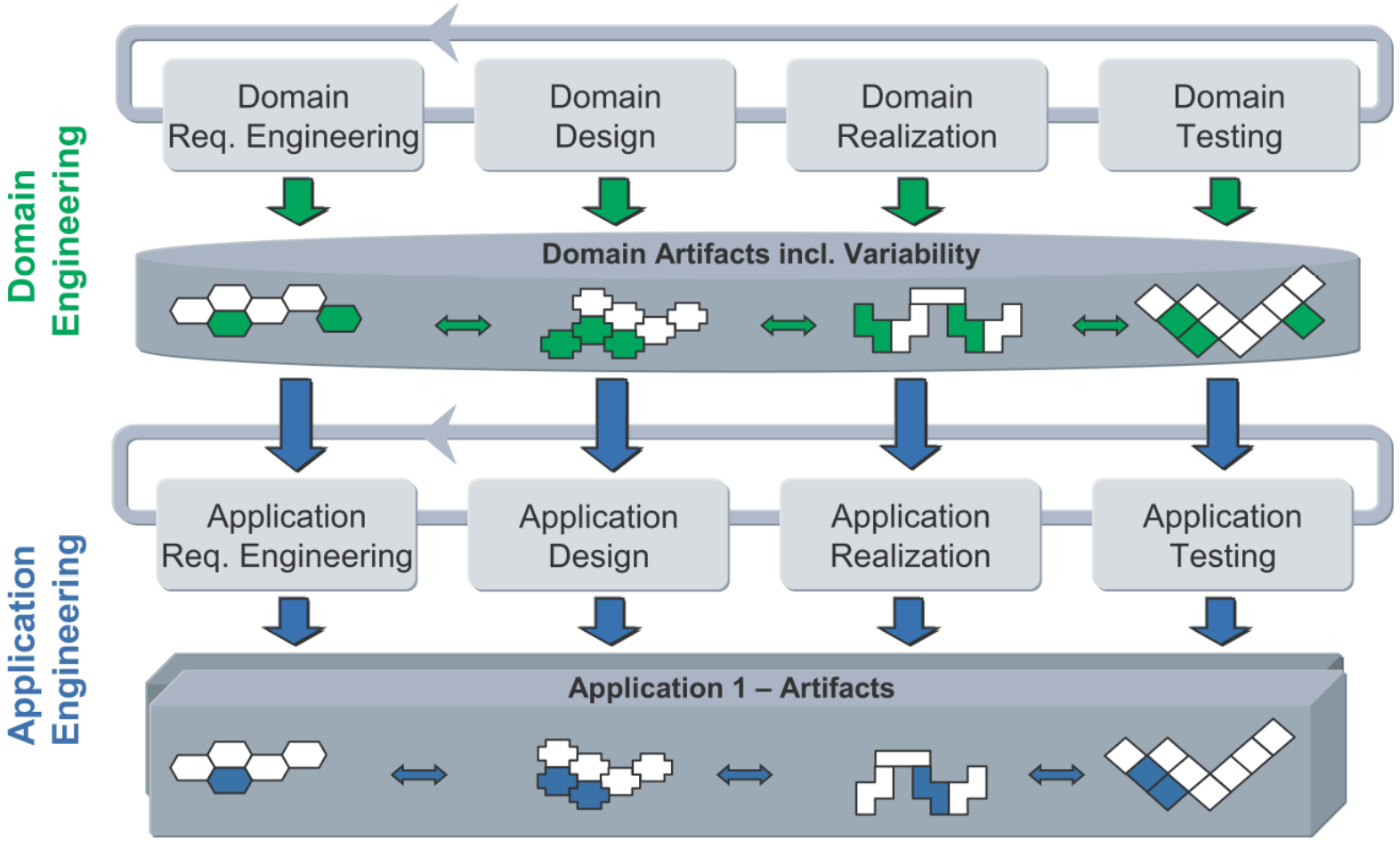


*« the investments required to develop the reusable artifacts during **domain engineering**, are outweighed by the benefits of deriving the individual products during **application engineering** »*

Jan Bosch et al. (2004)

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)

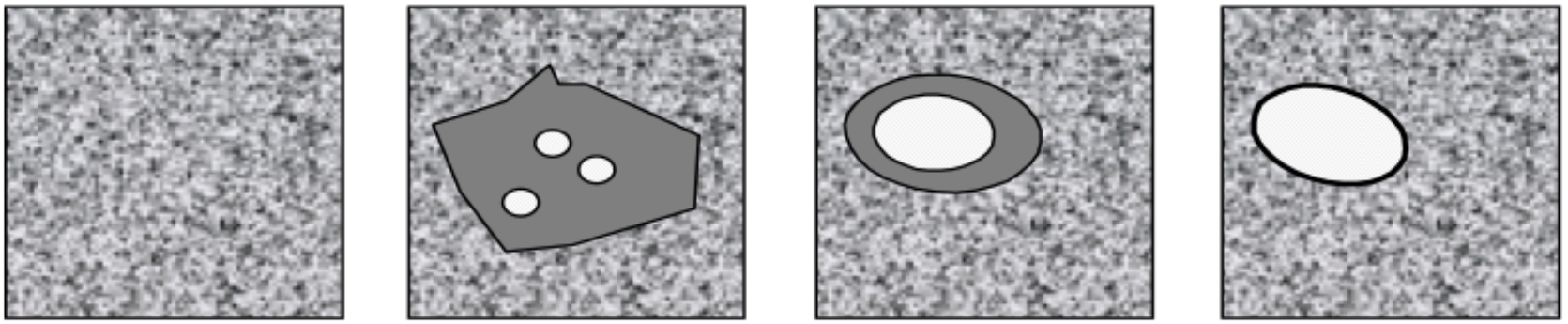
Domain Model

- 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

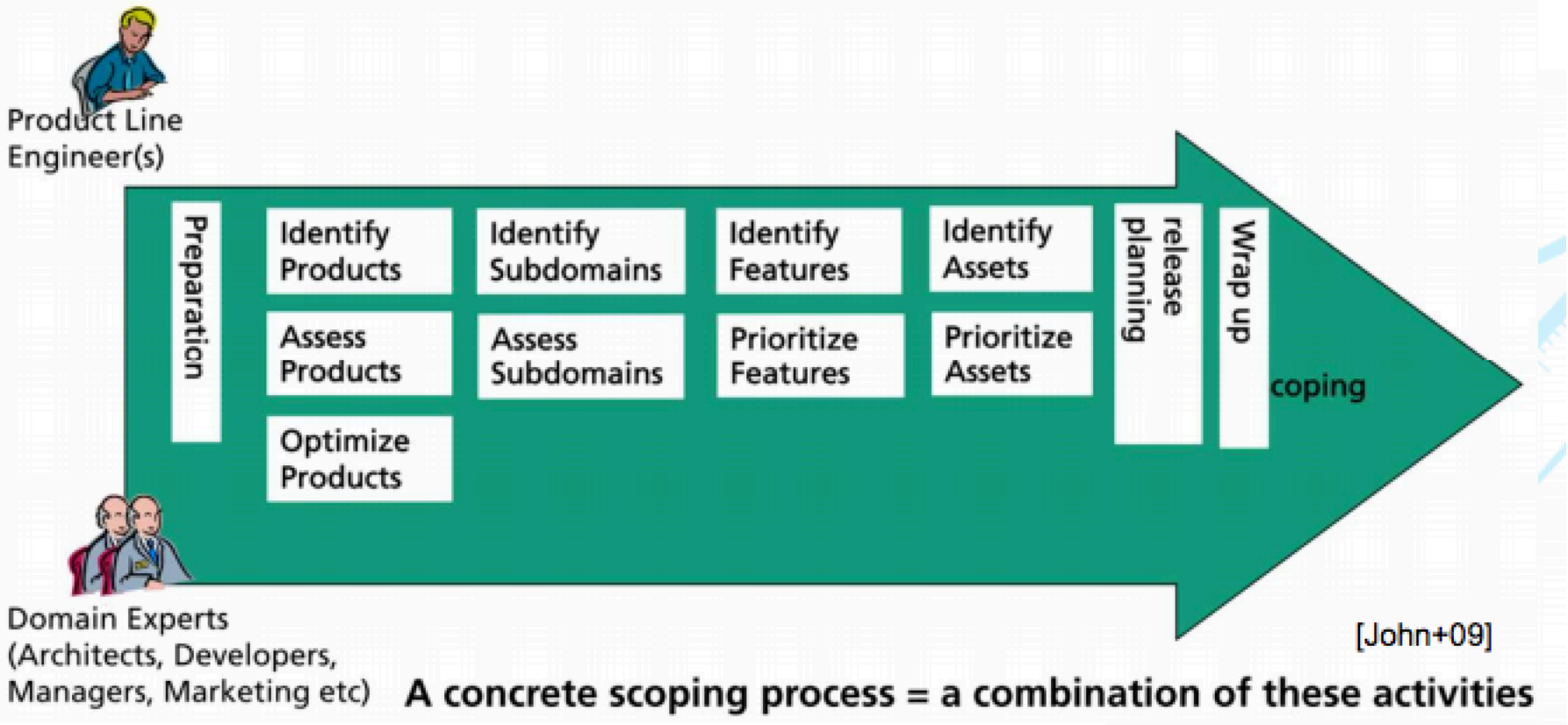


Clement et al. (2001)

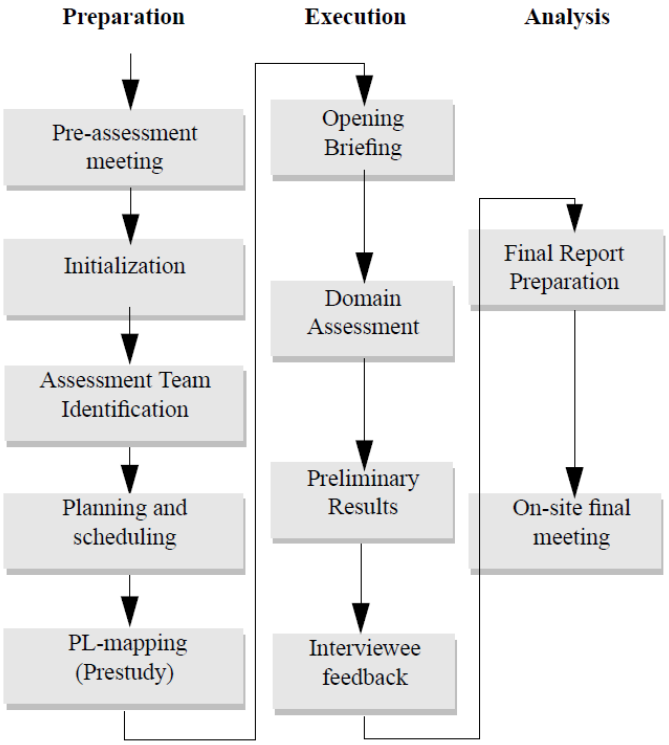

Unused flexibility

Illegal Variant

Scoping Activities



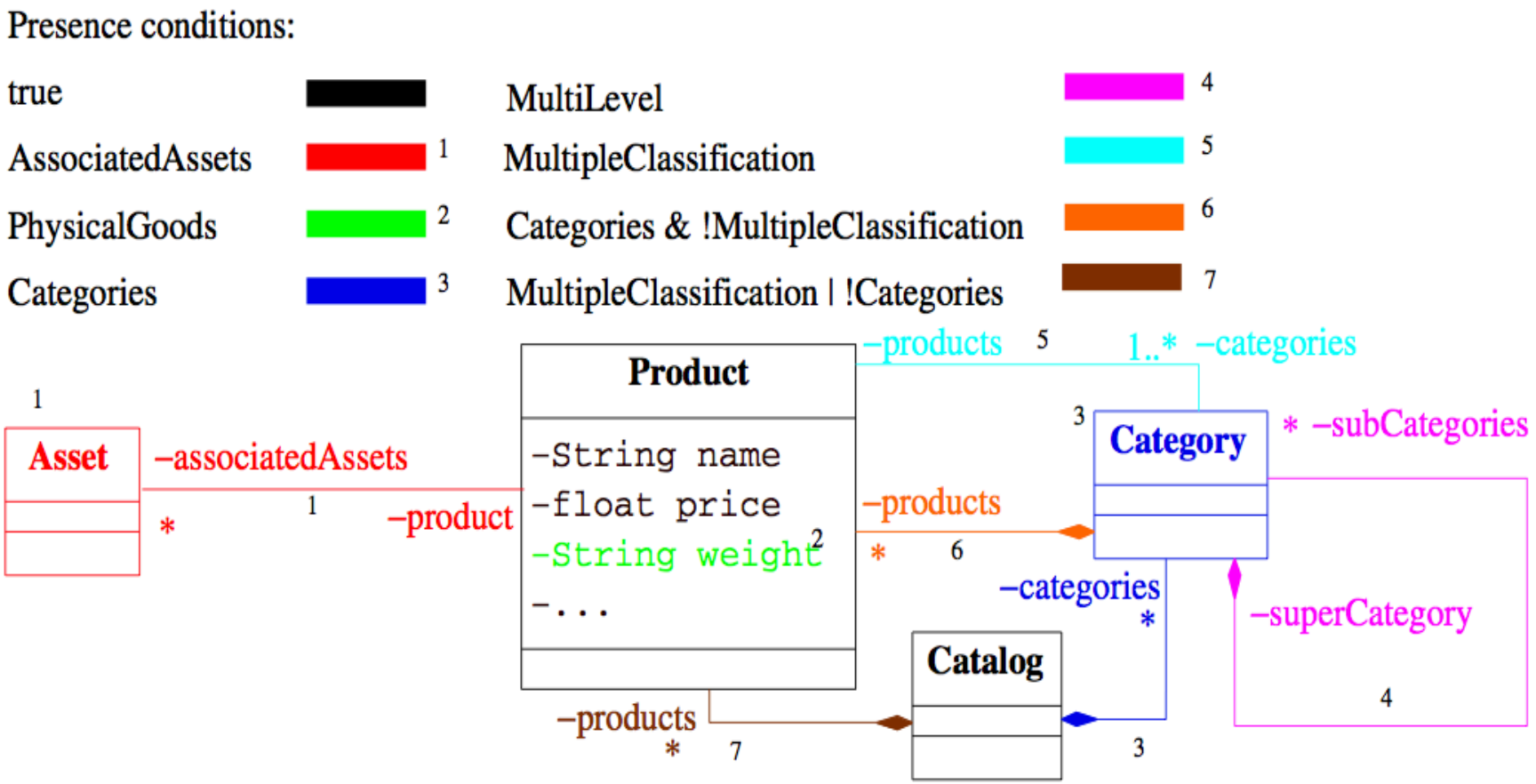
Domain/Product Line Scoping



Schmid 2002

			exist.	planned		potent.
			P1	P2	P3	P4
Domain 1	Sub-Domain 1.1	Feature 1.1.1	X	X	X	X
		Feature 1.1.2	—	X	X	X
		Feature 1.1.3	X	X	—	X
	...	...	...	...	...	...
	Sub-Domain 1.n	Feature 1.n.1	X	—	X	X
		...	...	...	...	...
Domain 2	Sub-Domain 2.1	Feature 2.1.1	—	X	X	—
	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	Feature m.1.1	—	X	—	X

Domain Design



Czarnecki et al. (GPCE'05)

Domain Implementation

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

- Many implementation techniques

Domain-specific language (DSL)

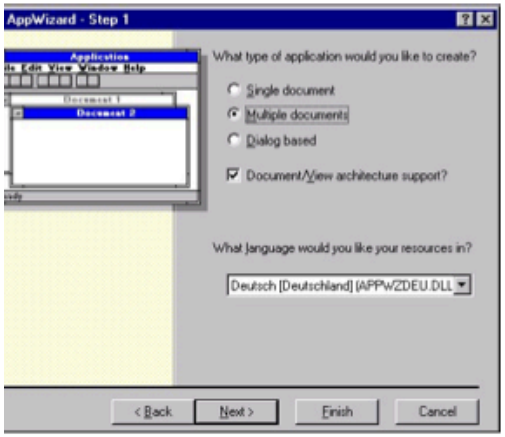
- Domain-specific **abstractions**
  - a DSL provides pre-defined abstractions to directly represent concepts from the domain.
- Domain-specific **concrete syntax**
  - a DSL offers a natural notation for a given domain and avoids syntactic clutter that often results when using a general- purpose language.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

DSLs

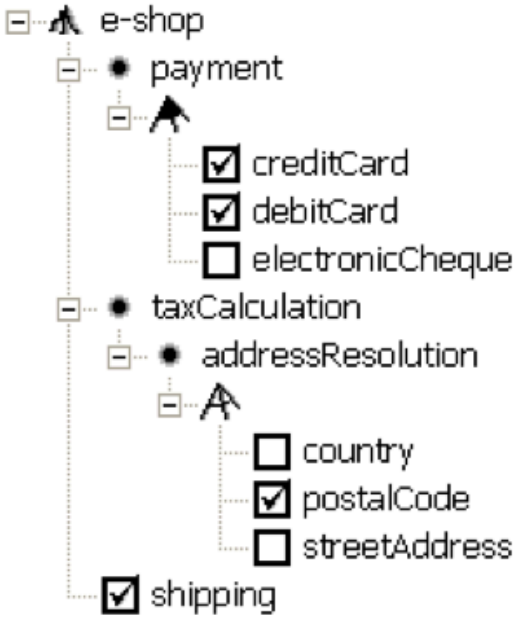
- textual languages (stand-alone or embedded in a general-purpose programming language),
- visual, diagrammatic languages
- form-based languages

Czarnecki et al. (2005)

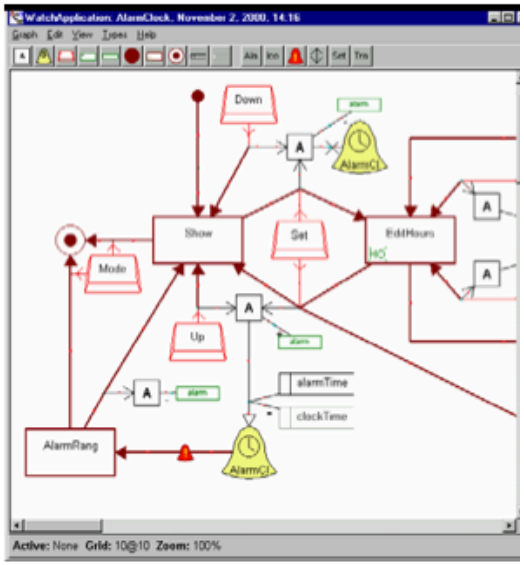
Routine configuration ⟷ *Creative construction*



Wizard



Feature-based configuration



Graph-like language

Dummy Feature Model

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

Voelter (SPLC'11)

Configuring Models and Code

```
{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 th
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>;
}
```

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

http://weblogs.java.net/blog/tball/archive/2006/09/munge_swings_se.html

```
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() { ... } }
```

Kastner's slide

Mapping: an example

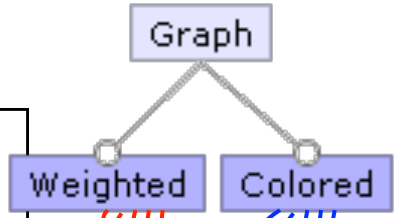
```
class Graph {
  Vector nv = new Vector(); Vector ev = new Vector();
  Edge add(Node n, Node m) {
    Edge e = new Edge(n, m);
    nv.add(n); nv.add(m); ev.add(e);
    e.weight = new Weight();
    return e;
  }
  Edge add(Node n, Node m, Weight w)
    Edge e = new Edge(n, m);
    nv.add(n); nv.add(m); 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]*/
  }
}
```

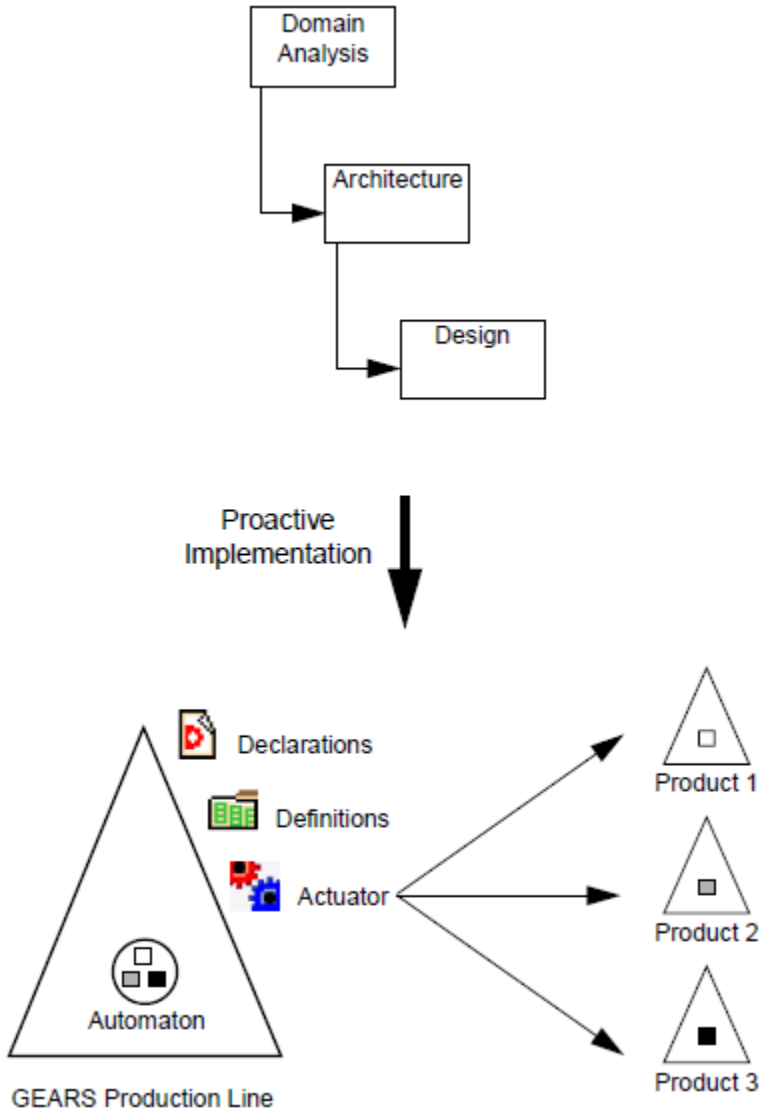
```
/*if[COLOR]*/
class Color {
  static void setDisplayColor(Color c) { ... }
}
/*end[COLOR]*/
```

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

Adoption and Strategies

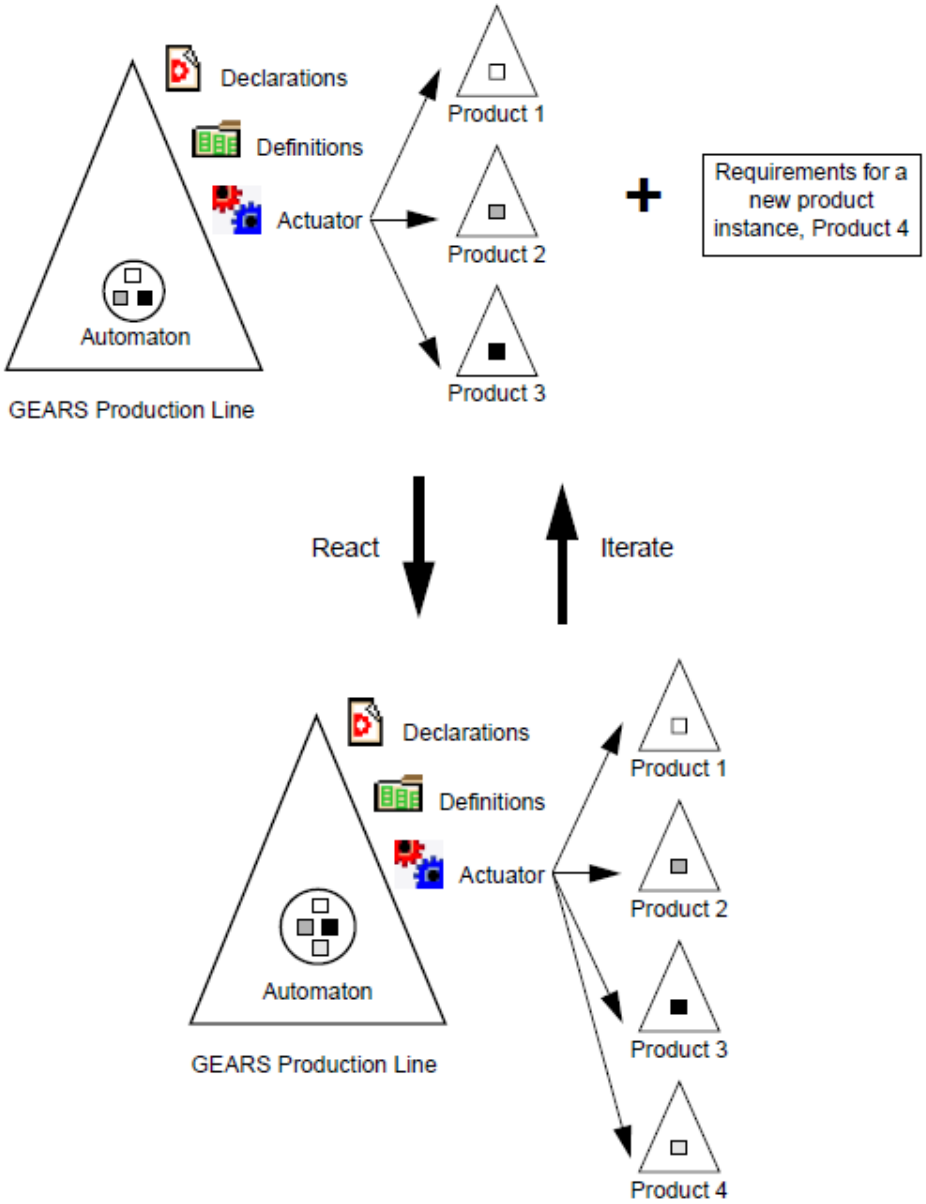
- **Proactive**
- **Reactive**
- **Extractive**

Proactive



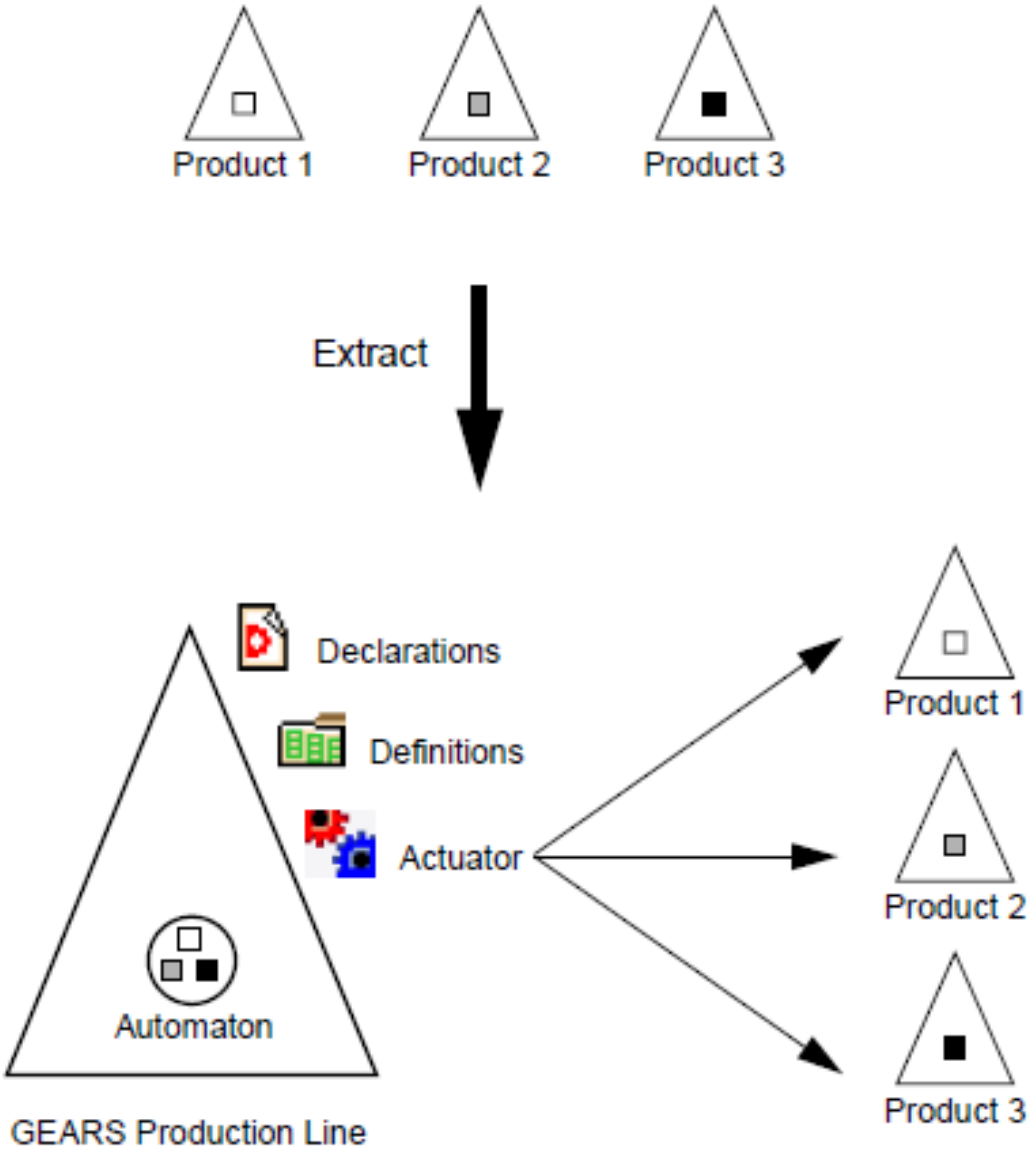
[Krueger 2002]

Reactive



[Krueger 2002]

Extractive



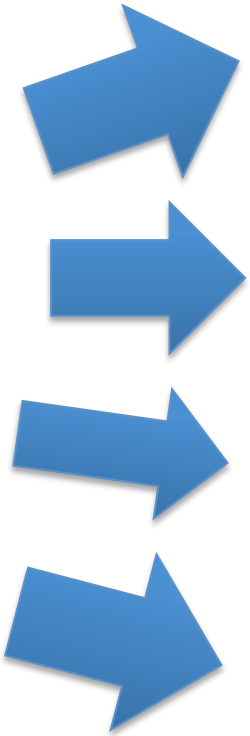
[Krueger 2002]

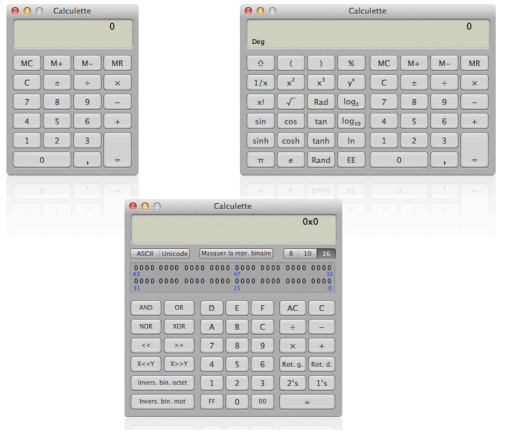
Software Product Line Engineering

Model-based perspective

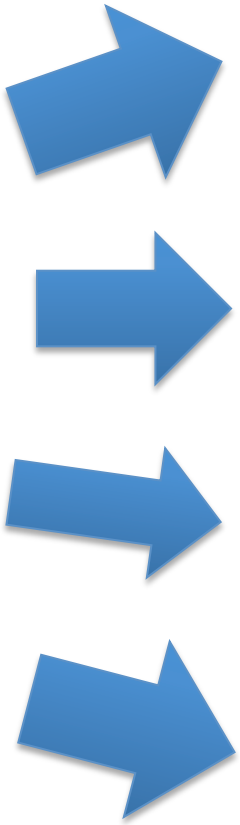
Modeling and implementing system families such that a desired system can be automatically generated from a specification written in one or more textual or graphical domain-specific languages.

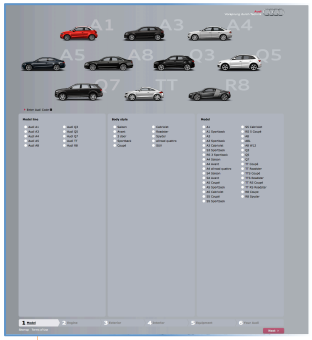
Models And Languages

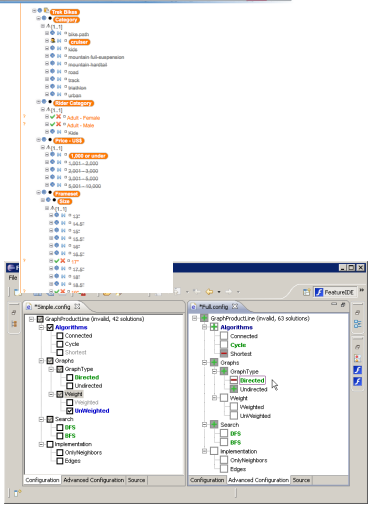


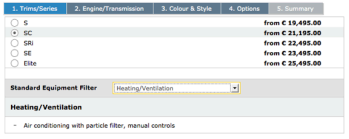


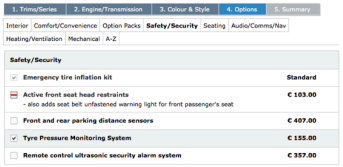
Models And Languages







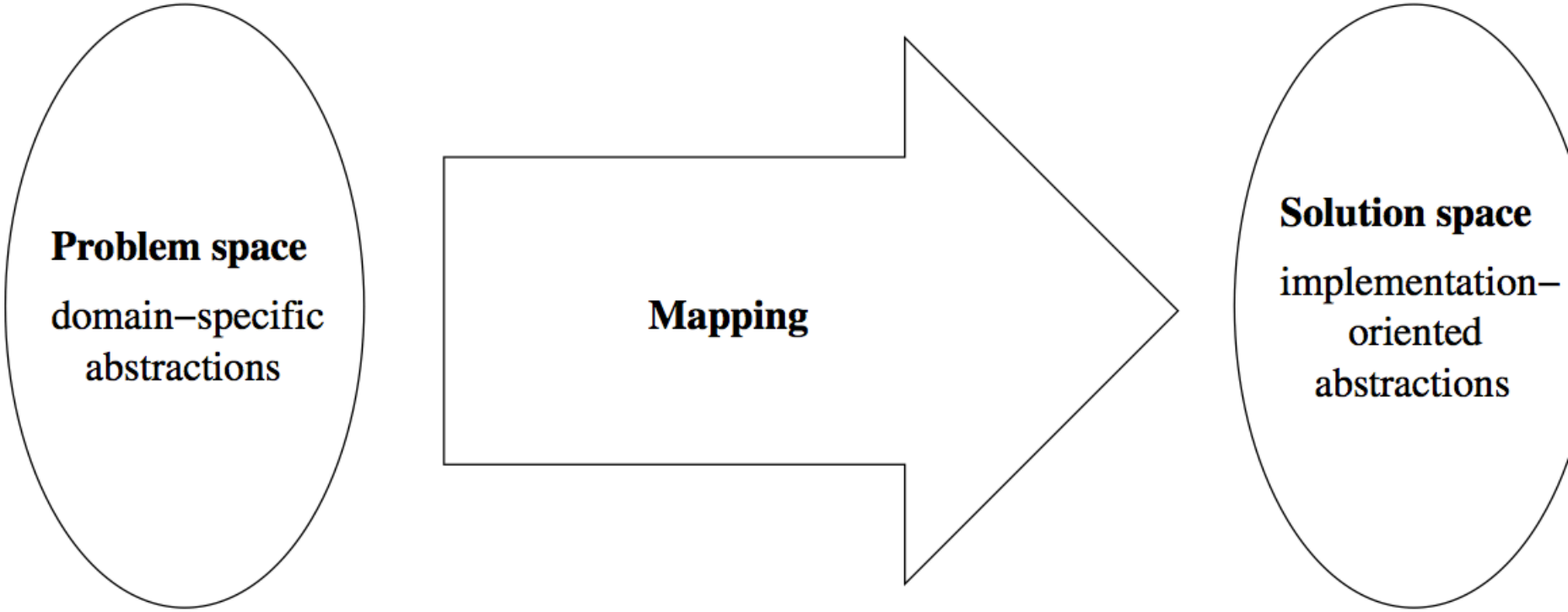




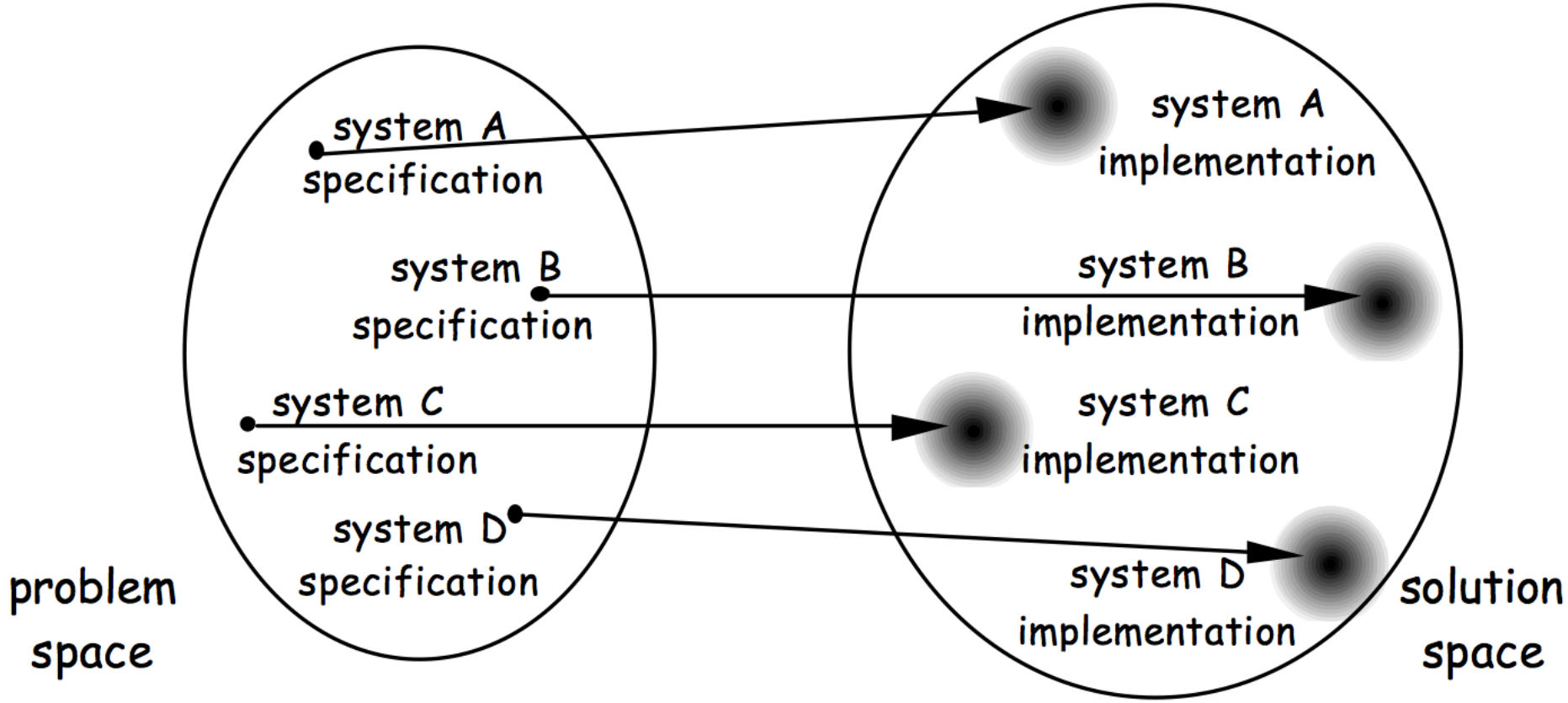
Generative approach

- 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

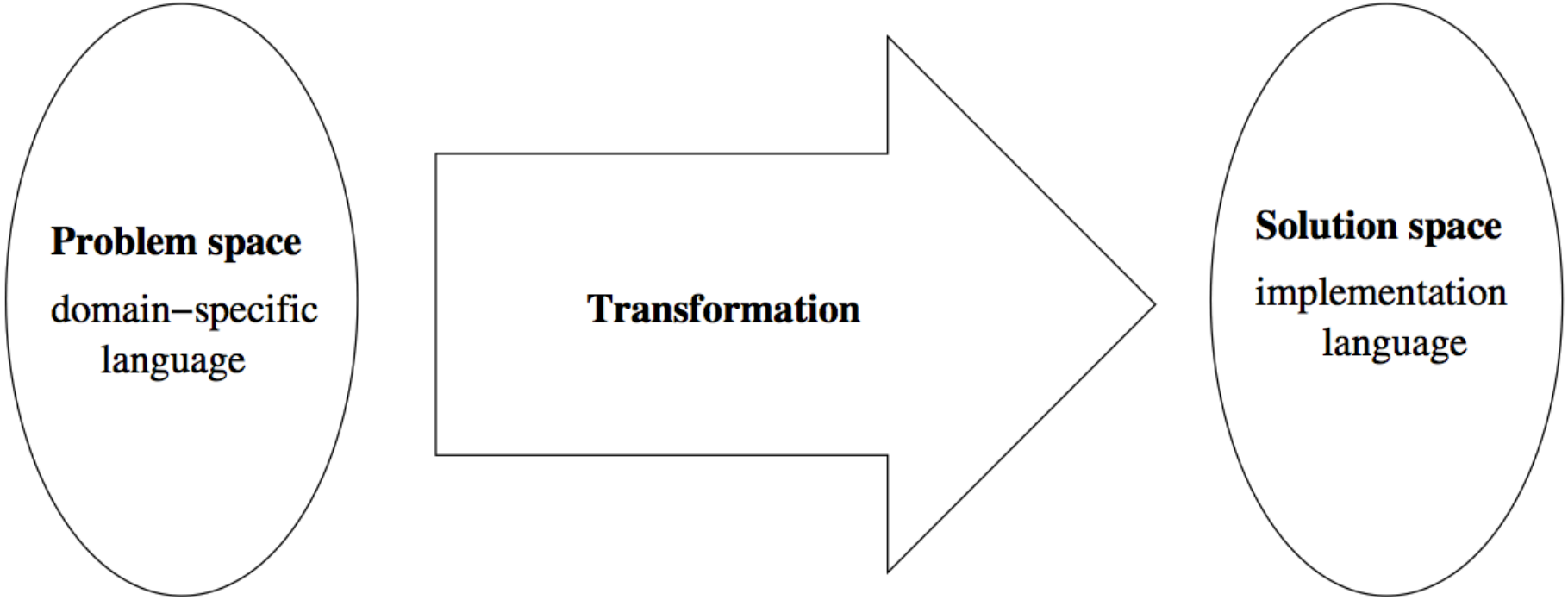


[Czarnecki and Eisenecker 2000]

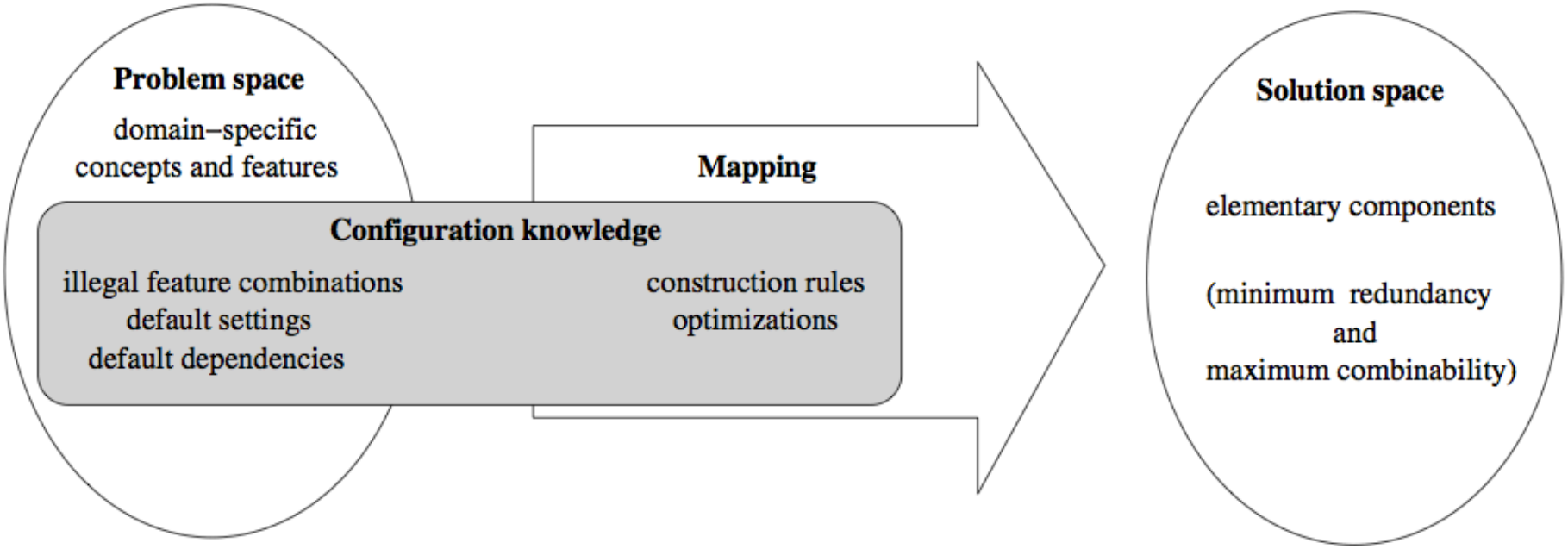


[Czarnecki, PhD thesis]

S



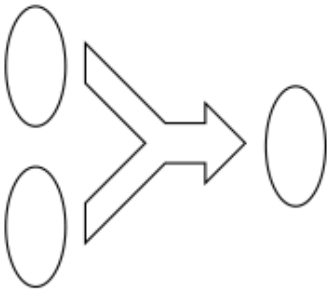
[Czarnecki and Eisenecker 2000]



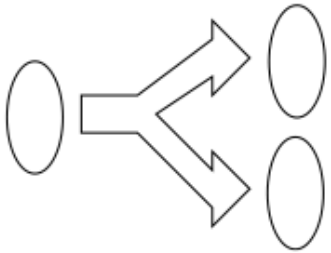
[Czarnecki and Eisenecker 2000]

[Czarnecki et al. 2005]

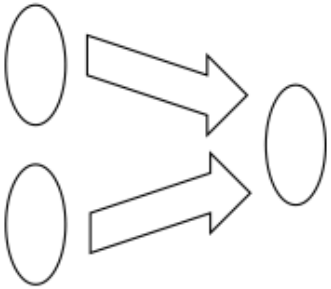
a. Chaining of mappings



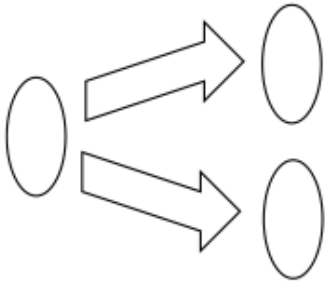
b. Multiple problem spaces



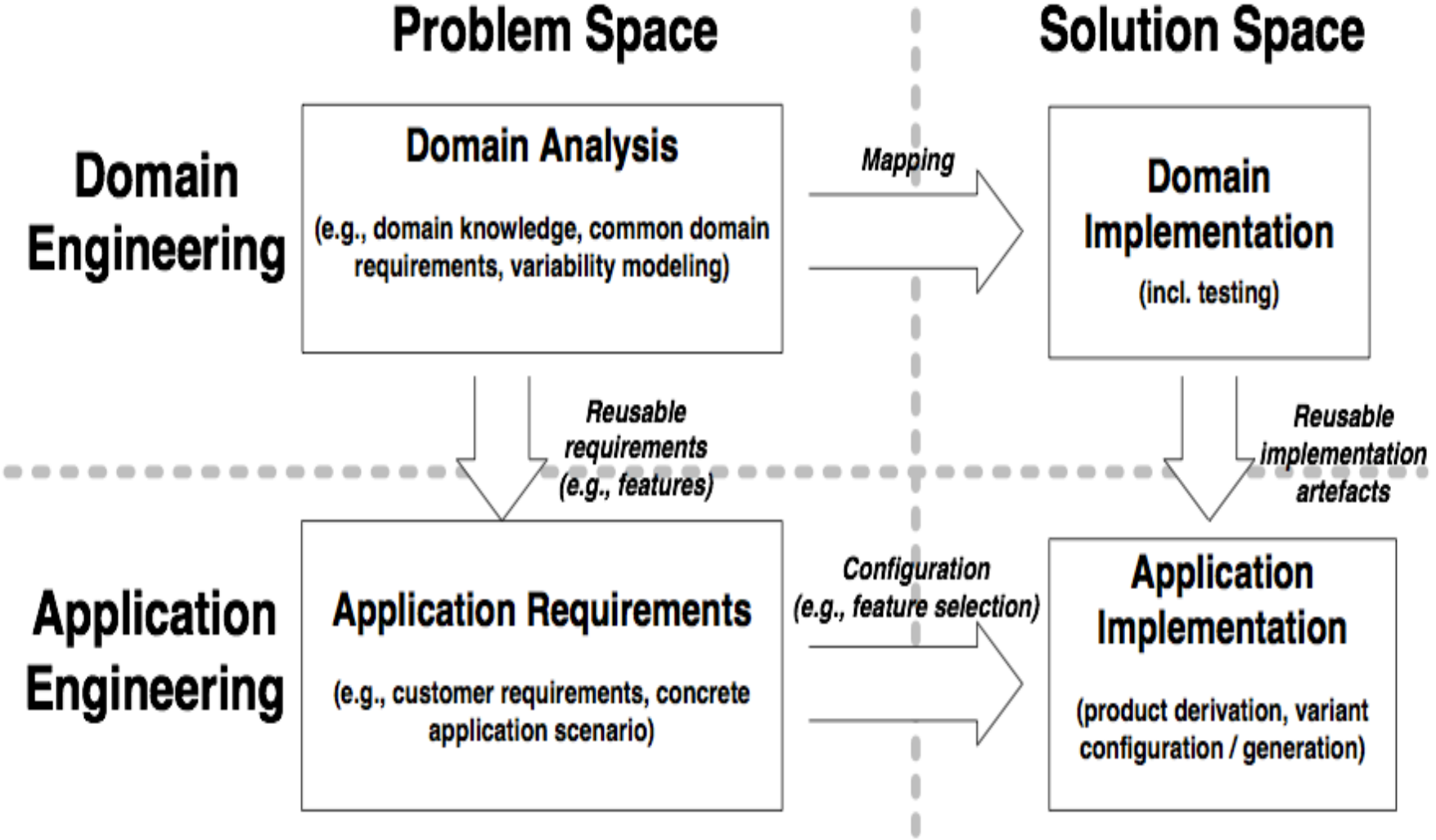
c. Multiple solution spaces



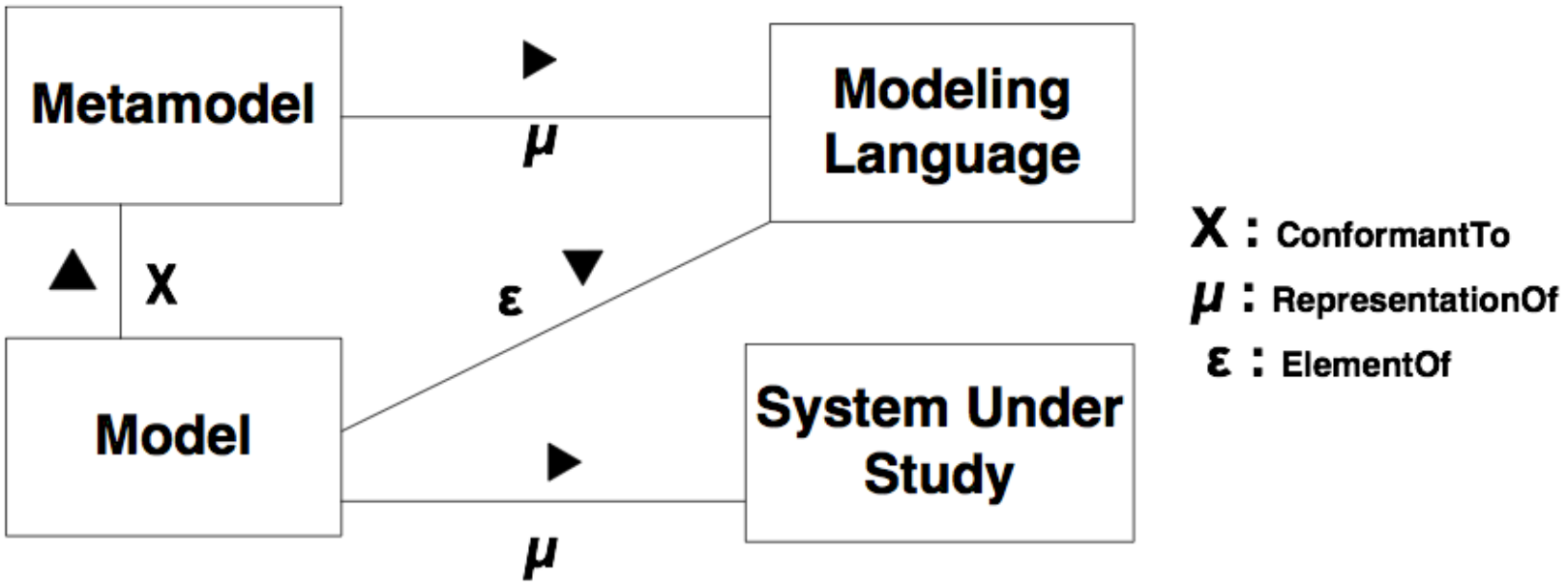
d. Alternative problem spaces



e. Alternative solution spaces


Problem Space
Solution Space
Domain Engineering
Domain Analysis
(e.g., domain knowledge, common domain requirements, variability modeling)
Mapping
Domain Implementation
(incl. testing)
Reusable requirements (e.g., features)
Reusable implementation artefacts
Application Engineering
Application Requirements
(e.g., customer requirements, concrete application scenario)
Configuration (e.g., feature selection)
Application Implementation
(product derivation, variant configuration / generation)

MDE


Metamodel
Modeling Language
Model
System Under Study
μ
μ
ε
X
X : ConformantTo
μ : RepresentationOf
ε : ElementOf

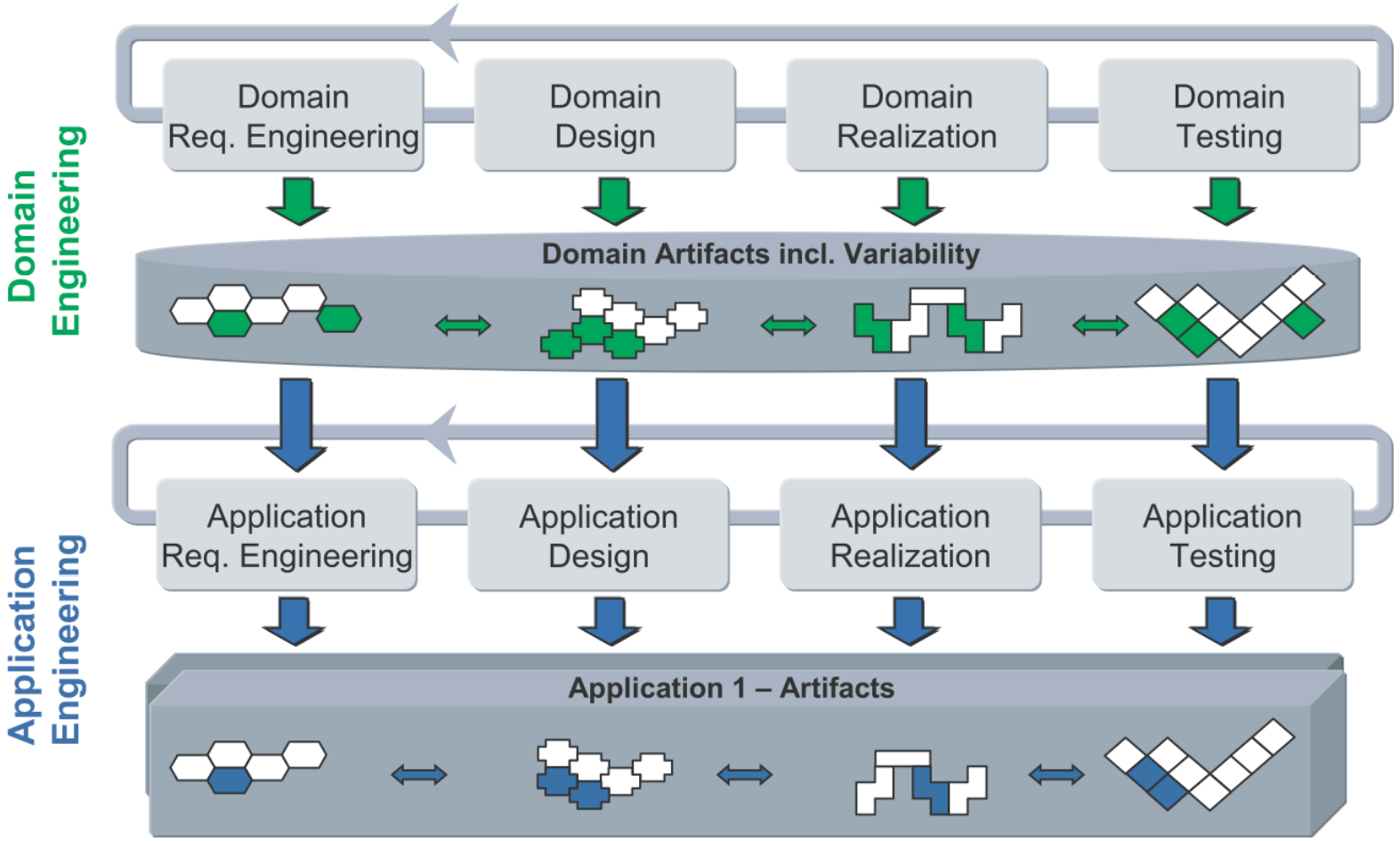
Model in a nutshell

- 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

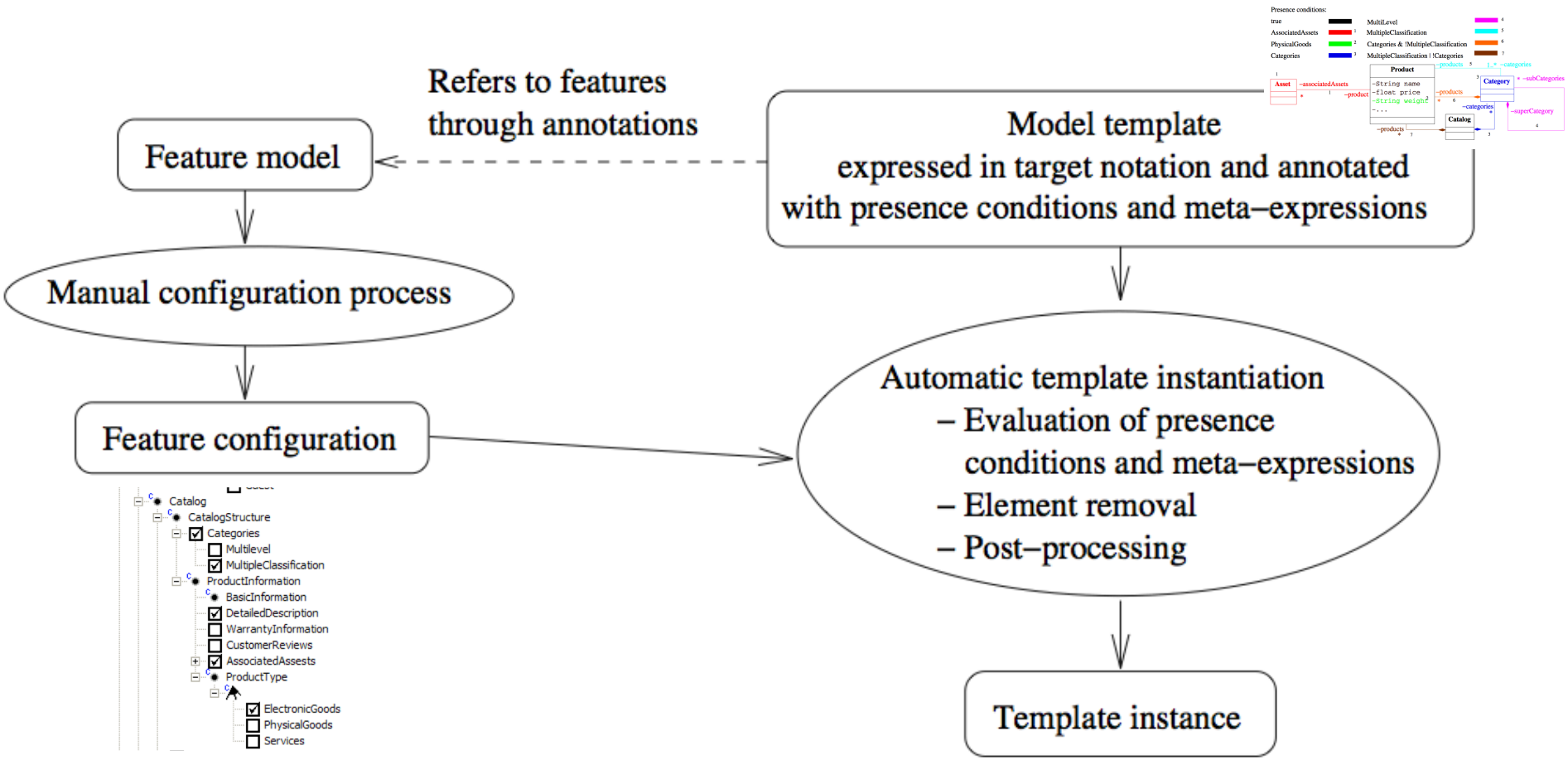
- Domain Engineering
  - Domain Models
  - Level of abstraction
  - Domain-specific modeling languages
    - (visual or textual) syntacs, precise semantics
    - analyzed (verification)
  - Traceability between the artefacts
- Application Engineering
  - Model transformations (automation)
- Reduce the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

Feature-based

Model Templates

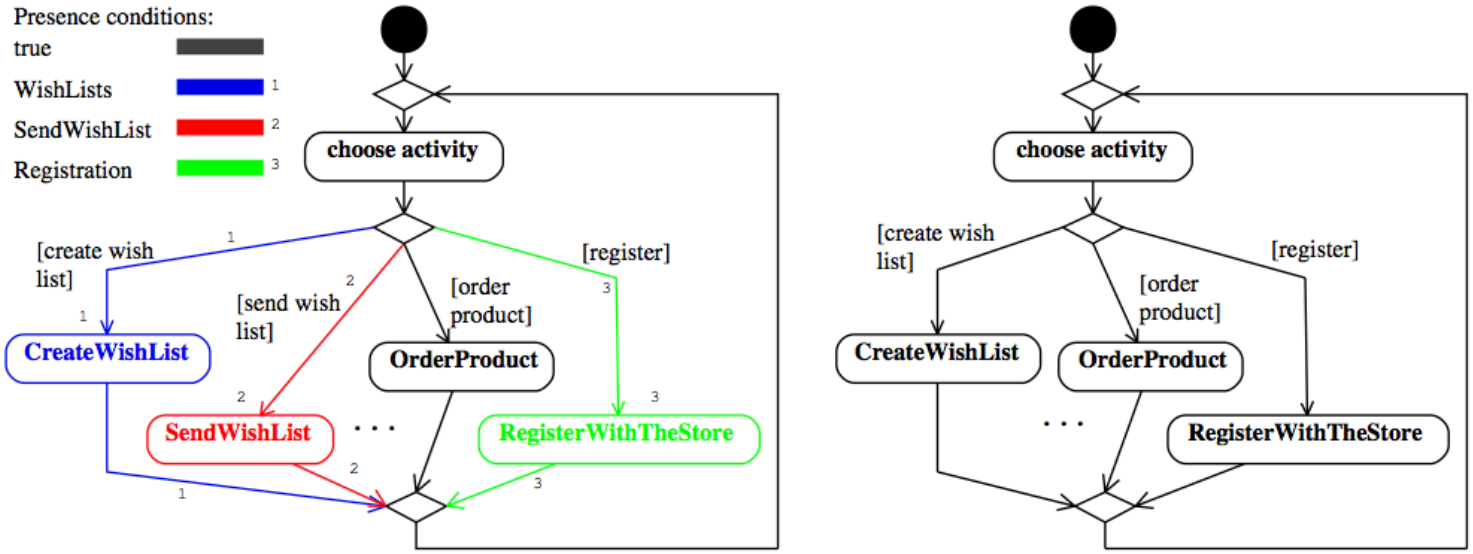
[Czarnecki et al. (GPCE'05)]

Realizing variability: derivation of models



[Czarnecki et al. (GPCE'05)]

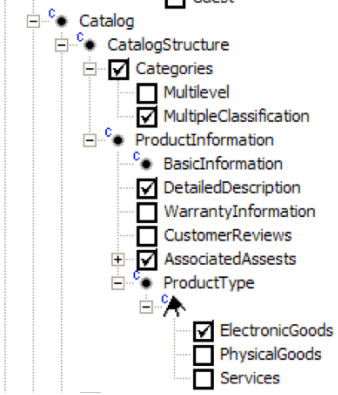
Example



(a) Storefront template

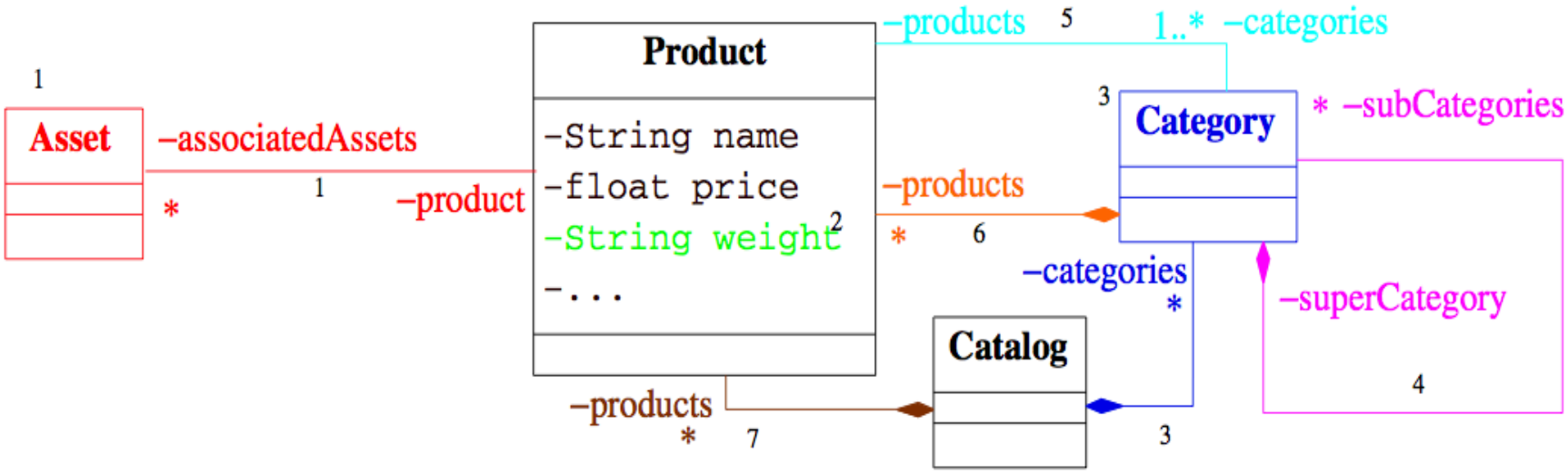
(b) Storefront instance

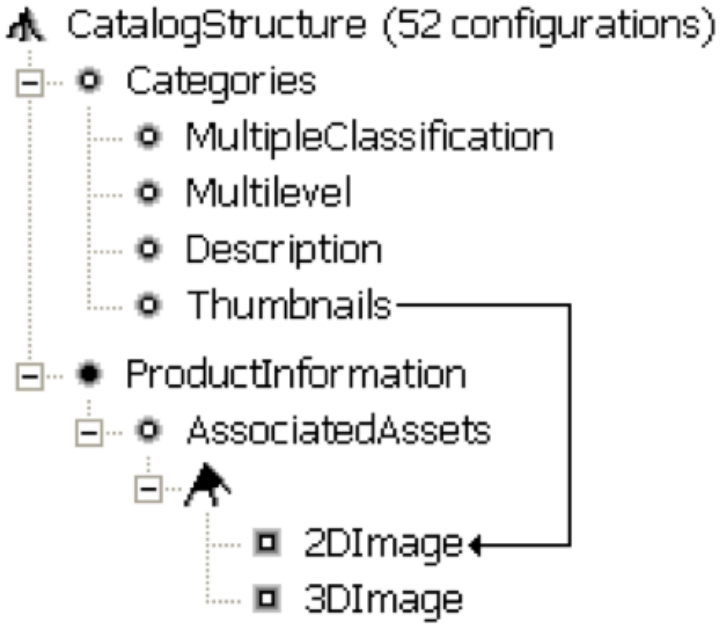
[Czarnecki et al. (GPCE'05)]

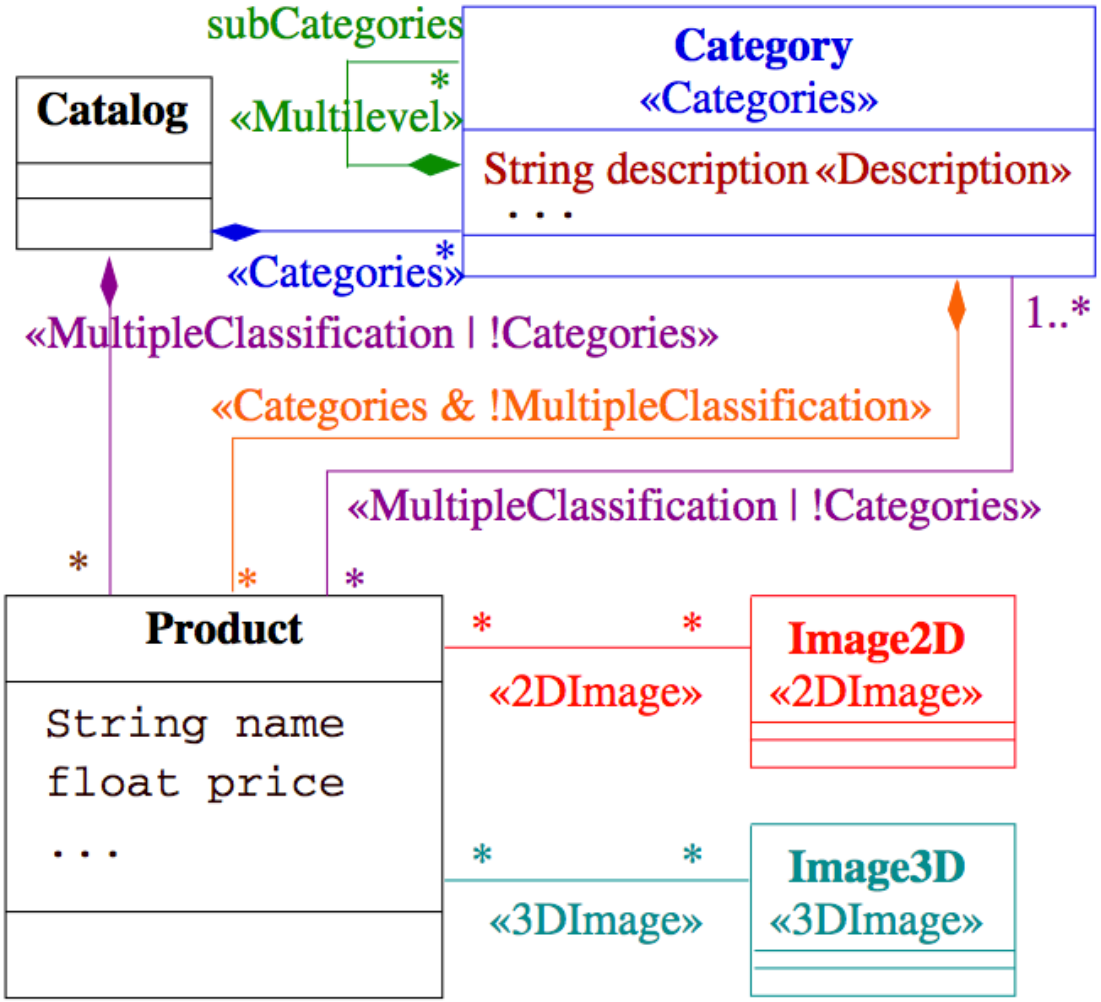


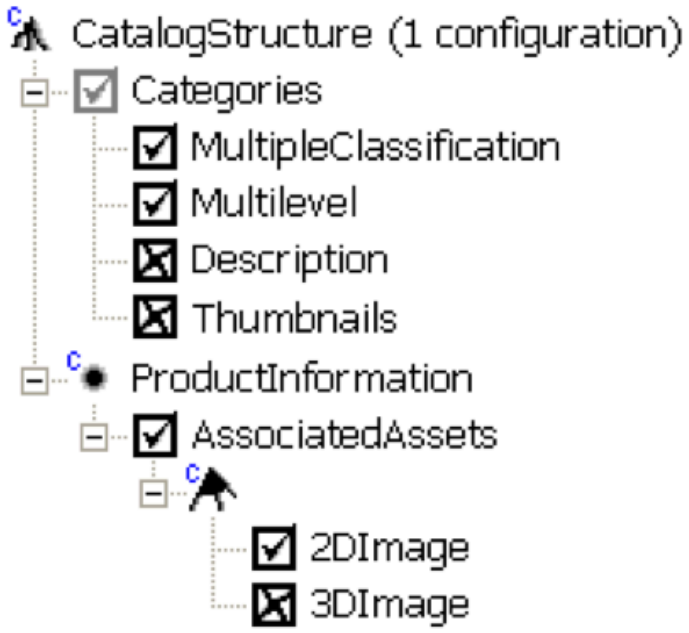
Presence conditions:

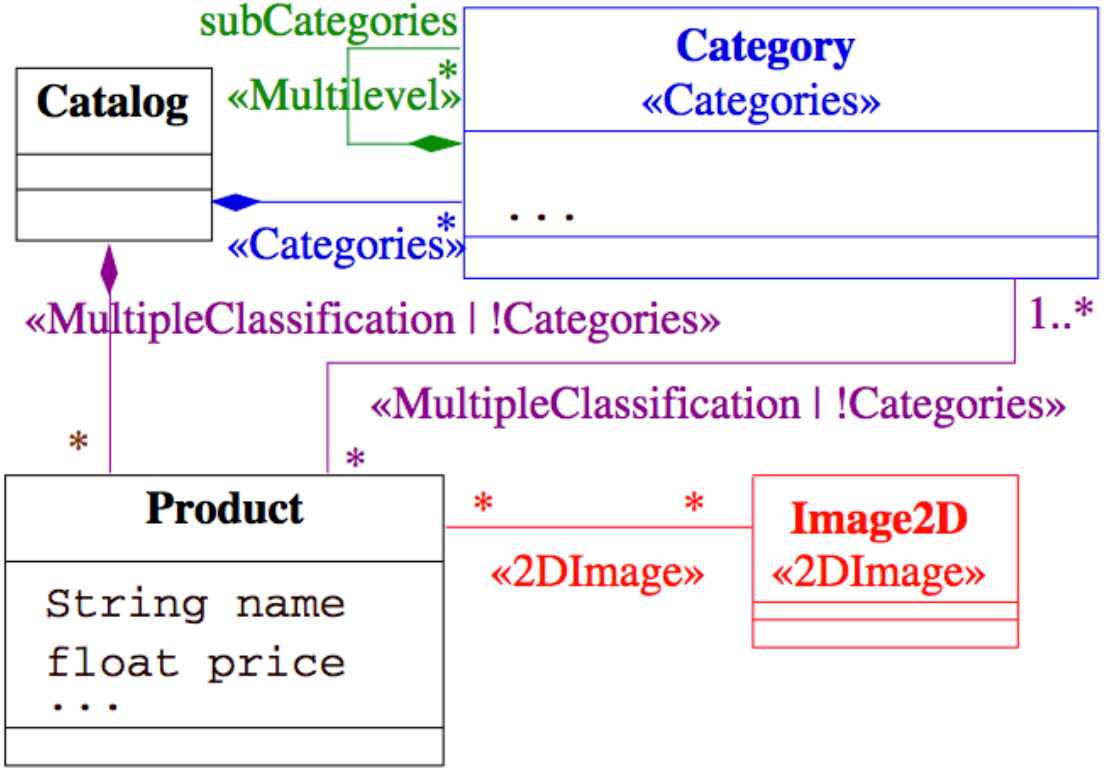
true		MultiLevel	4
AssociatedAssets	1	MultipleClassification	5
PhysicalGoods	2	Categories & !MultipleClassification	6
Categories	3	MultipleClassification | !Categories	7



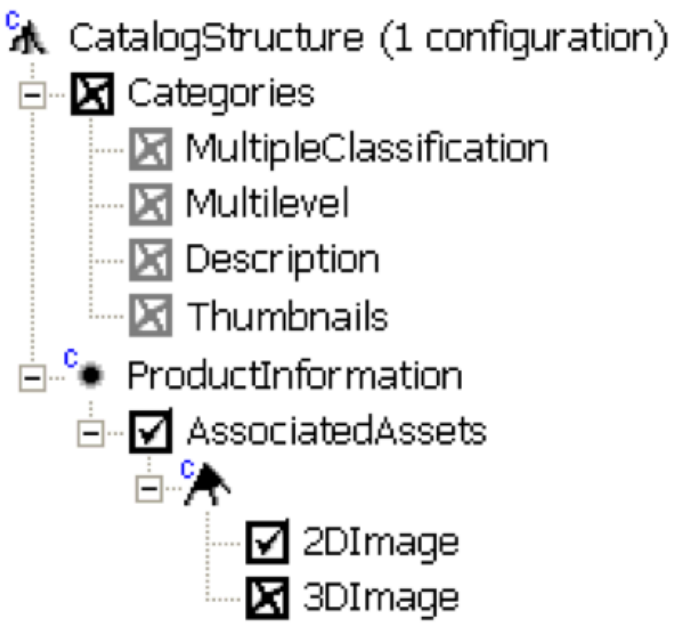

CatalogStructure (52 configurations)
Categories
MultipleClassification
Multilevel
Description
Thumbnails
ProductInformation
AssociatedAssets
2DImage
3DImage

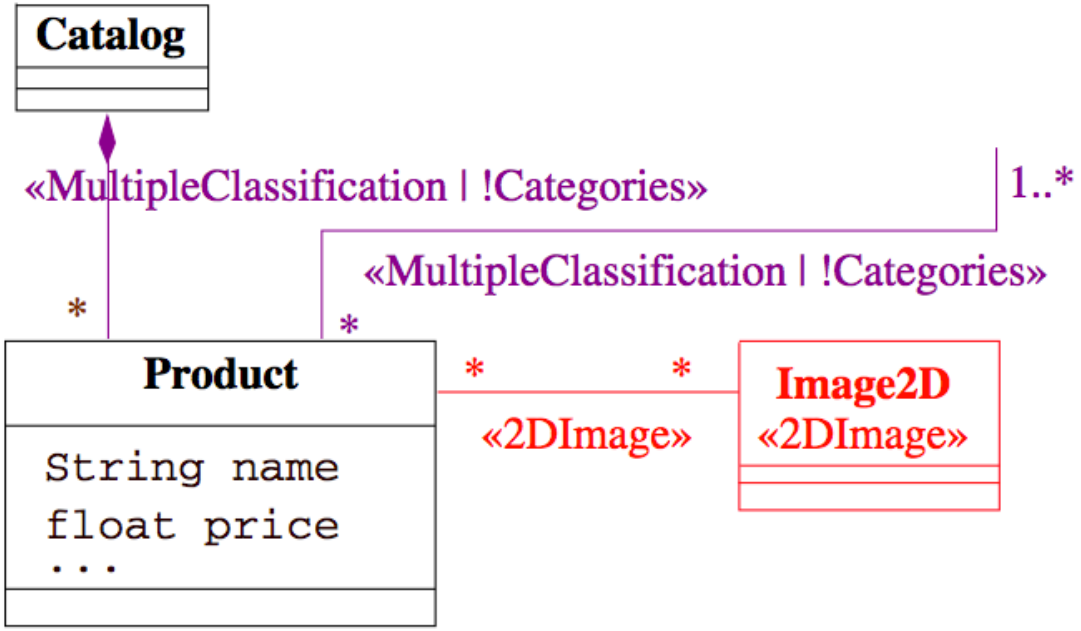

subCategories
Catalog
«Multilevel»
Category
«Categories»
String description «Description»
. . .
«Categories»
«MultipleClassification | !Categories»
1..*
«Categories & !MultipleClassification»
«MultipleClassification | !Categories»
Product
String name
float price
. . .
«2DImage»
Image2D
«2DImage»
«3DImage»
Image3D
«3DImage»


CatalogStructure (1 configuration)
Categories
MultipleClassification
Multilevel
Description
Thumbnails
ProductInformation
AssociatedAssets
2DImage
3DImage

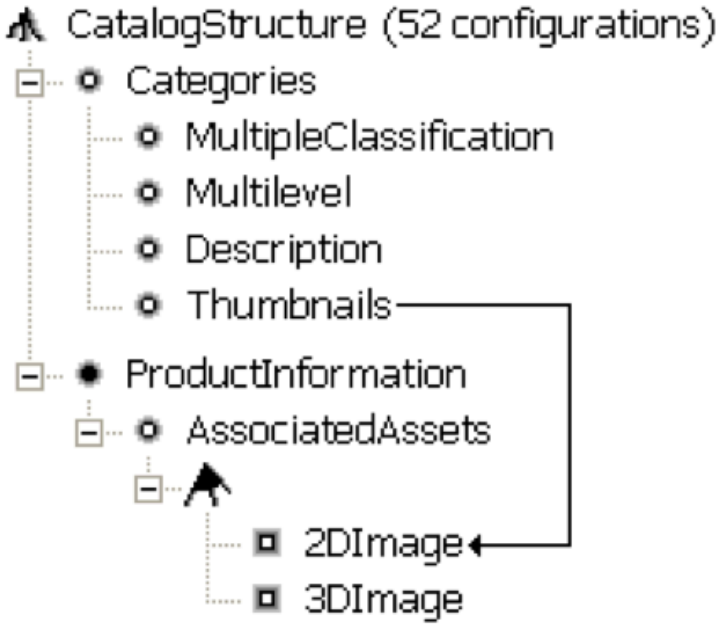

subCategories
Catalog
«Multilevel»
Category
«Categories»
...
«Categories»
«MultipleClassification | !Categories»
1..*
«MultipleClassification | !Categories»
Product
String name
float price
...
«2DImage»
Image2D
«2DImage»

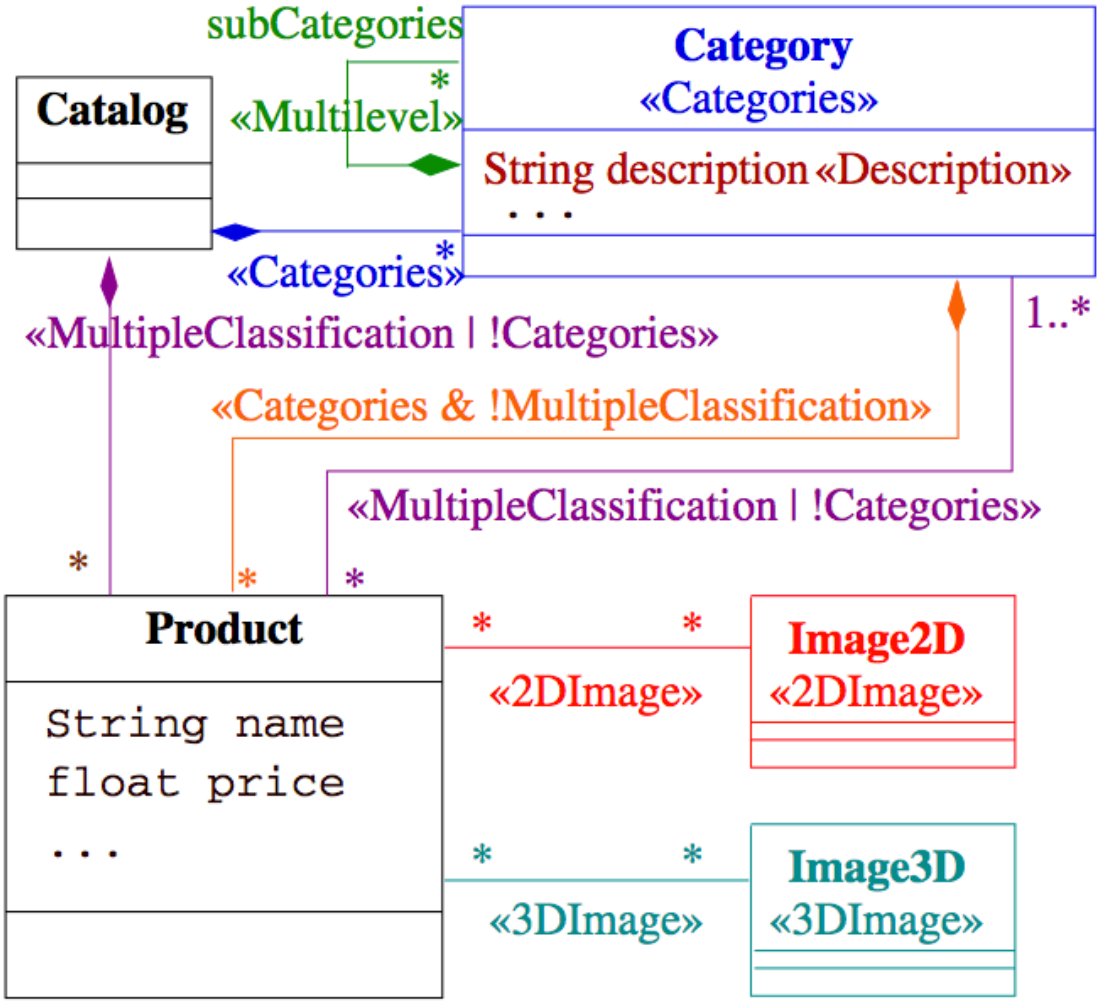
Ooops


CatalogStructure (1 configuration)
Categories
MultipleClassification
Multilevel
Description
Thumbnails
ProductInformation
AssociatedAssets
2DImage
3DImage


Catalog
«MultipleClassification | !Categories»
1..*
«MultipleClassification | !Categories»
*
*
Product
String name
float price
...
*
*
«2DImage»
Image2D
«2DImage»

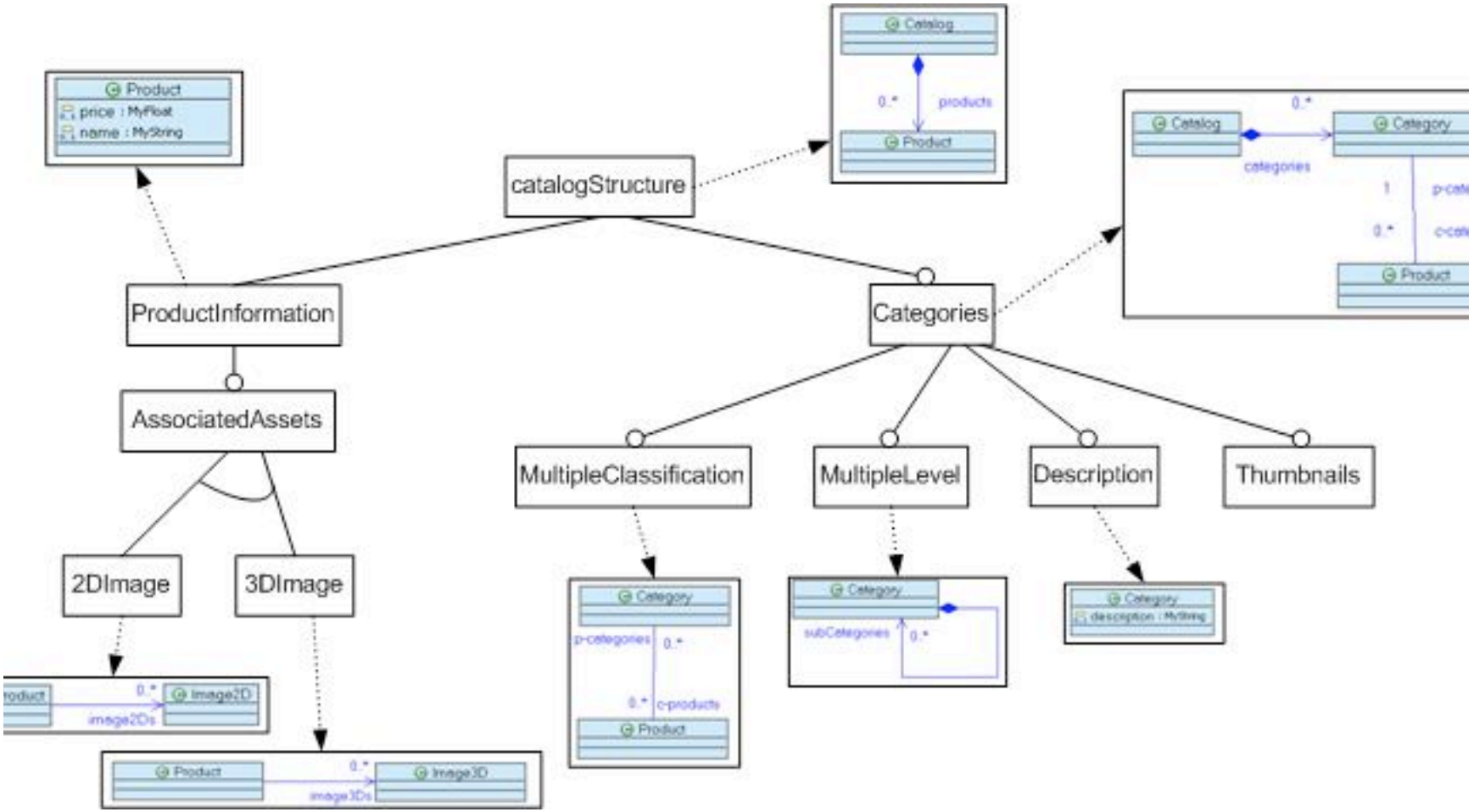
Safe composition? No!



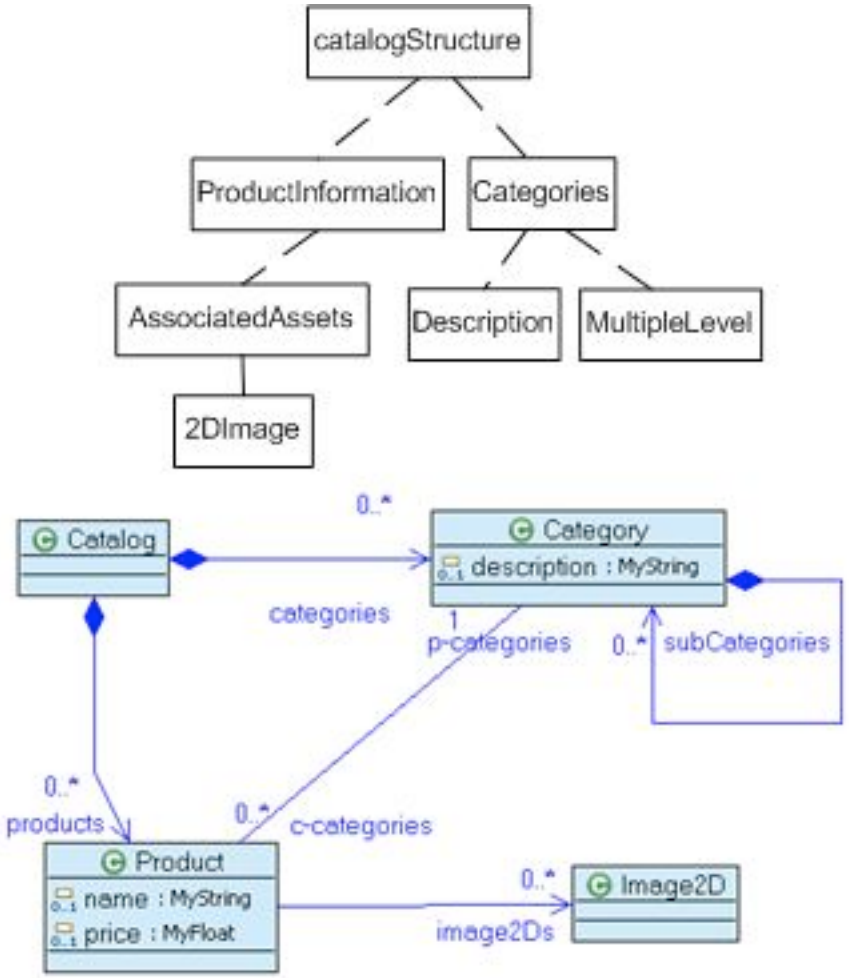


[Perrouin et al. (SPLC'08)]

Another approach



Composition

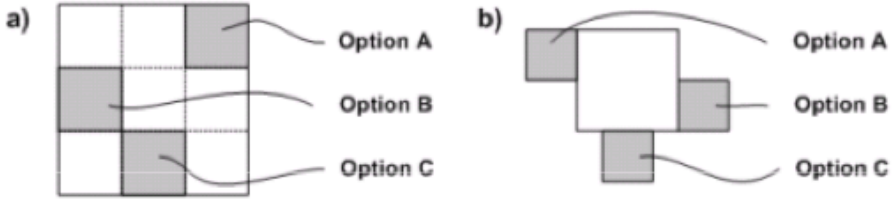


[Voelter et al., SPLC'07)]

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 perceptional 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)

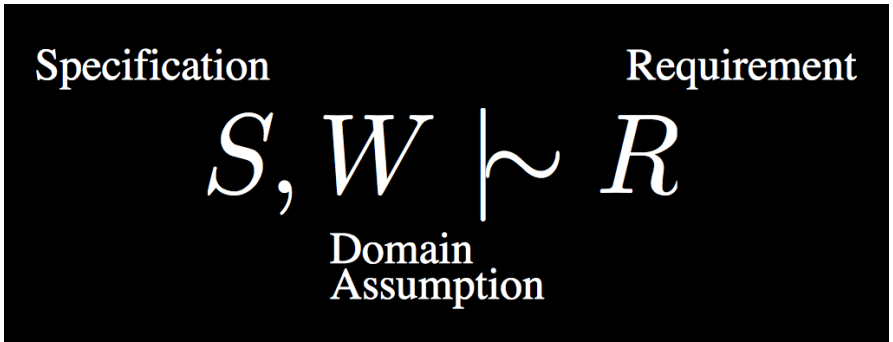
- **Features** are a convenient way to think about commonalities and differences between family members
  - e.g. "MP3 player" – some phones have it, some don't
- Features are usually **coarse-grained** abstractions
  - Abstract from detailed requirements
    - e.g. "when in shuffle mode, the player selects 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)

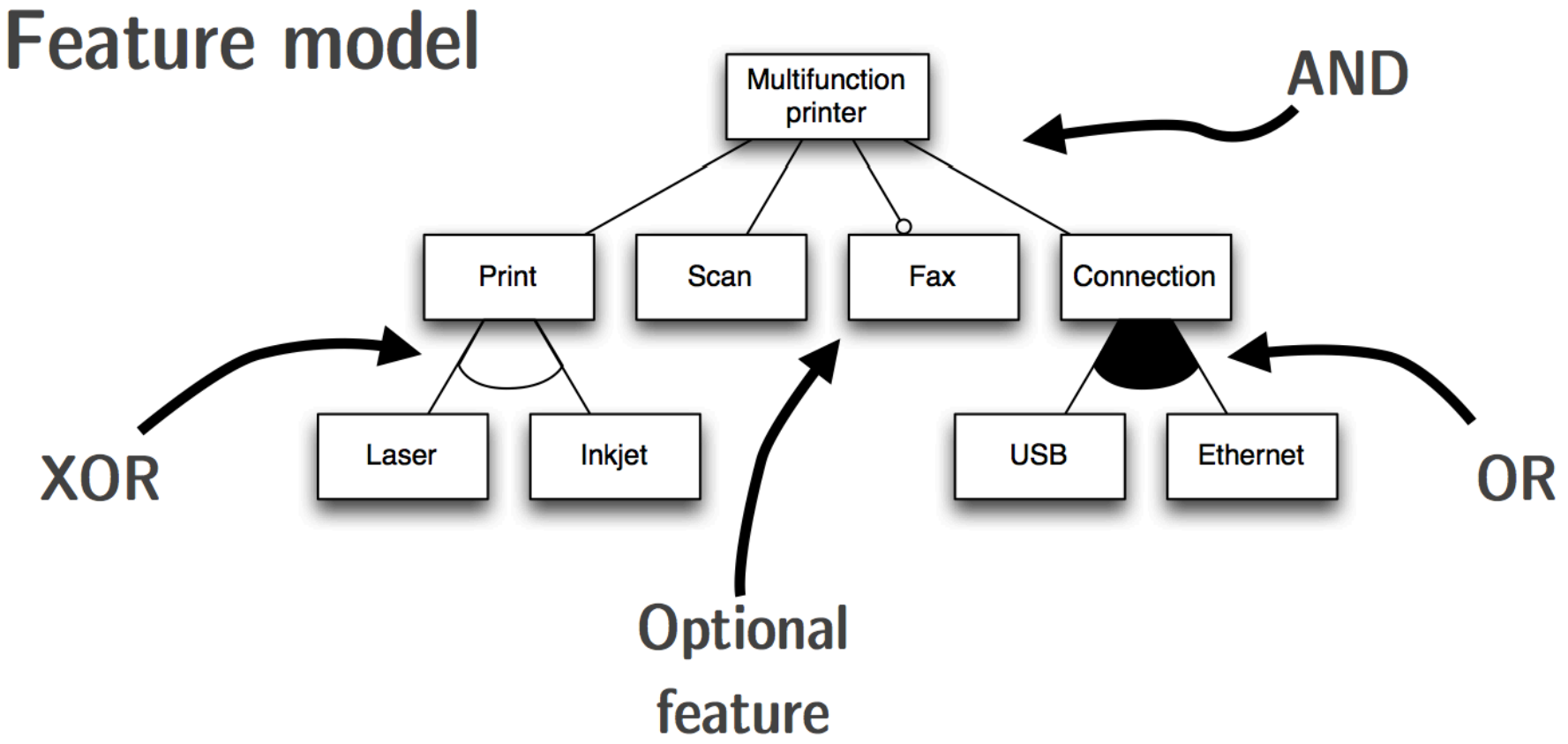
- *[Kang et al.]* "a prominent or distinctive **user-visible aspect, quality** or **characteristic** of a software system or systems"
- *[Kang et al.]* "distinctively identifiable **functional abstractions** that must be implemented, tested, delivered, and maintained"
- *[Eisenecker and Czarnecki]*. "**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*'

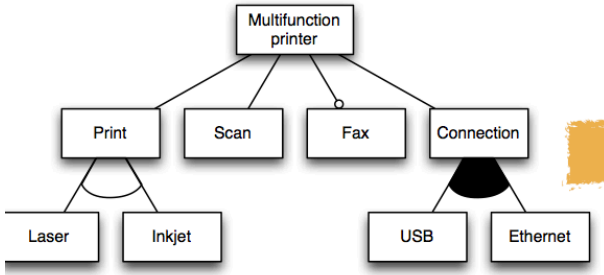


Variability Modelling



Feature Models Semantics

d



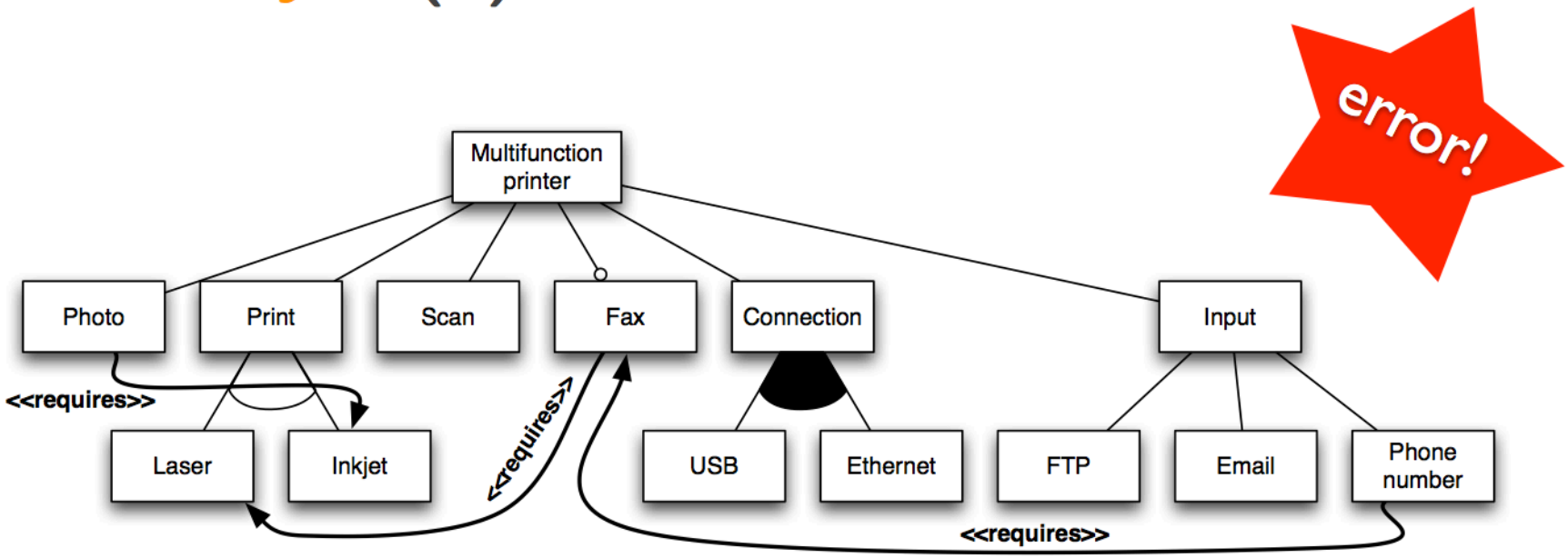
M(d)

{ {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} }

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

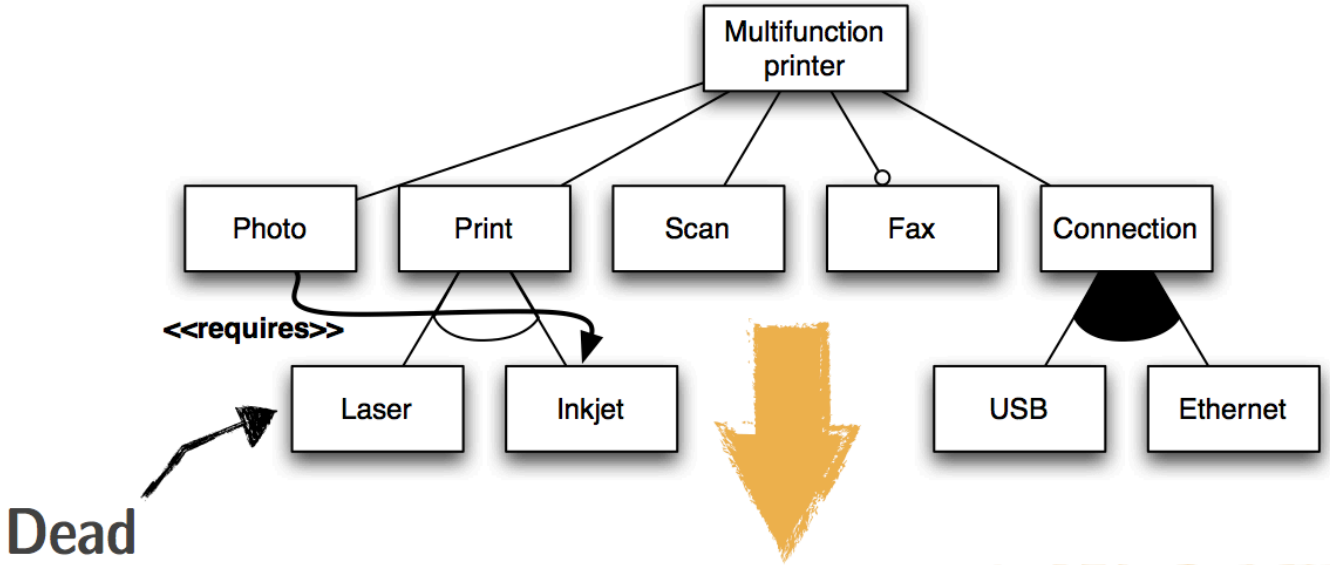
SPL Automations

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



SPL Automations

Dead features F \ ∪ M(d)?



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

SPL Automations

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

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

Product checking $\{f1,\ldots,fn\} \in M(d)$?

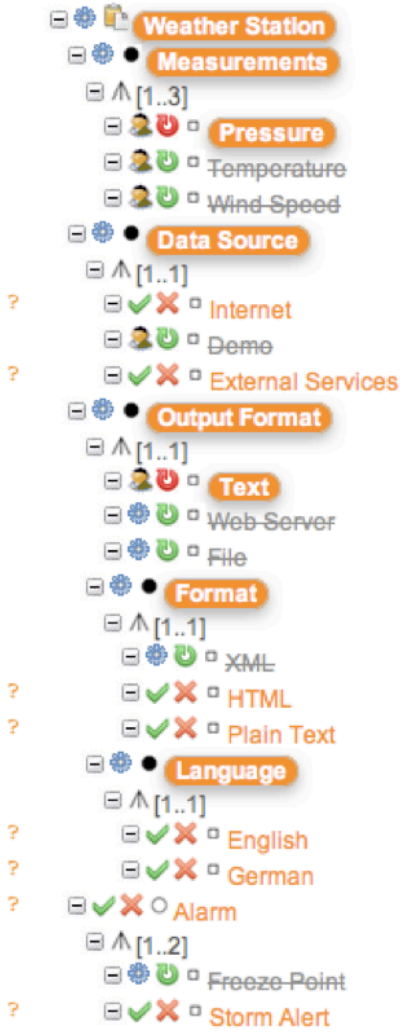
Product counting $\#M(d)$?

Equivalence $M(d1)=M(d2)$?

...

Interactive Configuration

Weather Station (23 features)



Configuration Steps [reset]

65 %

Step	Decision	#Decisions (cummulative)	#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

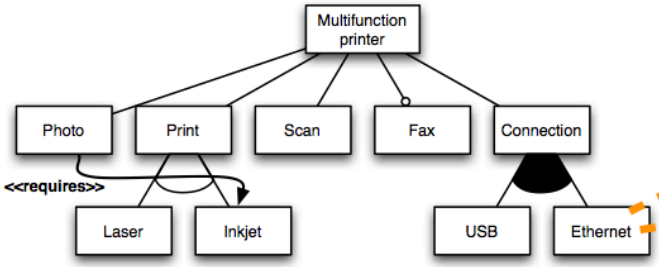
Auto-completion: Less Features | More Features

S.P.L.O.T.
Software Product Lines Online Tools

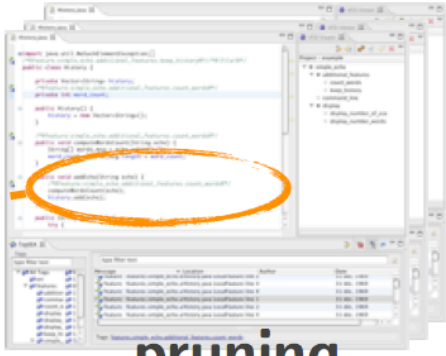
www.splot-research.org

Product Derivation

feature model



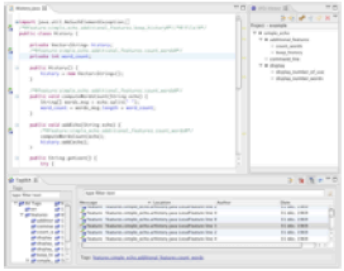
variable model and code assets



pruning,
composition,
weaving,
transformation
...

configuration

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

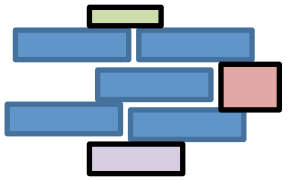


product spec

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