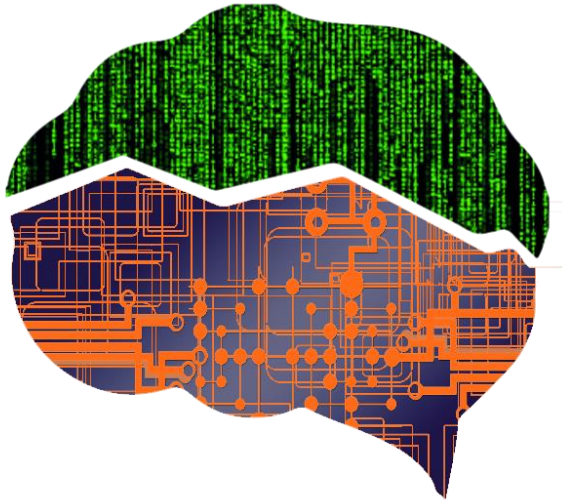
Software Product Line Engineering

Development Process and Variability Modelling

Christian Kästner (Carnegie Mellon University)
Sven Apel (Universität Passau)
Norbert Siegmund (Bauhaus-Universität Weimar)
Gunter Saake (Universität Magdeburg)



Bauhaus-Universität Weimar

Agenda

- Product Lines
- What is a feature?
- Domain engineering vs. application engineering
- Feature modelling

Product Lines

A software product line (SPL) is 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.

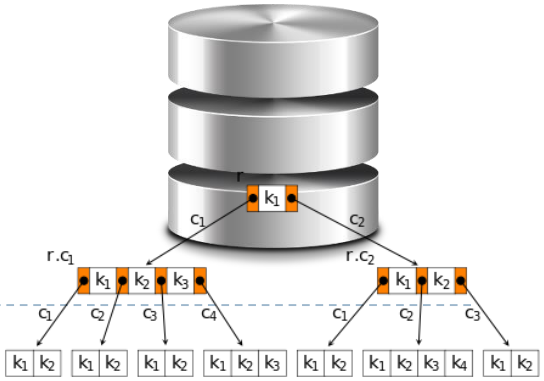
Software Engineering Institute
Carnegie Mellon University

Product Lines

- A set of program variants (software products),
- …that share a common set of functionalities (features),
- ...which are tailored to a common market segment (domain),
- ...with the goal of reusing the commonly shared software artifacts

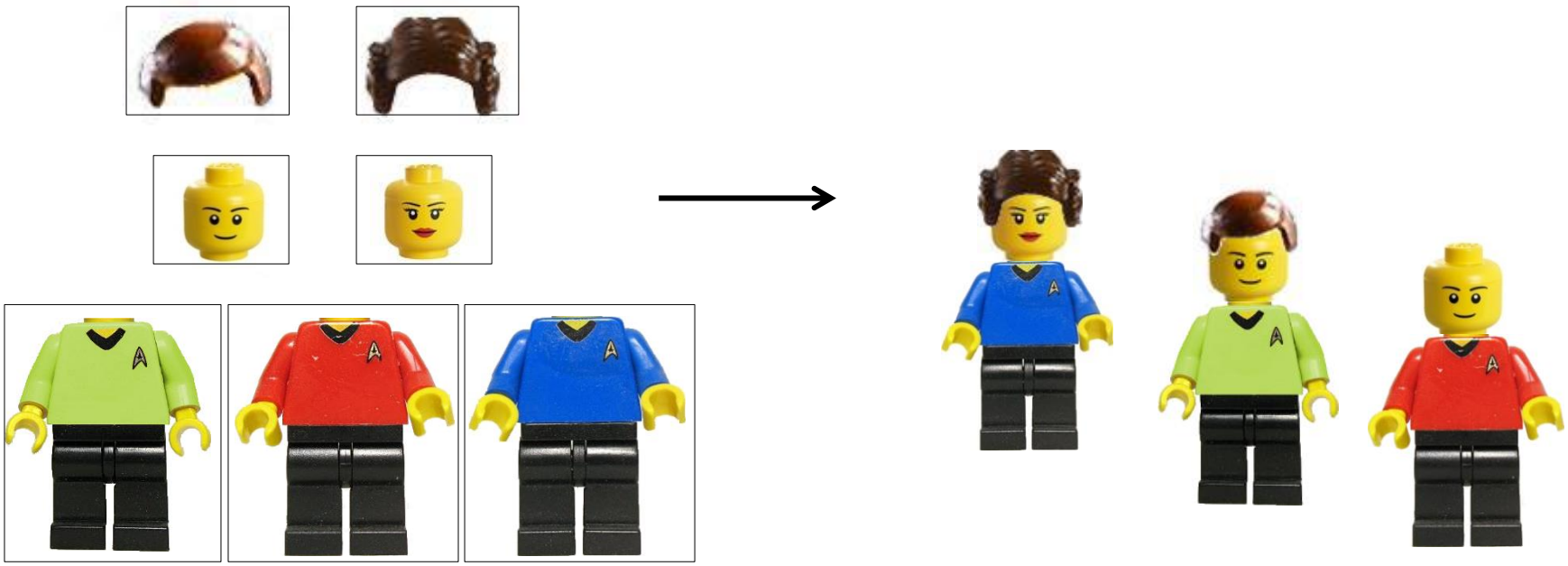
What is a Feature?

- Domain abstraction
- Features represent requirements, similarities and differences of program variants
- A mean to communicate between stakeholder
- A mean to specify program variants
  - Feature selection as input for program-variant construction



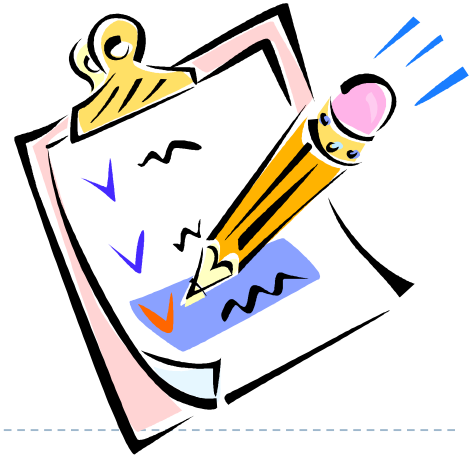
Features vs. Variants

- Features are the basic blocks of a product line (e.g., implemented via components, packages, etc.)
- Feature combinations represent individual variants



Example: Features in Database Systems

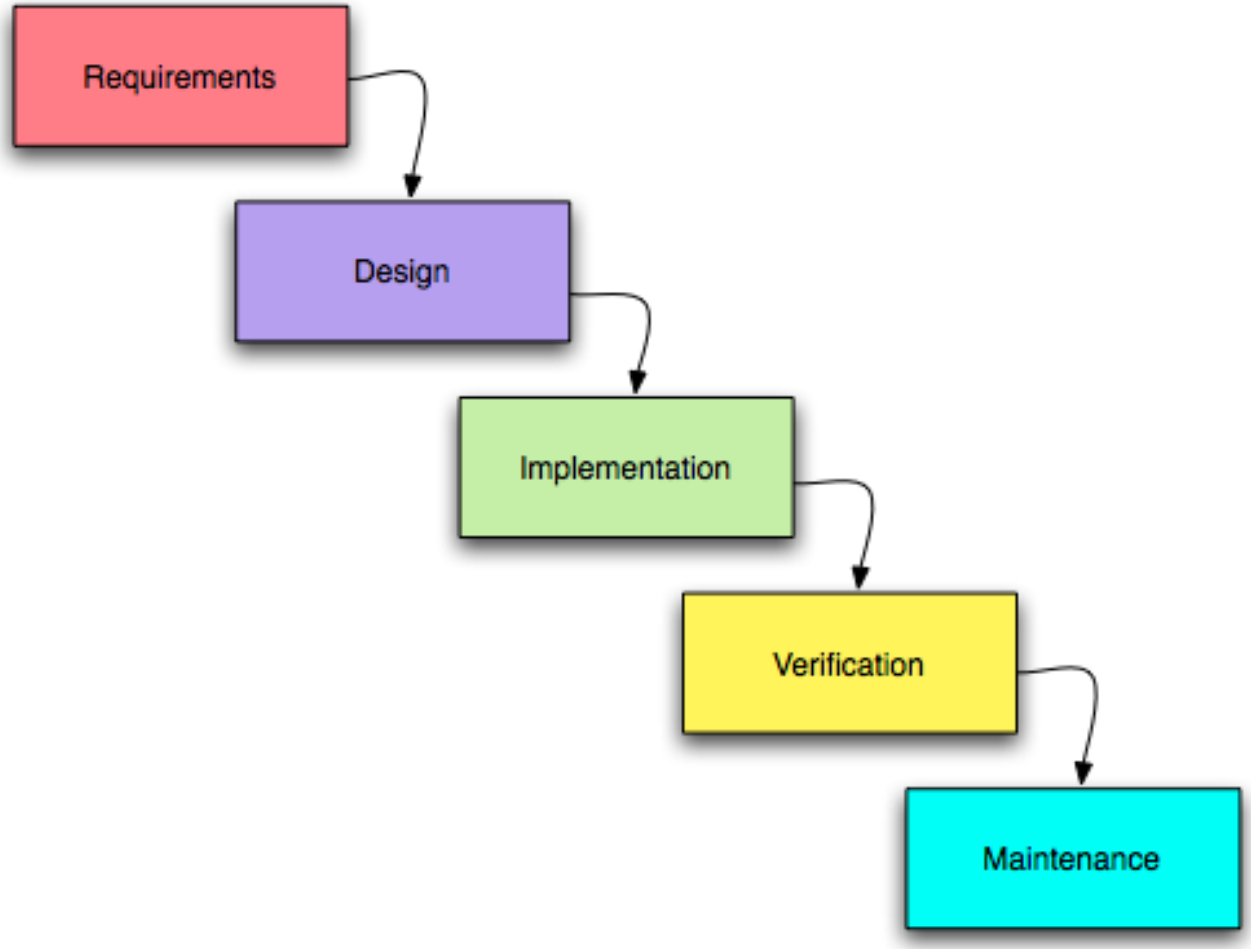
- Transaction Management
- Log & Recovery
- Write Access
- Persistency / In-Memory
- Page replacement strategies LRU / LFU / Clock /...
- Sorting algorithms
- Data types of variable length
- Group by, aggregation
- Windows / Unix / NutOS / TinyOS / …



Development of a Product Line

- Develop a product line instead of a single product
- Product line covers requirements of a whole domain
- Different from the classic software development process and life cycle
- Differences are in
  - Domain Engineering
  - Application Engineering

Classic Software Development Process

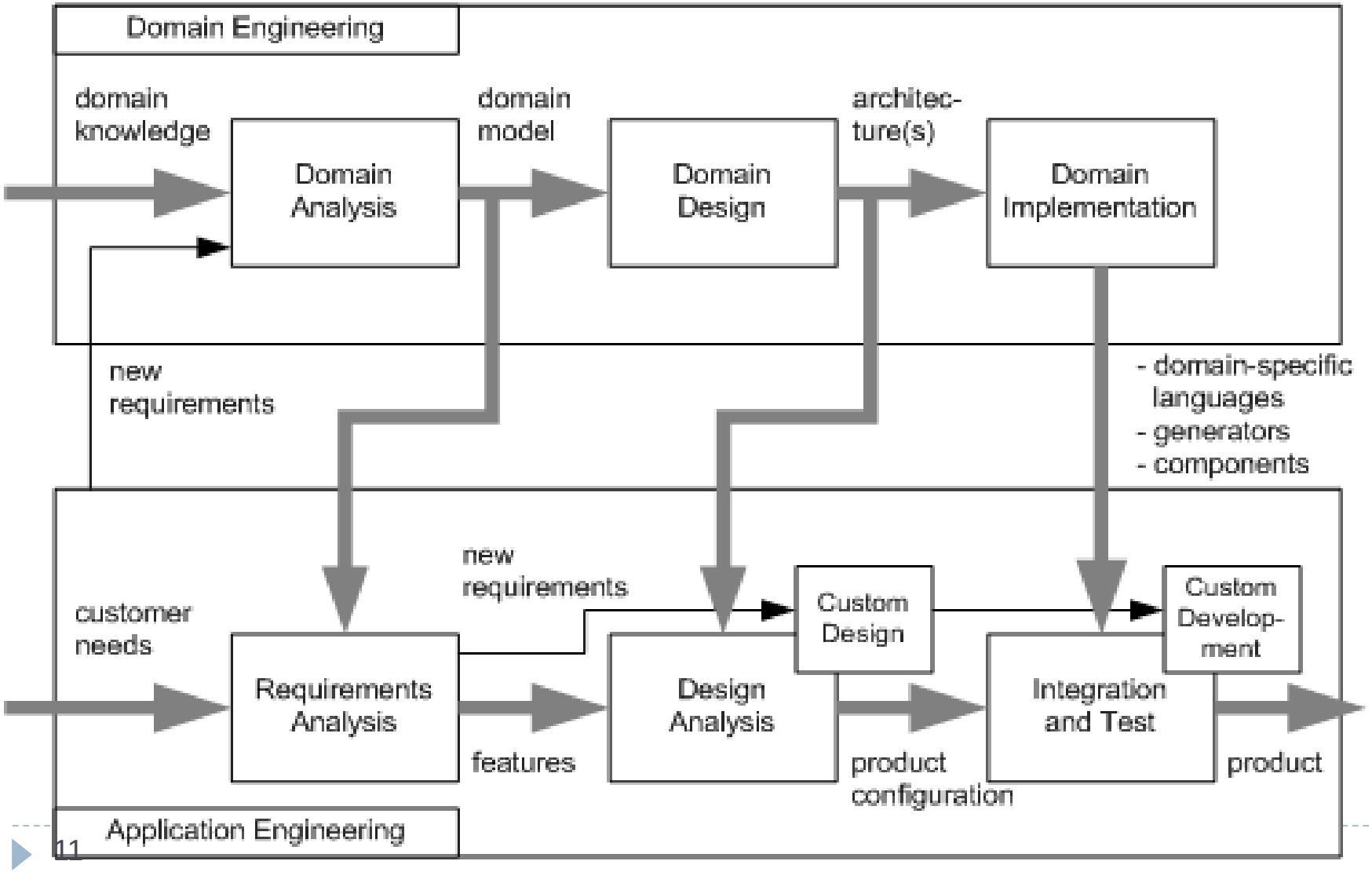


Domain Engineering

[...] 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

Application and Domain Engineering

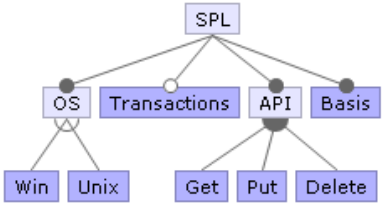


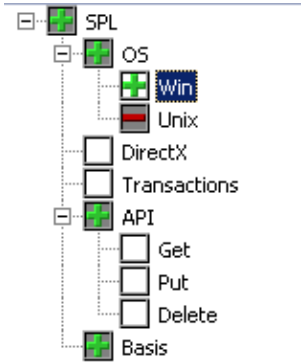
Application and Domain Engineering (simplified)

Domain Eng.

Application Eng.

Which features are provided?

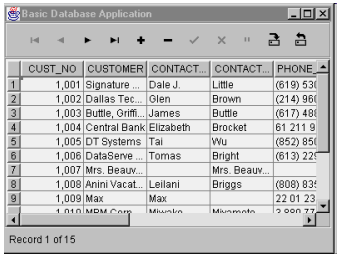




Which features are required?

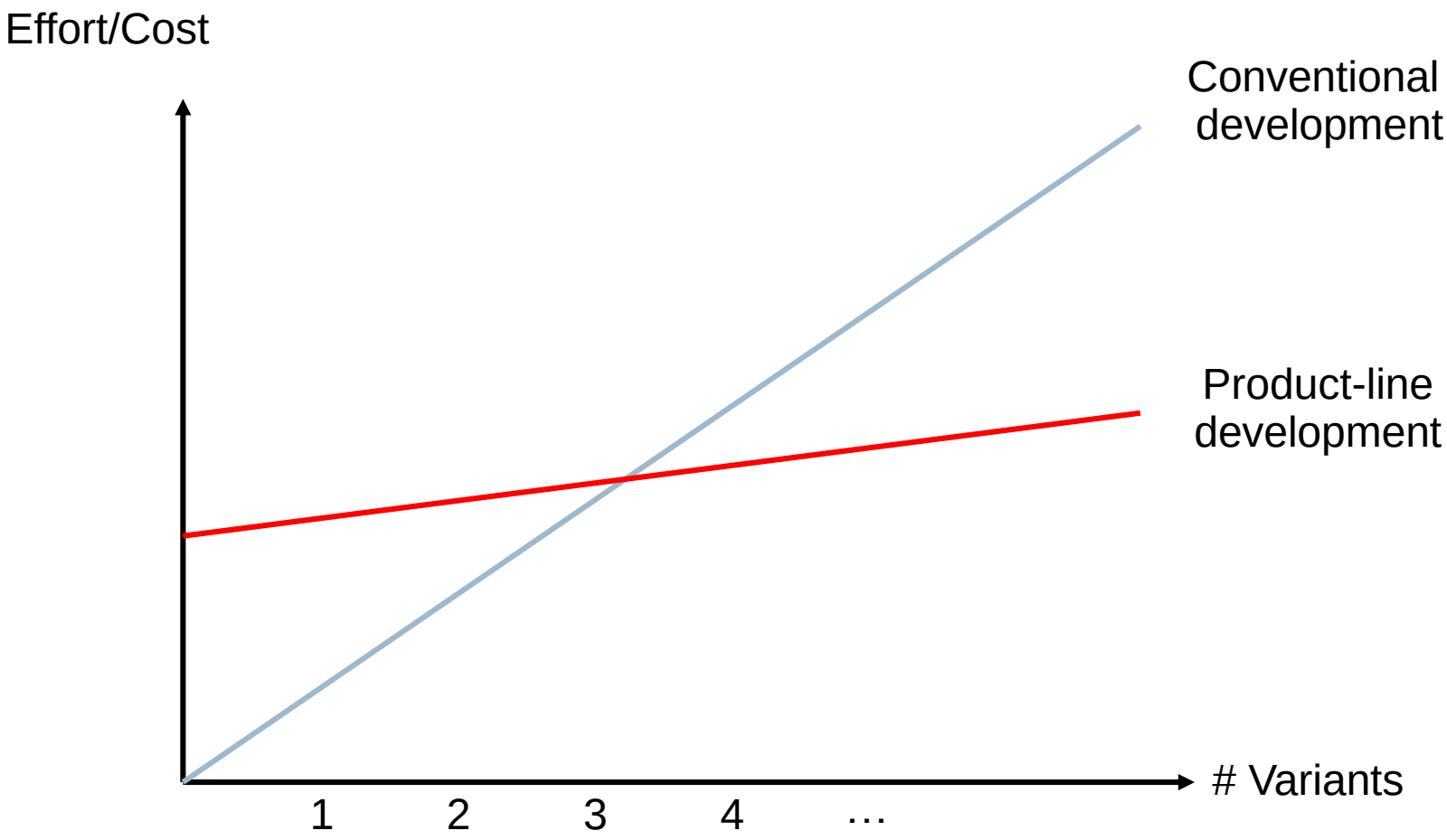
Feature implementations

Variant construction



Final product

Estimated Development Effort

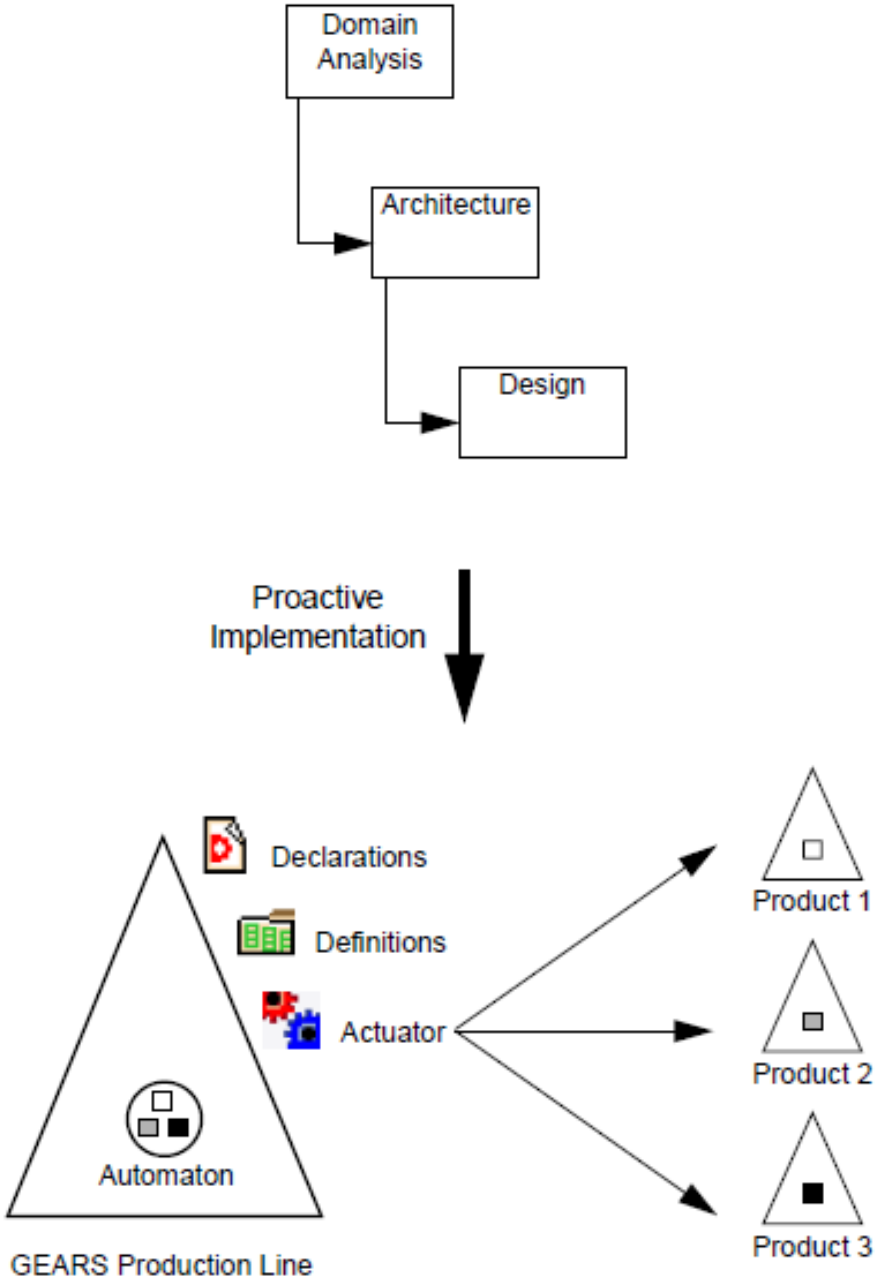


Approaches for Product Line Adoption

- Three approaches to create or adopt product lines
  - Proactive approach
  - Reactive approach
  - Extractive approach
- Applicable for all implementation techniques
- Choice depends on several factors (cost, risk, chances, ...)

Proactive Approach

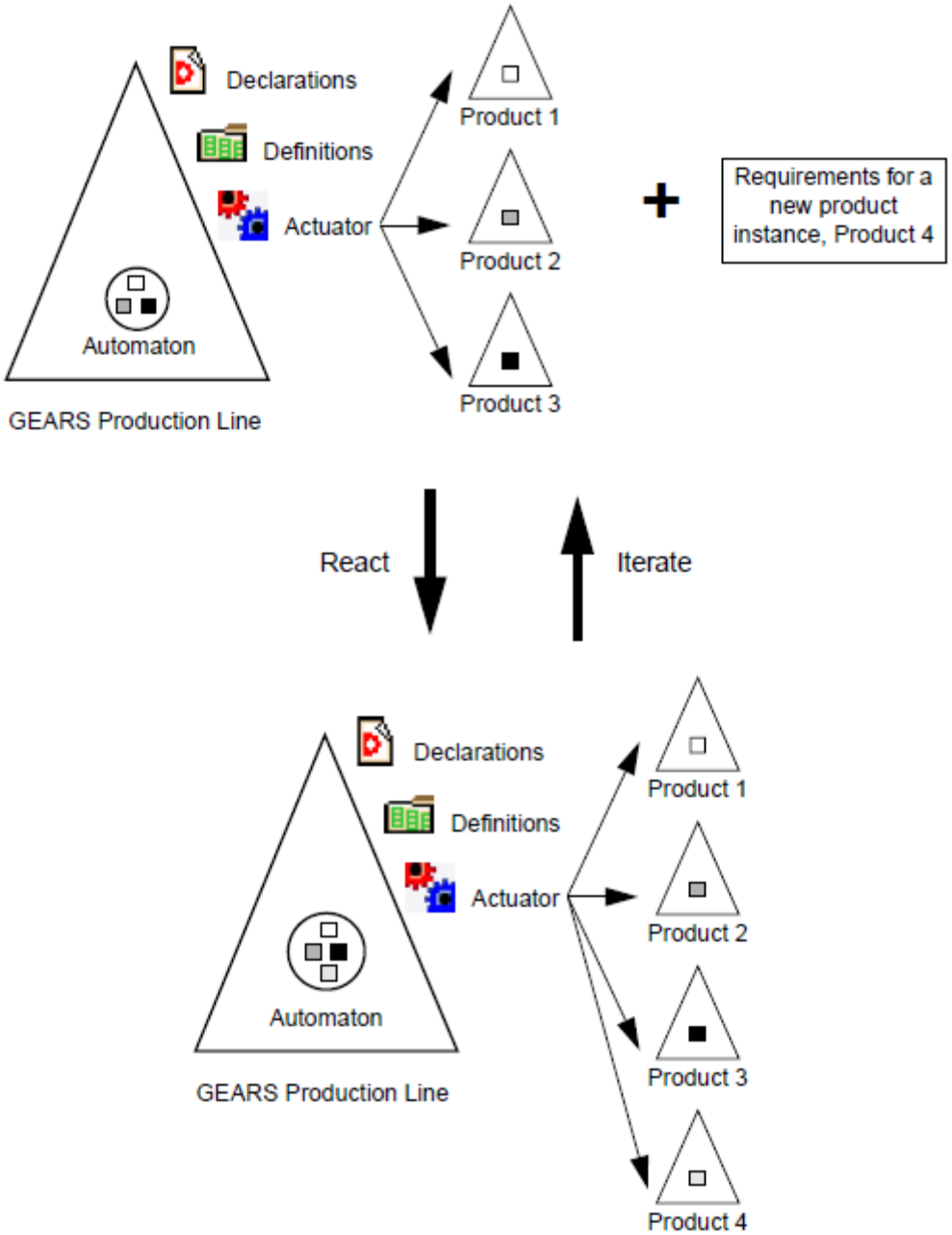
- Development a product line from scratch; as discussed so far
- Require a full domain analysis at the beginning
- Might disrupt existing development of software for months till the product lines has an equal set of functional as existing products
- High initial costs and high risk
- Beneficial if requirements are known and well defined



[Krueger 2002]

Reactive Approach

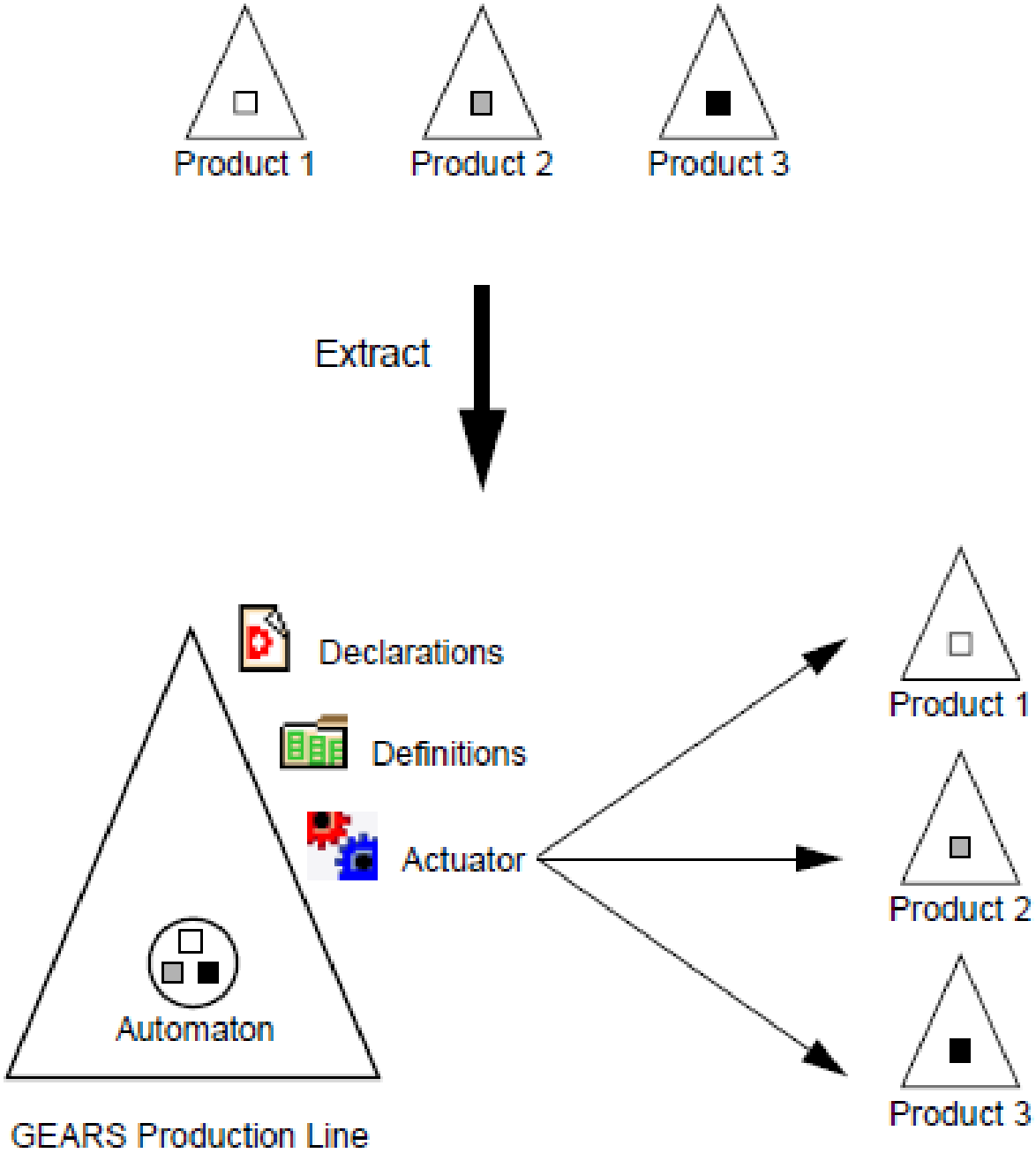
- Similar to the spiral development-process model or extreme programming
- Starts by implementing few program variants; incrementally adds more variants
- Beneficial when required variants are not known in advance and in case of unforeseen changes in requirements
- Smaller project size, low starting cost, less resources required, and faster results
- Requires substantial refactoring in later phases



[Krueger 2002]

Extractive Approach

- Uses existing products as base
- Extracts features from these products to enable the construction of new program variants
- Beneficial if a quick adoption of product line techniques based on traditionally developed programs is needed
- Relative low resource requirements and risk
- Requires sophisticated tools, languages, and expert knowledge, since the separation of an existing application is a complex task



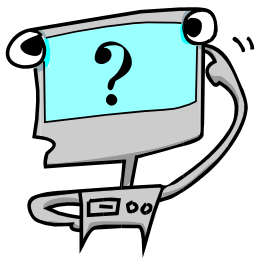
[Krueger 2002]

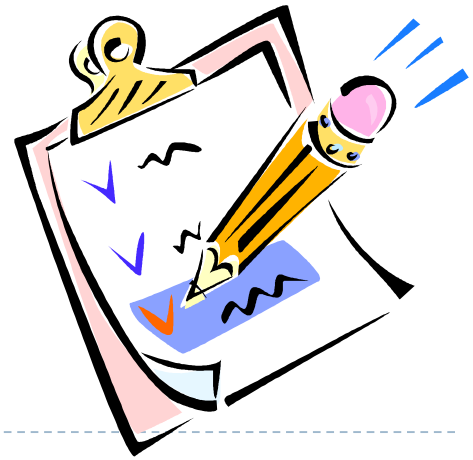
Feature Modelling

- Describes the features of a domain
- To visualize and communicate
- A feature model describes
  - The fundamental abstractions of a domain and their relationships
  - The set of program variants of a product line
- A feature diagram visualizes features and their relationships

Valid Feature Selection?

- Transaction Management
- Log & Recovery
- Write Access
- Persistency / In-Memory
- Page replacement strategies LRU / LFU / Clock /...
- Sorting algorithms
- Data types of variable length
- Group by, aggregation
- Windows / Unix / NutOS / TinyOS / ...





karoma - Poderosa

File Edit Console Tools Window Plug-in Help

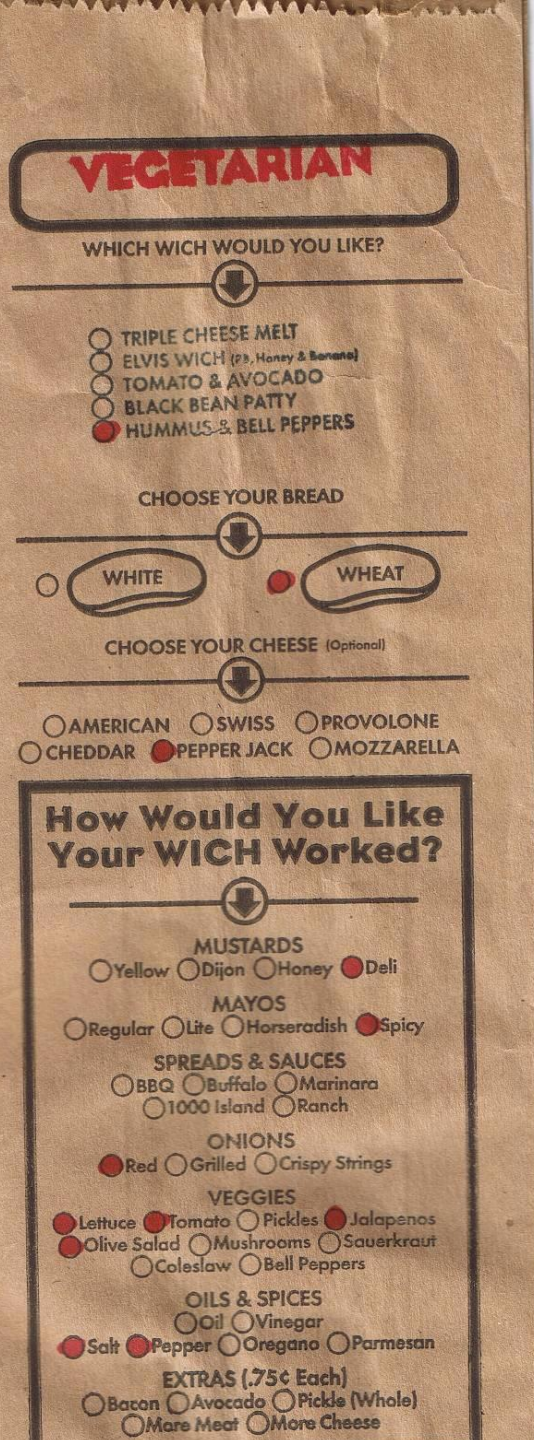
Line feed CR Encoding iso-8859-1 generic

1 karoma 2 karoma

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

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

                <Select>    < Exit >    < Help >
```



Feature Model: Example

- Features: *Basis*, *Txn*, *Write*, *Win*, *Unix*
- Constraints:
  - *Basis* must always be selected and requires the selection of either *Win* or *Unix*
  - *Win* must not be selected in combination with *Unix*
  - If *Txn* is selected, *Write* must be selected, too
- How many variants are valid?

  { *Basis*, *Win* },

  { *Basis*, *Unix* },

  { *Basis*, *Win*, *Write* },

  { *Basis*, *Unix*, *Write* },

  { *Basis*, *Win*, *Write*, *Txn* },

- 2^4 { *Basis*, *Unix*, *Write*, *Txn* }

Feature Model as Propositional Formula

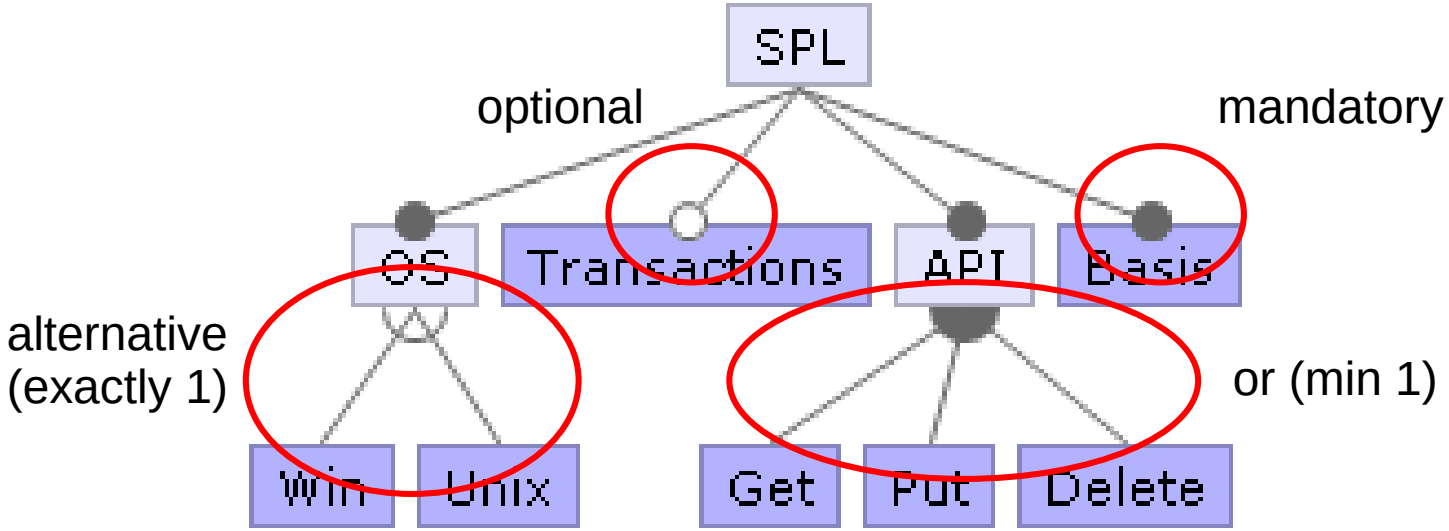
- Variable for each feature (true if selected)
- Formula describes feature model
- Formula yields true for a valid feature selection

$$Basis \wedge (Unix \vee Win) \wedge \neg(Unix \wedge Win)$$
$$\wedge (Txn \Rightarrow Write)$$

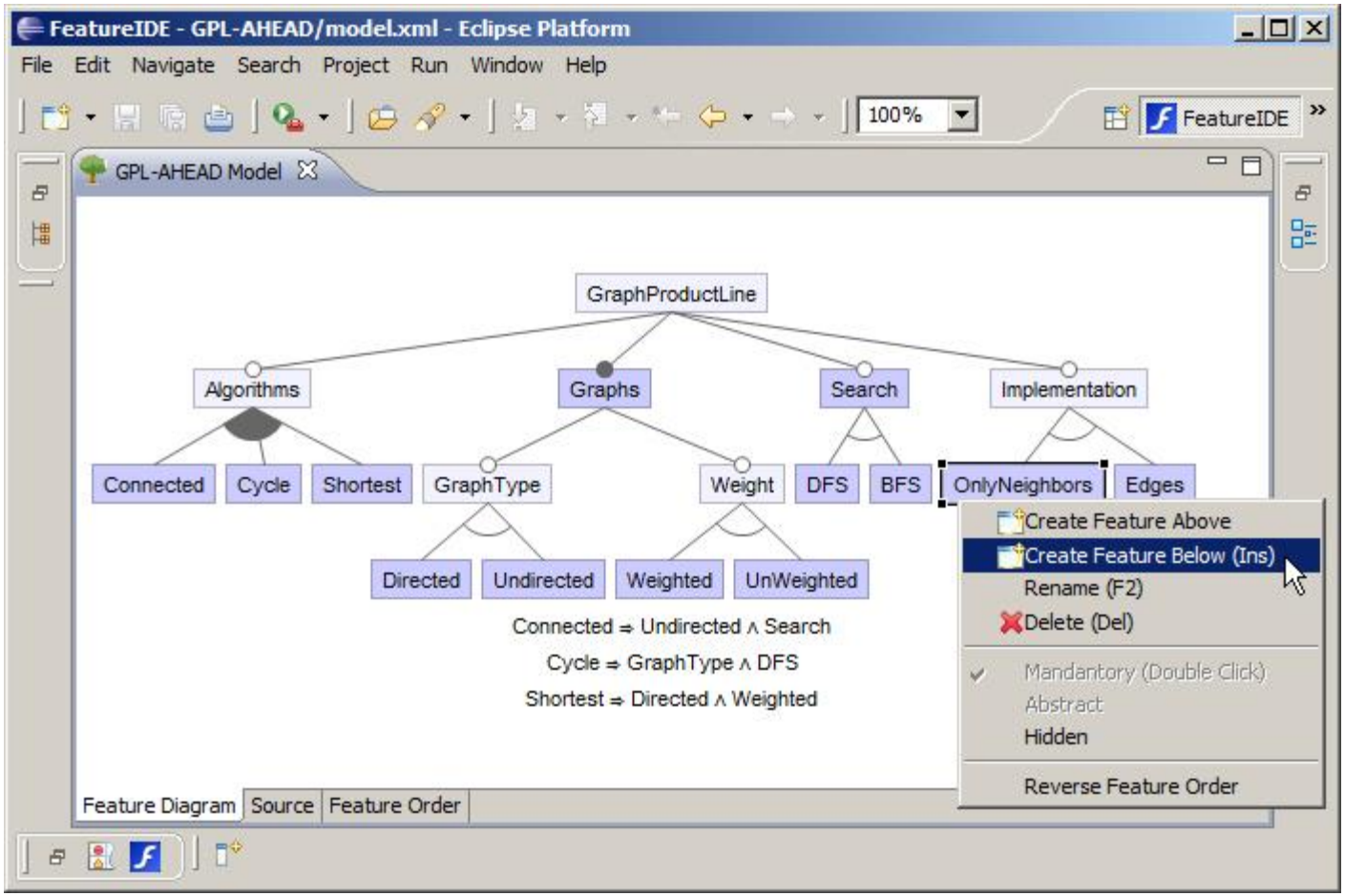
- Enables automated checking and enumeration of all valid program variants (satisfiability solver, binary decision diagram, …)

Feature Diagram

- Visualization of feature models
- Hierarchical structure
- Types of child nodes: optional, mandatory, OR, alternative
- Features in leaf nodes

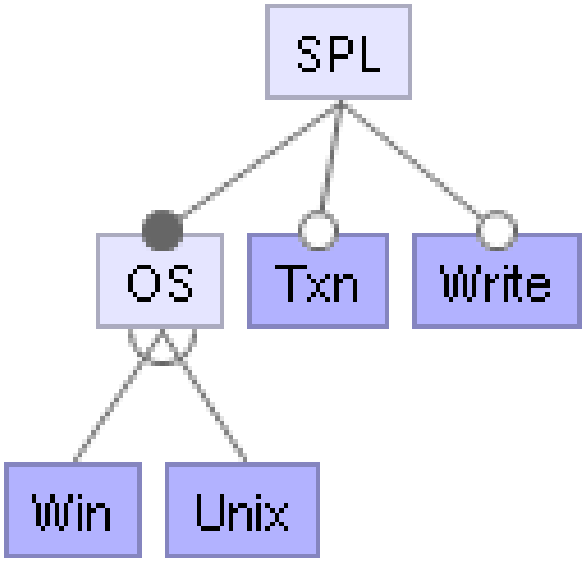


FeatureIDE


FeatureIDE - GPL-AHEAD/model.xml - Eclipse Platform
File Edit Navigate Search Project Run Window Help
100%
FeatureIDE
GPL-AHEAD Model
GraphProductLine
Algorithms
Graphs
Search
Implementation
Connected
Cycle
Shortest
GraphType
Weight
DFS
BFS
OnlyNeighbors
Edges
Directed
Undirected
Weighted
UnWeighted
Connected ⇒ Undirected ∧ Search
Cycle ⇒ GraphType ∧ DFS
Shortest ⇒ Directed ∧ Weighted
Create Feature Above
Create Feature Below (Ins)
Rename (F2)
Delete (Del)
Mandantory (Double Click)
Abstract
Hidden
Reverse Feature Order
Feature Diagram
Source
Feature Order

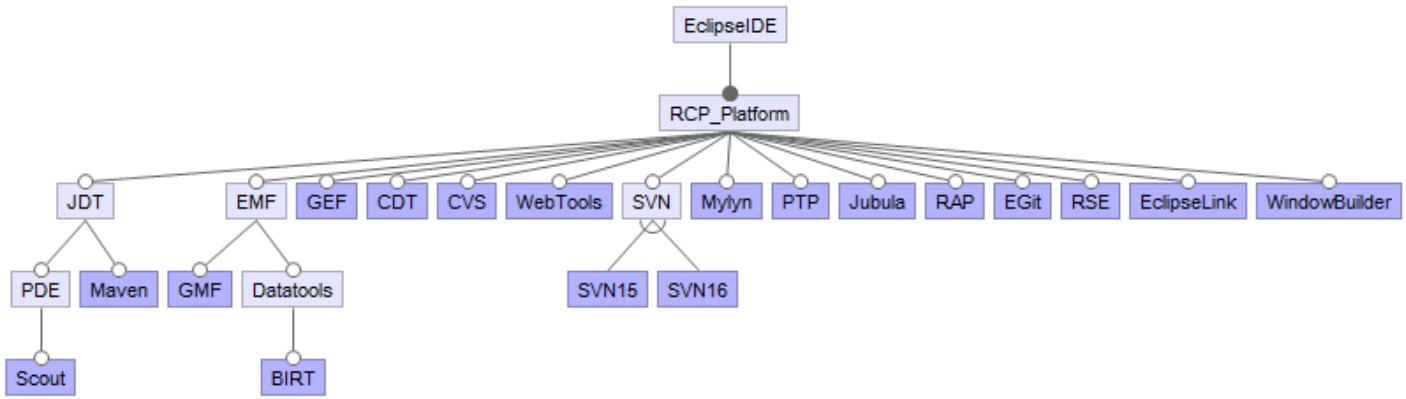
Feature Diagram vs. Formulas

- Improved readability compared to formulas ("management compatible")
- Less flexible → extra formulas possible and required
- Translation „diagram → formula" can be automated



$Txn \Rightarrow Write$

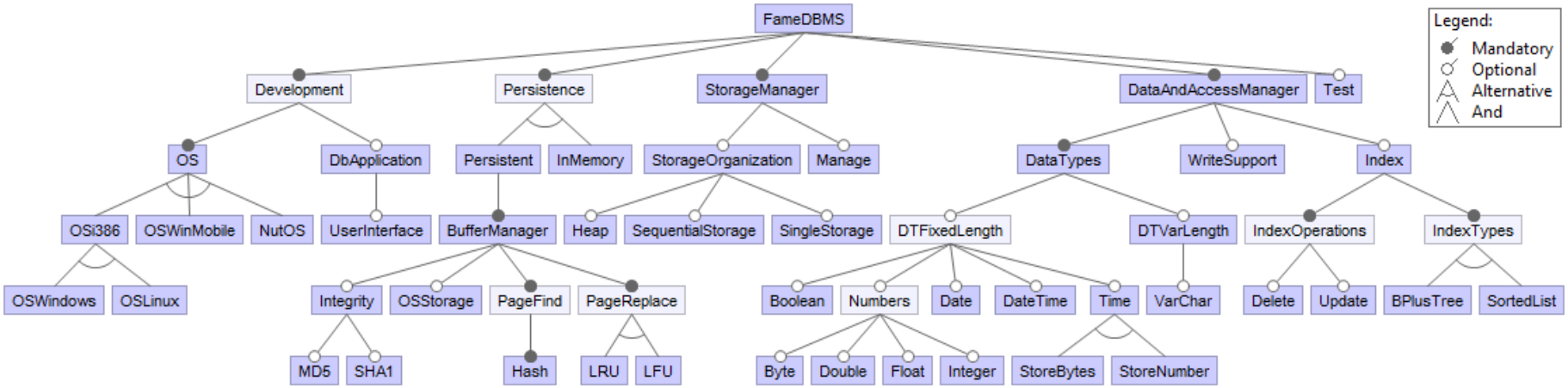
More…



$GMF \Rightarrow GEF$

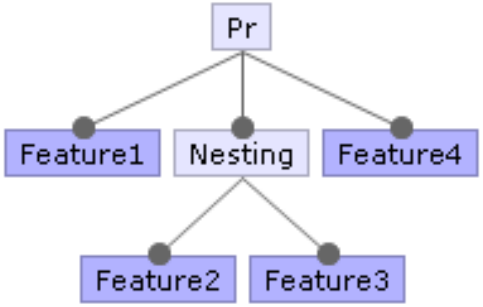
$Maven \Rightarrow EMF$

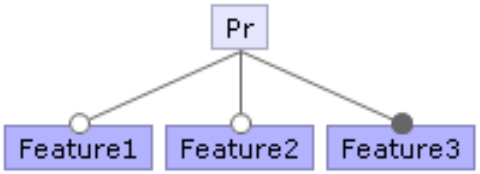
$BIRT \Rightarrow GEF \wedge JDT \wedge PDE$



Feature Models as Grammars

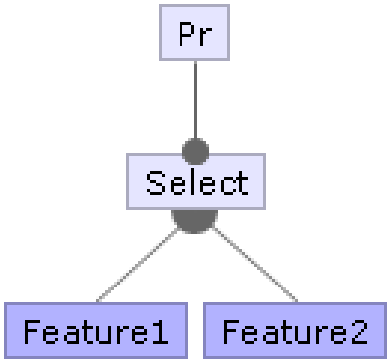
- Word in grammar is valid configuration






```
Pr ::= Feature1 Nesting Feature4
Nesting ::= Feature2 Feature3
```

```
Pr ::= [Feature1] [Feature2] Feature3
```

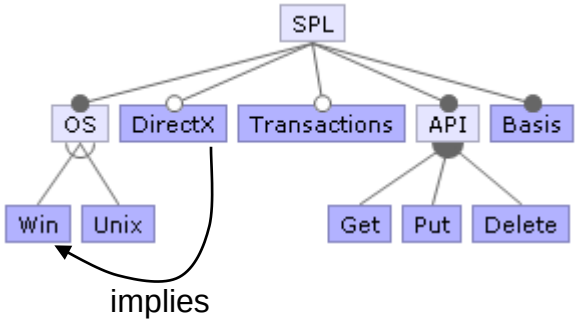


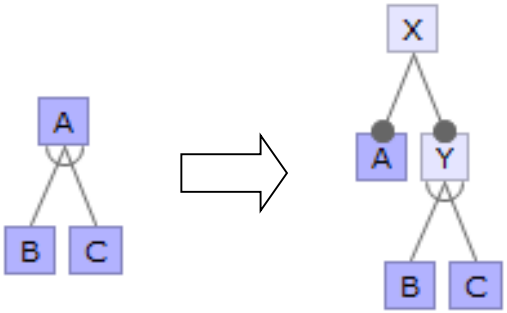

```
Pr ::= Select+
Select ::= Feature1 | Feature2
```

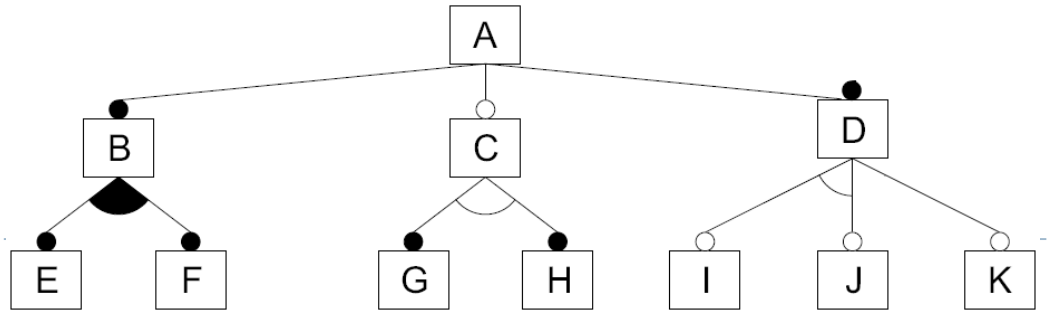

HowTo: `http://www.cs.utexas.edu/users/dsb/fopdocs/guidsl.html`

Feature Diagram – Variants

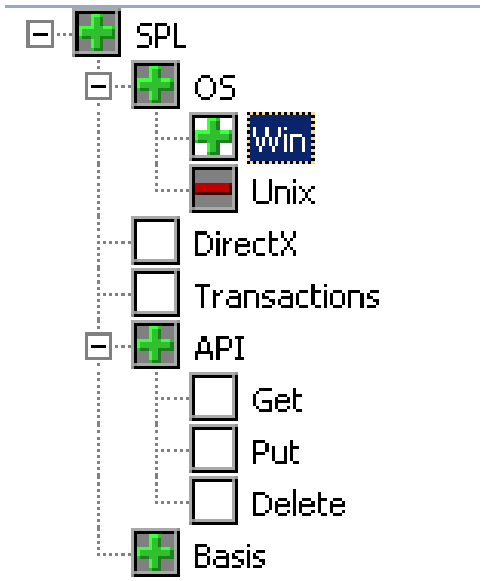
- Many different variants in literature, for example:
  - Inner nodes as features
  - Mandatory/Optional in or/alternative-Groups
  - Implies/Excludes arrows
- Conversion is usually possible



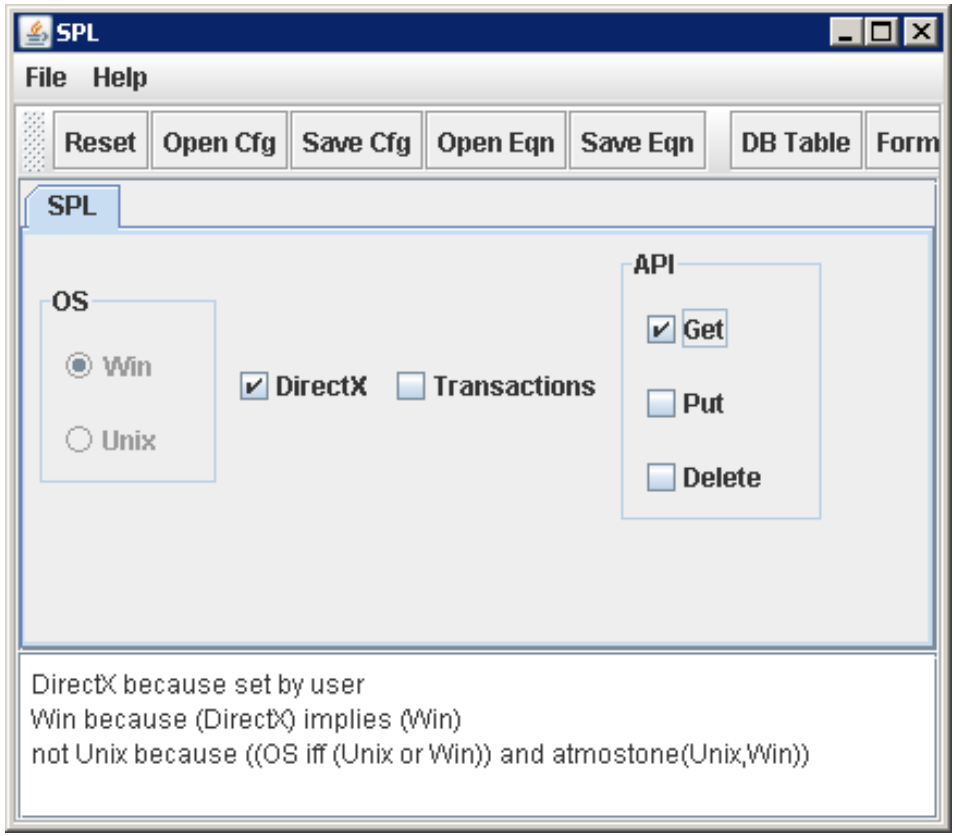




Configuring a Variant



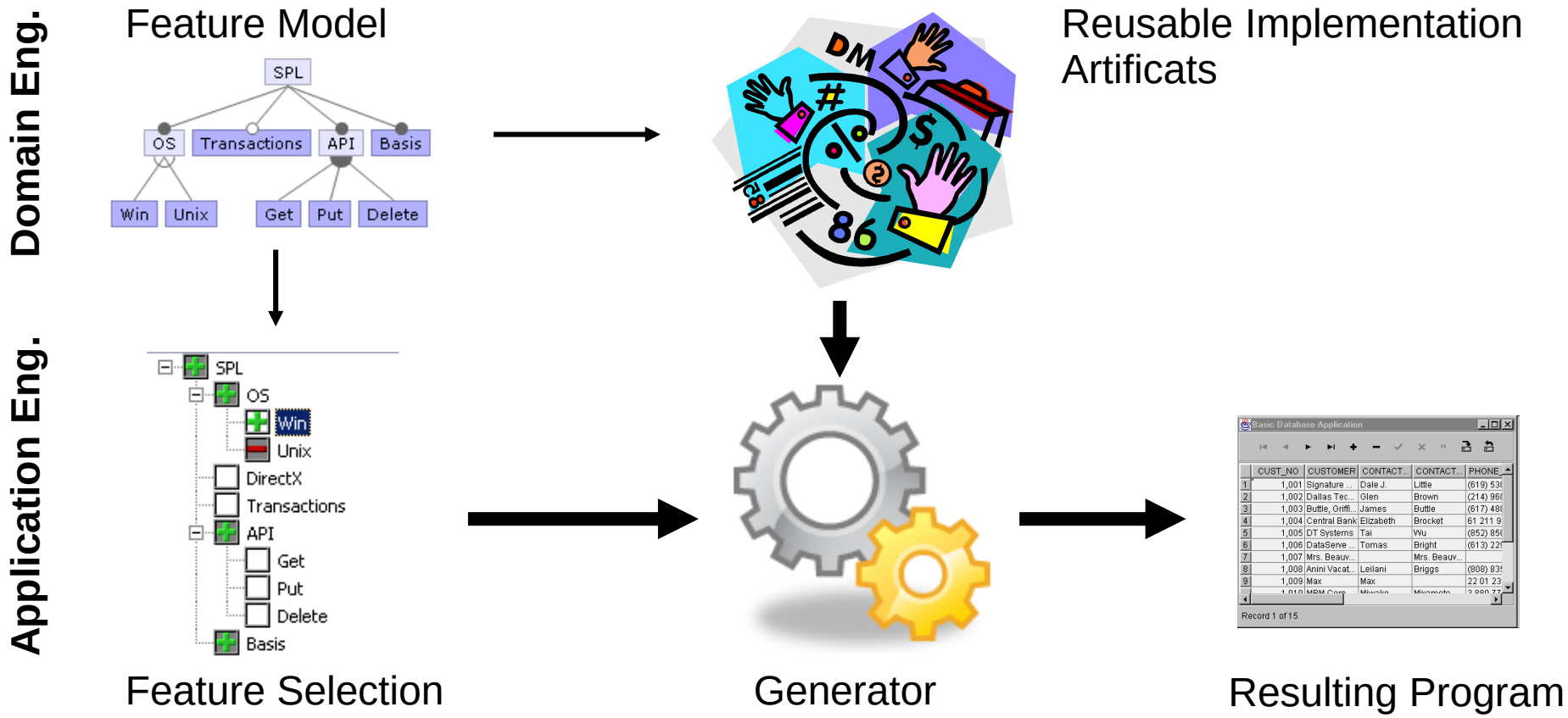
FeatureIDE



GUIDSL

Design and Implementation of Features

- After feature model follows the design and implementation



Summary

- Product line as a concept for systematic reuse
- Development is separated into two phases: domain engineering and application engineering
- Features represent domain concepts
- Variants of a product line share common features
- Feature models describe features and their constraints…
- …and are used for configuration

Outlook

- Next chapter is about methods, techniques, and tools for implementing product lines

Literatur I

- K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, SEI,1990.
  [Early ideas about domain analysis with feature models]

- K. Czarnecki and U. Eisenecker. Generative Programming: Methods, Tools, and Applications. Addison-Wesley, 2000.
  [Exhaustive description about domain engineering, feature diagrams and their normalizations]

Literatur II

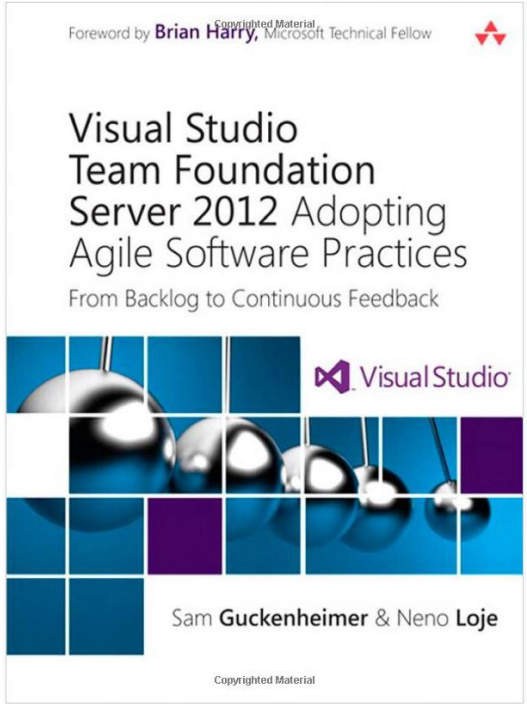
- D. Batory. Feature Models, Grammars, and Propositional Formulas, In Proc. of Software Product Line Conference (SPLC), 2005

  [Describes formally the relationship between feature models, grammars, and propositional formula]

- Books about product line development:
  - P. Clements, L. Northrop, Software Product Lines : Practices and Patterns, Addison-Wesley, 2002
  - K. Pohl, G. Böckle, F. van der Linden, Software Product Line Engineering: Foundations, Principles, and Techniques, Springer, 2005

A Real-World Example: Microsoft

- Development team
  - Visual Studio
  - .Net Framework
- Exerpirience report about the developing as a product line VS2005 vs. VS2008 (Chapter 9)
  - 3 500 Developers
  - 20 000 000 Source code files
  - 700 000 Tasks
  - 2 000 Nightly builds
  - 15 Terabyte data
  - Much variability (editors, components, …)



Initial State in 2005 (i)

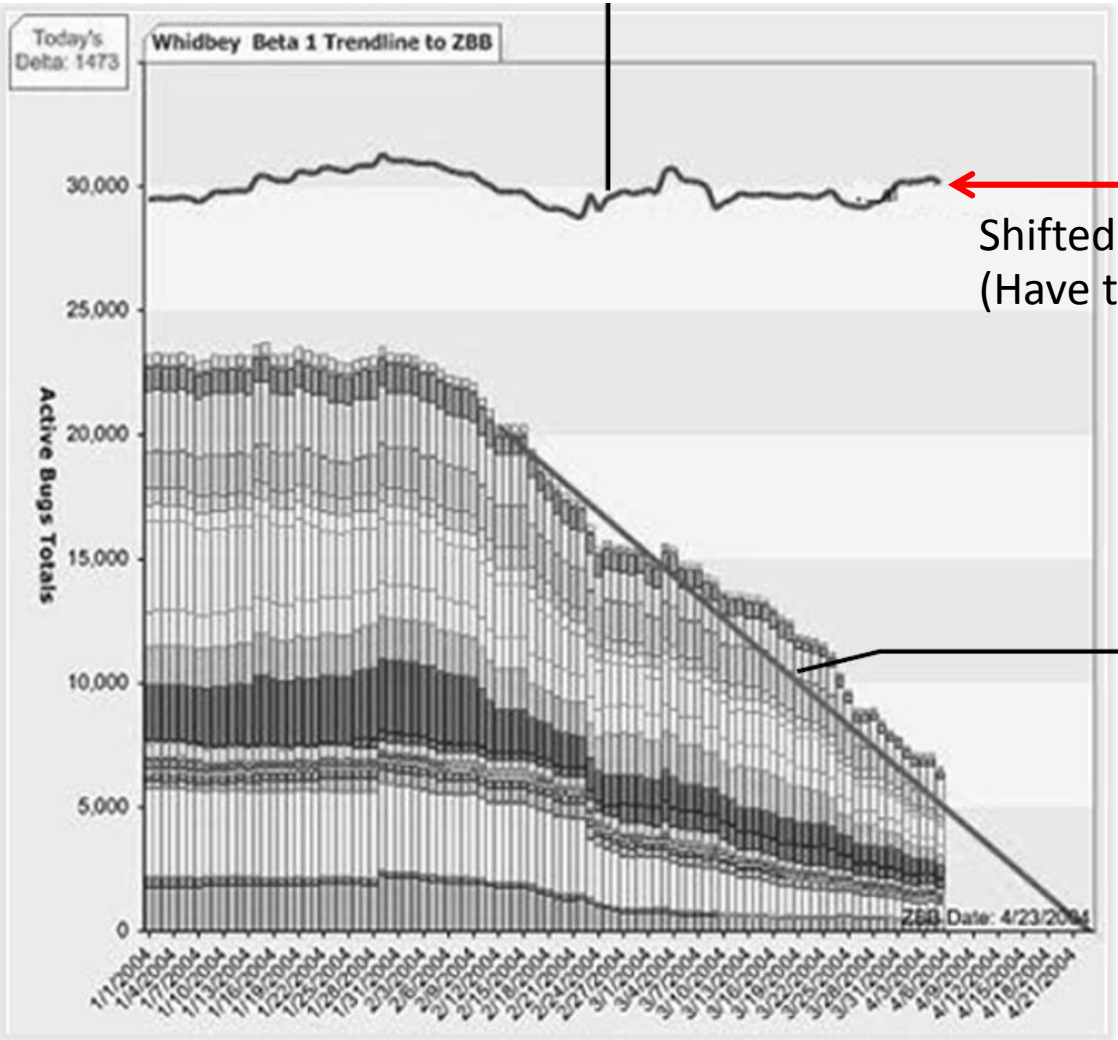
- 5 predecessor systems, non-interoperable
  - Code management, bug-tracking, build-automation, testing systemin and management
  - All isolated self-developments over decades
- Heterogeneous, economical context
  - Free software (Silverlight, Express-Versions, .Net) with the goal to bind customers and to increase development for Microsoft systems
  - Commercial systems (e.g., Visual Studio, MSDN) with rental model
  - Tension between economical goals, which were partly contradictionary

Initial State in 2005 (ii)

- No central authority
- No “Product-Backlogs” existing, which describe and summarize requirements
  - “Nobody had the ability to comprise a list of more than 1000 features.”
- 5 behavior patterns of different divisions
  - Nonaggression pact between managers
  - Schedule Chicken: No division could keep the time schedule, but everyone knew that also the others could not make it
  - Reference data and statistics are for others (but not for me)
  - Our customers are different (because of the broad range of the domain)
  - Our group is better

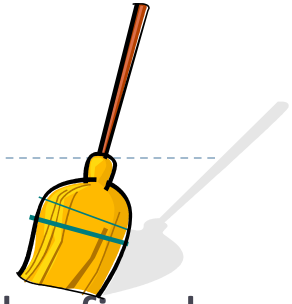
Waste of Resources

- Development of open bugs in the beta of VS2005



Shifted bugs to next version.
(Have to touched twice)

Improvements after 2005 (i)

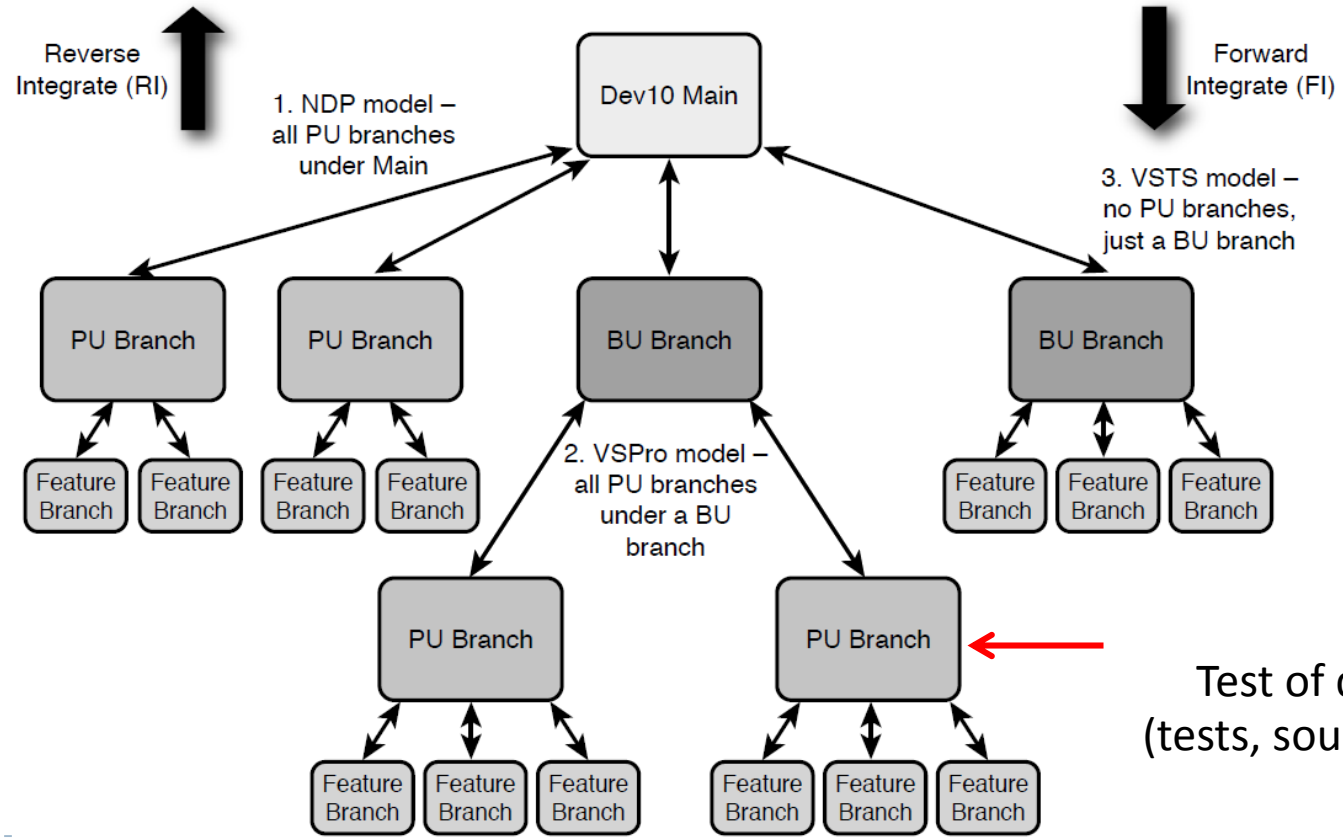


- Cleaning and keep order
  - Before implementing new features, all open bugs must be fixed
- Strict timeboxing
  - Instead of 3 month plan, 3 weekly sprints to implement features
  - Goal: After each sprint, there must be a potentially shippable product with increment in functionality

- Feature crews
  - Small, interdisciplinary teams (5—6 developers and testers + manager)
  - Implementation at isolated branches at the code base with special completion criteria (**modularity**!)

Feature Crews und Product Units (PU)

- Integration (in Main Branch) and Isolation (of changes of other crews)



Test of quality (tests, source code)

Improvements after 2005 (ii)

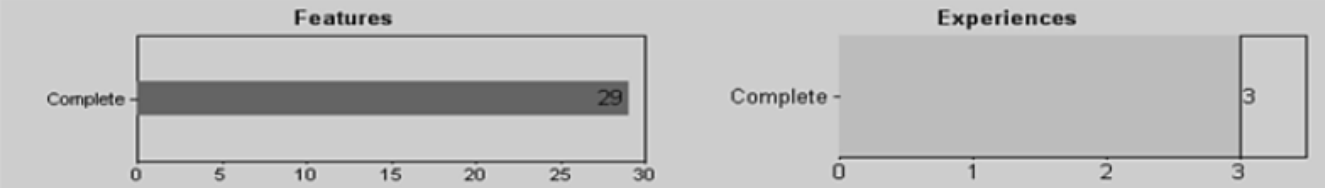
- Product-BackLogs

Coarse-grained, understandable for customers

ValueProp ID	ValueProp ValueProp Title	Experiences Complete	Features Complete	ValueProp Category	ValueProp State
⊟ 65581	Easily build solutions for the breadth of Office 12	205/380	29/29 (100.00 %)	Critical	Complete

Easily build solutions for the breadth of Office 12

Experience Summary

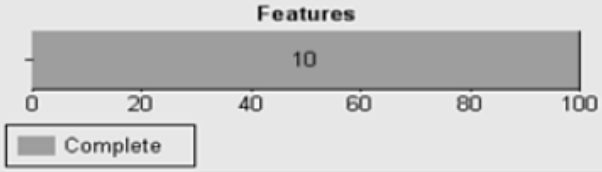


Experience ID	Experience Title	Feature Complete	Experience Category	Experience State
⊟ 85394	My add-ins use a standard framework that provides isolation and makes them independent of version changes in the host application	10/10 (100.00 %)	Critical	Complete

My add-ins use a standard framework that provides isolation and makes them independent of version changes in the host application

Feature Summary

Total Features: 10
Experience State: Complete

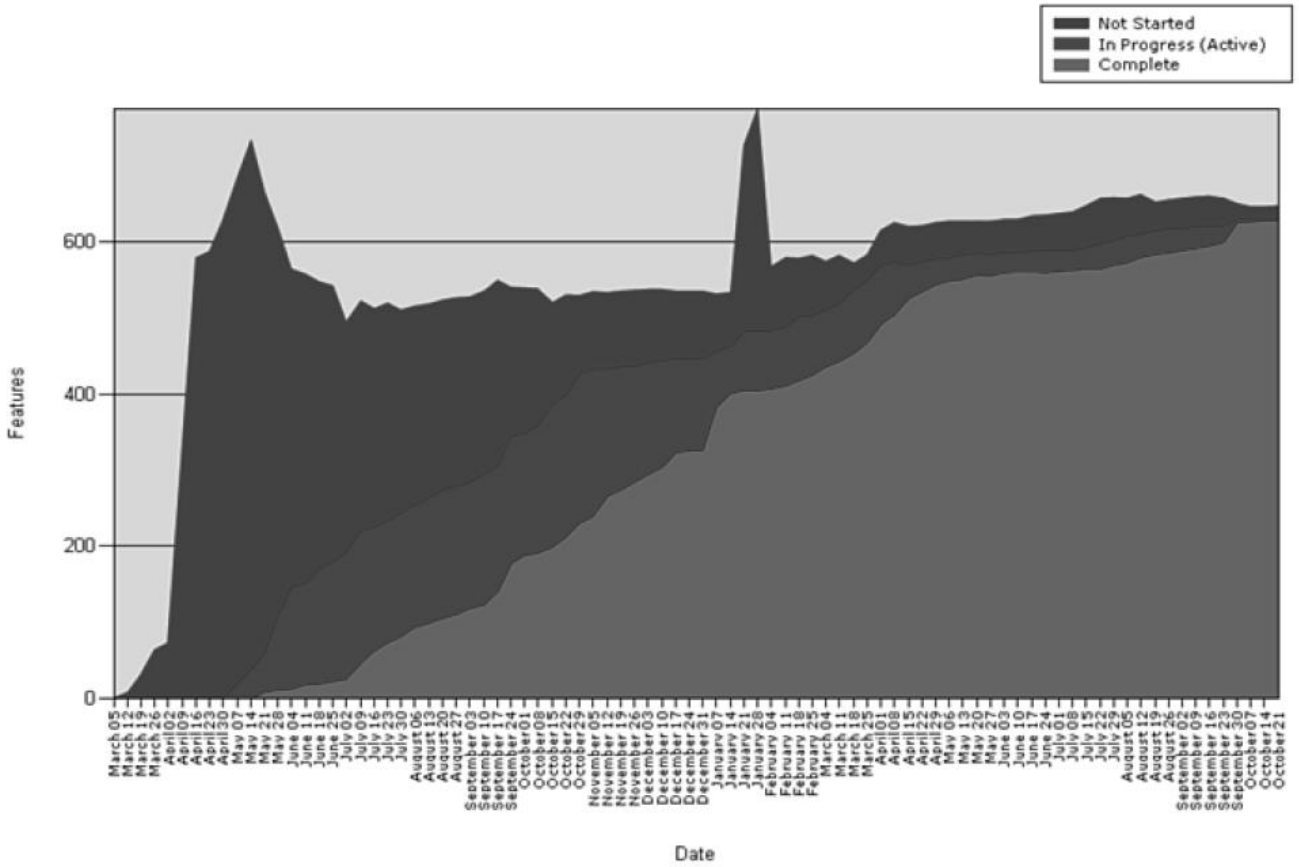


Feature ID	Feature Title	Feature Iteration Start	Feature Iteration End	Feature Category	Feature State
95892	VSTO Runtime versioning and stabilization	M2.2	MC2	Critical	Complete
95893	VSTO Loader Version Resilience	M2.2	MC2	Critical	Complete
95895	Runtime Support for IServiceProvider Interfaces	M2.1	M2.2	Critical	Complete

Fine-grained, precise for developers

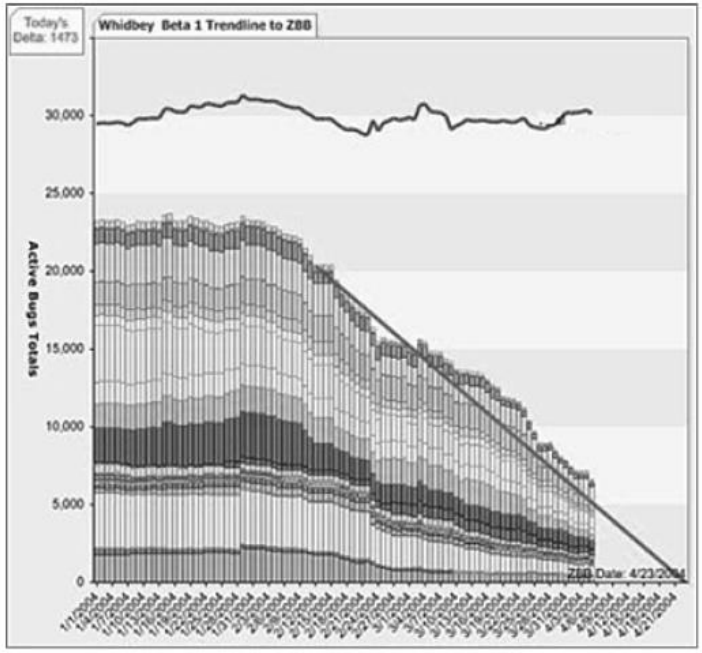
Results (i)

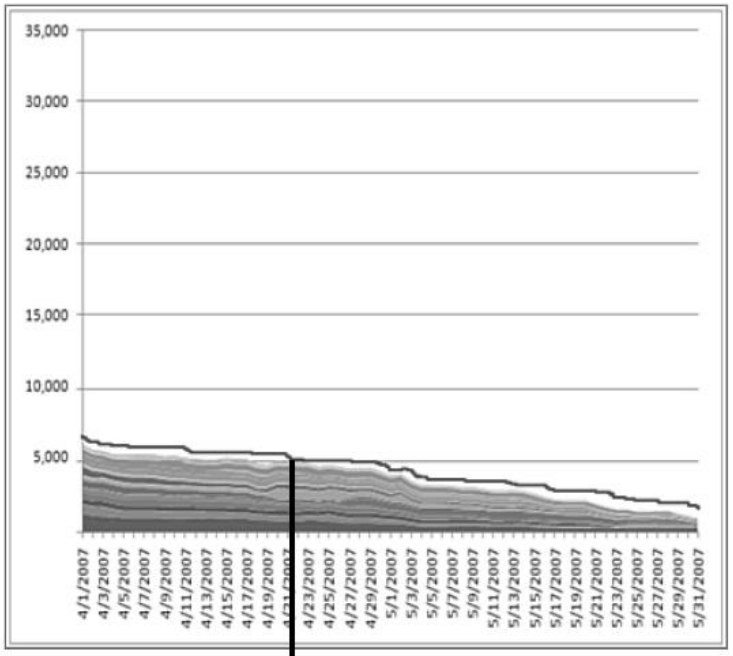
- Features were abstract enough to be interesting for customers or be important for another feature, and also detailed enough to be realizable within 3 weeks



Results (ii)

- Development time has cut in half till next release
- 15x less bugs
- Improved customer satisfaction



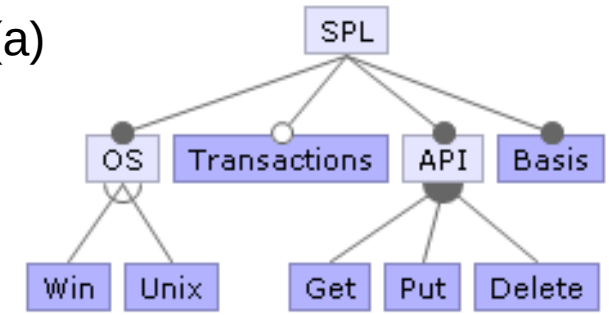


Total bug debt at Beta 1 of VS 2008.

Quiz

- How many variants exist? :

(a)



(b)

$$
\begin{aligned}
&Basis \\
&\wedge (Unix \vee Win) \\
&\wedge \neg(Unix \wedge Win) \\
&\wedge (Txn \Rightarrow Write) \\
&\wedge (BTree \vee HMap) \\
&\wedge (Write \Rightarrow BTree)
\end{aligned}
$$

- Is Android and the C++-Standard-Template-Library a product line? Explain!