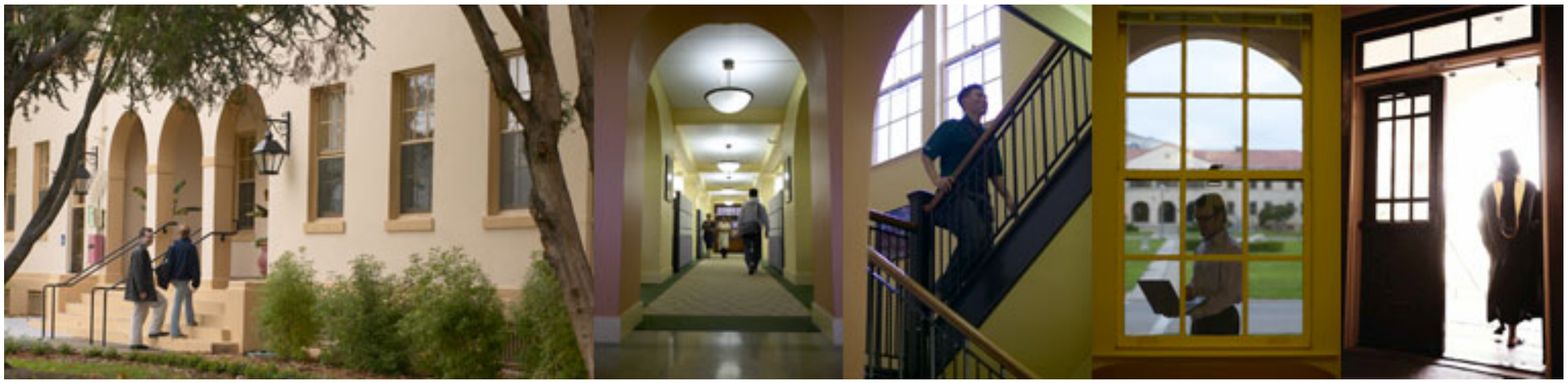


# Systematic Software Reuse

## *It Isn't What It Used to Be*

Martin Griss, PhD
Principal Research Scientist
Carnegie Mellon University, Silicon Valley Campus
&
Principal, Martin Griss Associates

## Agenda

- Background
- From Libraries to Factories
- Generative Reuse
- Agile Reuse
- Conclusions

## 40 Years Evolving Reuse Practice

- Software portability, LISP compilers, languages - U of Utah
- HP Reuse libraries, corporate reuse program, process
- Software Reuse: From Library to Factory
- (Hybrid) Domain Specific Kits
- UML 1.0 standards committee
- Reuse advice to HP divisions & customers
- RSEB: Software Reuse: Architecture, Process, & Organization for Business Success
- FeatureRSEB, Product Lines
- LogicLibrary, Flashline and TopCoder consulting
- Reuse Comes in Several Flavors
- Study of TopCoder crowdsourcing
- Agile Reuse

## Systematic Software Reuse

*Component-oriented software engineering*

### A simple idea

Use previously developed components, frameworks, other artifacts

### ... with complex execution ...

New component & framework & generator technology & methods
Architecture, process, organization, economics, cultural changes

### ... but with major benefits!

AT&T, GTE, Ericsson, HP, IBM, NEC, Rolls-Royce, Toshiba, Volvo,…
Significant cost and time reductions
Improved agility

# Reuse Body of Knowledge

*Many books & conferences on reuse & related topics*

- Architecture, aspects, patterns, frameworks, components, product lines, generators, domain engineering, management, organization

The Rational Unified Process An Introduction
UML Distilled Third Edition
Martin Fowler
Design Patterns
Elements of Reusable Object-Oriented Software
Software Reuse
Architecture, Process and Organization for Business Success
Ivar Jacobson • Martin Griss • Patrik Jonsson
Managing Software Reuse
Software Product Lines in Action
Software Product Lines
Practices and Patterns
Paul Clements
Linda Northrop
Software Factories
Pattern-Oriented Software Architecture
Domain Architecture
Software Architecture in Practice
Generative Programming
Aspect-Oriented Software Development
Building Systems from Commercial Components
Component Software
Composing Software Components

Carnegie Mellon University Silicon Valley

## Many Reuse Technologies

- Aspects
- Patterns
- Templates
- Parameters
- Components
- Frameworks
- Domain-specific languages
- Generators
- Services/SOA
- Agents

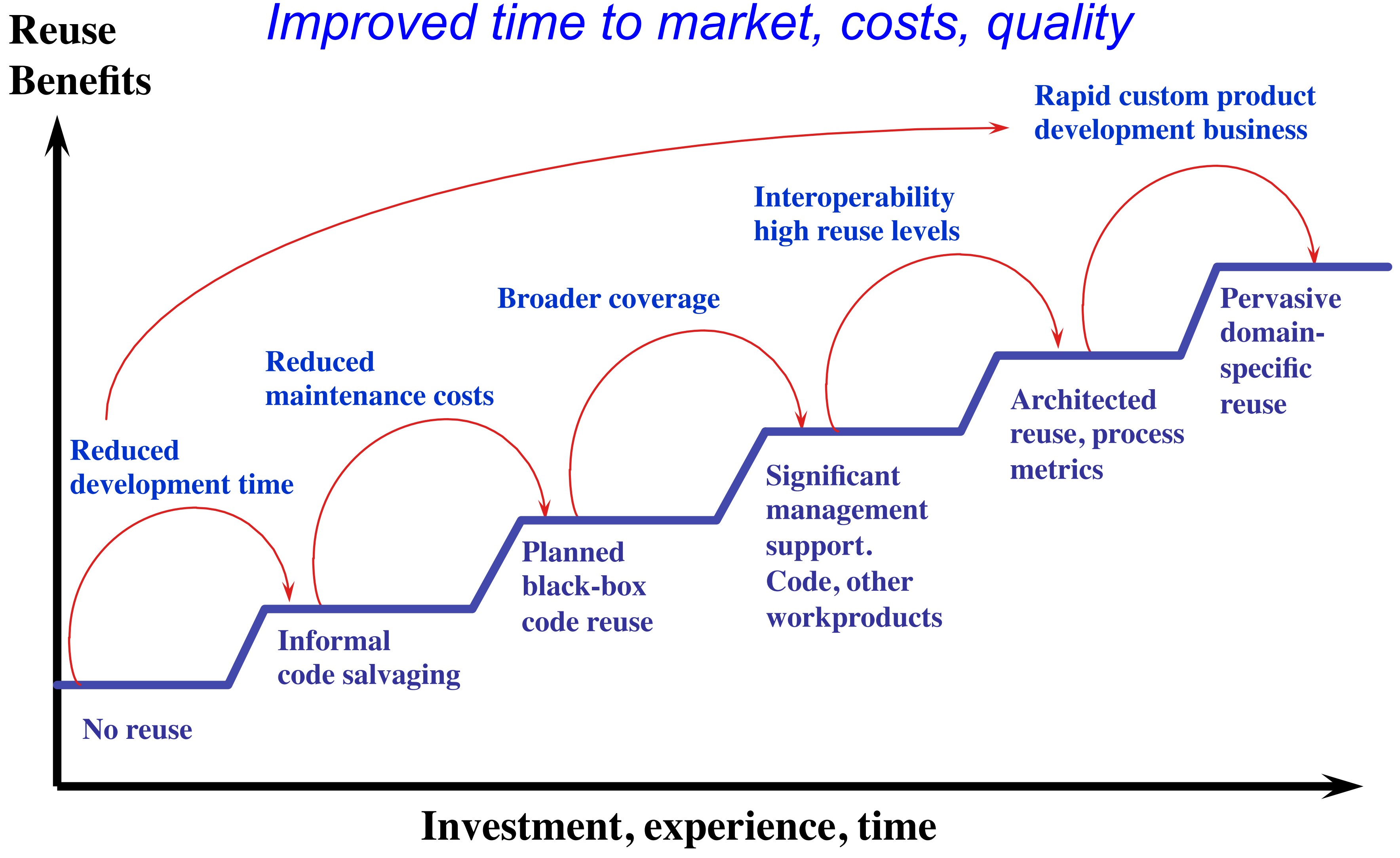
- Library system(s)
- Horizontal vs Vertical reuse
- Domain Engineering
- Variability Analysis
- Reuse-oriented Architecture
- Model-Driven Development
- Product Line Engineering
- Open Source/Corporate Source
- Crowd Source

## Many Reuse Questions

- What kind of reuse should we do?
- What  strategy of marketing, incentives for reuse?
- What is an appropriate organization model?
- Should we do full scale product line reuse?
- Should we do model-driven development
- Should we use generators and domain-specific languages
- What technologies and tools to focus on?
- How are assets and support funded?
- What kind of reuse pilots to do?
- How and when to scale up?
- How is reuse connected to other software initiatives: *architecture, SOA, process improvement, quality, metrics, open source, crowd source, …*

# (Staged) Adoption of Reuse

*Improved time to market, costs, quality*


Reuse Benefits
Rapid custom product development business
Interoperability high reuse levels
Broader coverage
Reduced maintenance costs
Reduced development time
Pervasive domain-specific reuse
Architected reuse, process metrics
Significant management support. Code, other workproducts
Planned black-box code reuse
Informal code salvaging
No reuse
Investment, experience, time

## Reuse May Vary Across Organization

Components, Libraries

- Ad hoc, random reuse

Architecture, Frameworks

- Powerful enablers and process enhancements

Platform, Services

- Strategic to company success

## Reuse "Flavors"

***4. Reuse-Driven Business*** – **Reuse central to all decisions**

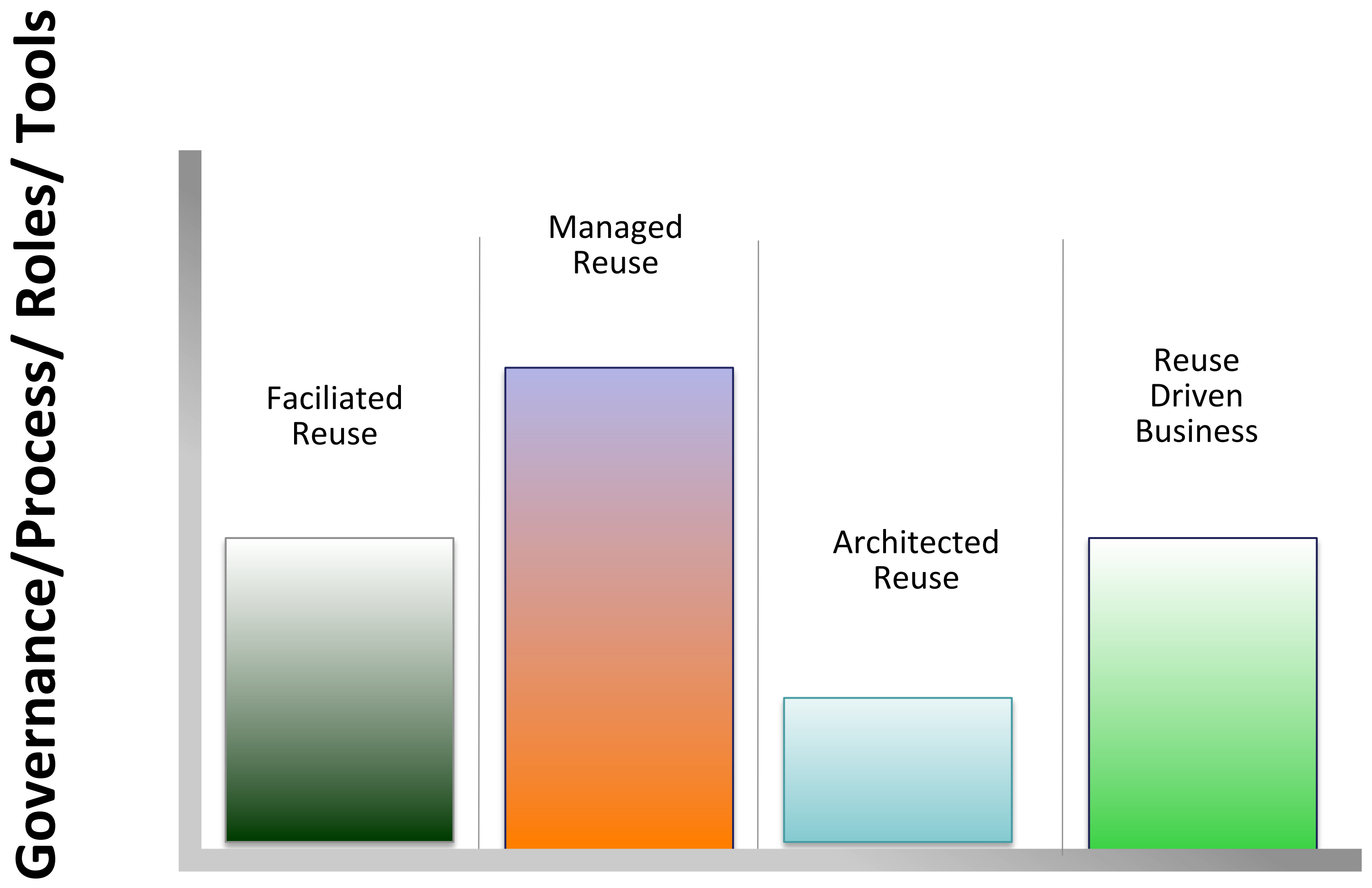
***3. Architected Reuse*** – **Architect, domain engineer assets for reuse, domain**

***2. Managed Reuse*** - **Require, enforce, control participation, use of assets**

***1. Facilitated*** – **Encourage, support, enable individual or team choice**

*ad hoc reuse* - **NONE**

# Mixing Reuse Flavors



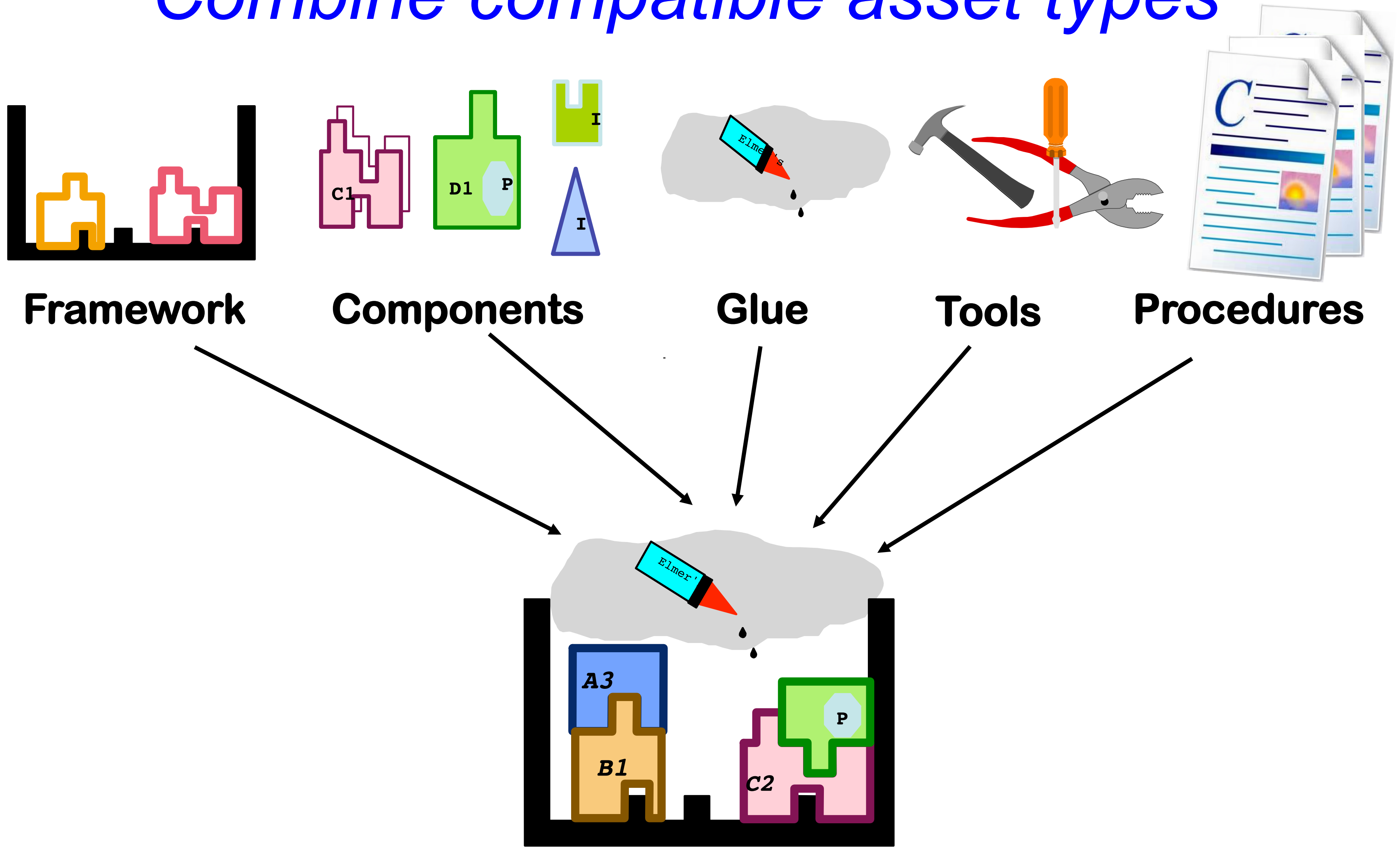
## Agenda

- Background
- ➡ From Libraries to Factories
- Generative reuse
- Agile reuse
- Conclusions

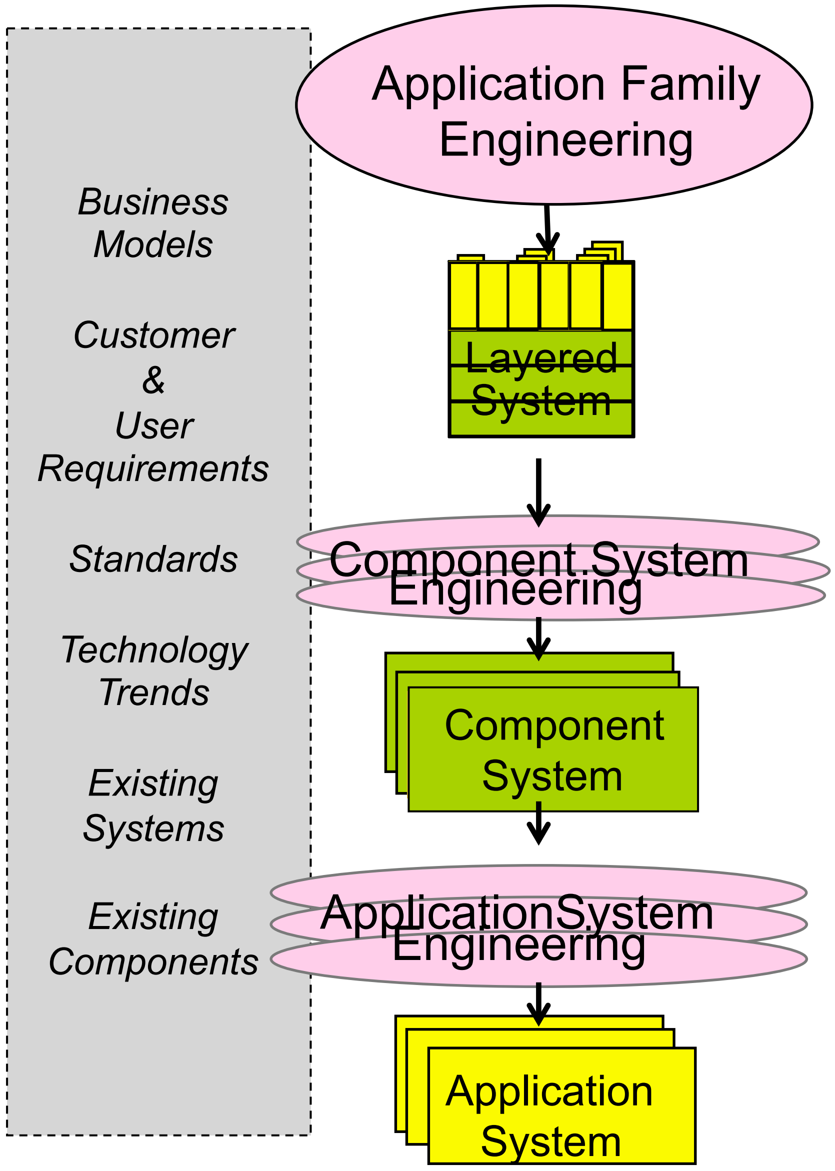
# From LEGO “components” to “kits”

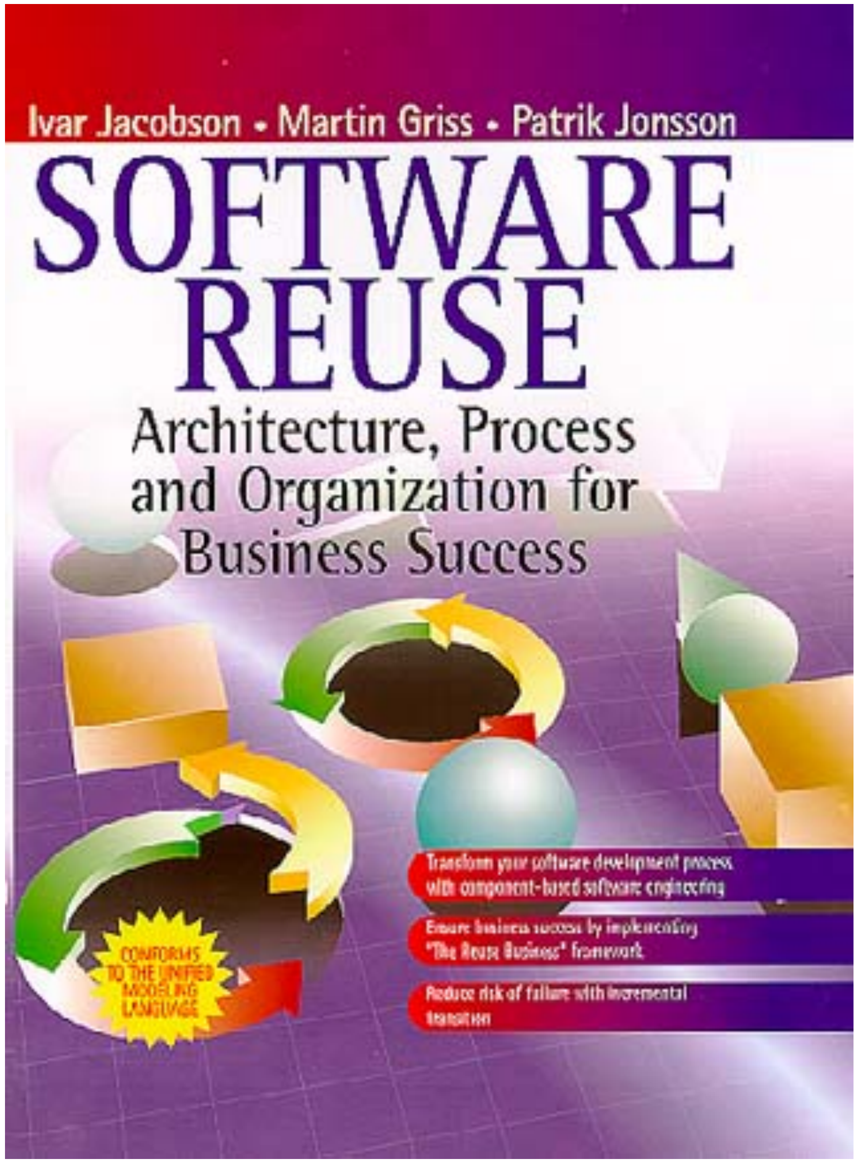
# (Hybrid) Domain-Specific “Kit”

## *Combine compatible asset types*

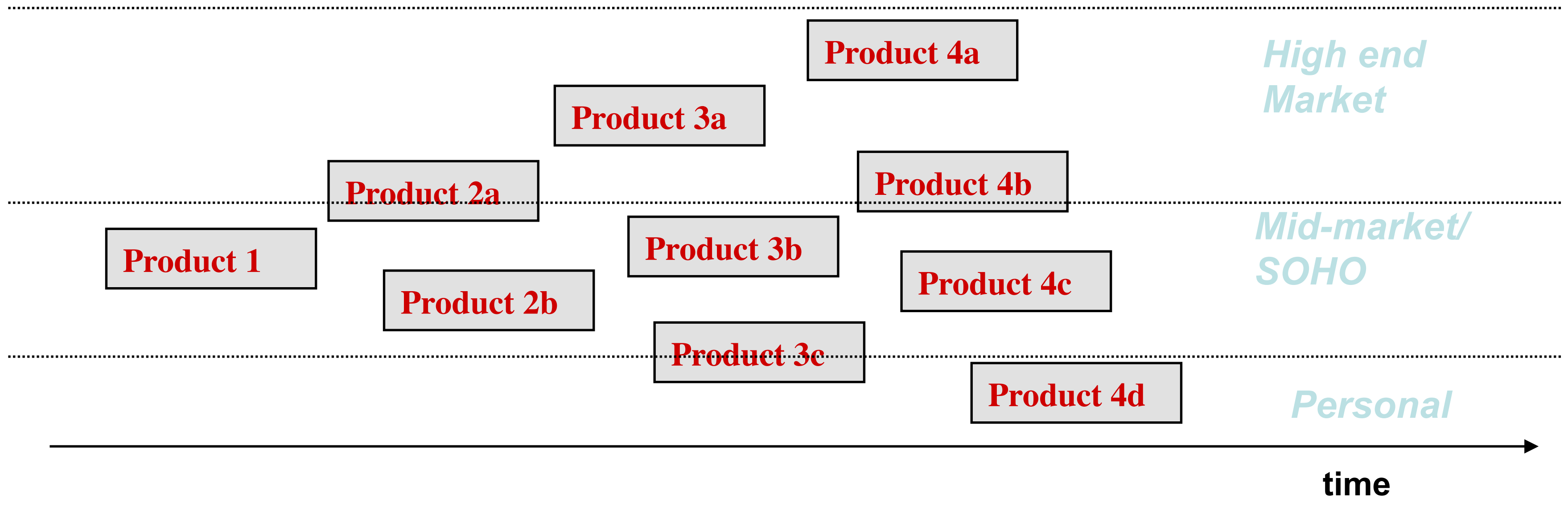


## RSEB


Application Family Engineering
Business Models
Customer & User Requirements
Standards
Technology Trends
Existing Systems
Existing Components
Layered System
Component System Engineering
Component System
Application System Engineering
Application System


Ivar Jacobson • Martin Griss • Patrik Jonsson
SOFTWARE REUSE
Architecture, Process and Organization for Business Success
Transform your software development process with component-based software engineering
Ensure business success by implementing "The Reuse Business" framework
Reduce risk of failure with incremental transition
CONFORMS TO THE UNIFIED MODELING LANGUAGE

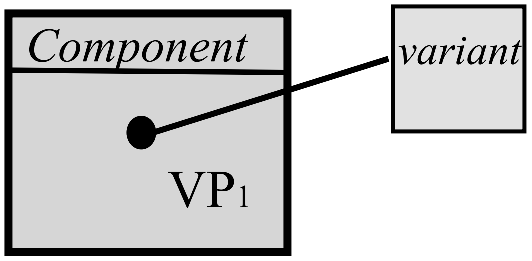
# Product Lines

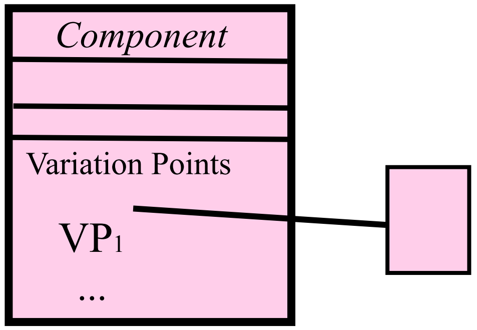


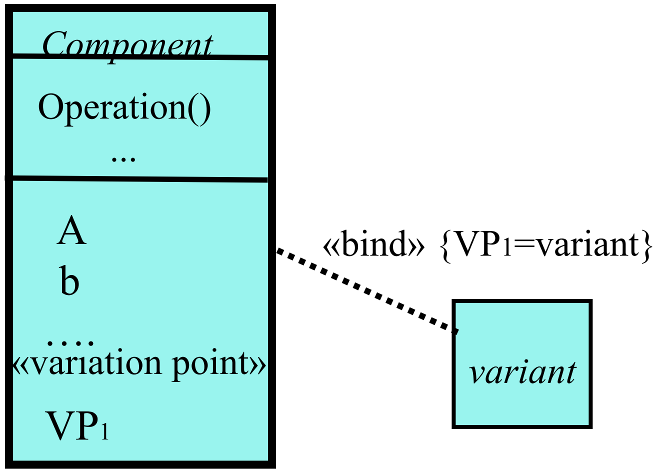
- *A* set of products sharing common set of requirements (*or features*), with significant variability
- *Feature* = product characteristic users, customers & developers use in describing/ distinguishing members of product-line.

# Expressing Variability

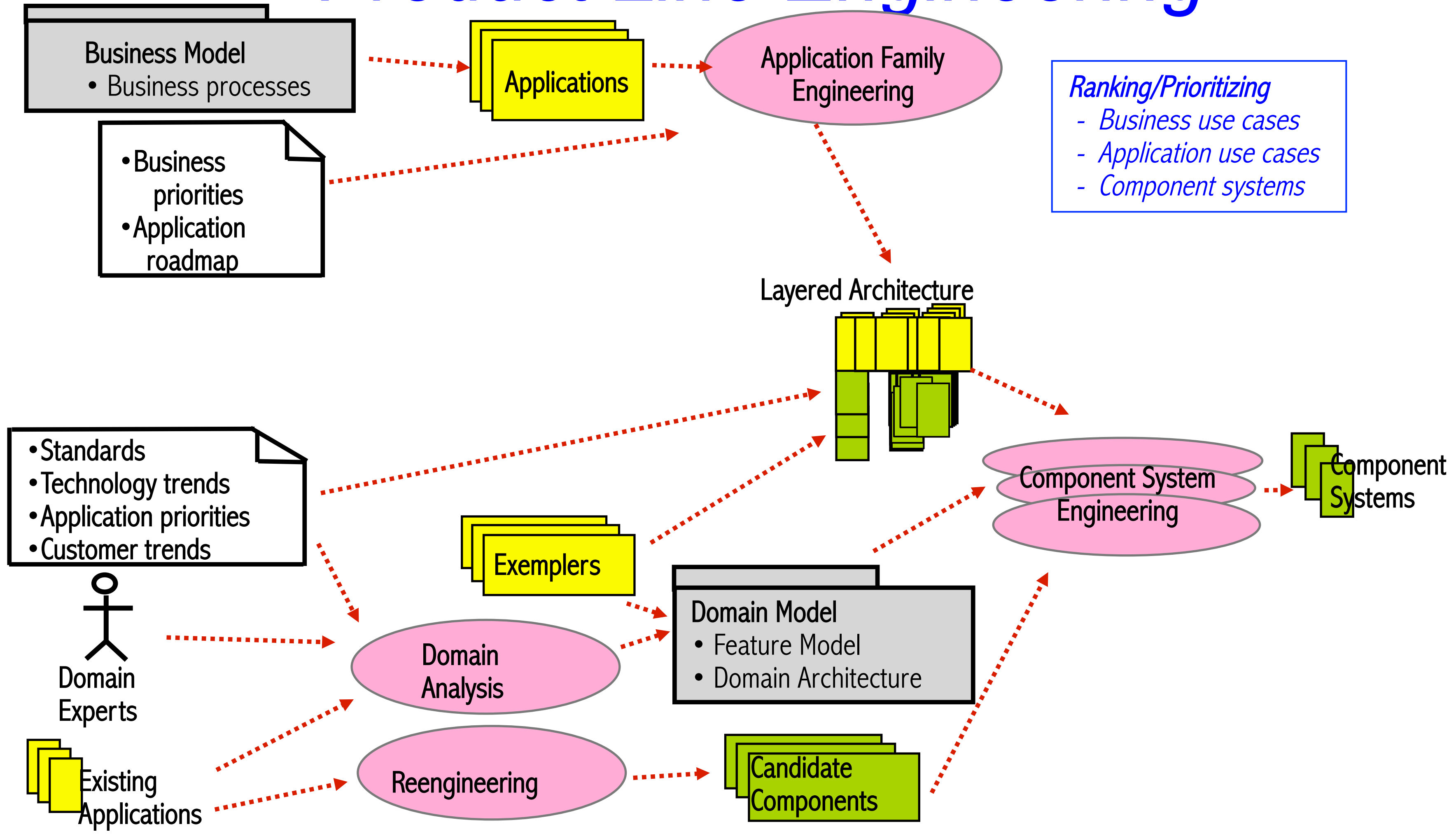
Components have *Variation Points* where they can be customized with *variants* using various mechanisms





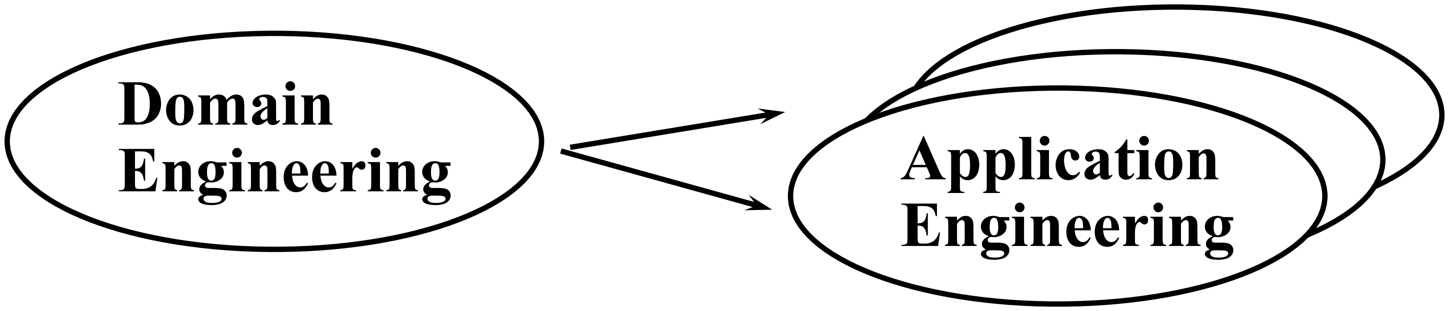


# RSEB
## *Product Line Engineering*



*Ranking/Prioritizing*
- *Business use cases*
- *Application use cases*
- *Component systems*

## Developing for Application Family
## *Domain-specific, architected, product-line*



**Provide:** ***Develop For Reuse***

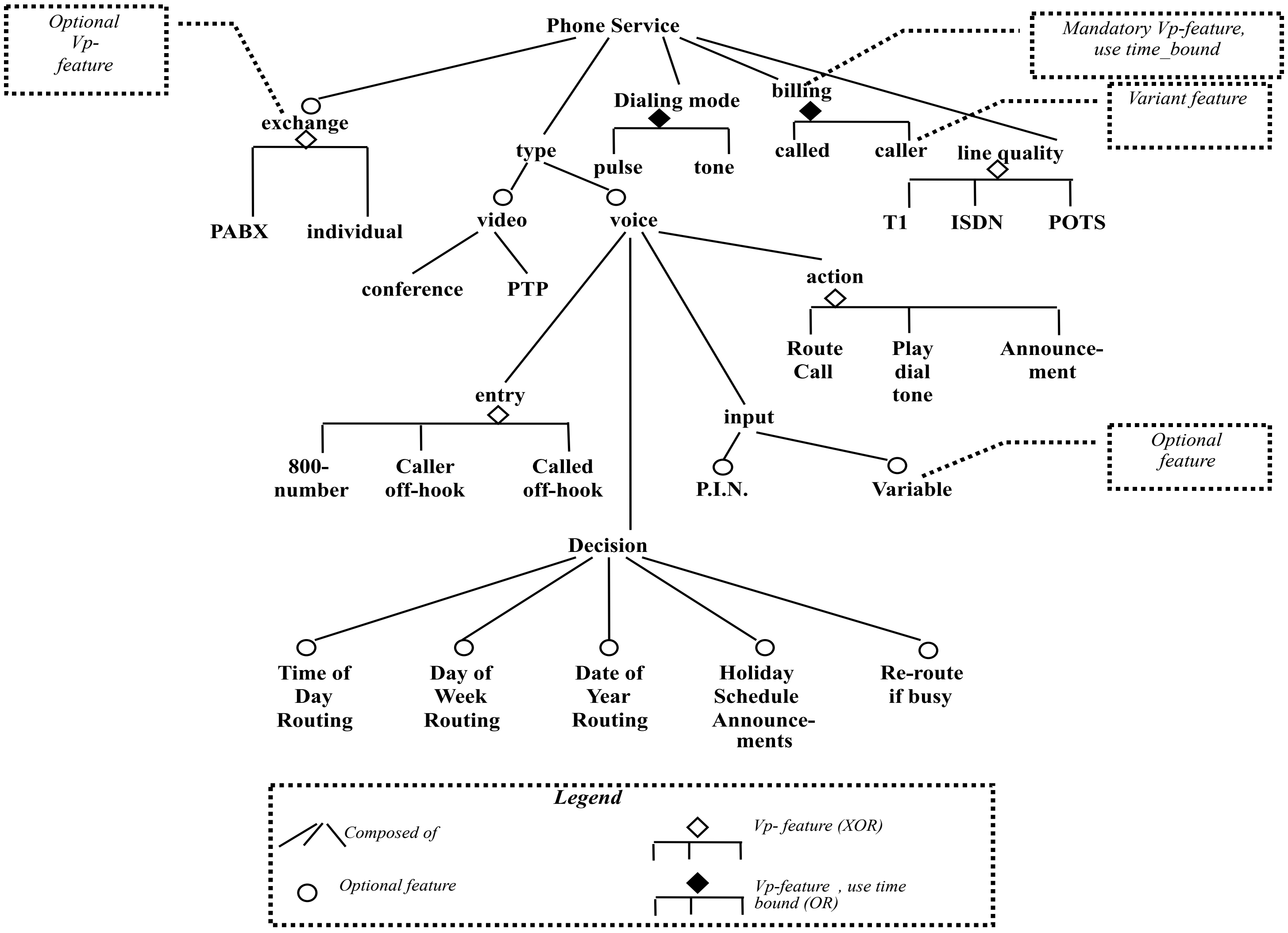
- **scope domain**
- **variability**
- **architecture**
- **components & frameworks**
- **DSL & generators**

**Utilize:** ***Develop With Reuse***

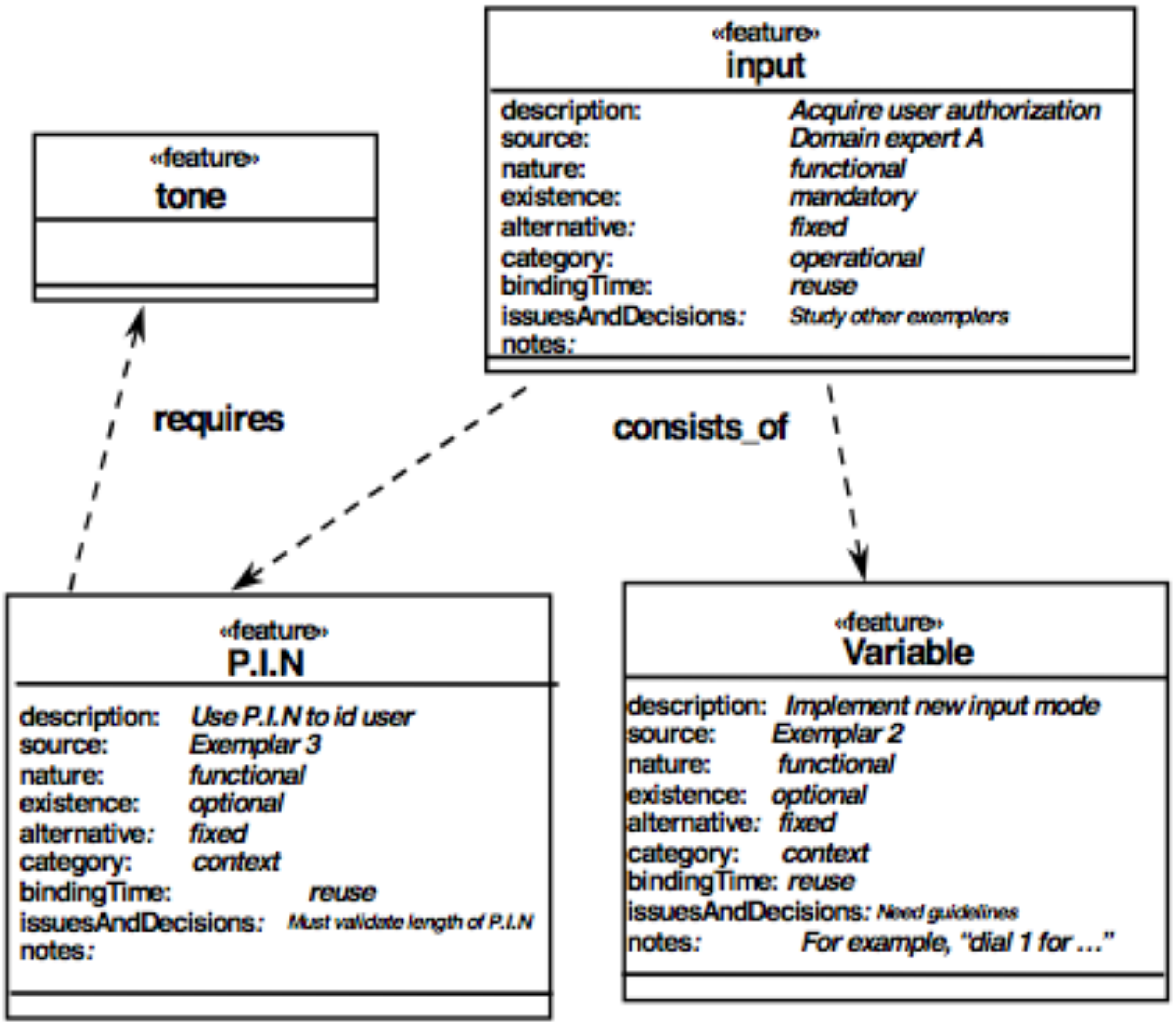
- **match to domain**
- **delta analysis**
- **select, adapt, integrate**

# FeatuRSEB

## *Combine RSEB, FODA, UML*



# FeatuRSEB


«feature»
tone
requires
«feature»
input
description: Acquire user authorization
source: Domain expert A
nature: functional
existence: mandatory
alternative: fixed
category: operational
bindingTime: reuse
issuesAndDecisions: Study other exemplers
notes:
consists_of
«feature»
P.I.N
description: Use P.I.N to id user
source: Exemplar 3
nature: functional
existence: optional
alternative: fixed
category: context
bindingTime: reuse
issuesAndDecisions: Must validate length of P.I.N
notes:
«feature»
Variable
description: Implement new input mode
source: Exemplar 2
nature: functional
existence: optional
alternative: fixed
category: context
bindingTime: reuse
issuesAndDecisions: Need guidelines
notes: For example, "dial 1 for …"

## Agenda

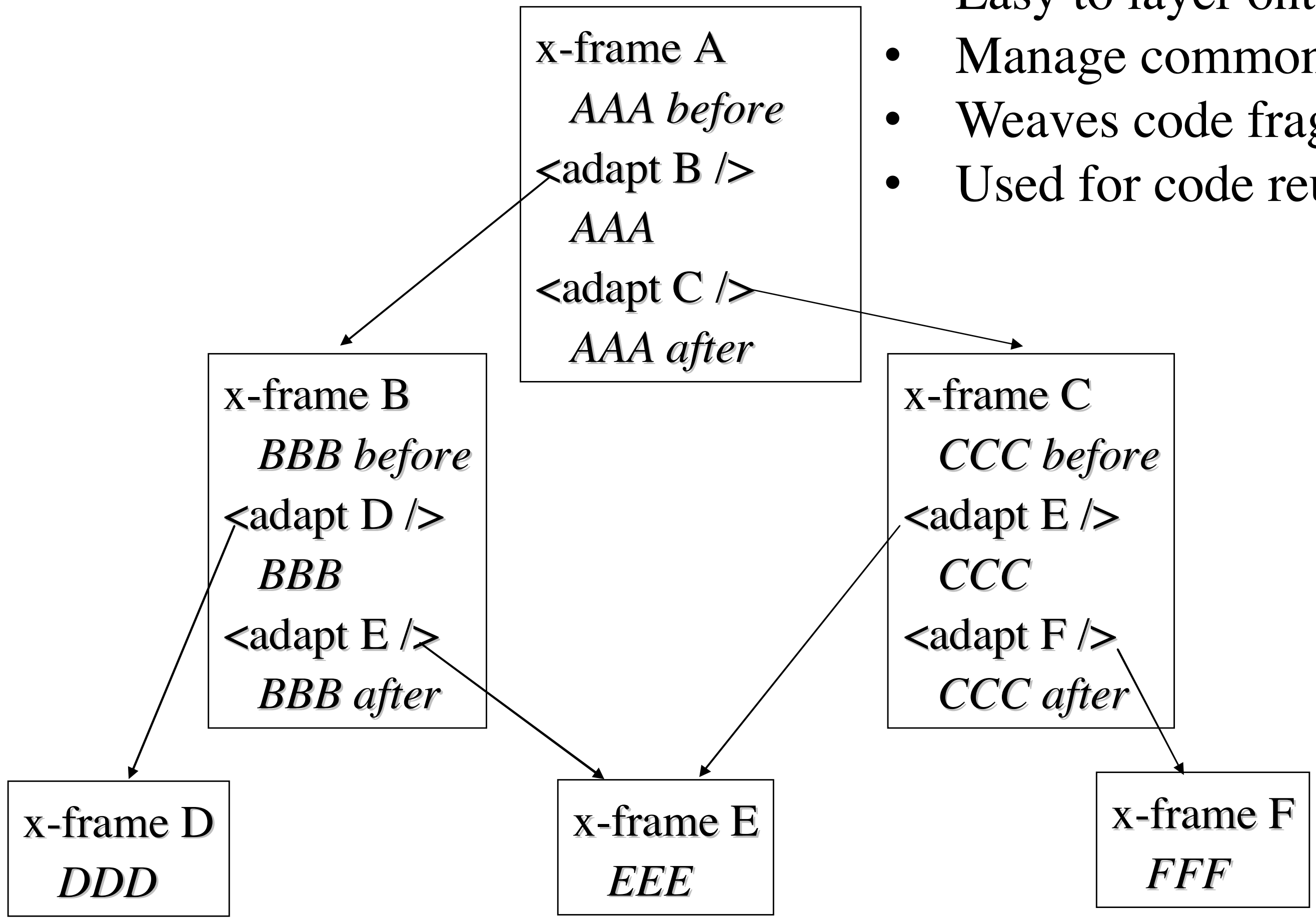
- Background
- From Libraries to Factories
- → Generative reuse
- Agile reuse
- Conclusions

## Generative Approaches

- Built-in to Language
  - C/C++ macros, LISP macros,  C++ templates, Java Generics, …
- General Purpose Macro Preprocessor
  - GPM, STAGE2, M4, Basset Frames (NETRON), XVCL, VCL, ..
- Extensible Languages
  - LISP, BALM/MBALM, Algol-68, EL1, …
- Domain Specific Languages/Kits
  - Via YACC, MetaLISP, BALM ,,,,. (e.g., PictureBALM), Visual Programming kit, OO Instrument Kits)
- Model-driven Generators
  - GenVOCA; MetaCASE; OMG MDA (UML for PSM/PIM), …
  - Aspects, …

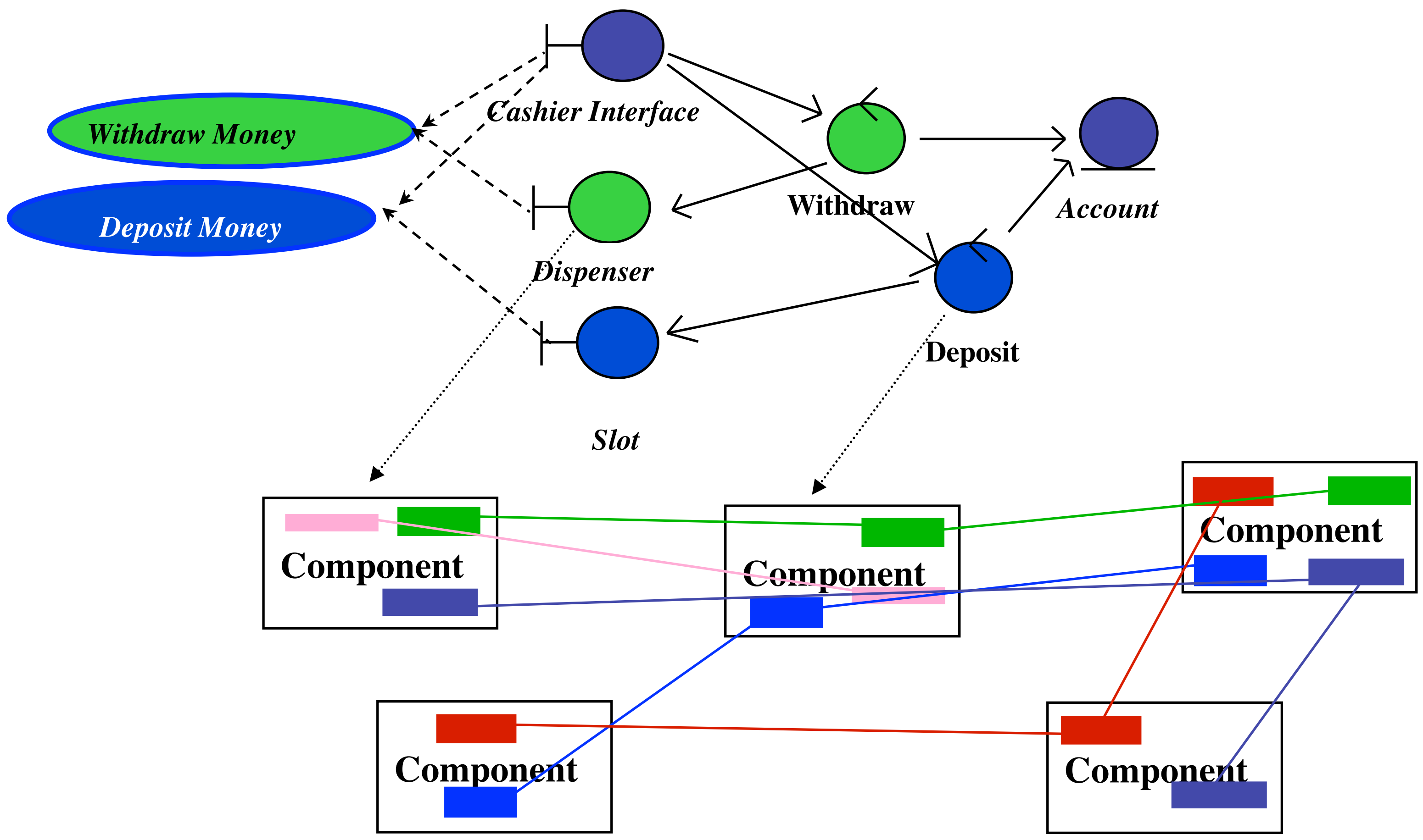
# XVCL/Bassett Frame Generator

- XML-based generator
- Template-based DSL
- Easy to layer onto existing software
- Manage commonality and variability
- Weaves code fragments (“aspects”)
- Used for code reuse  and  product lines



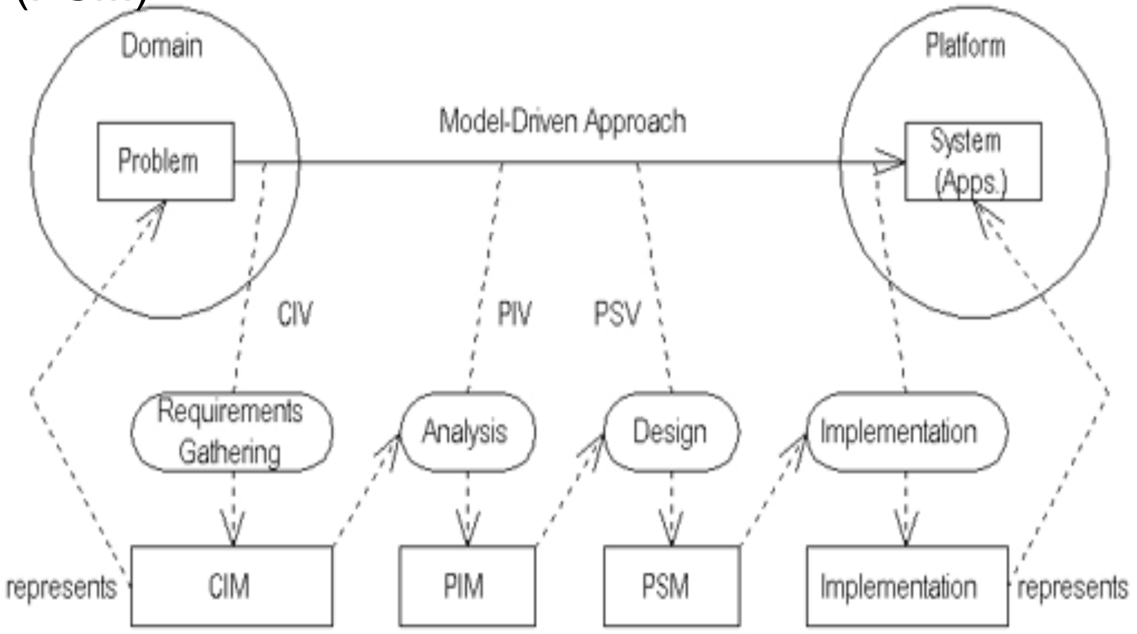
# Aspect-Oriented MDD

## *AOP, SOP, FOP, XVCL*



## OMG/UML Model Driven Architecture

- Use UML + <<stereotypes>> + OCL
- Create
  - Problem Independent Model (PIM)
- Generate
  - Problem Specific Model (PSM)
- Transformation rules



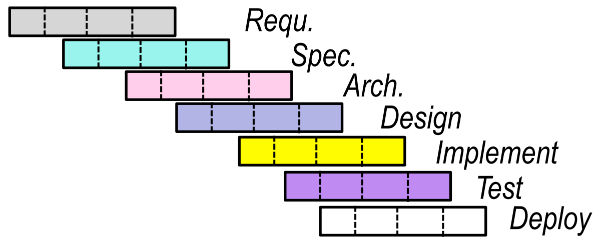
## Agenda

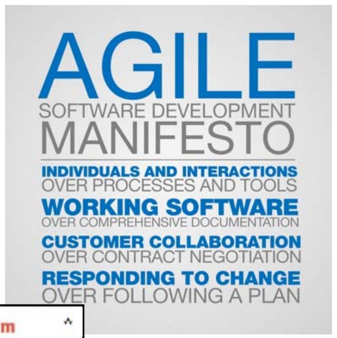
- Background
- From Libraries to Factories
- Generative reuse
- Agile reuse
- Conclusions

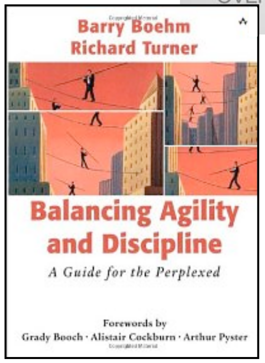
# Agile in the Enterprise

## *Plan-driven vs Agile vs Hybrid*

- Conventional plan-driven process
  - Large teams
  - Standardized models, architecture, documents and process
- Feature-oriented agile process
  - Small teams
  - Rapid development
  - Customer-oriented release and evolution
  - Expertise and tacit knowledge
  - Emergent architecture
- Hybrid approaches
  - Address scale, reuse, architecture




AGILE
SOFTWARE DEVELOPMENT
MANIFESTO
INDIVIDUALS AND INTERACTIONS
OVER PROCESSES AND TOOLS
WORKING SOFTWARE
OVER COMPREHENSIVE DOCUMENTATION
CUSTOMER COLLABORATION
OVER CONTRACT NEGOTIATION
RESPONDING TO CHANGE
OVER FOLLOWING A PLAN


Barry Boehm
Richard Turner
Balancing Agility and Discipline
*A Guide for the Perplexed*
Forewords by
Grady Booch · Alistair Cockburn · Arthur Pyster

## Approaches to “Agile” Reuse
## *Oxymoron? - YAGNI*

**Incremental Feature-Oriented Reuse**

- Leverage agile feature/story cards, SCRUM backlog
- Feed incremental Feature-Oriented Domain Engineering (FODA, FeatuRSEB)

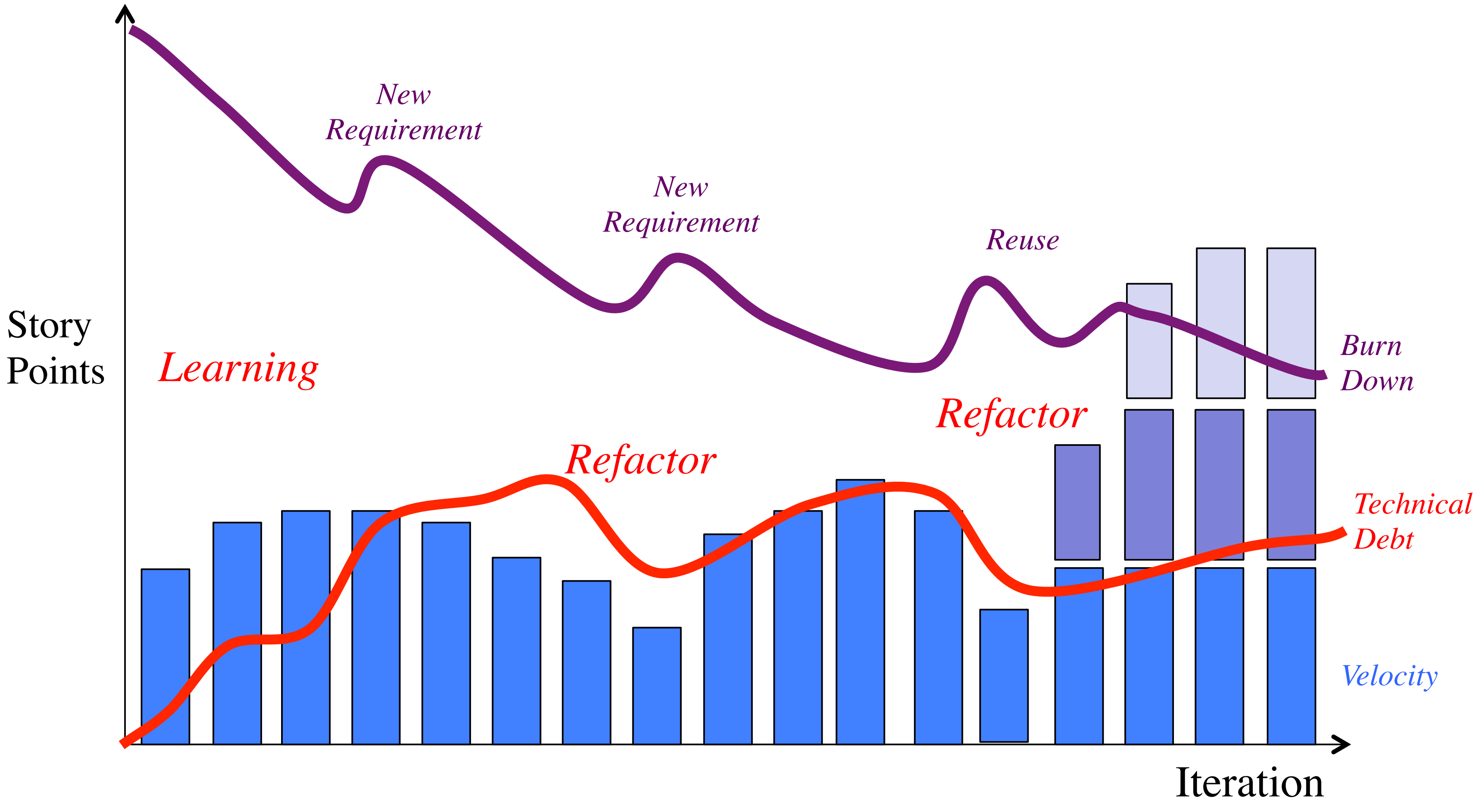
**Leverage Management of Technical Debt**

- Technical Debt accumulates with deferred decisions and work, coding shortcuts
- Incrementally pay off debt by re-factoring, re-engineering, re-architecting
- Economic/metrics models to help make decisions

**Domain Specific Languages**

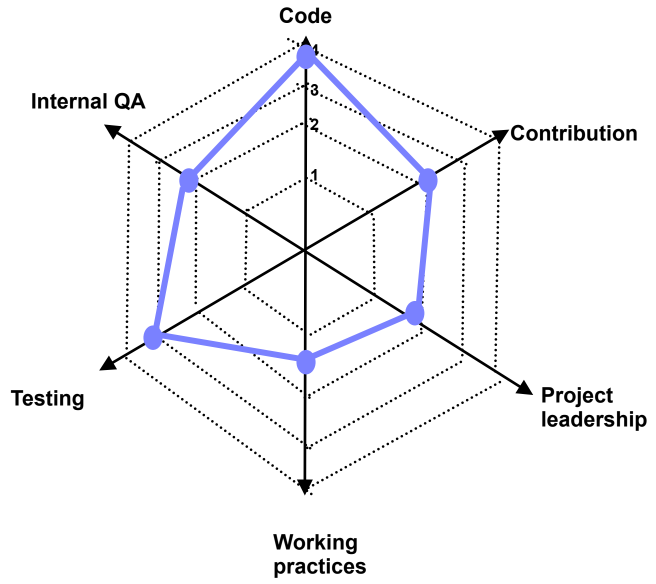
- Various levels of model-driven development

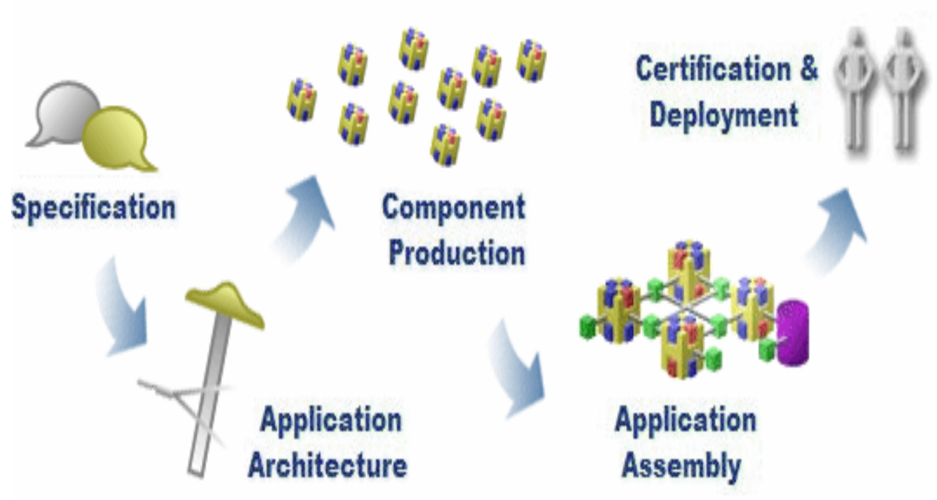
# Manage Technical Debt



## (New) Sourcing Models

- Open Source
  - Varying community and process management
- Corporate Source
  - Foster open source “style” in companies
- Crowd Source
  - Deliberate engagement of community





## Agenda

- Background
- From Libraries to Factories
- Generative reuse
- Agile reuse
- ➡ Conclusions

## Assess Reuse Readiness

1. Business goals that motivate reuse
   - Time, cost, quality, integration, agility, standards, ...
   - Urgency, importance, champion ...
2. Domain(s) readiness for reuse
   - Stability and variability, standards
   - Obvious, pervasive product line
3. Organizational readiness
   - Culture, process maturity, autonomy, standards
   - Conflicting initiatives, prior history, technology shifts,
4. Reuse experience
   - Current stage or flavor of (systematic) reuse
   - Reuse level, technology use, library use

## Conclusions

- Software reuse approaches keep evolving
- Assess reuse readiness before selecting reuse goal and flavors
- Identify opportunities for small DSL/MDD, generators and product-lines
- More work on agile reuse, SEMAT/reuse, open source/crowd source