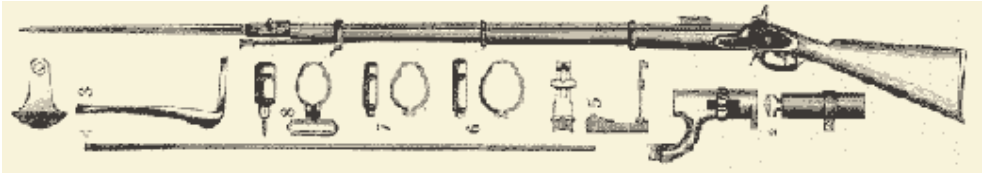
Software Architecture in Practice

Software Product Lines

1798 - Interchangeable Parts...



Reuse?

1969
- McIlroy: We need a component industry!

1994
- Booch: Why does component reuse not yet pervade industry?

2008
- ...?

Technical issues
- Lack of available components
- Low-level incompatibilities
- *Architectural mismatch [Garlan et al., 1995]*
  - E.g., mismatch on assumptions on the nature of component control such as push vs pull

Non-technical issues
- Organizational
- Economic
- Administrative
- Political
- Psychological

Reuse!

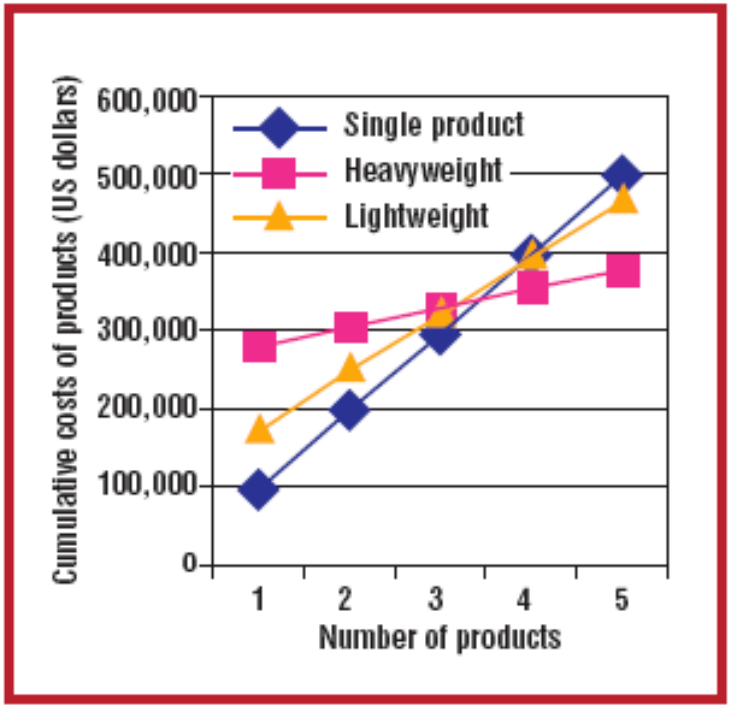
There are success stories

- Nokia, Cummins
- CelsiusTech

Reuse not of (only) code

- Reuse of software architecture
- Reuse of development artefacts
- Reuse of process artefacts
- ...

Effect of Software Product Lines



Heavyweight

- Proactive strategy
- Reusable assets are created before product

Lightweight

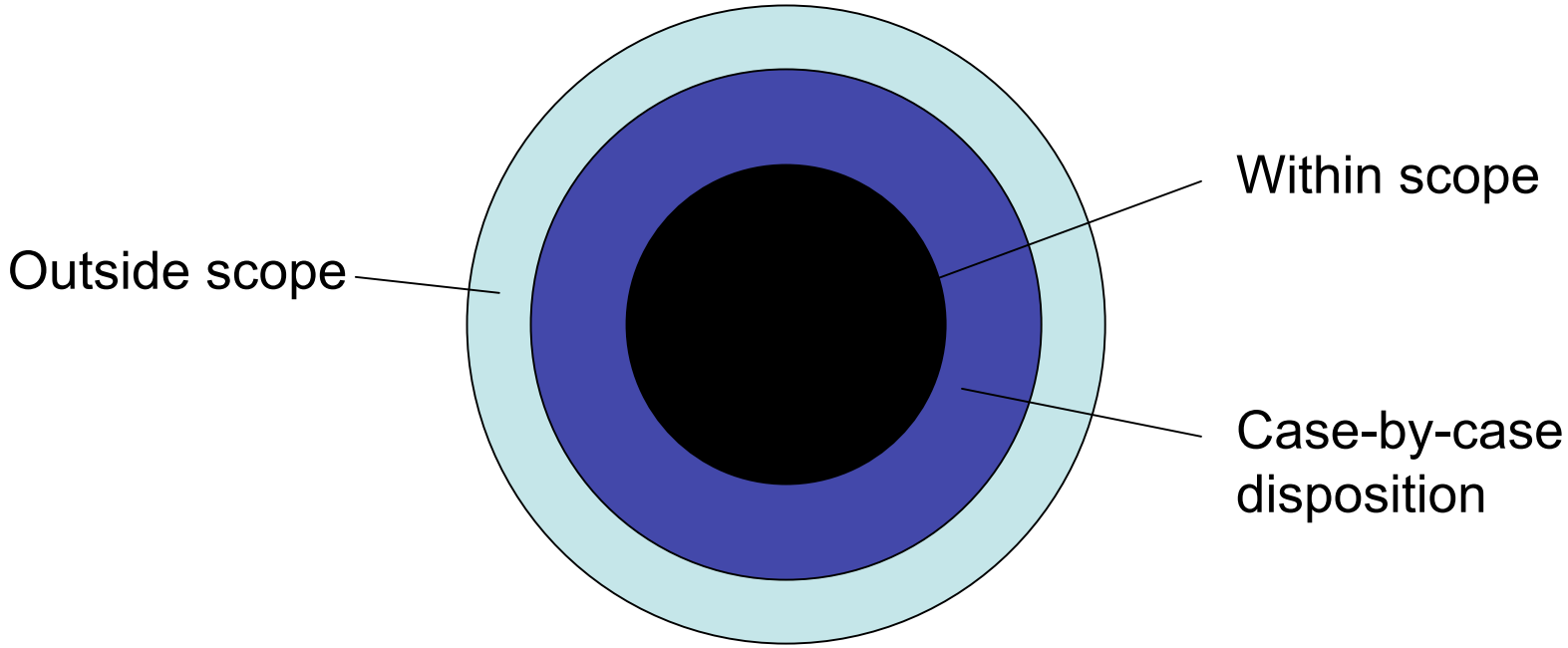
- Reactive strategy
- Products are mined for reusable assets

[McGregor, 2002]

Software Product Lines

[Clements et al., 2002]

- *a set of software-intensive systems*
- *sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission*
- *are developed from a common set of core assets in a prescribed way*



What Makes Software Product Lines Work?

Disciplined, strategic reuse of assets in producing a family of products

- Commonalities shared can be reused
- Variations need to be separately created

Planned growth of *core asset base*

- Requirements
- Design and code
- *Software architecture*
- Documentation
- Project management

Software product lines establish a strict context for reuse

- Defined architecture
- Defined functionality
- Defined quality attributes

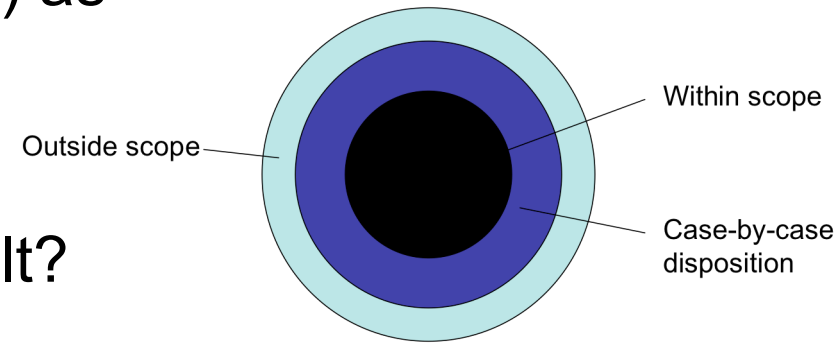
Scoping Software Product Lines

Product line scope

- A statement about what are in it and what systems are out
- What systems an organization is willing to build (and not build) as part of its line

Defining scope

- Which systems are to be built?
  - Commonalities and variations
- Market segmentation
- Types of customers



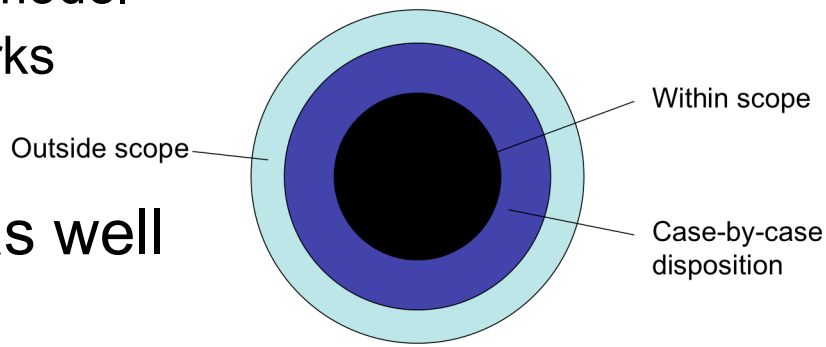
Scoping Software Product Lines

Narrow scope

- The products varies in a small number of features
- Building specialized tools to support specification of new products
  - E.g., domain-specific languages such as the Resource-Event-Agent model
  - E.g., domain-specific frameworks

Broad scope

- The products varies in kind as well as in features
  - E.g., CelsiusTech



Architectures for Product Lines

Software architecture has a central role in product lines

- What is expected to remain constant across products?
- What is expected to vary across products?

Need to consider

- Identifying variation points
- Supporting variation points
- Evaluating the architecture for product line suitability

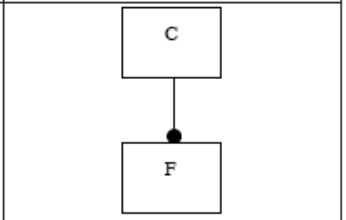
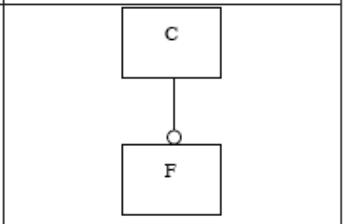
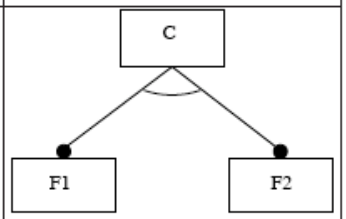
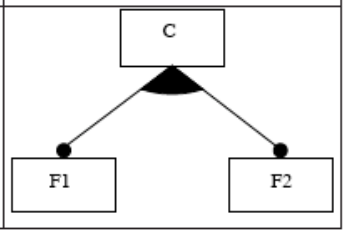
Identifying Variation Points

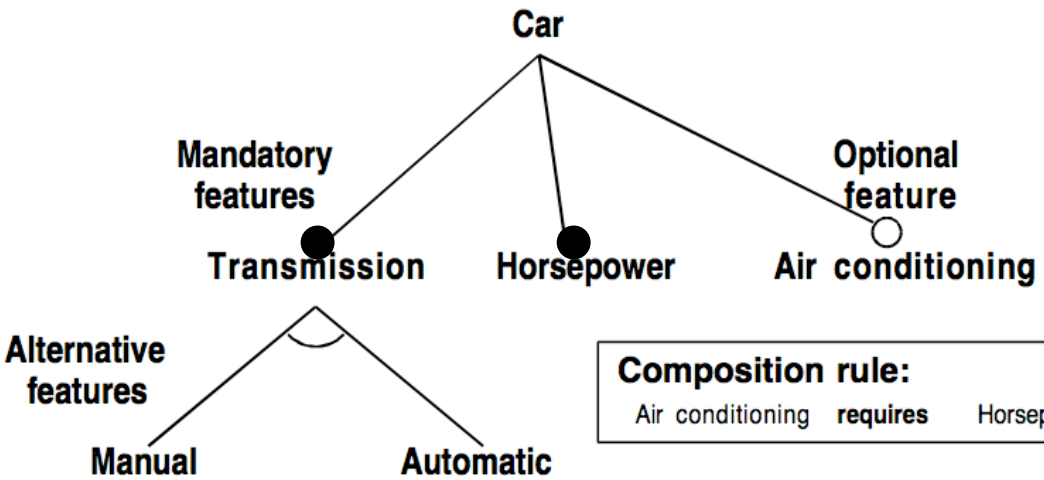
May be identified at various time of development

- Requirement elicitation
  - Features, platforms, user interfaces, qualities, target markets
  - May be interdependent
- Architecture design
  - Options for implementing variations
  - Deferance of decisions
- During implementation
  - Or during subsequent implementations

Feature Modeling

Features distinguish members of a product line
[Kang et al., 1990]

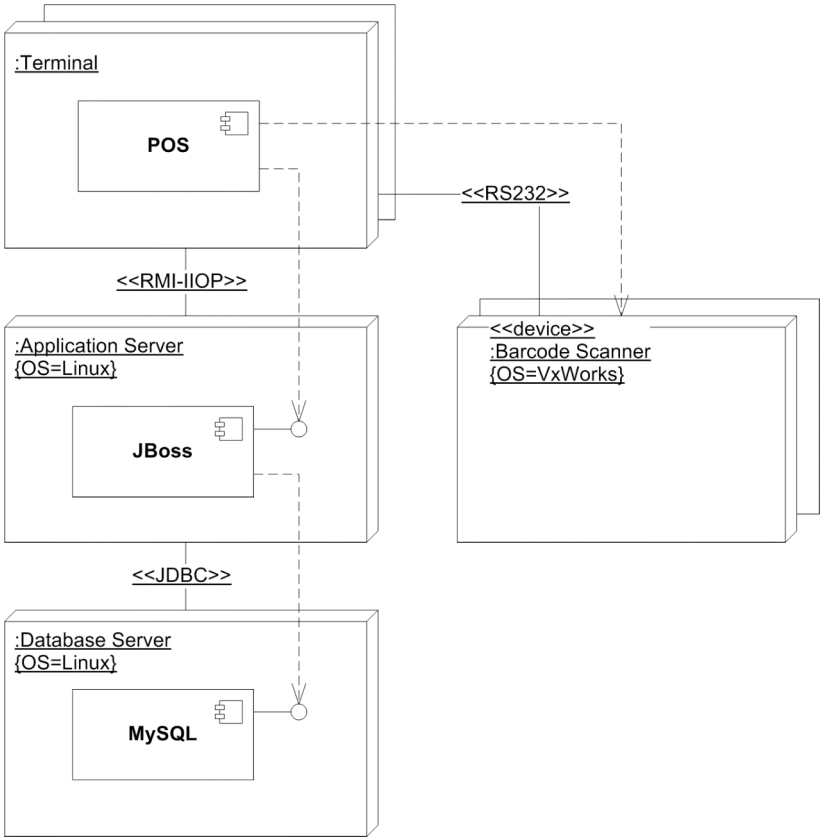
Type	Notation
Mandatory	
Optional	
Alternative	
Or	



Composition rule:
Air conditioning **requires** Horsepower > 100

Rationale:
Manual more fuel efficient

POS...?



Type	Notation
Mandatory	C — F
Optional	C — F
Alternative	C — F1, F2
Or	C — F1, F2

Supporting Variation Points

Changing structure

- Inclusion or omission of elements
  - E.g., through build procedures
- Inclusion of different number of replicated elements
  - E.g., choosing a set of replicated elements in build process
- Selection of elements with same interface but different characteristics
  - E.g., dynamic link libraries

Changing elements

- OO: ”specializing or generalizing”
- Explicit extension points
- Build-time parameters
- Computational reflection
- Overloading

Evaluating Product Lines

Architecture can be evaluated with respect to fitness for purpose

- Evaluation is a topic of next module :-)

Performed on an instance of the architecture

- Foucs on variation points
- Some evaluation artefacts may be reused
  - E.g., (quality attribute) scenarios may be part of reusable assets
  - E.g., checklists

Reevaluate product line architecture on out-of-scope products

Key Areas of Software Product Line

Adoption strategies

Creating products and evolving a product line

Organizational structure

Adoption Strategies

Direction of adoption

- *Top-down* where management decrees the use of product lines
- *Bottom-up* where developers and designers start using a product line approach
- Both approaches work; both need a *champion*

Growth of product line

- Proactive
  - Cf heavyweight
- Reactive
  - Cf lightweight

Evolving a Product Line

Evolution in core assets driven by various sources

External sources

- New versions of assets
- New technology
- User needs and competition change

Internal sources

- New functions added to product
- Evolution of existing products

Organizational Structure

Require an organization to manage asset base

Development department
- Product line practices reside in development unit
- May work for small units (30 people)

Business unit
- Business unit develops part of assets
- Share assets (?)
- Between 30 and 100 people

Domain engineering unit
- Special unit maintains and develops core assets
- Business units build products
- For more than 100 people

Hierarchical domain engineering unit
- Very large product lines may be subdivided hierarchically into subgroups
- Domain engineering unit per subgroup

Summary

Reuse is hard

Software product lines is a constrained way to achieve substantial reuse

- Software intensive systems
- Managed set of core assets
- Products developed based on assets in a prescribed way

Reuse is still hard

- Technically
- Organizationally
- Politically