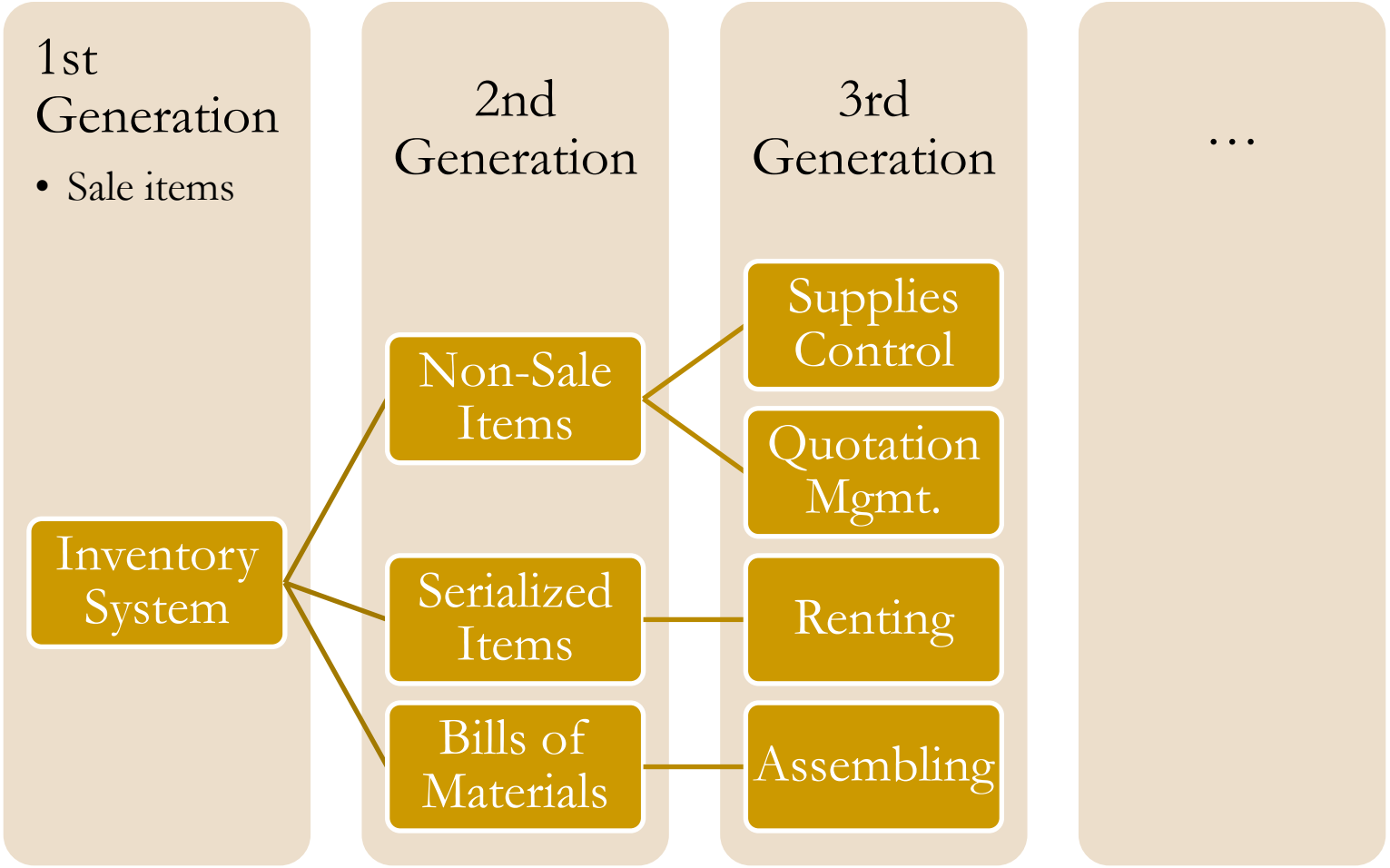
Software Product Lines

Some of the material in these slides is taken from *Software Architecture in Practice, 2nd edition* by Bass, Clements and Kazman.

Some of the material in these slides is taken from *Software Product Lines: Practices and Patterns* by Clements and Northrop.

Typical Software Evolution



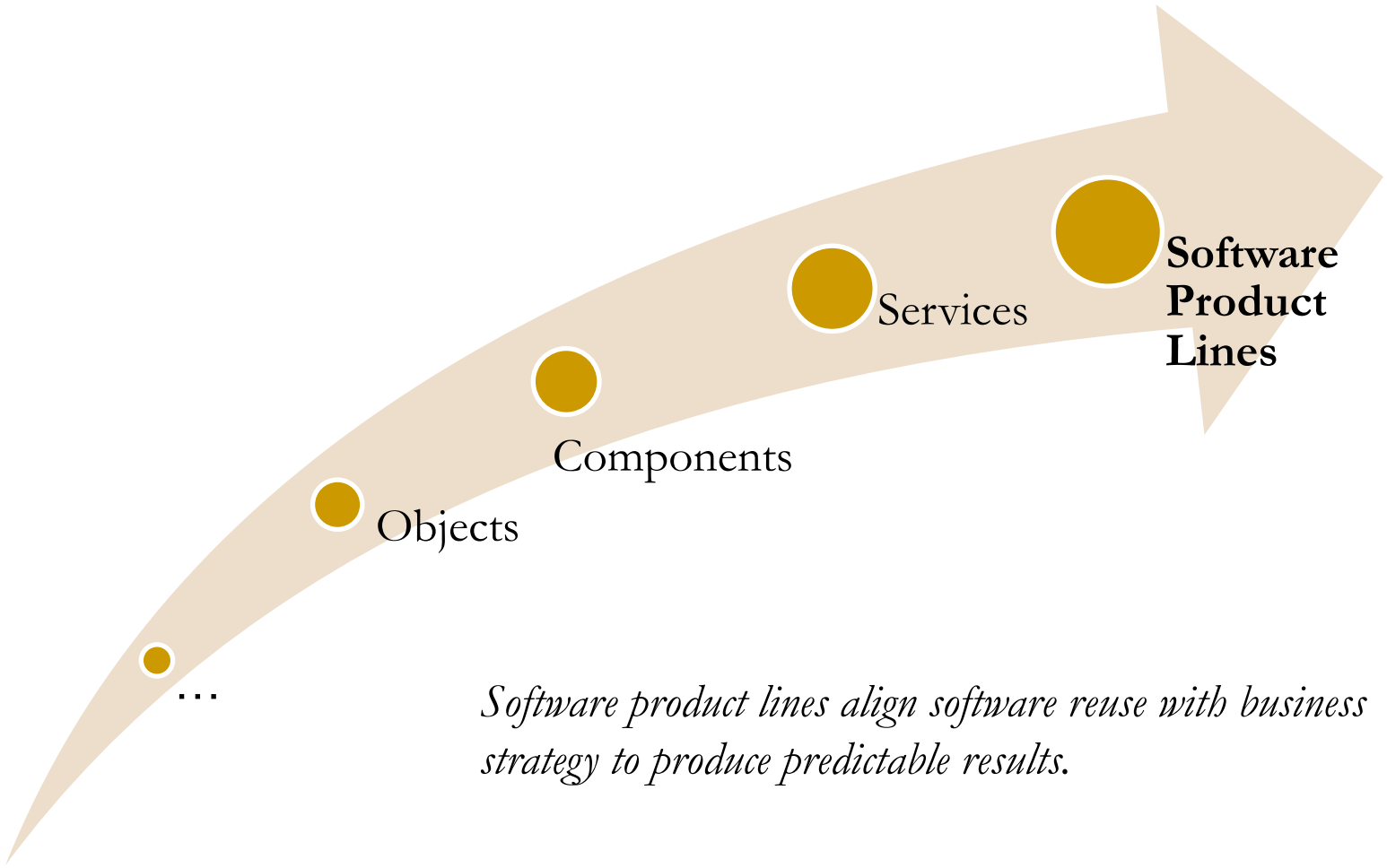
Re-Use Opportunities

- Requirements
- Architectural Design
- Software Elements
- Modeling and Analysis
- Testing
- Project Planning
- Processes, methods, tools
- People
- Exemplar Systems
- Defect Elimination

Failure of Re-Use

- Re-Use libraries
  - too sparse - not useful
  - too rich - too difficult to search
  - elements too small - easier to rewrite
  - elements too large - too difficult to use
  - unknown pedigree
  - architectural qualities – may not be the same

Strategic and Systematic Reuse

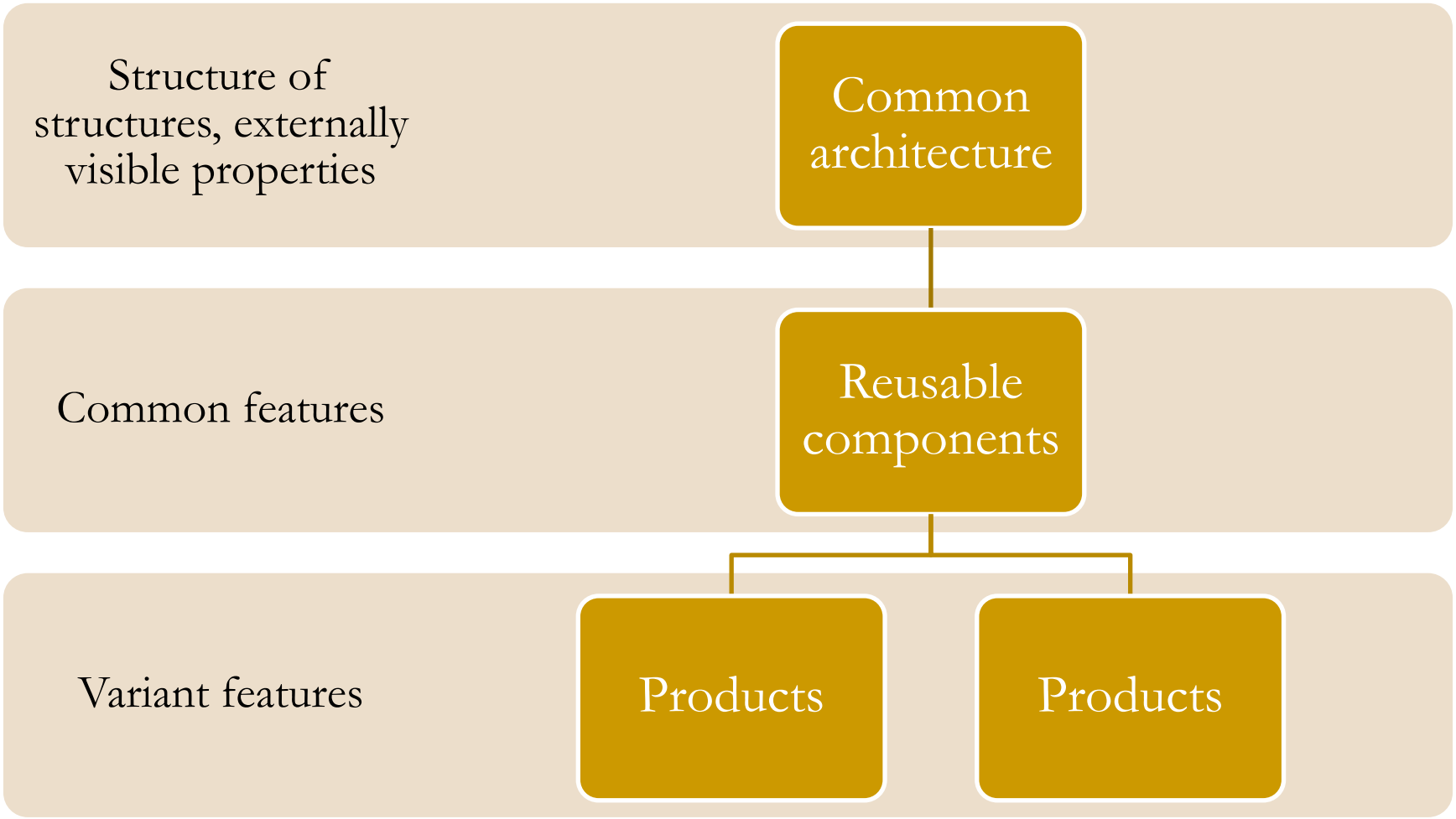


Software product lines align software reuse with business strategy to produce predictable results.

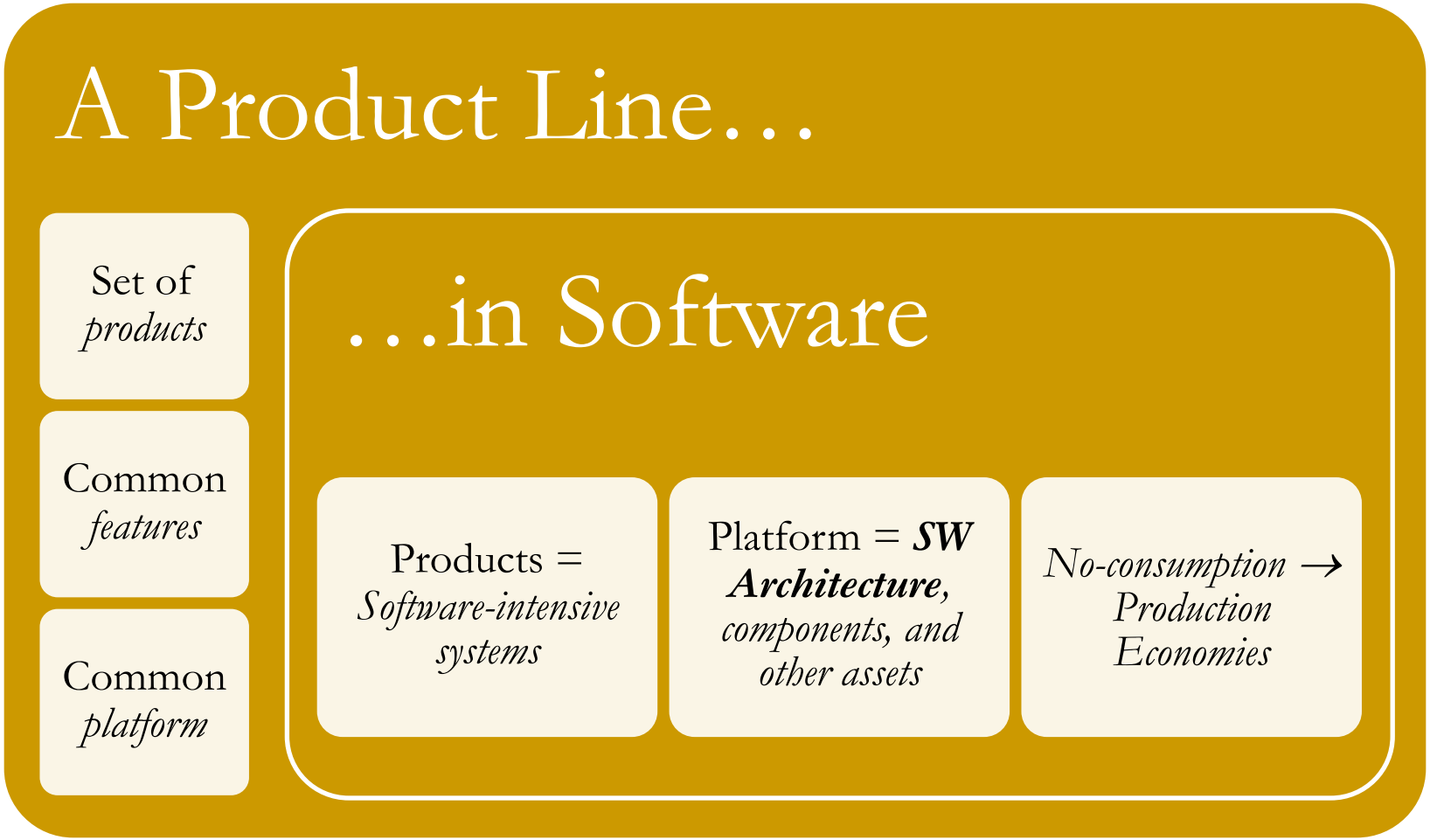
Examples – Successful Product lines

- Nokia – 25 to 30 phones per year (up from 4)
- Cummins, inc., - Reduce time to produce the software for diesel engine from a year to a week
- Motorola – 400% productivity improvement in a family of one-way pagers
- HP – reduced time to market by a factor of seven and productivity by six in printer systems

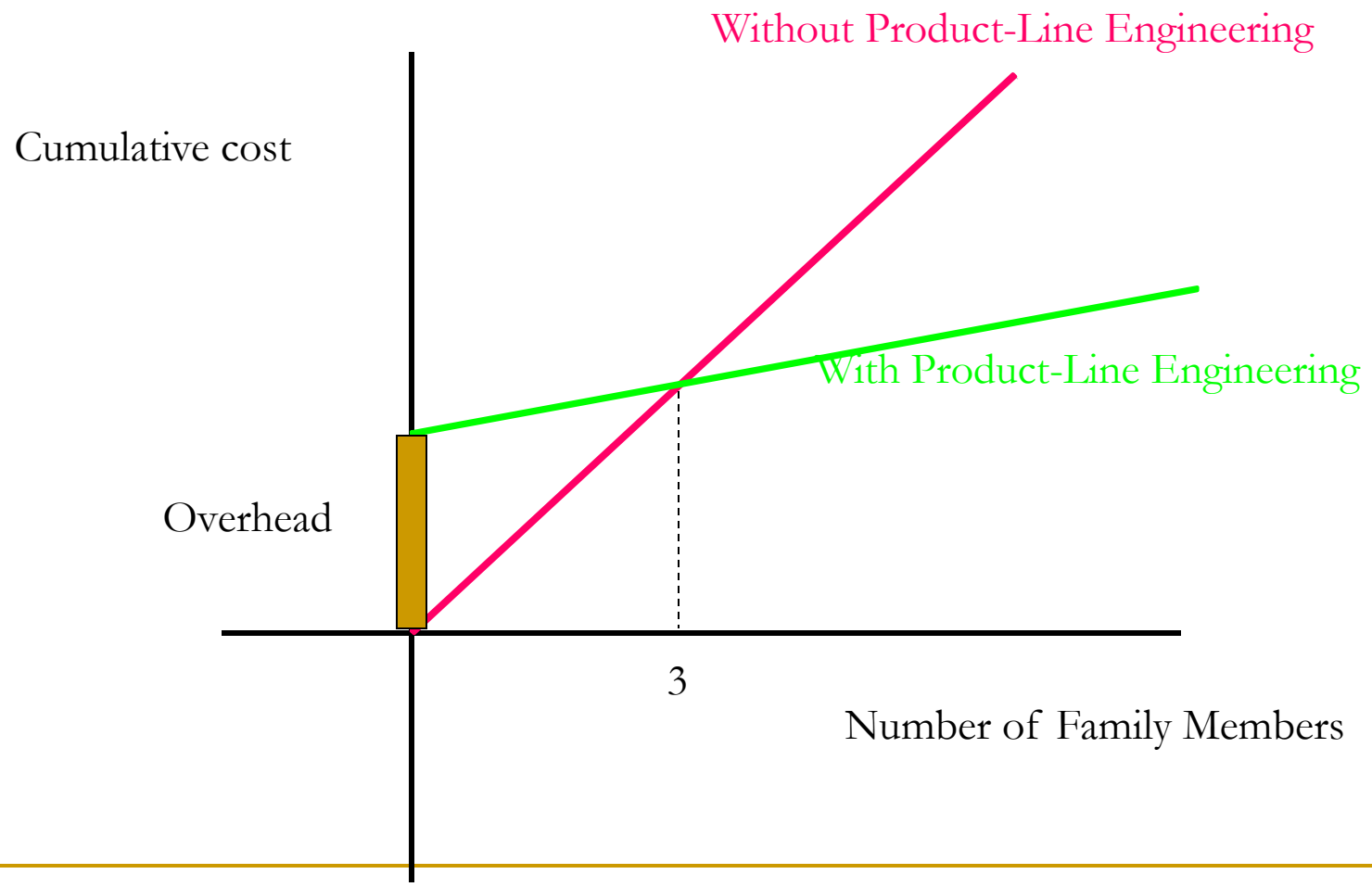
General Pattern of Product Lines



What is a Software Product Line?



Economics Of Families



Scope: How Big is the family?

Consider market segmentation (PL Variability)

- Customers should also have commonalities
- May need separate families for separate markets

Consider supporting technology (Technical Variability)

- Narrow scopes can be automated more easily
- Broader scopes require more process discipline

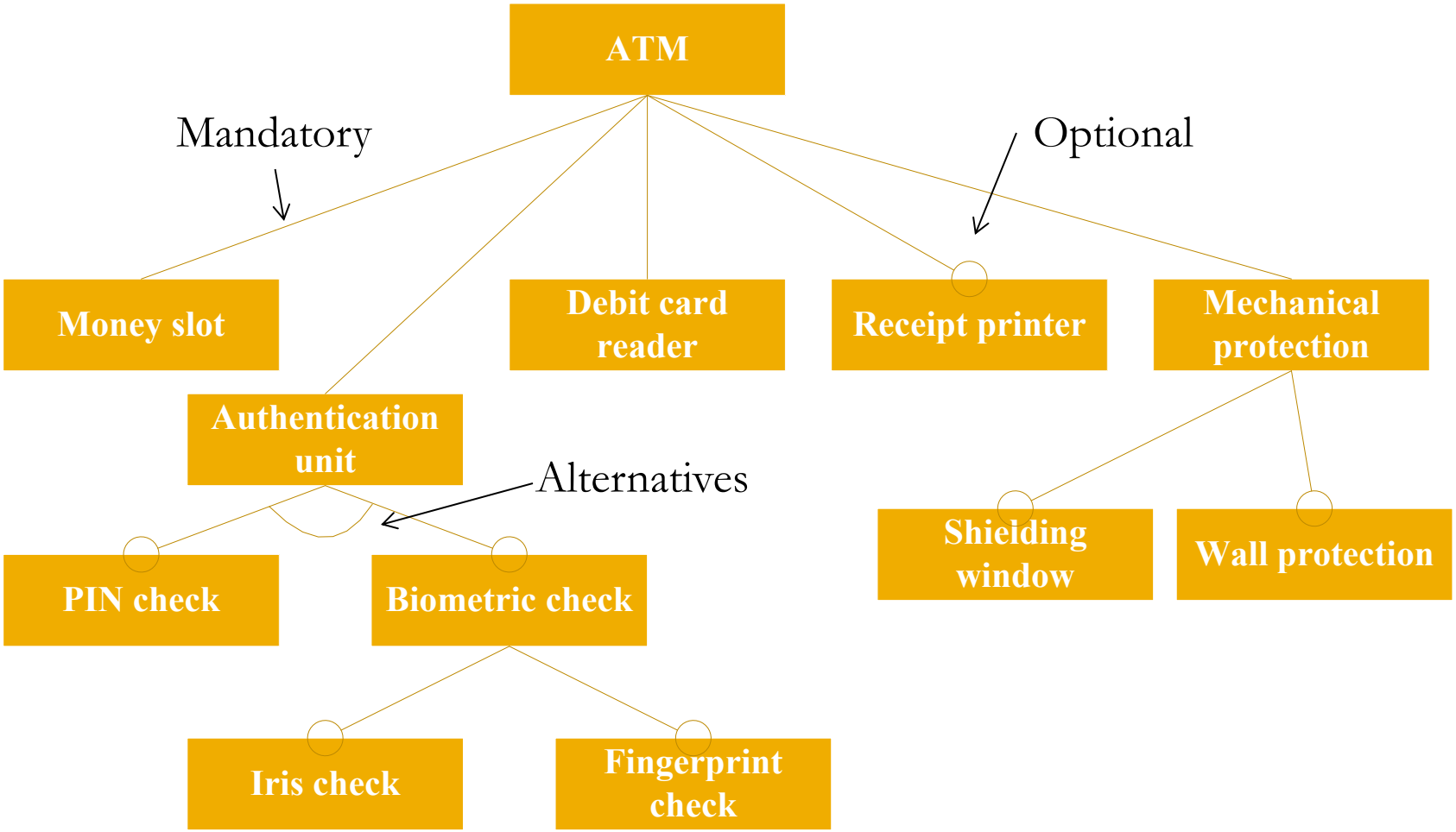
Scoping

- Product Lines are designed in terms of features
  - Logical units of behavior specified by a set of functional and quality requirements
  - Make easier to identify differentiating characteristics among products
- Specification techniques
  - Feature Matrix
  - Feature Model
  - Textual (SCV / Commonality Analysis)

Feature Matrix

Product/Feature	F1	F2	F3	F4	…
S1	X				
S2		X	X		
S3	X	X		X	
S4	X		X		
…					

Feature Model



Commonality Analysis Sections

Dictionary	• Technical vocabulary of the domain
Commonalities	• Assertions about every member of the family
Variabilities	• Assertions about variation across the family
Parameters of Variation	• Type and binding time of variabilities
Issues and To-Do List	• Parking lot for divisive issues or incomplete sections

Commonality Analysis Sections

- Dictionary: Technical vocabulary of the domain
- Commonalities: Assertions about every member of the family
- Variabilities: Assertions about variation across the family
- Parameters of Variation: Type and binding time of variabilities
- Issues and To-Do List: Parking lot for divisive issues or incomplete sections

Dictionary of Terms

- Lists the key terms for the family
  - *unit*, *status*, *action* for configuration control
- Defines technical terms precisely
  - unit - the set of *circuits* and a set of associated protocols (e.g. CLNKs, QLNKs) that comprise a single configurable entity
- Separates *technical terms* from ordinary vocabulary
  - technical terms will appear *italicized* in later text

Commonalities

- Assumptions that are true for every family member
- "Every X has the following attributes..."
  - Every *unit* has a *status*.
  - Every *unit* has a list of allowed *actions* which must include at least: remove and restore.

Variabilities

- Assumptions on how family members differ
- Examples:
  - A *child unit* has one or more *parent units*
  - A *unit* may have 0 or 1 switch *physical unit groups*

Parameters of Variation

link	name	values	default	binding time
V1	*unit* name	alphanumeric strings	none	specification
V5	inhibit state	boolean	false	specification
V23, V24, V25	*input request*	<unit name, unit number, action, ... >	required	run

Issues and To-Do List

- Issues
  - document items that need further thought or investigation, chronologically arranged
  - record decisions
  - document alternatives that were considered

- To-do list
  - tasks that remain, usually with assignments to team members
  - useful when changes leave document in inconsistent state

Key Aspects Peculiar to a SPL Architecture

Identification of Variation Points	• What variations the architecture must support?
Support of Variation Points	• What variation mechanism should be used for each VP?
Integration/ Derivation	• How to ensure the variation mechanism is provided reliably?

Variation Mechanisms

- Inheritance
- Component substitution
- Plug-ins
- Parameterization
- Extension Points
- Aspects
- …

Activity