Product Line Engineering: Introduction

KV Product Line Engineering (343.354)

Dr. Roberto Lopez-Herrejon

Dr. Rick Rabiser

JOHANNES KEPLER UNIVERSITY LINZ

ISSE

2.3.2016

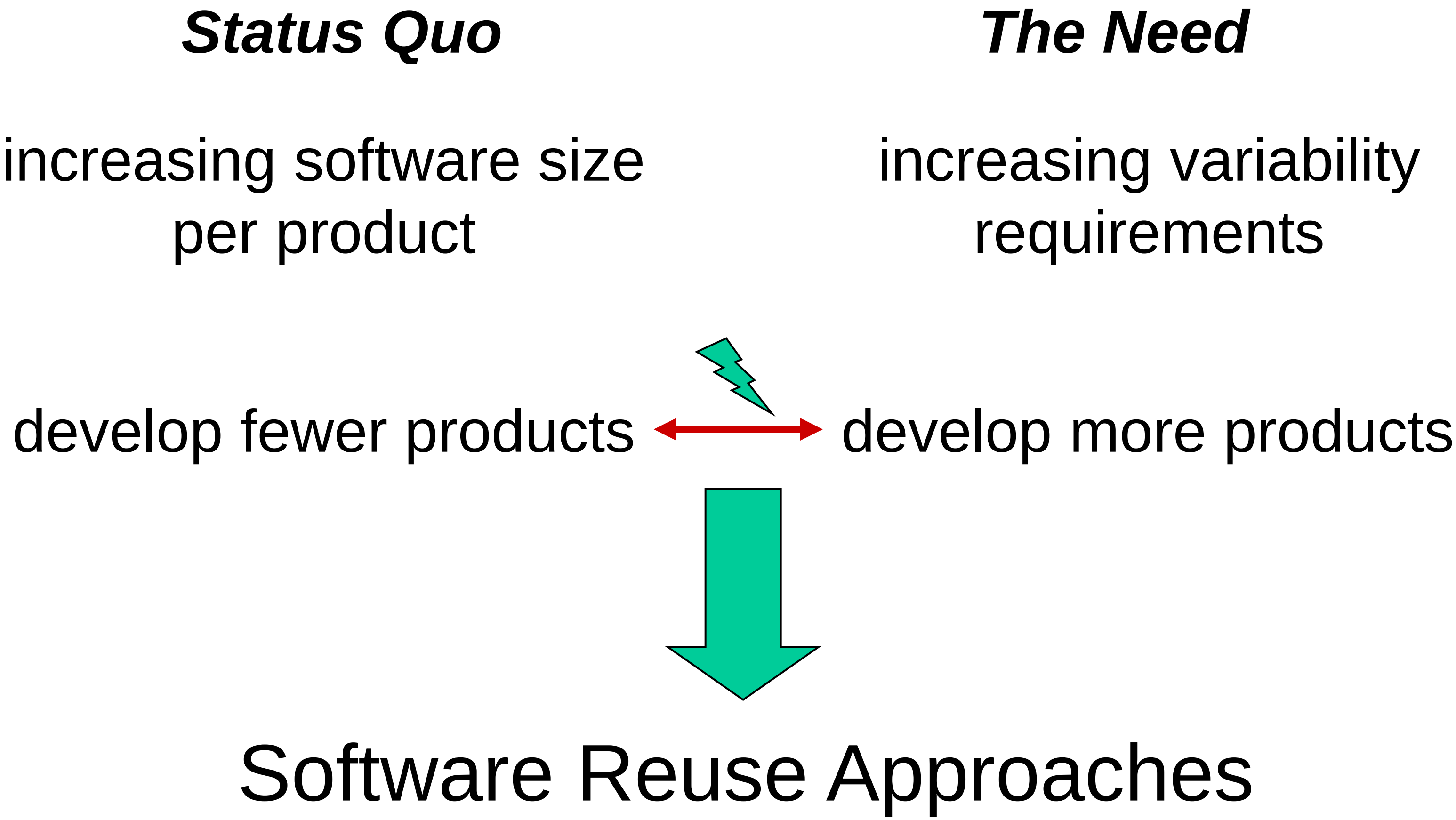
Outline for the lecture

- 2.3. Introduction/Overview (RR/RL)
- **9.3.** Variability Modeling/Management/Implementation + Hand out of **Exercise 1** (RR)
- 16.3. Feature Modeling (RL)
- *23.3./30.3. Easter Holidays*
- **6.4.** Feature-Oriented PLE + **Exercise 2.1** (RL)
- **13.4.** Aspect-Oriented PLE + **Exercise 2.2** (RL)
- 20.4. Product Derivation (RR)
- 27.4. PL Tool Support (RR)
- *4.5. Landespatron St. Florian*
- 11.5. PL Testing (RL)
- **18.5. Invited Talk** Dipl.-Ing. Daniela Rabiser: PLE at Keba AG
- 25.5. PL Evolution (RR)
- **1.6.** PL Case Studies + **Exercise 3** (RR) -- finish presentations at home
- **8.6.** PL Case Studies **Presentations by Students** (Exercise 3) and PL Scoping (RR)
- 15.6. Reverse Engineering SPLs (RL)
- **22.6. Exam (RR/RL)**
- *29.6. Substitute Appointment (in case we have to leave out one lecture, e.g., due to sickness, otherwise: no lecture on this day)*

Outline for Today

- ✓ Outline for KV PLE
- ✓ General information on KV PLE
- What are Software Product Lines?
- History of SPLs
- Product Line Engineering
- Software variability
- Typical industry problems and experiences

Software Challenge



Status Quo

The Need

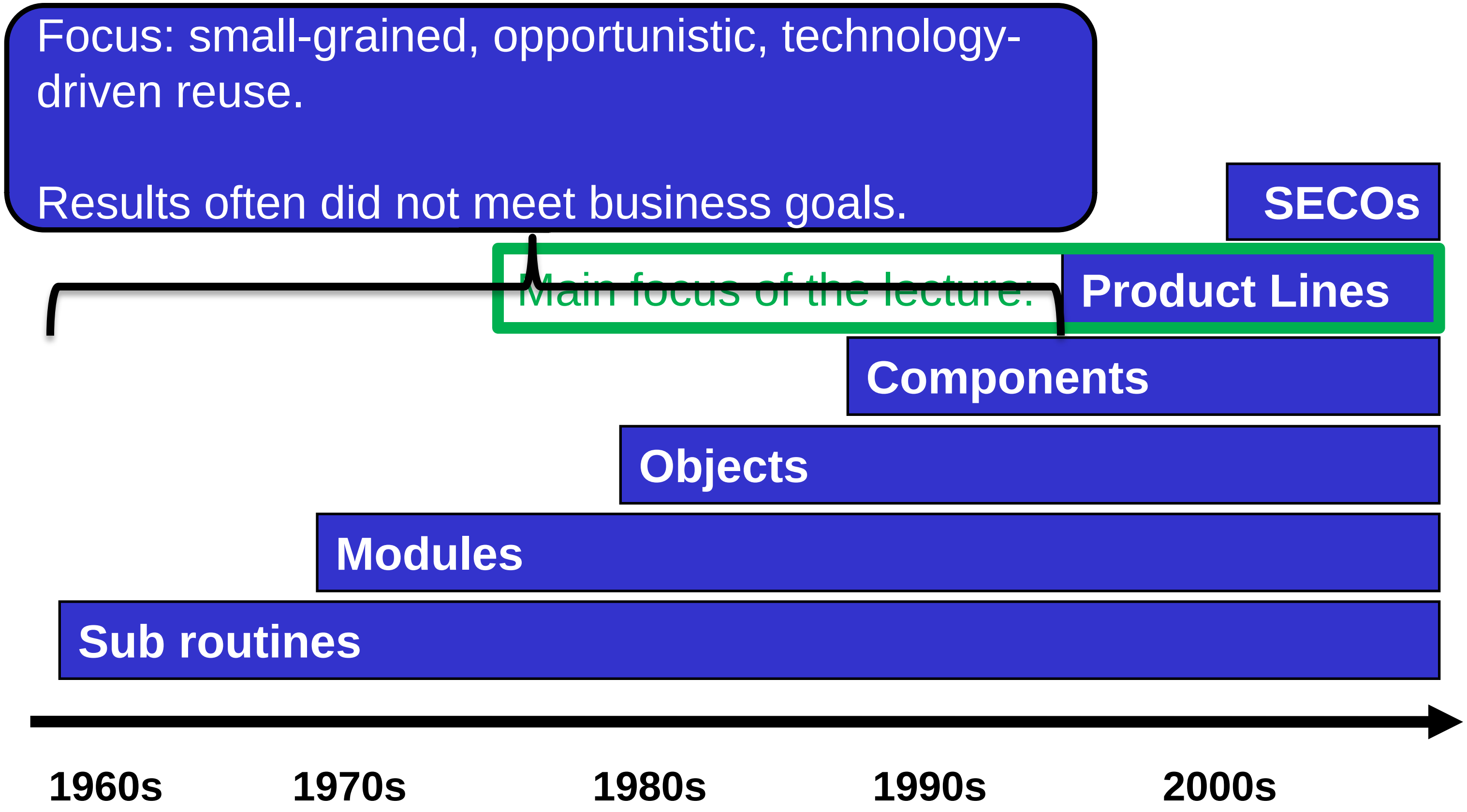
increasing software size per product

increasing variability requirements

develop fewer products ↔ develop more products

Software Reuse Approaches

History of Reuse



Focus: small-grained, opportunistic, technology-driven reuse.

Results often did not meet business goals.

Main focus of the lecture:

SECOs

Product Lines

Components

Objects

Modules

Sub routines

1960s

1970s

1980s

1990s

2000s

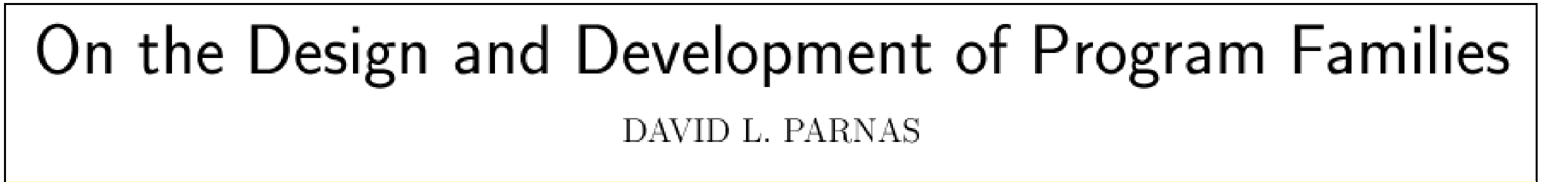
From Reusable Components to Product Lines

Product Lines

Building blocks / Assets

What are Software Product Lines?

- The move from one of a kind software development to large-scale, planned reuse
- First touted by Dijkstra and Parnas in the early 1970s (as program families)

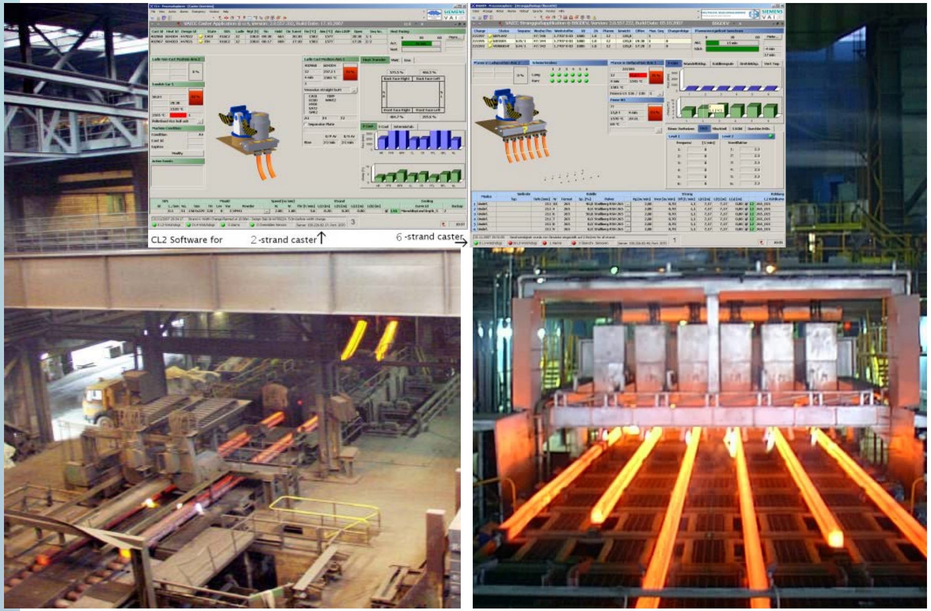


On the Design and Development of Program Families

DAVID L. PARNAS

Sets of programs whose common properties are so extensive that it is advantageous to study the common properties of the programs before analyzing individual members

SPL Definition



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.

Clements, Northrop: 'Software Product Lines – Practices and Patterns', Addison-Wesley, 2001

Marketed SPLs (vs. Engineered SPLs)

- Definition 1:
  - A set of products that are **marketed together** as **sharing a common set of concepts or features**.
  - Example: **different Windows versions**
- Definition 2:
  - A set of products that share some concepts and **complement each other in terms of their capabilities**.
  - Example: Word, Excel, etc. are part of the **office suite**

- ***Both definitions are independent of the engineering approach!***

Engineered SPLs

- A **set of products** that are **engineered together** so as **to share major parts of their implementation**.
  - this definition is **independent of the marketing** of the product line!
  - different products in the product line may belong to different marketed product lines
  - Example: different systems by a producer sold under different brands

- In this lecture we will mainly focus on the Engineered (Software) Product Line!

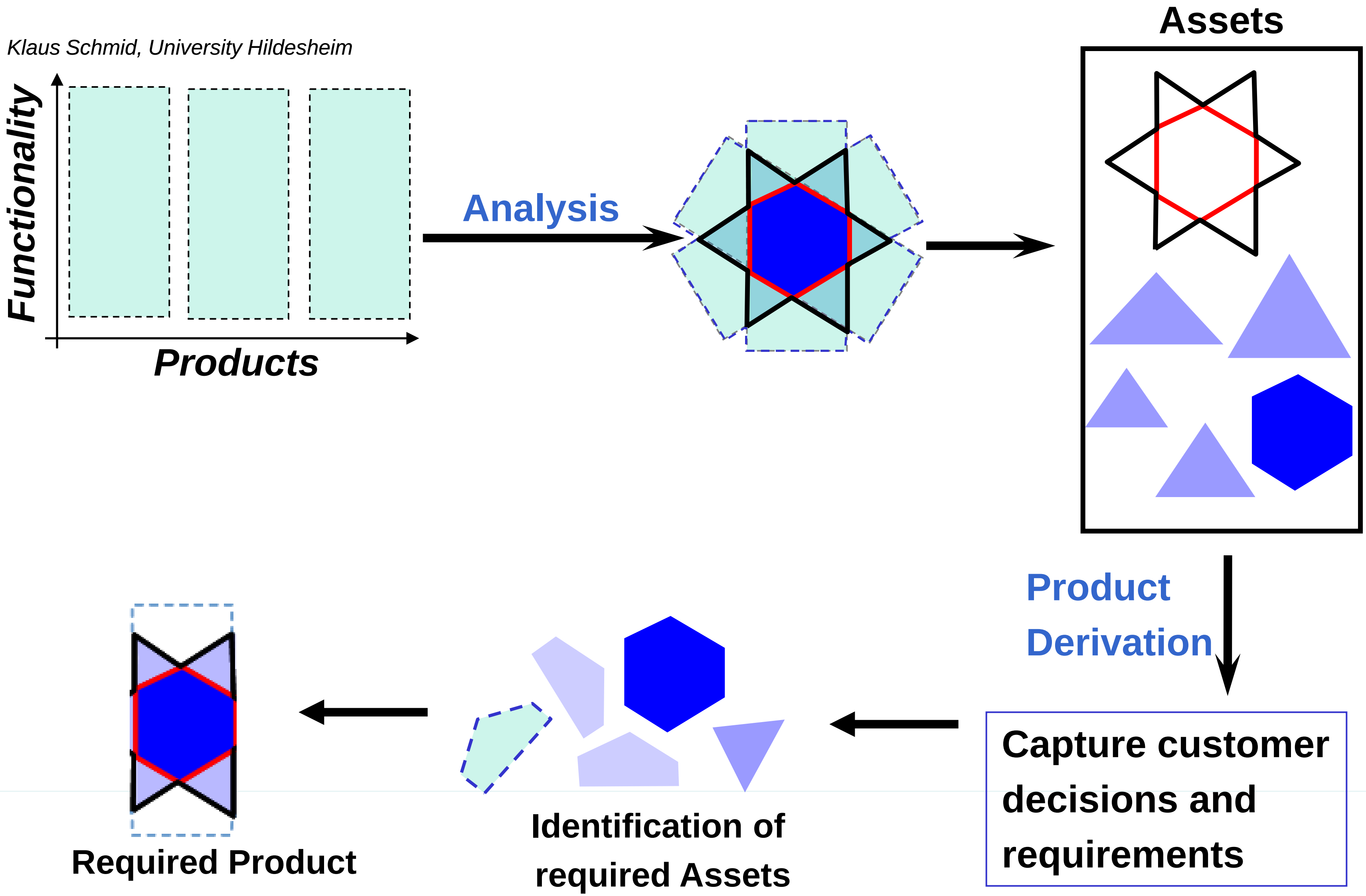
What Product Lines are NOT

- Fortuitous small-grained reuse
  - In PLE, reuse is **explicitly planned, enabled, and enforced**
- Single-system development with reuse
  - In PLE, assets are **designed/developed for reuse**
- Just component-based development
  - PLE defines **assembly of components**, documents **variability**, and integrates **management** aspects
- Just a reconfigurable architecture
  - Just **one level of interest** in PLE
- Just a set of technical standards
  - Can be input to PLE, but again, this is just one level of interest

Clements, Northrop: 'Software Product Lines – Practices and Patterns', Addison-Wesley, 2001

The 2 Life Cycles

(c) Klaus Schmid, University Hildesheim



Functionality

Products

Analysis

Assets

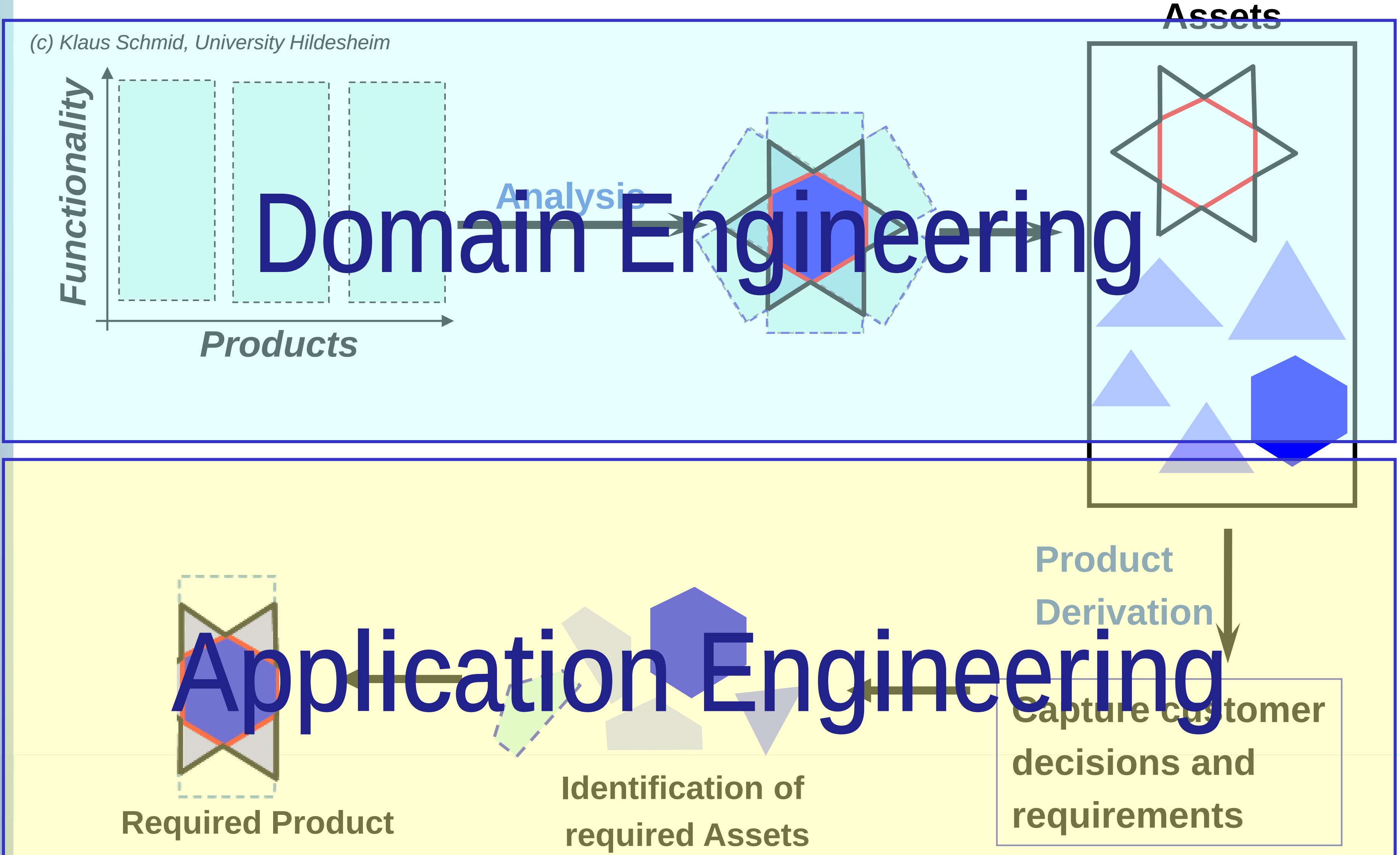
Product Derivation

Capture customer decisions and requirements

Identification of required Assets

Required Product

The 2 Life Cycles



Both Lifecycles revolve around Variability

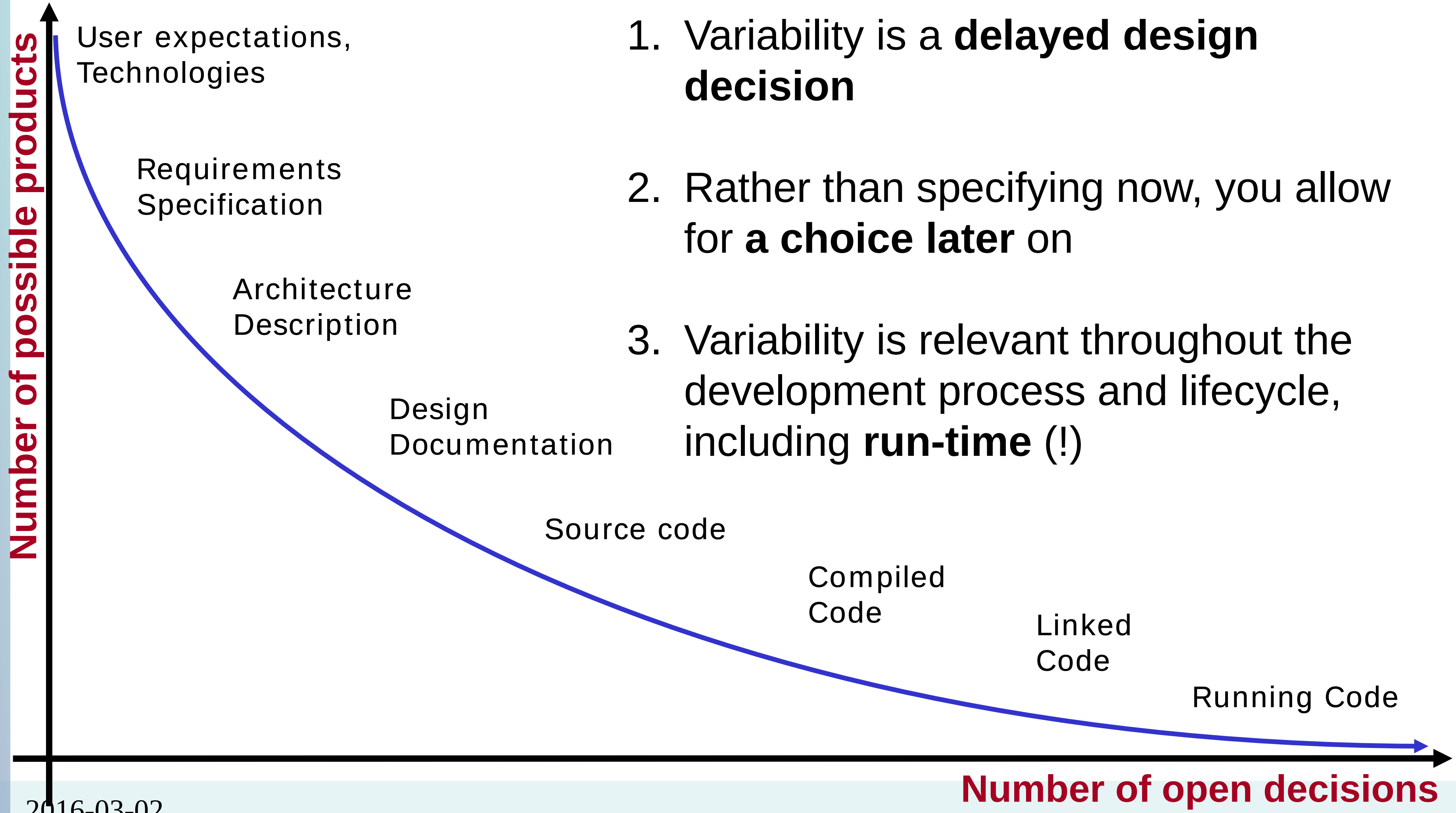
- Reuse = using an existing piece of software in a different context
  - Must be possible to adapt software to new contexts
  - Ad-hoc reuse does not work, the software needs to be prepared for reuse

- Software without variability is not reusable

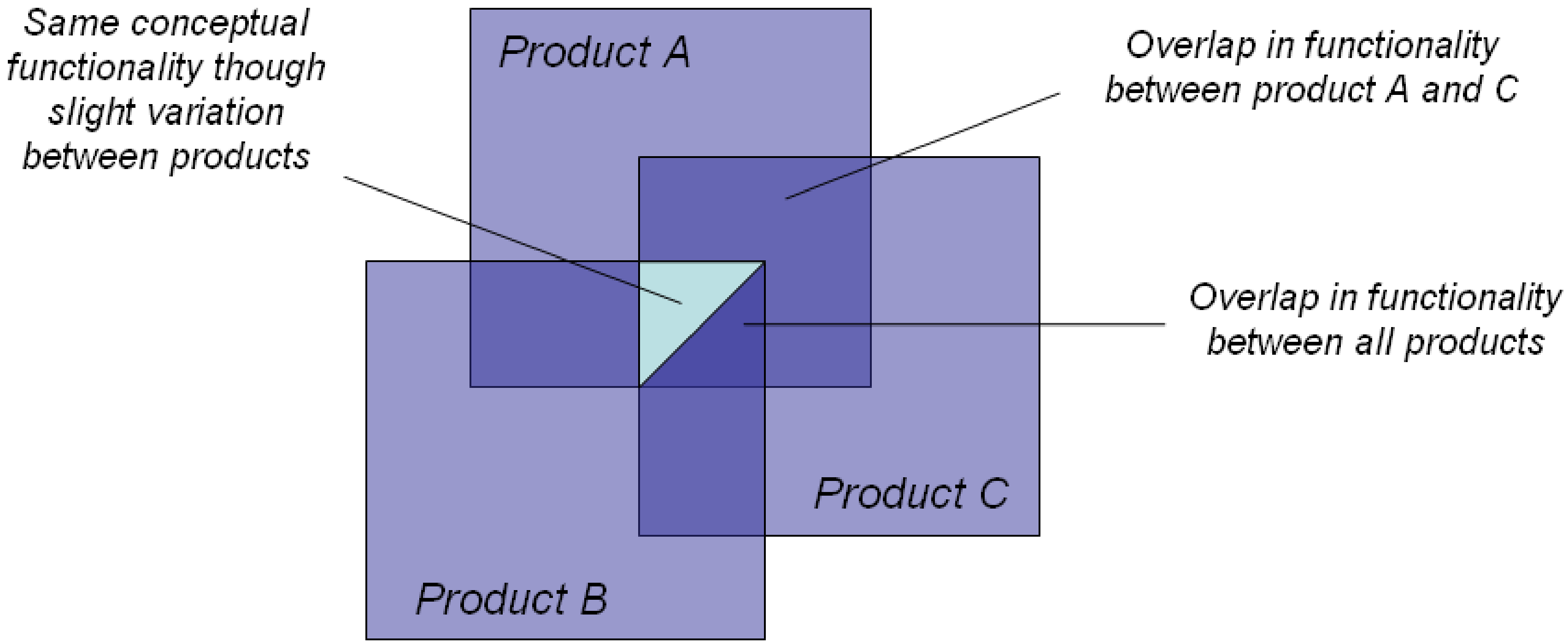
- **Achieving reuse relies on understanding, documenting, and managing variability**

What is Variability?

1. Variability is a **delayed design decision**
2. Rather than specifying now, you allow for **a choice later** on
3. Variability is relevant throughout the development process and lifecycle, including **run-time** (!)

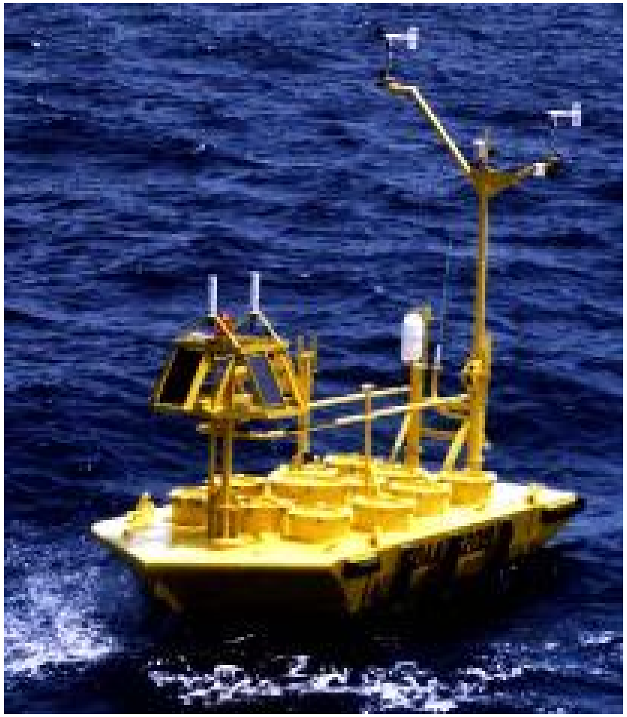


Commonality and Variability Analysis


Same conceptual functionality though slight variation between products
Product A
Overlap in functionality between product A and C
Overlap in functionality between all products
Product C
Product B

Example: Floating Weather Stations

- Commonality Analysis:
  - **Commonalities**
    - “All FWS shall report the current temperature.”
  - **Variability**
    - “Some FWS may report the wind direction.”
  - **Constraints**
    - “Any FWS that reports the wind direction must also report the wind speed.”



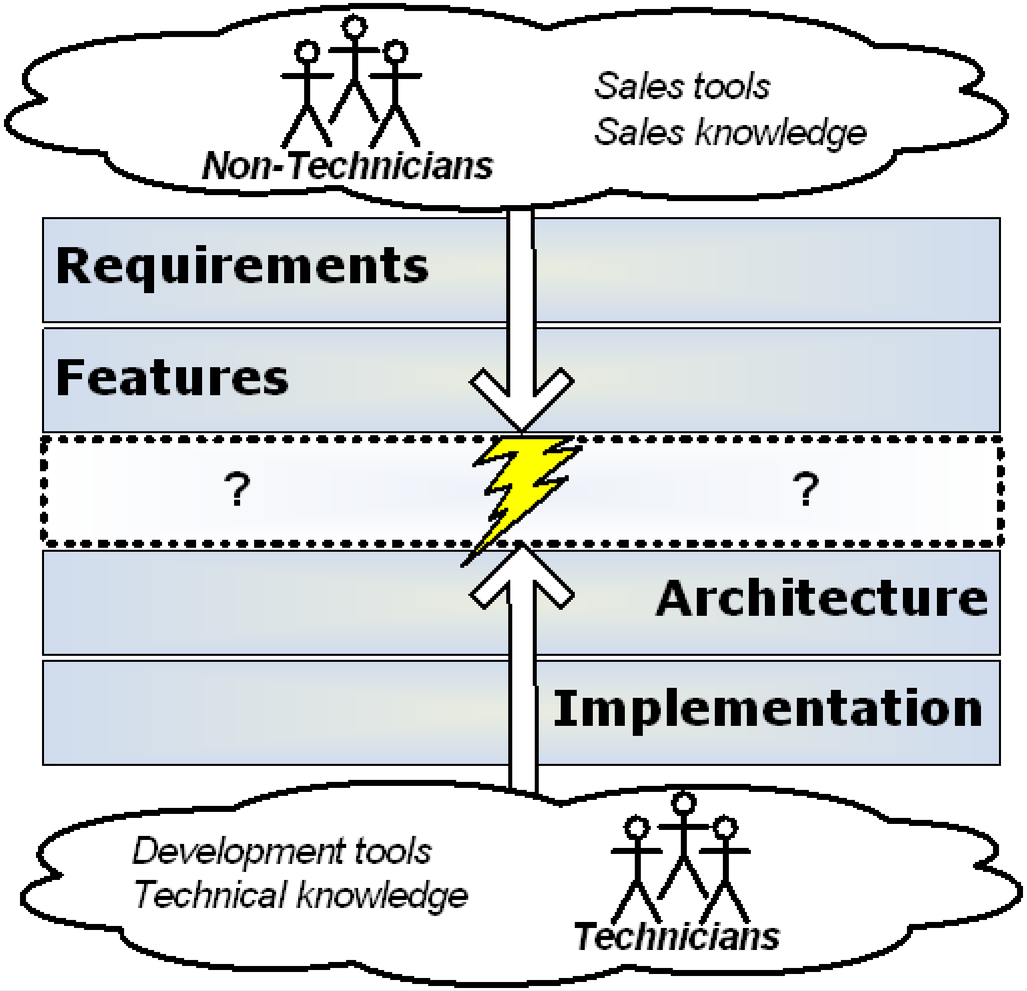
Source: National Data Buoy Center

External vs. internal variability

- **External** variability as **visible to customers**
  - manual vs. automatic transmission
  - the camera of your smartphone may or may not have an auto-focus and you may have different resolution options
  - …

- **Internal** variability managed **under the hood**
  - battery technology in hybrid electric car
  - communication protocol
  - …

Bridging internal and external variability is difficult



Different

- Views
- Languages
- Knowledge
- Notations
- Objectives
- ...

Typical Industry Problems

- Understanding and modeling variability
  - Specifications, code, configuration files, documentation, …
  - Requires know-how and experience
  - **Knowledge about variability is concentrated in few people**

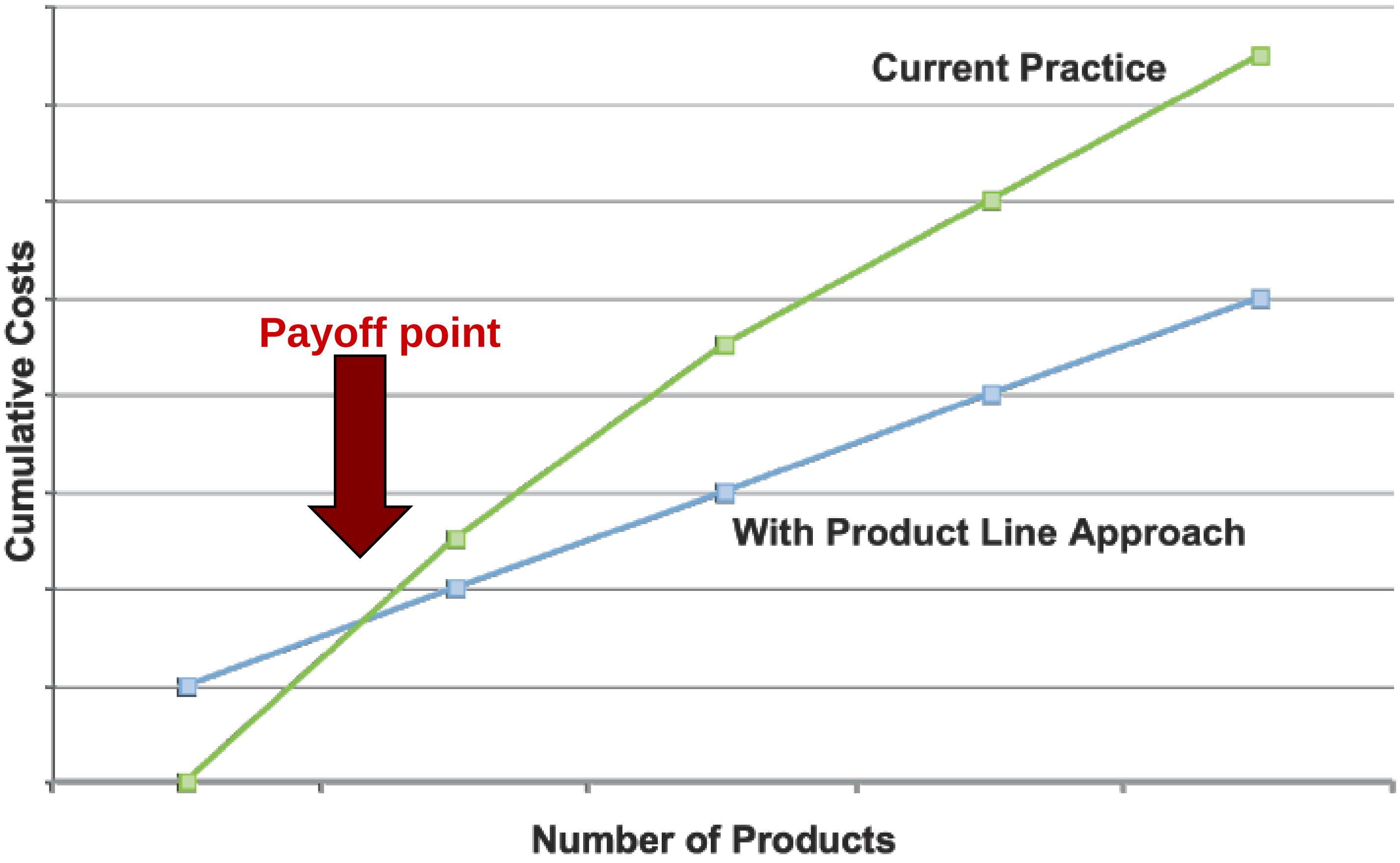
- Integration of existing variability mechanisms
  - Isolated view on variability, effects of decisions are not visible for different user groups
  - **Linking different views is difficult**

- Relationship between project decisions and technical configuration
  - **Inadequate coordination between engineering and sales**
  - Leading to redundant data-entry and -management

PLE Industry Experiences

- Experience Reports from bigger and smaller companies
    - http://www.softwareproductlines.com/
- SEI’s Product Line Hall of Fame
    - http://www.sei.cmu.edu/productlines/plp_hof.html
  - Short experience papers and links to further information
    - Boeing
    - Siemens
    - Ericsson Telecommunication
    - General Motors Powertrain
    - Lucent / AT&T
    - Nokia Mobile Phones
    - Philips Consumer Electronics
    - Philips Telecommunication
- Software Product Line Conferences
  - http://www.splc.net

Economics of Product Lines



Weiss. D.M. & and Lai, C.T.R..
Software Product-Line Engineering: A Family-Based Software Development Process
Reading, MA: Addison-Wesley, 1999.

SPL Value Proposition

- Increased quality by as much as 10x
- Decreased cost by as much as 60%
- Decreased labor needs by as much as 87%
- Decreased time to market (to field, to launch...) by as much as 98%
- Ability to move into new markets in months, not years

PLE Approaches

- 3-Tiered Methodology *(Krueger, BigLever, Inc.)*
- Cardinality-based Feature Modeling and Staged Configuration *(Czarnecki et al., Univ. Waterloo)*
- COVAMOF *(Deelstra, Sinnema, et al., Univ. Groningen)*
- DOPLER *(Dhungana, Rabiser, Grünbacher et al, JKU)*
- KobrA *(Atkinson et al., Fraunhofer IESE)*
- Kumbang/Koalish *(Männistö et al., Helsinki Univ. of. Tech.)*
- Orthogonal Variability Modeling *(Pohl et al., Univ. Duisburg-Essen)*
- PLUS *(Gomaa, George Mason University)*
- PuLSE *(Bayer et al., Fraunhofer IESE)*
- SEI Product Line Practice Initiative *(Northrop et al., SEI CMU)*
- ...

Existing PLE Tools

- Commercial
  - pure::variants (www.pure-systems.com)
  - GEARS (www.biglever.com)
- Research
  - Feature IDE (http://wwwiti.cs.uni-magdeburg.de/iti_db/research/featureide/)
  - DOPLER (ase.jku.at/dopler)
  - Kumbang (http://www.soberit.hut.fi/KumbangTools/)
  - http://www.splot-research.org/
  - etc.

Summary

- Software reuse is essential
- Systematic variability management!
- Product Line Engineering has developed a wide array of methods and tools
- Tool support is essential

References

Books

- P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*: SEI Series in Software Engineering, Addison-Wesley, 2001.
- **K. Pohl, G. Böckle, and F. van der Linden, *Software Product Line Engineering: Foundations, Principles, and Techniques*: Springer, 2005.** **Hand-out**
- F. van der Linden, K. Schmid, and E. Rommes, *Software Product Lines in Action - The Best Industrial Practice in Product Line Engineering*: Springer Berlin Heidelberg, 2007.

Overview Papers (IEEE Software)

- A. Birk, G. Heller, I. John, K. Schmid, T. von der Maßen, and K. Müller. *Product Line Engineering: The State of Practice*. IEEE Software, 20(6), pp. 52-60. 2003.
- F. van der Linden. *Software Product Families in Europe: The Esaps & Cafe Projects*. IEEE Software, 19(4), pp. 41-49. 2002.
- L. Northrop. *SEI's Software Product Line Tenets*. IEEE Software, 19(4), pp. 32-40. 2002.

Next Week

- 9.3. Variability Modeling, Management, and Implementation (RR)

- Exercise 1

Questions?

- Now or later to rick.rabiser@jku.at | roberto.lopez@jku.at