Software Evolution in the Financial Industry

Carl Worms

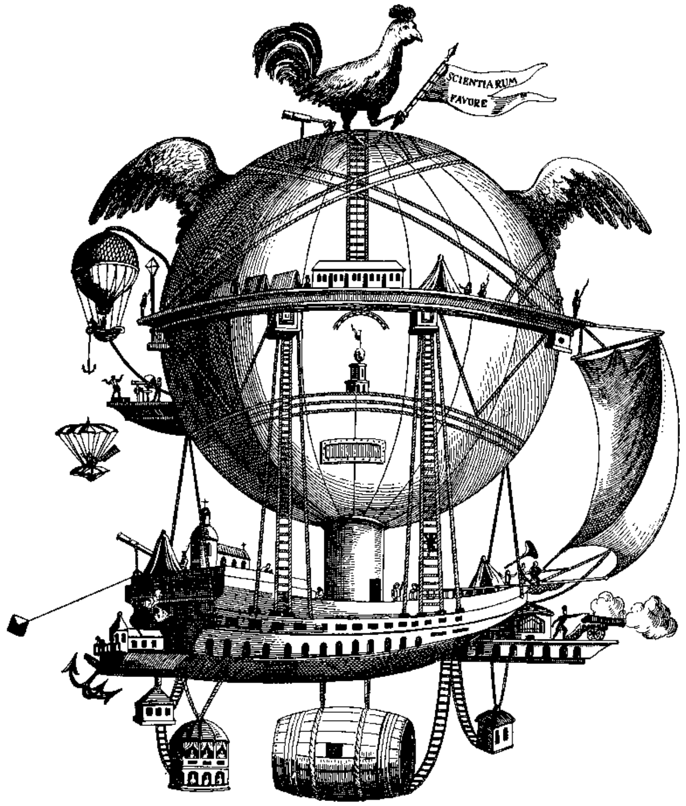
Guest Lecture 2014

IAM Bern

Who I am

- Studied Computer Science in Karlsruhe
- Lived from programming for 15 years
- Walter Masing awardee (DGQ)
- IT Architect/SWE Process Architect at a major Swiss bank
- PC member of IEEE conferences since 2007

IT in Large Banks?



IT in large banks

- Huge global infrastructure
  - >100.000 PC/Laptops
  - ~30.000 Servers and a big mainframe
  - A multitude of technologies (HW, OS, DB, …)

  - ~5.000 Applications, >100m SLOC
  - ~2.000 SOA like Services
  - ~50.000 Data Bases, >100.000 data attributes

IT in large banks

- Huge global IT organization
  - ~25-30% of the bank's staff is in IT
  - The big Swiss banks are the major IT employers
  - Global distributed SWE
  - ~50 high level processes areas (one is SW design)
  - A multitude of SWE methodologies
  - A multitude of cultures

IT in large banks

- High performance
  - >100m transactions/month
  - >10 tons of prints (accounts, tax reports)/year
- High security
- High regulatory constraints

- Extreme change rate (monthly/quarterly)
- >20.000 bug fixes/year in a major hub

Question 1

If you are the CIO (Chief Information Officer) of a large bank - what’s your major concern?

Answer

Question 2

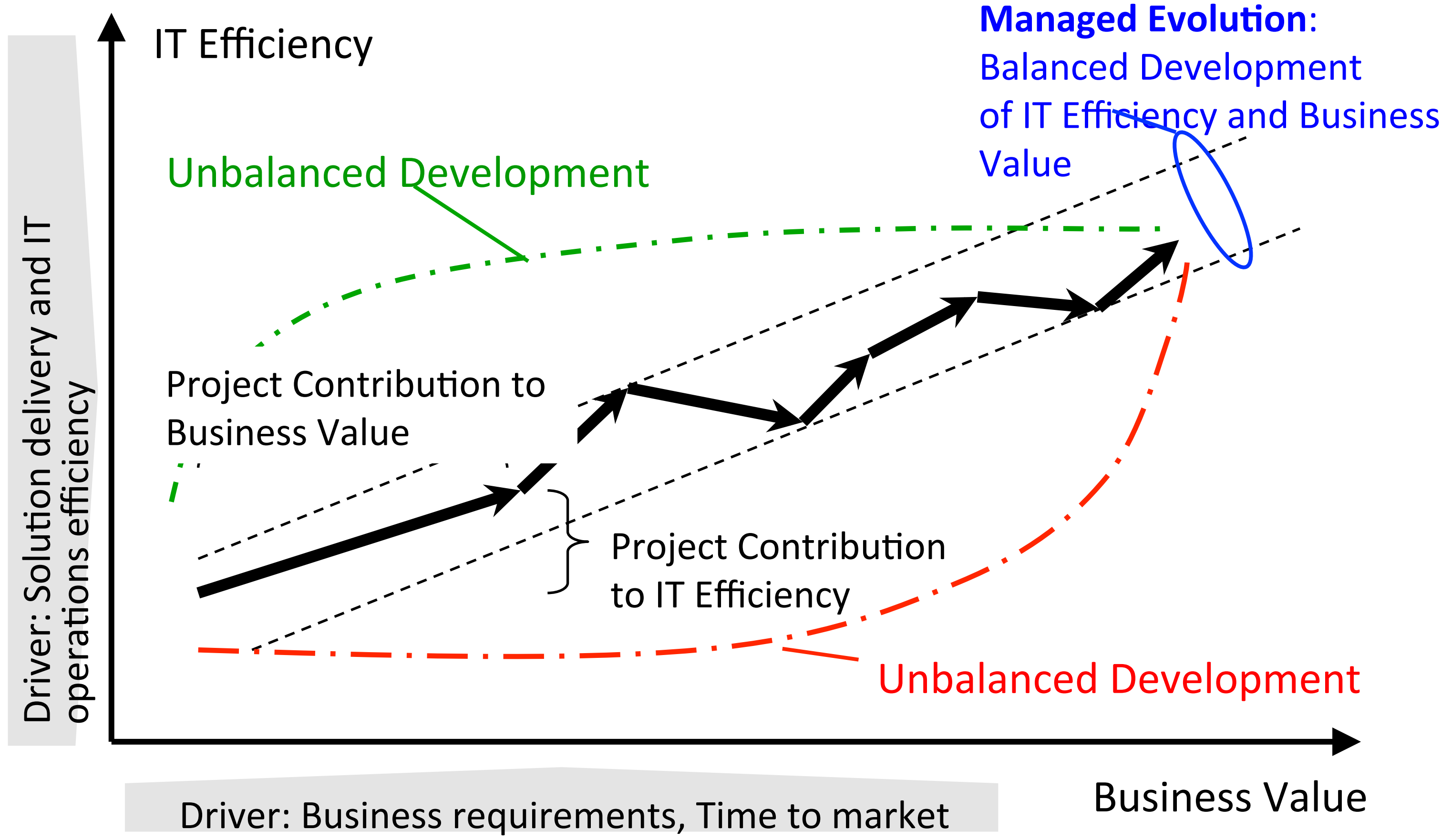
If you are the Chief Architect of a large bank - what’s your long-term vision?

Answer

Managed Evolution

From: Murer/Bonati/Furrer, Managed Evolution; Springer 2010

Managed Evolution


IT Efficiency
Managed Evolution: Balanced Development of IT Efficiency and Business Value
Unbalanced Development
Project Contribution to Business Value
Project Contribution to IT Efficiency
Unbalanced Development
Driver: Solution delivery and IT operations efficiency
Driver: Business requirements, Time to market
Business Value

From: Murer/Bonati/Furrer, Managed Evolution; Springer 2010

Software Evolution Management called IT Architecture

All systems have an architecture, be it an implicit or an explicit one

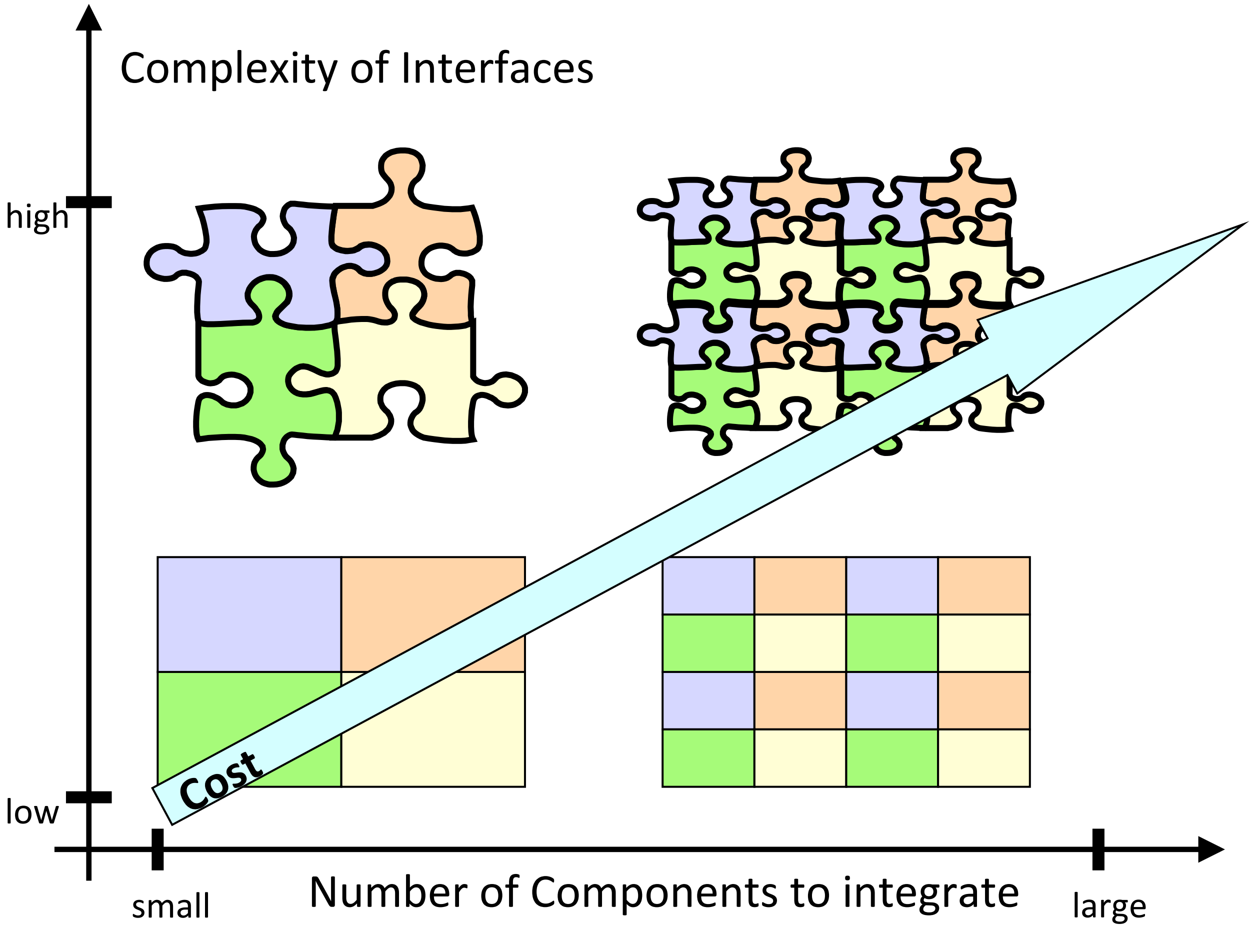
If the architecture is implicit, we have no way to control, analyze, reason about, evolve, and communicate it

It is the role of the architect to establish an explicit architecture of the system

Software Evolution Management Principles

- **Flexibility** with regard to business organization and expansion
  - applications tend to live longer than the organization …. Thus, we keep the architecture flexible (-> Multiple Channels, Multiple Countries, …..)
- **Componentization**
  - The whole platform is decomposed into components with well-defined interfaces among each other.
  - Components encapsulate data and related functionality
  - Application domains serve as high-level components
  - The decomposition into components is supported by an adequate piece of integration infrastructure

Software Evolution Management Principles - Componentization



Software Evolution Management Principles

- Accept and handle **trade-offs**
  - Architecture is not an exact science
  - There are conflicting goals and interests

  => necessity of architecture process to define standards…and to handle exceptions
- The **importance of standards**
  - The main property of a standard is its widespread acceptance and support in the market
  - The discussion of architecture will generally focus on standards rather than products
  - It is an important duty of IT Architecture to enforce guidelines that restrict the use of the product to standard features
- **Avoid unnecessary technology diversity**
  - each infrastructure functionality is covered by exactly one product

    … but temporary overlaps when old standard is being replaced by new one

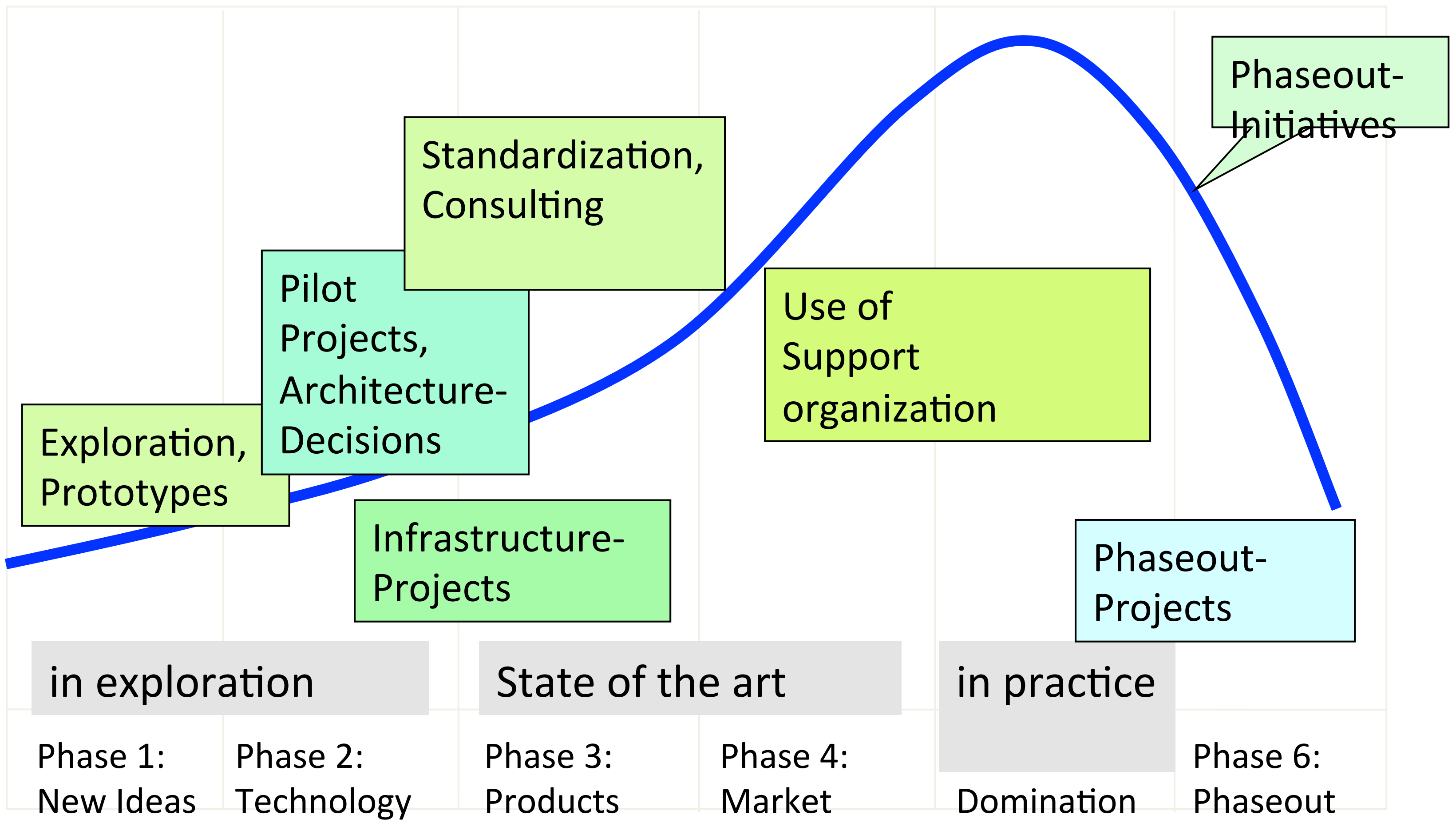
    … exception: exert price pressure on vendors

Software Evolution Management Principles

- **Be properly positioned in the technology life cycle**
  - there is a lot of hype around life cycle
    … a careful technology portfolio management is important
  - Managed Evolution makes it possible to predict the replacement of a specific technology and to allow for an adequate transition phase

- **Stay in themainstream**
  - Generally our technology strategy will be an ‘early follower strategy’ of mainstream products
  - exceptions: Security, Channel technology, Large system integration

Software Evolution Management Principles – Lifecycle Management



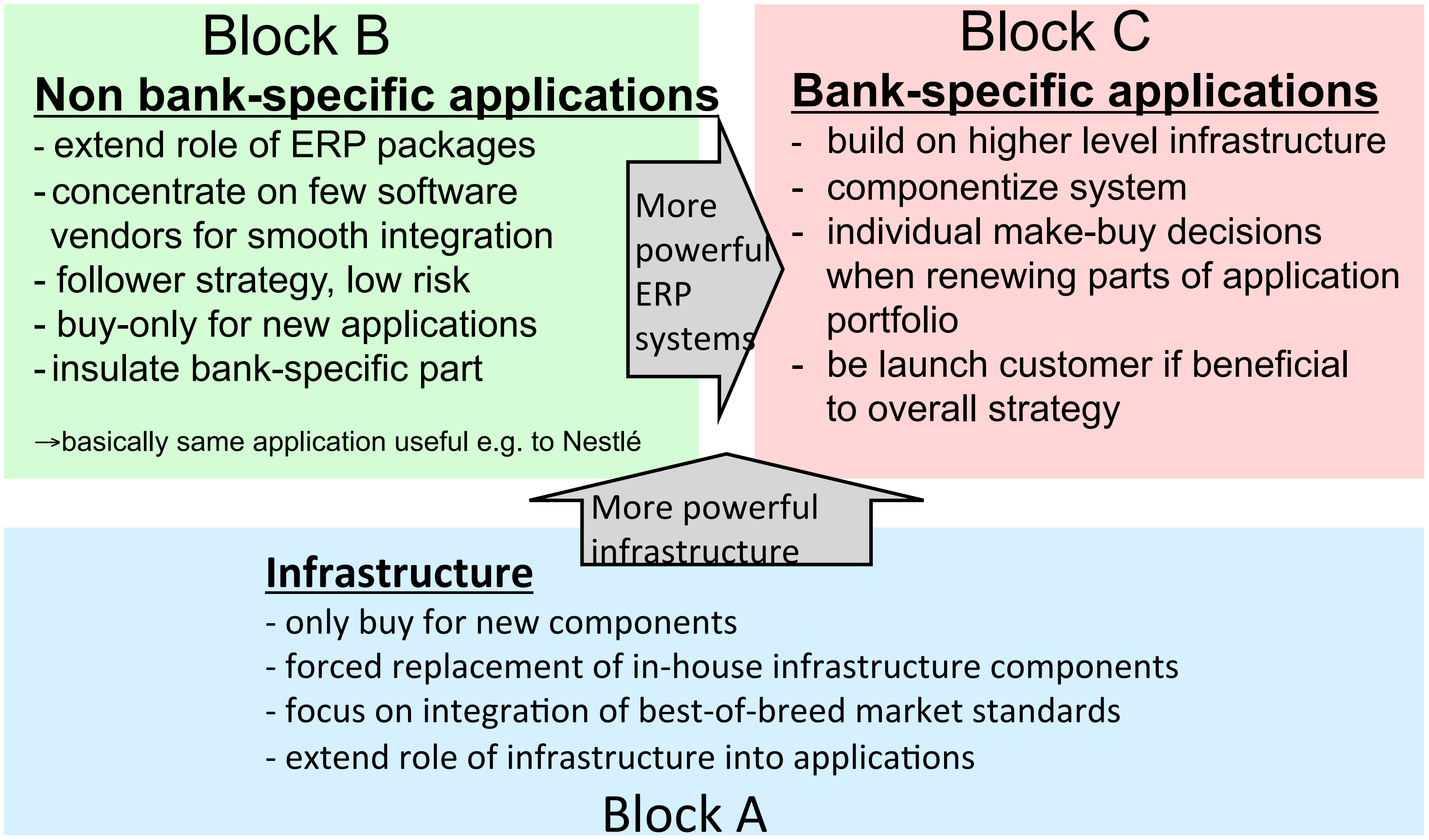
Software Evolution Management Principles

- **Adequate make or buy decisions**
  - Purchase Non bank-specific applications and technology infrastructure
  - Development activity should be concentrated on core business functions
  - ERP systems/Standard Software will pick up bank functionality

➢ business processes have to be adapted to given processes of bought applications

➢ A highly developed integration architecture is key for successful standard software projects

Software Evolution Management Principles – Focus on Core Business



Block B

Non bank-specific applications

- extend role of ERP packages
- concentrate on few software vendors for smooth integration
- follower strategy, low risk
- buy-only for new applications
- insulate bank-specific part

→basically same application useful e.g. to Nestlé

Block C

Bank-specific applications

- build on higher level infrastructure
- componentize system
- individual make-buy decisions when renewing parts of application portfolio
- be launch customer if beneficial to overall strategy

Infrastructure

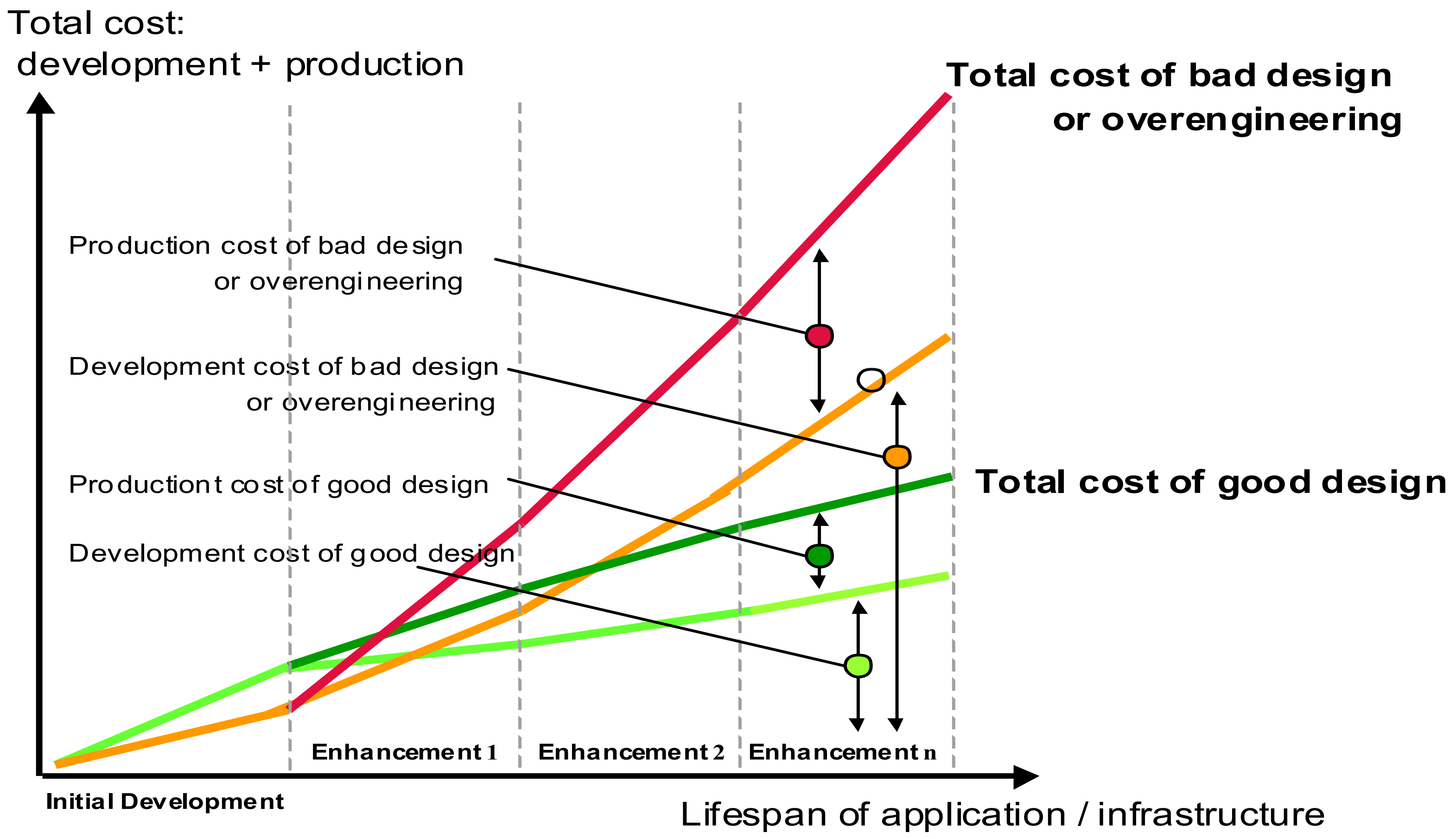
- only buy for new components
- forced replacement of in-house infrastructure components
- focus on integration of best-of-breed market standards
- extend role of infrastructure into applications

Block A

Software Evolution Management Principles

- **Make it as simple as possible but not simpler**
  - Choose the simplest solution that fulfills the requirements
  - No “overengineering”
- **Design for low-cost production and simple maintainability**
  - the cumulated maintenance and production cost of an application by far exceed its development cost

Software Evolution Management Principles – Design for Maintenance



Software Evolution Management Questions & Answers