TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

0101seda010100

software engineering dependability

Quality Management of Software and Systems: Reuse

Contents

Problems

- Problems
  - Normally, each software development reinvents the wheel several times
  - This luxury cannot be afforded any longer

- /Weber 92/
  - The reusability of concepts, products, and procedures is the key concept of advanced industries
  - Implementing reusability aims for cost reduction and quality improvement of products
  - Not using existing products as components or not estimating market opportunities for products and components before development is industrial stone age

Problems

- »Golden Rule of Reusability:
    - Before you can reuse something, you need to
    - 1st find it
    - 2nd know what it does
    - 3rd know how to reuse it.«

Goals

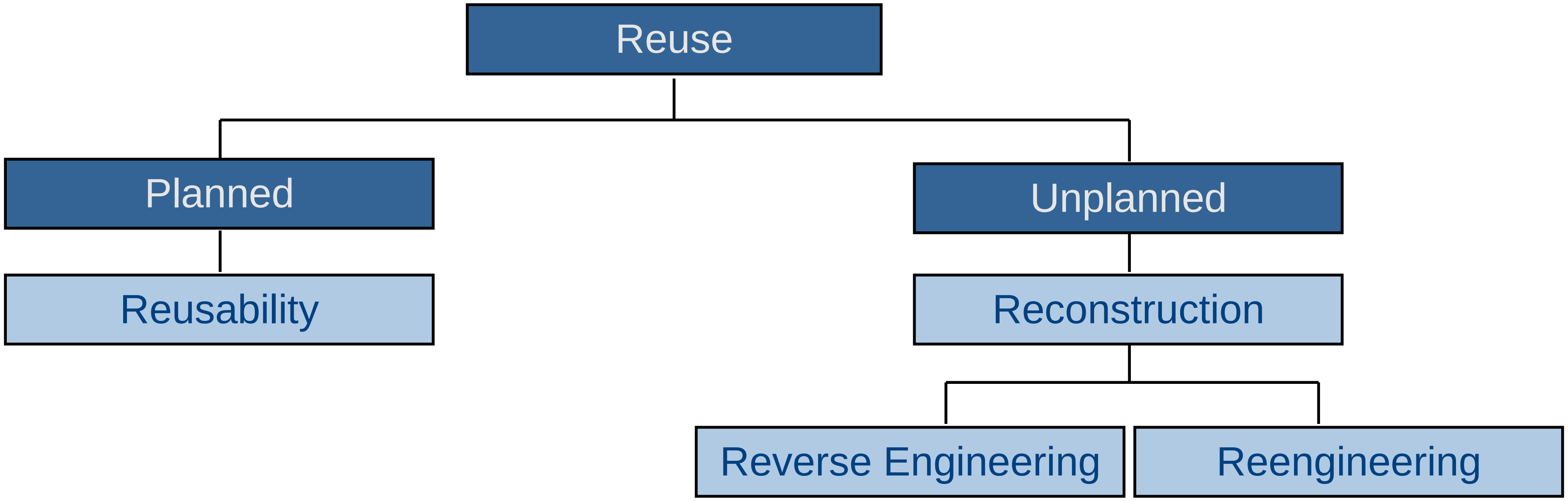
- Goals of reuse
  - Significant improvement of productivity
  - Quality improvement
  - Reducing development time
  - Cost reduction

Reusability vs. Reuse

- Reuse
  - Reusing software components
  - Highly reusable software components ease the reuse of these components
- Reusability
  - Development and provision of reusable software
- Precondition: Optimal cooperation of
  - Technicians / Developers
  - Organization
  - Management

Reuse

- Planned vs. unplanned Reuse



Reuse

- Today: Unplanned reuse
  - Existing software systems
    - Have to be redesigned due to additional requirements or changed hardware/software platforms
  - Original systems
    - Not designed for reuse in other contexts
  - Usage of Reverse Engineering, Reengineering and Reconstruction
    - To identify and prepare software components for reuse

Reuse

- Planned reuse
  - Design software components in the first place in a way that they can be easily reused
  - Good reusability imply a software component to be
    - Highly generic
    - High quality
    - Well documented
  - Software component
    - Each explicit, physical existent development result
    - Each result, offering easy solutions for later projects

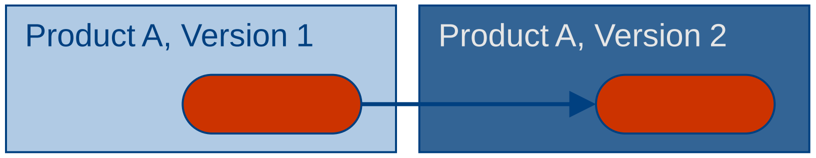
Reuse

- Types of reuse
  - white-box-reuse
    - The developer modifies, adapts, and tests the reused components
  - black-box-reuse
    - The reused components are not changed

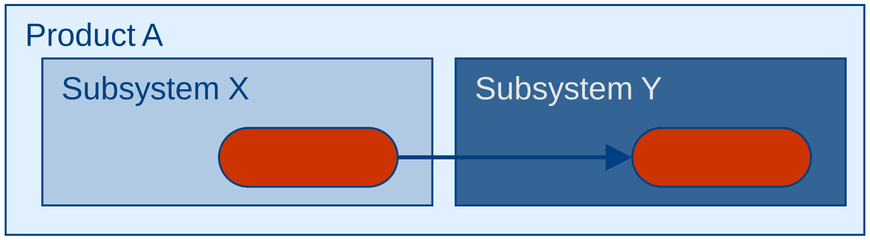
Reusability

- Four types of reusability

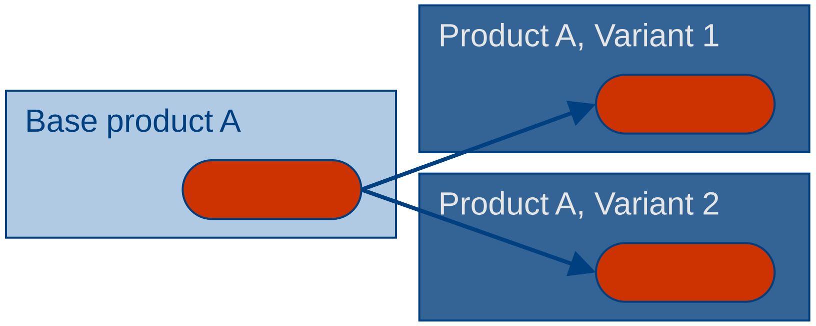
Inter-version Reuse



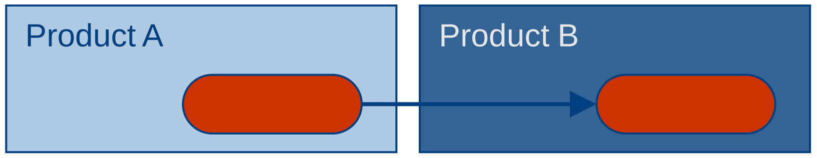
Intra-product reuse



Reuse on variants



Inter-product reuse



Key

⬭	Reused component

Reusability

- Reuse on versions and variants
  - Already good understood
  - Especially, using object-orientation
- Intra- and inter-product reuse
  - Hardly seen
- Layers of reusability
  - Product definition
  - Product design
  - Product Implementation
    - Nowadays: Reuse of code components is prevalent

Reusability

- Differentiation by application domain
  - Vertical differentiation
    - Within the same domain
  - Horizontal differentiation
    - In different domains
      - Example: Libraries with scientific routines
      - Example: Libraries with base classes

Technique

- Techniques for reusing according to the layer of abstraction:
  1. Functional abstraction
     (functions, procedures, methods)
  2. Data abstraction
     (abstract data objects and abstract data types)
  3. Classes using inheritance and polymorphism (Object oriented techniques)

Technique

1. Functional abstraction
   - Definition of reusable functions, procedures and methods.

- Advantages and disadvantages:
  - + Suitable for specific application domains, e.g. mathematical libraries
  - – Low layer of abstraction
  - – Functional perspective not broad enough
  - – Separation of value, state, and processing
  - – Inflexible parameter mechanism
  - – Only some programming languages support generic functional abstraction

Technique

2. Data abstraction
   - Attributes and operations aggregated to one unit (encapsulation and information hiding)
   - Removes disadvantages of functional abstraction

- Advantages and disadvantages:
  - + Suitable for many different application domains, especially for fundamental data structures (e.g. stack, queue, list, etc.)
  - + Highly generic and good adaptability through type parameterization
  - + Easily understandable
  - – Applicable only with strongly typed languages
  - – Not as generic as inheritance

- Comprehensive reusable libraries despite the disadvantages available.

Technique

3. Object oriented components
   - Best options to achieve high reusability
   - Depends on the proper usage of options
   - Wrong usage may decrease reusability

- Advantages and disadvantages:
  - + Inheritance permits specialization, super class can be reused as it is (black-box-reuse)
  - + Adding additional features with little new code
  - – For the most part system inherent complexity shifts to system dynamics
  - – Deep inheritance structures are hard to control

Organization

- Reuse
  - Is not automatically implemented through the use of appropriate technique
  - It must be organized

- To implement reuse, and organization should:
  1. Develop, install, and operate a reusability repository
  2. Include a mechanism to evolutionary promote reuse
  3. Have a process model, which supports reusability

Organization
Reusability repository

Motivation:

- Today, reuse often fails, because existing components cannot be found in a fast, easy, and reliable way
- Example:
  - A software developer is at the beginning of a software project. He would like to reuse software and he asks his colleagues for similar developments. They recall that two years ago there was a similar software project. Therefore, they start searching the corresponding old project files. After a long search, some source code files were found. Unfortunately these files only included comments. They did not include further documentation of the code written therein. Depressed, the developer starts implementing his own solution: the wheel is reinvented.

  - Reference to a certain file within the streamer backup of the project
  - Search for the streamer within the archive
  - Needed compiler version cannot be obtained

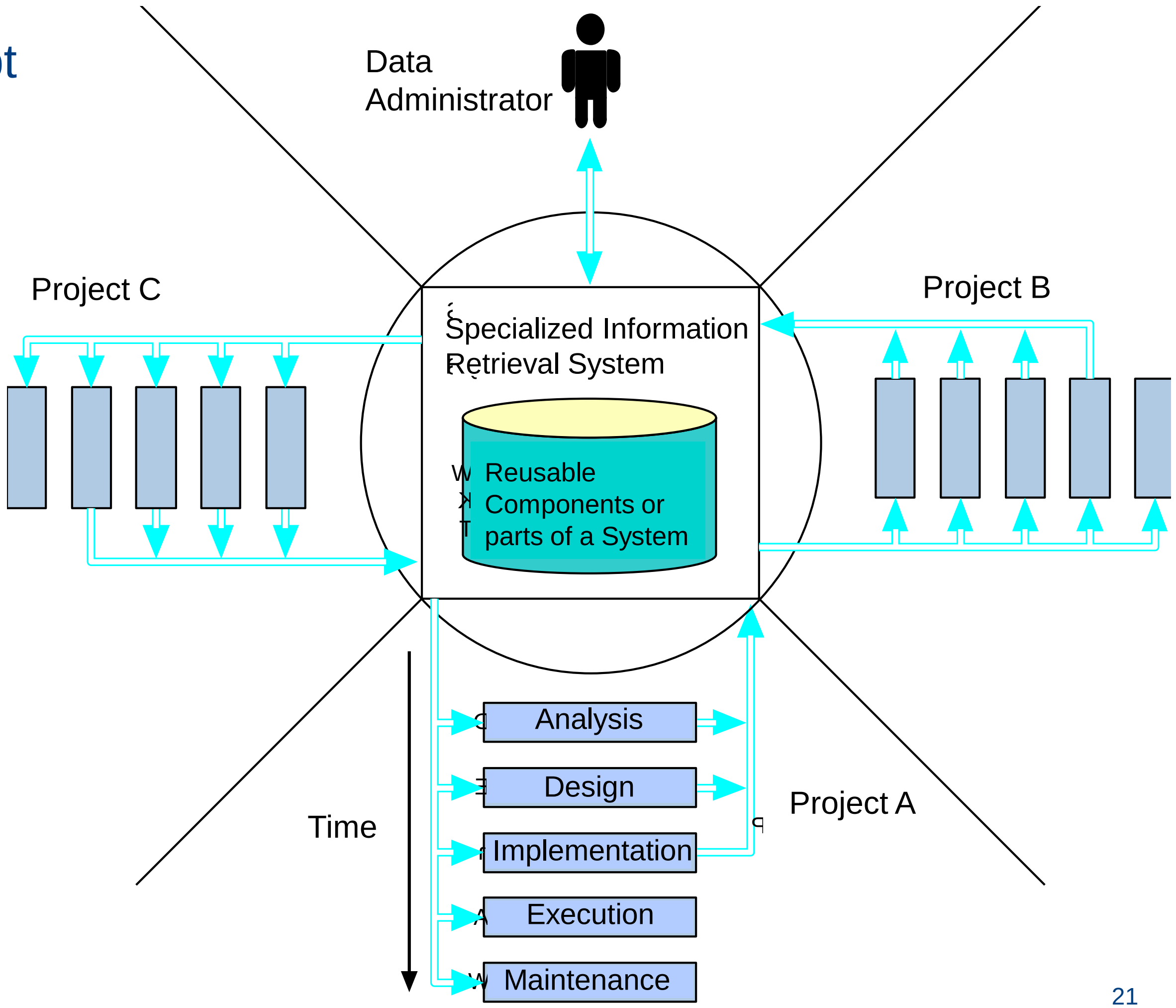
Organization
Reusability repository

- Requirements:
  - Project independent
  - Easily accessible
  - Each employee needs access at any time directly from his desktop
- This repository can be:
  - Independent: with or without connections to a CASE environment
  - Integrated in a reuse oriented CASE environment

- Retrievability must be constructed
  - Reusable components and subsystems must be classified
- Classification
  - Research: Many classification systems for storing and retrieving reusable components
  - Important method: facet classification

Organization
Reusability repository

- Reusability repository Concept



Organization
Reusability repository

- Tasks of a repository
  - Development of classification systems
  - Classification and retrieving of components and subsystems
  - Exporting/importing of components to/from other management systems
  - Documentation of classification systems and classified components

Organization
Maturity levels of reuse

- In order to continuously improve and promote reuse in an organization, five maturity levels of reuse can be distinguished:

Level	Reuse in percent	Requirements
5 Domain-based reuse	80 – 100 %	Domain analysis and architectures
4 Consistent reuse	50 – 70 %	Libraries, processes, metrics, training
3 Planned reuse	30 – 40 %	Approval and support of management, motivation programs, libraries
2 Exploit existing applications	10 – 50 %	Luck and maintenance problems
1 No or ad-hoc reuse	< 20 %	No organizational changes

Organization
Maturity levels of reuse

Level 1: Ad hoc-reuse

- Unsystematic reuse
  - Occasional, dependent on the project staff’s work style, uncoordinated, normally undocumented
  - Rule: copy, adapt, divergence
  - Combined versioning with the original does not occur
- Multiple maintenance and bugfixing
- Better part of today's reuse is in this way
- Better than no reuse
- Simple, with only short-term benefit
  - Benefit compensates if too many undocumented copies circulate uncontrolled, leading to software that is hardly maintainable

Organization
Maturity levels of reuse

Level 2: Reuse of available software

- Available and reusable software must be collected and documented in a structured way
- System parts with different change frequencies are localized
  - Variants and versions of correspondent systems can be developed more efficiently
- Purchasing software that is available on the market
- Development of modular software systems
  - Basis for the reuse of system components
- Commercially available and internal standards are met
  - Portable systems

Organization
Maturity levels of reuse

Level 3: Development for reuse

- Domain-oriented software development
- Establish appropriate infrastructure for reuse-oriented development
- Develop software components, not with respect to the current application but with respect to future applications
- Reusable software components were developed incrementally as subassemblies
- Functionality of such software can be enhanced without changing the original source code
- Incrementally extendable software eases the development of versions and variants
- Development of subassemblies project independently or separated of domain specific projects
- Subassemblies based on an application domain analysis
  - Goal: domain model, describing a generic and therefore reusable architecture for products of the application domain

Organization
Maturity levels of reuse

Level 4: Usage of domain models and statistical process control

- Application development using domain models
- Control of reuse-oriented software development based on statistics
- Provide a domain model for each well defined application domain
  - The model should describe a generic and therefore reusable architecture for products in the application domain
- Domain analysis permits integration to software development processes
- Each domain model consist of
  - Architecture
  - Integrated components and subassemblies including self developed and commercially available components
- Focal point of software development
  - Application modeling
  - Not programming

Organization
Maturity levels of reuse

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Level 5: Organization wide reuse orientation

- All division activities completely reuse oriented
- Alike level 4, applications are developed by assembling existing and widely standardized subassemblies
- **New:** Divisions that do not develop software, act reuse oriented
- All sales activities with available components in mind
- Each software component is considered company asset
- Amount of assets and life cycle length incorporated in
  - Decisions for software development
  - Strategic decisions of the company management

Management

Change of management tasks:

- Software manager has to develop a software product today
  - Crucial question: »How can I solve the problem with my available resources?«
- Prognosis
  - Crucial question: »Where has somebody solved a similar problem, and how can I obtain the solution?«
- Main management task in the future
  - Ensure reusability of components

Management

Reusability:

- Today not primary a technical problem
- Organizational and management problem
- Who decides today to reuse software, can start tomorrow

- Task of software management:
  - Provide an adequate organizational environment
  - Establishing of a **culture of reusability**

Management

- Example 1:
  - Employee develops software component during project
  - During the development he has ideas to generalize the component for the usage in other contexts
  - Under high deadline pressure, only a project specific solution is developed
- Scenario shows
  - In nearly all software companies the appeal to develop generic solutions is missing
- Thus, a bonus system shall be developed for:
  - Providing reusable components
  - Reusing of such components

Costs and Benefits of Reuse

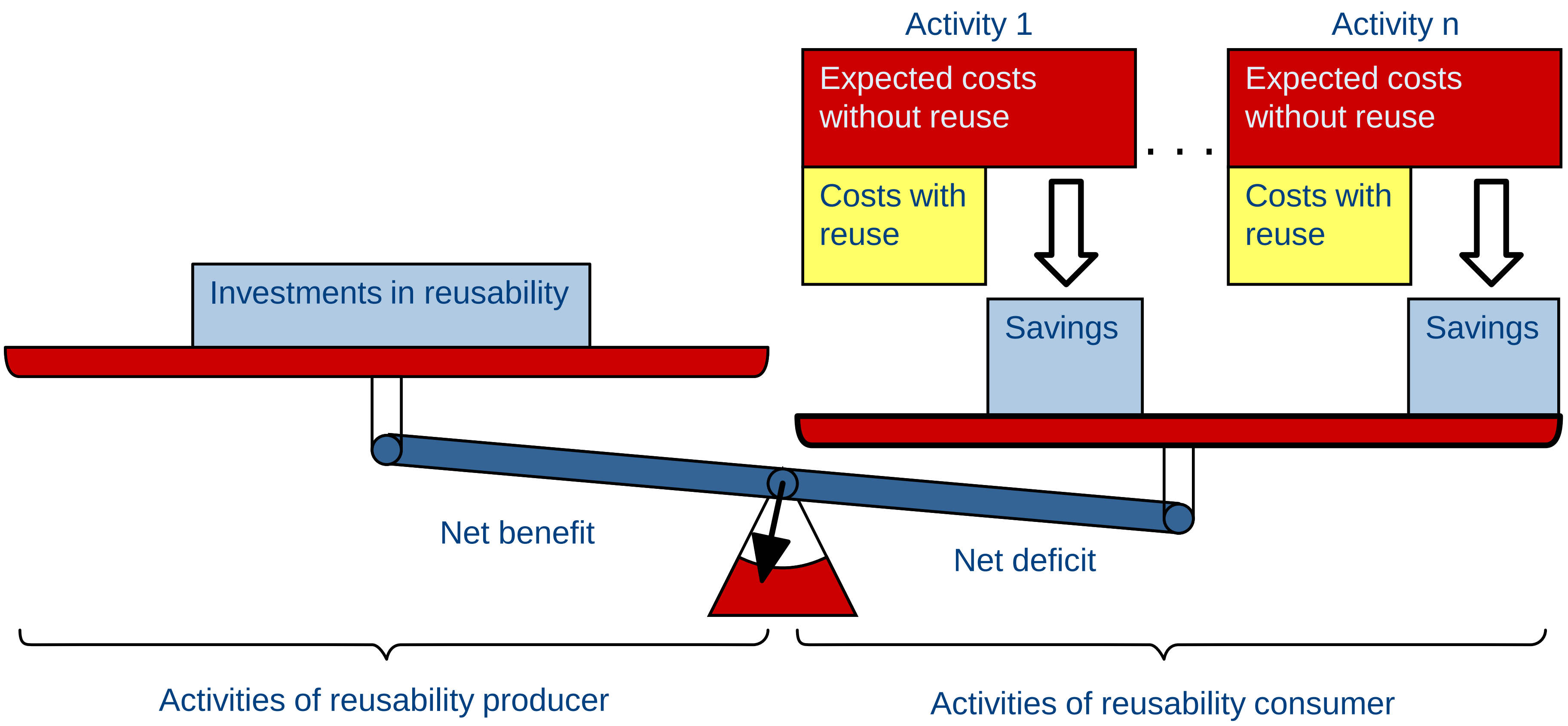
- Japanese experiences
  - Not primarily a technical question
  - Since the end of the 1980ies reuse has a high priority
  - Results through firm organization and strict compliance of methods
  - NEC
    - 6.7 times better productivity
    - 2.8 times better quality of economic applications
    - Achieved by
      - Identification and standardization of 32 logical templates and 130 common algorithms
      - Strict use of CASE environment
  - Fujitsu
    - After establishing a reusability repository 70% on schedule
    - Before only 20%

Costs and Benefits of Reuse

- American experiences
  - Raytheon
    - Analysis and revised design of 5,000 COBOL programs in use
    - 50% productivity increase in six years
    - Within new systems approximately 60% code reuse
  - Independent American study
    - > 70% shorter delivery times because of reuse
  - Hewlett-Packard
    - Project 1
      - Error reduction: 51%
      - Productivity increase: 57%
    - Project 2
      - Error reduction: 24%
      - Productivity increase: 40%
      - Decrease of development time: 42%

Costs and Benefits of Reuse

- Cost/Benefit ratio of reuse


Activity 1
Activity n
Expected costs without reuse
Expected costs without reuse
. . .
Costs with reuse
Costs with reuse
Investments in reusability
Savings
Savings
Net benefit
Net deficit
Activities of reusability producer
Activities of reusability consumer

Costs and Benefits of Reuse

- A reusability investment is cost-effective, if K < N
  - K = Overall costs of the investment
  - N = Savings

- Benefit N

$$N = \sum_{i=1}^{k} n_i = \sum_{i=1}^{k} (o_i - m_i)$$

  - n_i: Savings for activity i
  - o_i: Expected costs for activity i without reuse
  - m_i: Costs for i with reuse
  - k: Number of activities affected by the investment

Costs and Benefits of Reuse

- The Return-On-Investment relation is given by

$$R = \frac{N}{K}$$

- R < 1 means deficit
- R >> 1 means good profit
- 3 possible strategies for R >> 1
  - Increase reuse rate
  - Reduce average reuse costs
  - Reduce investment costs needed to gain a reuse benefit

Implementing Reuse: Impediments during the introduction of reuse

- Economic
  - Missing Commitment
  - Vague company strategy
  - Investment level
  - Losing Deal
  - Missing rights of use and patent rights
- Organizational
  - Not scheduled in the process
  - Responsibilities not assigned
  - Missing promoter
  - Missing infrastructure

Implementing Reuse: Impediments during the introduction of reuse

- Sociological
  - Not-invented-here syndrome
  - Resistance against changes
  - Existential fear »Re-Use is a Job-Killer«
  - Self-image of the developer / changed role
- Technical
  - Missing experience with practical applications
  - Missing Know-how
  - Weaknesses in the Software-Engineering process
  - Missing tools