

Quality Management of Software and Systems

Reuse

Contents

- ☐ Problems
- ☐ Reusability and Reuse
- ☐ Technique
- ☐ Organization
- ☐ Management
- ☐ Costs and Benefits of Reuse
- ☐ Implementing Reuse

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.«

Problems

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

QMS - Reuse

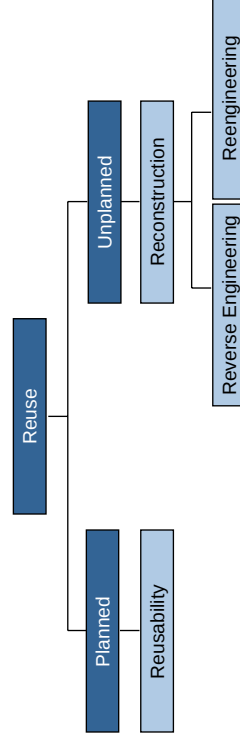
Reusability and Reuse

- ☐ Reusability
 - Development and provision of reusable software
- ☐ Reuse
 - Operation of reusable software
 - Highly reusable software components ease the reuse of these components
- ☐ Precondition: Optimal collaboration of
 - Technique
 - Organization
 - Management

QMS - Reuse

Reusability and Reuse

- ☐ Planned vs. unplanned Reuse



QMS - Reuse

Reusability and 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

QMS - Reuse

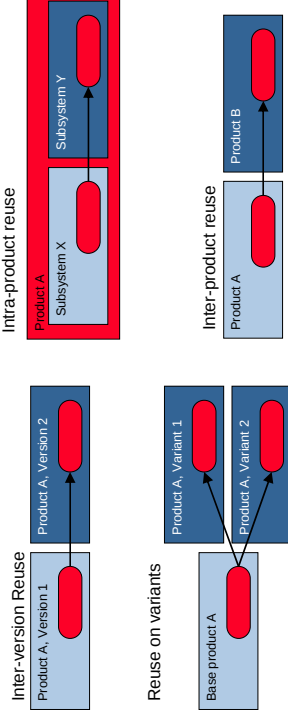
Reusability and Reuse

- ☐ 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

QMSS - Reuse

Reusability and Reuse

- ☐ Four types of reusability



Key

Reused component

QMSS - Reuse

Reusability and Reuse

- ☐ 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

QMSS - Reuse

Reusability and Reuse

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

QMSS - Reuse

Technique

- ☐ Reuse with respect to the layer of abstraction
 - Functional abstraction (functions, procedures, methods)
 - Data abstraction (abstract data objects and abstract data types)
 - Classes using inheritance and polymorphism
 - Additionally, all three abstraction in a generic way (except abstract data objects)

QMS - Reuse

Technique

- ☐ Functional abstraction
 - + 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

QMS - Reuse

Technique

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

QMS - Reuse

Technique

- ☐ Data abstraction
 - Comprehensive reusable libraries despite the disadvantages
 - Example: Ada-Library by /Booch 87/
 - Many class libraries do not use inheritance, but use only options given by generic class concepts
- ☐ Object-oriented components
 - Best options to achieve high reusability
 - Depends on the proper usage of options
 - Wrong usage may decrease reusability

QMS - Reuse

Technique

- ☐ Object-oriented components
 - + 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
- ☐ Reusability in larger units
 - Frameworks
 - Analysis and design patterns
 - Componentware

QMS - Reuse

Organization

- ☐ Reuse
 - Not automatically through the use of appropriate technique
 - Must be organized
- ☐ Organization must include
 - Development, installation, and operation of a reusability repository
 - Evolutionary advancement of reuse
 - Including reuse in process model

QMS - Reuse

Organization

- ☐ Reusability repository
 - Today, reuse often fails, because existing components cannot be found in a fast, easy, and sure way
 - Example
 - Software developer wants to reuse software
 - Question to colleagues for relevant developments
 - Remembrance of a similar problem two years ago in another project
 - Reference to a certain file within the streamer backup of the project
 - Search for the streamer within the archive

QMS - Reuse

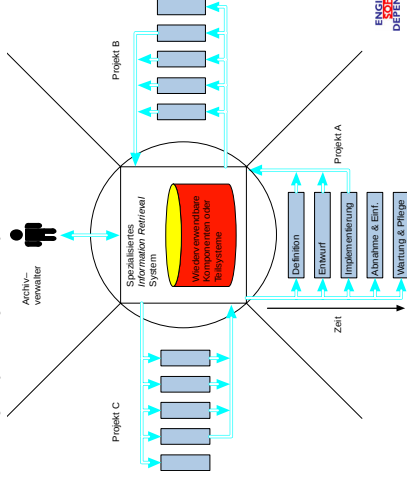
Organization

- ☐ Requirements: reusability repository
 - Project independent
 - Easily accessible
 - Each employee needs access at any time directly from his desktop

QMS - Reuse

Organization

- Reusability repository concept



QMS - Reuse

Organization

- Reusability repository can be
 - Independent repository
 - With or without connections to a CASE environment
 - Integrated in a reuse oriented CASE environment
- Retrieval 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

QMS - Reuse

Organization

- Services of a repository administration
 - Development of classification systems
 - Classification and retrieving components and subsystems
 - Export of components to other management systems
 - Imports from other systems
 - Documentations about classification systems and classified components
 - Saves not only links to sources, but sources itself
 - E.g., if an OOA-model is found, the graphical representation must be accessible

QMS - Reuse

Organization

- Requirements for reuse-oriented CASE environments
 - Support for assembling applications from components
 - black-box-reuse
 - Management of an "use"-relation between component and using application
 - Changes at the component can be communicated to all applications using the component
 - white-box-reuse
 - Additionally, each application must be able to derive and enhance their own version from the original component
 - Application needs test environment (i.e., test scope, test cases, test protocols) of the component

QMS - Reuse

Organization

- Online management of all results from all tools during the whole development cycle
 - All results need to be manipulable with all their relations
- Configuration and change management of the reusable components

QMS - Reuse

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 unmaintainable software

QMS - Reuse

Organization Maturity levels of reuse

- ☐ Level 2: Reuse of available software
 - Systematized through implementation of appropriate arrangements
 - Basis for the development of reusable components were established
 - 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
 - Acquisition of market-available software based on the process
 - Specifically, software development oriented
 - Development of modular software systems
 - Basis for the reuse of system components
 - Commercially available and internal standards are met
 - Portable systems

QMS - Reuse

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 the future applications
 - Reusable software components were developed incrementally as subassemblies

QMS - Reuse

Organization Maturity levels of reuse

- 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

QMSS - Reuse

Organization Maturity levels of reuse

- Additional necessary activities
 - Establish a reusability repository
 - Arrange reusability related trainings
 - Creation of an appropriate in-house communication structure (e.g., Bulletin Boards)
 - Provision of standards for selection, evaluation, and adaptation of components

QMSS - Reuse

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

QMSS - Reuse

Organization Maturity levels of reuse

- 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
- Applications development through assembly of subassemblies
- Application developer develops only application specific parts

QMSS - Reuse

Organization Maturity levels of reuse

- Additional activities
 - Implementing life cycle for development, certification, and usage of components
 - Establish reusability-oriented CASE environments
 - Project progress statistically controlled by characteristics and cost/benefit models
 - Characteristics calculated from measurements

QMS - Reuse

Organization Maturity levels of reuse

- 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

QMS - Reuse

Organization Maturity levels of reuse

- Reuse levels

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

QMS - Reuse

Organization

- 3 level maturity model
 - Practical experience with financial and assurance domain
- Level I: Maintainability
 - Reuse within a single project
- Level II: Balance
 - Reuse between similar projects
- Level III: Standardizing
 - Basis for domain comprehensive reuse

QMS - Reuse

Organization

- ☐ Reuse-oriented process model
 - Necessary requirement to establish reuse
 - All development phases
 - Adequate components are searched
 - New reusable components were classified and put into the repository
 - Reusable components applicable in
 - Definition phase
 - Design phase
 - Implementation phase
 - Developed intern or obtained from extern

QMSS - Reuse

Organization

- Constant observation of software market needed
- Task for buying components
 - Specifying of all today's and future requirements for the own application
 - Examination
 - Components sufficiently parameterized in order to adapt to changing requirements
 - Can the components be evaluated, adapted, and maintained by CASE- and Reengineering-tools
 - Clarification of legal aspects of the components usage

QMSS - Reuse

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

QMSS - Reuse

Management

- ☐ Reusability
 - Today not primary a technical problem
 - Organizational and management problem
 - Who decides today to reuse software, can start tomorrow
 - ☐ Culture of reusability
 - Task of software management
 - Provide an adequate organizational environment
 - Establishing of a culture of reusability

QMSS - Reuse

Management

- Example
 - 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: Development of bonus systems for
 - Provide reusable components
 - Reuse of such components

QMS - Reuse

Management

- Example
 - Nippon Novel pays each software developer 5 Cent per line of code, if he puts a component in the repository or reuses a component
 - Additionally, the developer of a reusable component gets 1 Cent per line of code for each reuse
 - Management has to answer the following questions
 - Who pays for the development of a reusable component?
 - Who is responsible for maintenance?
 - What is the gain of providing a reusable component?

QMS - Reuse

Costs and Benefits of Reuse

- ☐ Qualitative statements
 - Rule of thumb: »Formula 3«
 - Software has to be developed three times until it can be developed for reuse
 - Not until the software is reused three times can the reuse yield fruit
 - Break-even at approximately three reuses of a component specifically developed for reuse
 - Afterwards, the 30-50% higher development costs can be earned

QMS - Reuse

Costs and Benefits of Reuse

- 60% higher development costs
 - 25% for additional generalization
 - E.g., parameterization, design, review
 - 15% for additional documentation
 - 10% for additional testing
 - 5% for repository and maintenance

QMS - Reuse

Costs and Benefits of Reuse

Reuse and productivity	0%	25%	50%
Total time in MM	81,5	45	32
Number of Employees	8	6	5
Costs per LOC	40,74	22,5	16
LOC per MM	165	263	370
Savings	0%	45%	61%

- Examined were three real applications, consisting of 10,000 line of COBOL-Code

Costs and Benefits of Reuse

- Reuse and productivity

	0%	10%	30%	50%	80%
Total time in MM	200	176	131	89	33
Savings	0%	12%	34%	56%	84%

 - Examined were Objective-C programs consisting of 20,000-30,000 LOC
- Cost/Benefit ratio for class libraries
 - Library supply has serious impact on the cost/benefit ratio
 - If no library classes are available at the start, the cost/benefit ratio is <1 not until the 3rd project

Costs and Benefits of Reuse

- Benefit prevails after the sixth project has finished, if
 - Library size, starting with ten classes, continually increases and
 - Development costs for reusable classes are twice as high as for project specific classes
- Potential reuser has higher efforts
 - He has to search, find, understand, install, test, parameterize, and integrate a component
 - Chosen component may inadequate
- Effort may pay off, as developers strongly underestimate the development effort

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

Costs and Benefits of Reuse

- Fujitsu
 - After establishing a reusability repository 70% on schedule
 - Before only 20%
- 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

QMS - Reuse

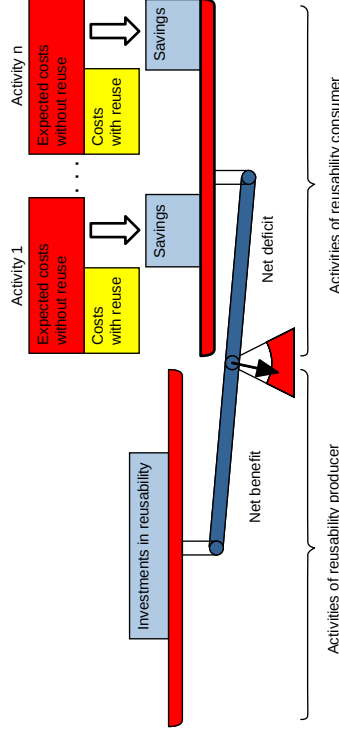
Costs and Benefits of Reuse

- Hewlett-Packard
 - Project 1
 - Error reduction: 51%
 - Productivity increase: 57%
 - Project 2
 - Error reduction: 24%
 - Productivity increase: 40%
 - Decrease of development time: 42%
- Quantitative Statements
 - Each reusable software component has to be seen as company asset
 - Afterwards, reuse can be calculated as normal financial investment

QMS - Reuse

Costs and Benefits of Reuse

- Cost/Benefit ratio of reuse



QMS - Reuse

Costs and Benefits of Reuse

- A reusability investment is cost-effective, if $K < N$
 - K = Overall costs if 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

QMS - Reuse

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 $R \gg 1$
 - Increase reuse rate
 - Reduce average reuse costs
 - Reduce investment costs needed to gain a reuse benefit

Costs and Benefits of Reuse Metrics to measure reuse

- 6 categories
 - Metrics for the cost/benefit analysis
 - Metrics for the maturity classification
 - Metrics for the reuse rate
 - Metrics to calculate reuse impediments
 - Metrics to estimate the reusability of a component
 - Metrics for the usage of the reusability repository

Costs and Benefits of Reuse Metrics to measure reuse

- Metrics for the maturity classification
 - No metric in the stricter sense
 - More like a characteristic to help an organization to identify the current maturity level
- Metrics for the reuse rate
 - Used, to estimate the reuse rate and to track the development over time

$$\frac{\text{Number of reused components}}{\text{Number of components}}$$

Costs and Benefits of Reuse Metrics to measure reuse

- Metrics to calculate reuse impediments
 - Determine impediment factors while trying to reuse components
 - Impediment factors are
 - Number of tries to reuse a components
 - Component does not exist
 - Component is not available
 - Component was not found
 - Component was not understood
 - Component is not valid
 - Component cannot be integrated

Costs and Benefits of Reuse

Metrics to measure reuse

- Each impediment factor are assigned impediment causes
 - E.g.: the factor 'Number of tries to reuse a components' are assigned the following causes
 - Resource restrictions
 - No reuse incentive
 - Missing training
- Usage of such a metric
 - Organization collects data about impediment factors and causes
 - She uses this information to improve reusability activities

Costs and Benefits of Reuse

Metrics to measure reuse

- Metrics to estimate the reusability of a component
 - Measure component attributes, indicating potential reusability
 - Problems arise, as attributes depend on the component type and the programming language
 - Example
 - The following attributes are good indicators for a high black-box reusability of Ada components
 - Less calls per LOC
 - Less input/output parameters per LOC
 - Less read/write statements per LOC
 - More comments in relation to code
 - More auxiliary functions per LOC
 - Less lines of code

Costs and Benefits of Reuse

Metrics to measure reuse

- Metrics for the usage of the reusability repository
 - Quality of classification scheme
 - Costs
 - Search effectiveness
 - Understandability
 - Quality of components
 - Number of reuses within 3 months
 - Judgment of reused components
 - Usage of reusability repository
 - System availability
 - Number of users
 - Executed archive functions
 - Number of available components, etc.

Implementing Reuse Procedure

- ☐ Introduction of Reuse
 - Typical introduction of innovations, with all associated characteristics
- ☐ Orientation at maturity model of reuse
 - 1. Task: Determination of current state
 - Afterwards, plan to iteratively reach next level
- ☐ Define realistic and measurable goals
- ☐ Time needed to achieve a reuse ratio of 20%
 - 1 year from experience

Implementing Reuse Procedure

- ☐ Requirements for reusable components initially not too demanding
 - If only tested, ready components are allowed in the repository, developers do not dare adding their software
- ☐ Solution
 - Divide repository
 - Tested components
 - Provided components, but to be tested and to be completed
- ☐ Introduction of reuse leads to investments and changes in the organization

QMS - Reuse

Implementing Reuse Procedure

- ☐ Lasting practice of reuse
 - Has to enter the company's culture
 - Has to be trained daily
- ☐ Change of development tasks
 - From a writing developer to a reading, evaluating, and creative composing software architect
- ☐ Developers
 - Show distinct resentments against software, they have not developed themselves
 - Fear poor quality and feel stroked at their creative self-image

QMS - Reuse

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

QMS - Reuse

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

QMS - Reuse