Faculty of Computers and Information
Joint Master In Software Engineering Program (JMSE)

Dr. Lamia Abo Zaid

د. لمياء أبوزيد

Software Evolution : TOC

1. Introduction to Software Evolution
2. Taxonomy of Software Maintenance and Evolution
3. Evolution and Maintenance Models
4. Reuse and Domain Engineering
5. Program Comprehension
6. Impact Analysis
7. Refactoring
8. Reengineering
9. Legacy Information Systems

Reuse Capability

- Reuse capability concerns gaining a comprehensive understanding of the development process of an organization with respect to reusing assets and establishing priorities for improving the extent of reuse.
- The concept of reuse opportunities is used as a basis to define reuse efficiency and reuse proficiency.
- An There are two broad kinds of reuse opportunities:

1. **Targeted reuse opportunities** are those reuse opportunities on which the organization **explicitly spends much efforts**.
2. **Potential reuse opportunities** are those reuse opportunities which **will turn into actual reuse**, **if exploited**. Not always a targeted opportunity turns into a potential opportunity.

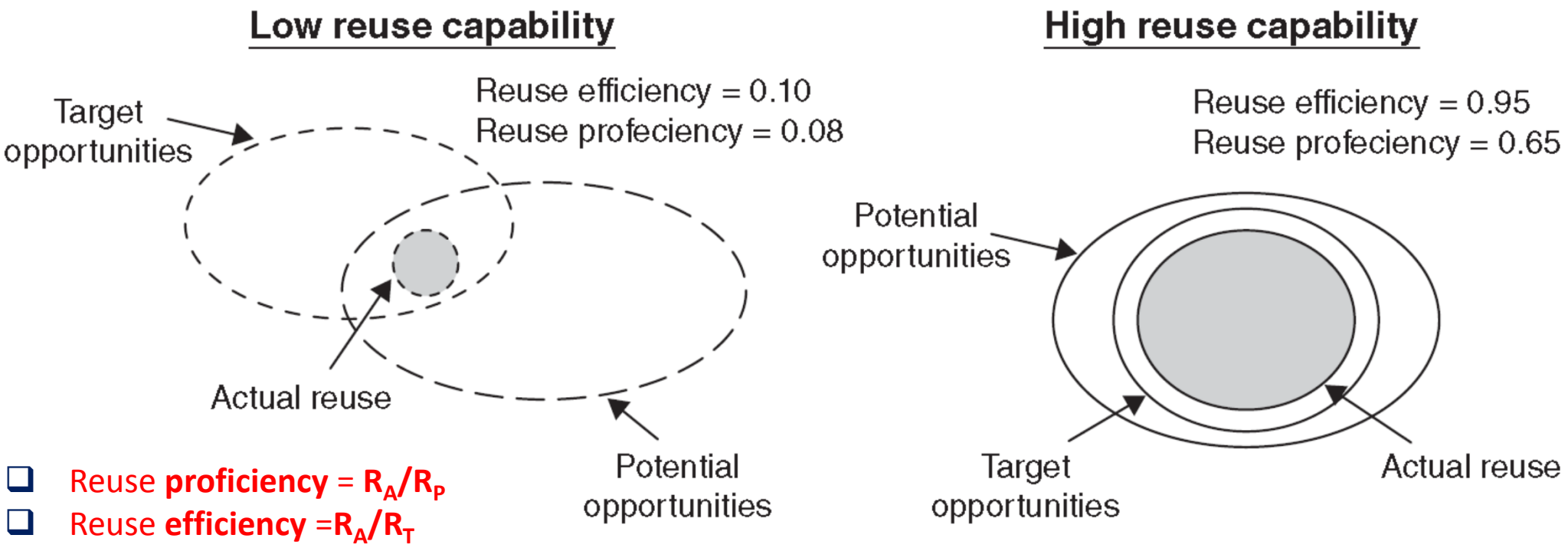
Reuse Capability

- We define reuse proficiency and reuse efficiency by means of R_A, R_P , and R_T , where
  - R_A counts the actual reuse opportunities exploited.
  - R_P counts the potential opportunities for reuse.
  - R_T counts the targeted opportunities for reuse.
- Reuse **proficiency** = $\mathbf{R_A/R_P}$.
- Reuse **efficiency** =$\mathbf{R_A/R_T}$.

Reuse Capability

- **Reuse effectiveness** is represented as:
- $N\ (C_{NR} - C_R)/C_D$, where:
  - N = number of products, systems, or versions developed with the reusable assets.
  - C_{NR} = cost of developing new assets without using reusable assets.
  - C_R = cost of utilizing, that is, identifying, assessing, and adapting reusable assets.
  - C_D = cost of domain engineering, that is, developing assets for reuse and building a reuse infrastructure.

Reuse Capability



- Reuse **proficiency** = R_A/R_P
- Reuse **efficiency** = R_A/R_T

* Assume that the areas of the ovals denote the counts of the assets corresponding to those opportunities.

Maturity Models

- A reuse maturity model is an aid for performing planning and self-assessment to improve an organization's capability to reuse existing software artifacts.
- Maturity model can be used in planning systematic reuse. Organizations development and maintenance
- Maturity models can be divided to:
  1. Reuse Maturity Model
  2. Reuse Capability Model
  3. RiSE Maturity Model

Reuse Maturity Model

- The model comprises five levels and ten dimensions of reuse maturity
- An organization evaluates its current maturity level on each of the 10 dimensions before embarking on a reuse improvement program.
- Maturity improves on a scale from 1 to 5, where level 1 corresponds to Initial/ Chaotic state and level 5 corresponds to the Ingrained state.
- The five scales for maturity are : Initial/ Chaotic , monitored , coordinated , planned Ingrained state
- This model was not applied in real case studies, but are considered as the key insights for the reuse capability model

	Reuse Maturity Levels				
Dimension of Reuse	1. Initial/Chaotic	2. Monitored	3. Coordinated	4. Planned	5. Ingrained
Motivation/Culture	Reuse discouraged	Reuse encouraged	Reuse incentivized reinforced rewarded	Reuse indoctrinated	Reuse in the way we do business
Planning to reuse	None	Grassroots activity	Targets of opportunity	Business imperative	Part of strategic plan
Breadth of reuse	Individual	Work group	Department	Division	Enterprise wide
Responsible for making reuse happen	Individual initiative	Shared initiative	Dedicated individual	Dedicated group	Corporate group with division liaisons
Process by which reuse is leveraged	Reuse process chaotic; unclear how reuse comes in	Reuse questions raised at design reviews (after the fact)	Design emphasis placed on off-the-shelf parts	Focus on developing families of products	All software products are genericized for future reuse
Reuse assets	Salvage yard (no apparent structure to collection)	Catalog identifies language- and platform-specific parts	Catalog organized along application specific lines	Catalog includes generic data processing functions	Planned activity to acquire or develop missing pieces in catalog
Classification activity	Informal, individualized	Multiple independent schemes for classifying parts	Single scheme catalog published periodically	Some domain analyses done to determine categories	Formal, complete consistent timely classification
Technology support	Personal tools, if any	Many tools, but not specialized for reuse	Classification aids and synthesis aids	Electronic library separate from development environment	Automated support integrated with development environment
Metrics	No metrics on reuse level, payoff, or costs	Number of lines of code used in cost models	Maturity tracking of reuse occurrences of catalog parts	Analyses done to identify expected payoffs from developing reusable parts	All system utilities, software tools and accounting mechanisms instrumented to track reuse
Legal, contractual accounting considerations	Inhibition to getting started	Internal accounting scheme for sharing costs and allocating benefits	Data rights and compensation issues resolved with customer	Royalty scheme for all suppliers and customers	Software treated as key capital asset

Reuse Capability Model (RCM)

- Reuse Capability Model comprises two models:
  1. An assessment model.
  2. An implementation model.
- An organization can use the Assessment Model to:
  - Understand its current capability to reuse artifacts.
  - Discover opportunities to improve its reuse capability.
- The success factors are described as goals that an organization uses to evaluate the present state of their reuse practice.
- The organization can apply the implementation model in prioritizing the critical factor goals by grouping them into stages.

Reuse Capability Model- Assessment Model

- Success factors in the assessment model are grouped into four categories: **application development, asset development, management, and process and technology**

Application Development Factors	Asset Development Factors	Management Factors	Process and Technology Factors
Asset awareness and accessibility	Needs identification	Organizational commitment	Process definition and integration
Asset identification	Asset interface and architecture definition	Planning and direction	Measurement
Asset evaluation and verification	Needs and solution relationships	Cost and pricing	Continuous process improvement
Application integrability	Commonality and variability definition	Legal and contractual constraints	Training
	Asset value determination		Tool support
	Asset reusability		Technology innovation
	Asset quality		

Reuse Capability Model- Implementation Model

❑The goals are divided into four stages: opportunistic, integrated, leveraged, and anticipating

1. **Opportunistic**

- A common reuse strategy does not fit all projects so each project develops its own strategy to reuse artefacts. The strategy includes:
  - □ defining reuse activities in the project plan.
  - □ using tools to support the reuse activities.
  - □ identifying the needs of the developers and developing or acquiring reusable artefacts.
  - □ identifying reusable artefacts throughout the lifecycle of the project.

Reuse Capability Model- Implementation Model

2. Integrated

- The organization defines a reuse process and integrates it with its development process.
- It is important for the organization to support the reuse process by means of policies, procedures, resource allocation, and organizational structure.

3. Leveraged

- To extract the maximum benefits from reuse in groups of related products, a strategy for reuse in product lines is developed.

4. Anticipating

- Reusable assets are acquired or developed based on anticipated customer needs.

RiSE Maturity Model

- The RiSE maturity model includes:
  - reuse practices grouped by perspectives and in organized levels representing different degrees of software reuse achieved.
  - reuse elements describing fundamental parts of reuse technology, such as assets, documentation, tools and environments.
- The five maturity levels are as follows:
  - Level 1: Ad-hoc reuse.
  - Level 2: Basic Reuse.
  - Level 3: Initial Reuse.
  - Level 4: Integrated Reuse.
  - Level 5: Systematic Reuse.

RiSE Maturity Model

1. Level 1: Ad-hoc reuse.
   - software is developed in a traditional way, without reusing assets
   - management does not support reuse
   - reuse practices might be performed as an individual initiative by developers.
2. Level 2: Basic Reuse.
   - Goal 1: Best practices in reuse are adopted in the design and implementation of the software product.
   - Goal 2: Use the technical assets, namely, code and documentation, to build software.
3. Level 3: Initial Reuse.
   - Goal 1: Separate business-specific aspects from domain-related aspects.
   - Goal 2: Use defined process.

RiSE Maturity Model

4. Level 4: Integrated Reuse.
   - Goal 1: Enhance the organization's competitive advantage (domain engineering is performed).
   - Goal 2: Integrate reuse activities in the whole software development process (developing families of products).
   - Goal 3: Ensure efficient reuse performance (via standardized metrics).
5. Level 5: Systematic Reuse.
   - Goal 1: Reuse is "the way we do business."
   - Goal 2: Establish and maintain complete reuse-centric development. (automated reuse support integrated with their development process)

RiSE Maturity Model

- In the RiSE Maturity Model there are fifteen factors divided into four perspectives: organizational, business, technological, and processes.

Factors of Influence	Levels				
	1. Ad-hoc	2. Basic	3. Initial	4 Organized	5. Systematic
Product family approach	- Isolated products - No family product approach	- Common features and requirements across the products - Commonalities and reuse possibilities were identified	- Product line domain analyses performed	- Focus on developing families of products - Domain engineering performed	- Domain analysis performed across all product lines - Product family approach
Software reuse education	- Chaotic development process unclear where reuse comes in	- Reuse questions raised at design reviews (after the fact) - Development process defined (some reuse activity indications)	- Design emphasis placed on reuse of off-the-shelf parts - Product line domain analyses performed - Shared understanding of all the activities needed to support reuse	- Focus on developing families of products - Reuse-based processes are in place to support and encourage reuse - Domain engineering performed	- All software products generated for future reuse - Domain analyses performed across all product lines - Product family approach

RiSE Maturity Model Business factors

Economic Models of Software Reuse

- Project managers can use the general economics model of software reuse in their planning for investments in software reuse.
- The project managers need to estimate the costs and potential payoffs to justify systematic reuse.
  - Increased productivity is an example of payoff of reuse.
- Many cost models exist, of which are:
  - **Gaffney and Durek**
  - application system cost model of Gaffney and Cruickshank
  - business model of Poulin and Caruso.

Gaffney and Durek Cost Model

- Two cost and productivity models proposed by Gaffney and Durek for software reuse are:
  - First Order Reuse Cost Model. - estimates cost of reusing software components
  - Higher Order Cost Model. – estimates the cost of developing reusable assets.
- First Order Reuse Cost Model
  - we assume the following conditions:
    - The reused software satisfies the black-box requirements in the sense that it is stable and reliable.
    - Users of the reusable components have adequate expertise in the context of reuse.
    - There is adequate documentation of the components to be reused.
    - The cost of reusing the components is negligible.

Gaffney and Durek Cost Model - First Order Reuse Cost Model

- The effective size S_e, is an adjusted combination of the modified source code and the new source code, as given in the following equation:

$$S_e = S_n + S_o(A_d \times F_d + A_i \times F_i + A_t \times F_t)$$

where:

- A_d = is a normalized measure of design activity,
- A_i = is a normalized measure of integration activity,
- A_t = is a normalized measure of testing activity, and
- $A_d + A_i + A_t = 1$.

- Letting S_r denote the estimated size of reusable components, the relative sizes of reusable components is given by R, where R is expressed as follows:

$$R = S_r / (S_e + S_r)$$

Gaffney and Durek Cost Model - First Order Reuse Cost Model

- Let C be the cost of software development for a given product relative to that for all new code (for which C = 1)
- Let R be the proportion of reused code in the product as defined earlier ($R \leq 1$).
- Let b be the cost, relative to that for all new code, of incorporating the reused code into the new product. Note that b = 1 for all new code.
- The relative cost for software development is:

  [(relative cost of all new code) * (proportion of new code)] +

  [(relative cost of reused software) * (proportion of reused software)].

Therefore: $C = (1)(1 - R) + (b)(R) = (b - 1)R + 1$ and the associated relative productivity is:

$$P = \frac{1}{C} = \frac{1}{(b-1)R + 1}$$, b must be < 1 for reuse to be cost effective.

Questions

?