Reuse and Domain Engineering

Isaac Griffith

CS 4423 and CS 5523
Department of Computer Science
Idaho State University

ROAR

Outcomes

After today's lecture you will be able to:

- Understand and describe reuse maturity models
- Understand and describe reuse cost models

Reuse and Domain Engineering

CS 4423/5523

ROAR

Idaho State University
Computer Science

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 helps the organization's understanding of their existing and future goals for reuse activities.
- Maturity model can be used in planning systematic reuse. Organizations developing and maintaining.
- In this section we discuss briefly three maturity models:
  - Reuse Maturity Model
  - Reuse Capability Model
  - RiSE Maturity Model

ROAR

Reuse Maturity Model

- In circa 1991, Koltun and Hudson presented the first reuse maturity model (RMM).
- The model provides a concise form of obtaining information on reuse practices in organizations.
- The model comprises five levels and ten dimensions of reuse maturity
- 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.
- This model was not applied in real case studies, but are considered as the key insights for the reuse capability model developed by Ted Davis

Idaho State University Computer Science

Reuse Maturity Model

Dimension of Reuse	Reuse Maturity Levels				
	1. Initial/ Chaotic	2. Monitored	3. Coordinated	4 Planned	5. Ingrained
Motivated/ Culture	Reuse discouraged	Reuse encouraged	Reuse incentivized re-enforced rewarded	Reuse indoctrinated	Reuse in the way we do business
Planning for 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, pay-off, 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	Inhibitor 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

Table 9.1: Reuse Maturity Model [43] (©[1996] ACM).

ROAR

Reuse Capability Model

- RCM comprises two models, namely
  - an assessment model
  - 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
- A set of critical success factors are at the core of the assessment model.
- 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.

Assessment Model

- The 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		

Table 9.2: Critical Success Factors

Assessment Model

- Each critical success factor is defined in terms of one or more goals.
- The goals describe what is to be achieved – and not how those goals can be realized.
- Therefore, there is much flexibility in achieving those goals.
- As an example, the needs identification factor has the following goals:
  - Identify the current needs for solutions of the developer.
  - Identify the anticipated needs for solutions of the developer.
  - Identify the current needs for solutions of the customer.
  - Identify the anticipated needs for solutions of the customer.
  - Use the identified needs as a reference to develop or acquire reusable assets to meet the specified needs.

Implementation Model

- The goals are divided into four stages:
  - Opportunistic
  - Integrated
  - Leveraged
  - Anticipating

ROAR

Opportunistic

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

Implementation Model

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.

Leveraged

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

Anticipating

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

RiSE Maturity Model

- The RiSE maturity model was developed during the RiSE project through discussions with industry partners.
- The RiSE maturity model includes:
  1. reuse practices grouped by perspectives and in organized levels representing different degrees of software reuse achieved.
  2. 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.

Idaho State University Computer Science

RiSE Maturity Model

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

Factors of Influence	Levels				
	1. Ad-hoc	2. Basic	3. Initial	4 Organized	5. Systematic
Planning for reuse	- Nonexistent	- Grassroots activity - Reuse is viewed as single-point opportunities - Individual achievements are rewarded	- Targets of opportunity - Organization responsible for reuse - A key business strategy	- Business imperative - Reuse occurs across all functional areas	- Part of a strategic plan - Discriminator in business success
Software reuse education	- Lack of expertise by staff members - Frequent resistance to reuse	- Basic definitions of reuse are agreed upon	- The staff has the expertise and how to obtain benefits with reuse	- The staff members know the reuse vocabulary and have reuse expertise	- All definitions, guidelines, standards are in place, enterprise-wide
Legal, contractual, accounting considerations	- Inhibitor to getting started	- Internal accounting scheme for sharing costs allocating benefits	- Data rights and compensation issues resolved with customer	- Royalty scheme for all suppliers and customers	- Software treated as key capital asset
Funding, costs, and Financial Features	- Costs of reuse are unknown	- Costs of reuse are "feared"	- Payoff of reuse is "known" and understand for a given domain - Investments made in reuse, payoffs expected - Costs of reuse are "known"	- All costs associated with an asset's development and all savings from its reuse are reported and shared	- All costs associated to a product line or a particular asset and all saving from its reuse are reported and shared
Rewards and incentives	- Reuse is discouraged by management	- Reuse is encouraged	Reuse is motivated reinforced, rewarded	- Reuse is indoctrinated	- Reuse is "the way we do business"
Independent reusable assess development team	- Individual initiative (personal goal as time allows)	- Shared initiative	- Dedicated individual	- Dedicated group	- Corporate group (for visibility not control) with division liaisons

Table 9.3: RiSE maturity model levels: organizational factors [42]

ROAR

RiSE Maturity Model

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

Table 9.4: RiSE maturity model levels: business factors [42]

Factors of Influence	Levels				
	1. Ad-hoc	2. Basic	3. Initial	4 Organized	5. Systematic
Repository systems usage	- Salvage yard (No apparent structure to collection)	- Catalog identifies language and platform specific parts - Simple structure like concurrent versions systems - Considered mainly source code	- Catalog includes generic data processing functions - Considered software components, reports and document models	- Catalog organized along application specifications - Have all data needed decide which assets to build/acquire - Considered screen generators database elements and test cases	- Planned activity to acquire or develop missing pieces in catalog - Considered all artifacts of software development life cycle
Technology support	- Personal tools, if any	- A collection of tools, e.g., CM, but not specialized to reuse - General-purpose analyzers combined to assess reuse levels	- Classification, aids and synthesis aids - Standardization on components and architecture - Tools customized to support reuse	- Digital library separate from development environment	- Automated support integrated with development system - Fully integrated with development and reporting systems

Table 9.5: RiSE maturity model levels: technological factors [42]

ROAR

Idaho State University Computer Science

RiSE Maturity Model

Factors of Influence	Levels				
	1. Ad-hoc	2. Basic	3. Initial	4 Organized	5. Systematic
Quality models usage	- No quality model adoption	- Some quality activities were incorporated in the software development process	- Software development process guided by a quality model	- High quality model usage in the engineering department	- Quality model completely adopted in the organization activities
Software reuse measurement	- No metrics on level of reuse, payoff or cost of reuse	- Number of lines of reused code factored into cost models	- Manual tracking of reuse occurrences of catalog parts	- Analyses performed to identify expected payoffs from developing reusable parts	- All system utilities, software tools, and accounting mechanisms instrumented to track reuse
Systematic reuse process	- No reused-based process	- Some reuse activities were adopted in the development process - Planning to adapt the software development process of the organization for a reuse-based process	- Development process of the organization is adapted to reuse concepts	- Reuse benefits and concepts are clear for the engineering team - Development process is reused-based	- Systematic reuse process is enterprise-wide
Origin of the reused asses	- No reuse assets	- Build from scratch, some times indirectly	- Build from existent products; adapting existing products	- Build from existing products; extracted through a reengineering process	- Planning the design and building of reusable assets according to product family
Previous development of reusable assets	- No development of reusable assets	- Parallel with development	- Before development	- Before development	Before development

Table 9.6: RiSE maturity model levels: processes factors [42]

ROAR

Economic Models of Software Reuse

- Project managers and financial managers can use the general economics model of software reuse in their planning for investments in software reuse.
- The project manager needs to estimate the costs and potential payoffs to justify systematic reuse.
- Increased productivity is an example of payoff of reuse.
- We will discuss:
  - **cost model** of Gaffney and Durek,
  - **application system cost model** of Gaffney and Cruickshank
  - **business model** of Poulin and Caruso.

Gaffney & Durek Cost Model

- Two cost and productivity models proposed by Gaffney and Durek for software reuse are:
  - first order reuse cost model.
  - higher order cost model.
- The cost of reusing software components has been modeled in the first order reuse cost model.
- Whereas the higher order cost model considers the cost of developing reusable assets.

First Order Reuse Cost Model

- In this 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.
- Three broad categories of program code are used in a project:
  - S_n: It represents the new code added to the system.
  - S_o: It represents the original source code from the pre-existing system. So includes both lifted code and modified code. Lifted code means unchanged, original code taken from past releases of a product. The source code from modified (partial) parts are not considered as reused code.
  - S_r: It represents the reuse source code that are not developed or maintained by the organization. The reuse code is obtained from completely unmodified components normally located in a reuse library.

Idaho State University Computer Science

First Order Reuse Cost Model

- The effective size, denoted by 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:
  $R = S_r/(S_e + S_r)$

ROAR

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:
  - $C = (1)(1 - R) + (b)(R) = (b - 1)R + 1$
  - the associated relative productivity is: $P = 1/C = 1/(b - 1)R + 1$
  - b must be < 1 for reuse to be cost effective.

First Order Reuse Cost Model

Activity	Activity Code	Activity Cost
Requirements	Req	0.08
Design	Des	0.37
Implementation	Imp	0.22
Test	Test	0.33

Table 9.7: Relative costs of development activities

Component Type	Activities to be Completed	Relative Reuse Cost (b)
Requirements	Des, Imp, Test	0.92
Design	Req, Imp, Test	0.63
Code	Req, Test	0.41
Requirements, Design, Code,	Test	0.33

Table 9.8: Relative reuse cost (b)

- If we want to reuse a requirements component, the relative cost to reuse requirements is $b = (0.37 + 0.22 + 0.33) = 0.92$
- If code is reused, then the additional tasks will involve requirements and testing, the relative cost to reuse code is $b = (0.08 + 0.33) = 0.41$

Higher Order Reuse Cost Model

- Estimating the cost of developing reusable components is key to formulating a reuse cost model.
- By combining the development cost of the reusable components into the economic model, we have:

$$C = (1 - R) \times 1 + \left(b + \frac{a}{n}\right) R$$

- where:
  - a is the cost of developing reusable components relative to the cost of building new non-reusable components from the scratch
  - n is the number of uses over which the cost of reusable components is amortized
- Now, the model can be rewritten as:

$$C = \left(b + \frac{a}{n} - 1\right) R + 1$$

Application System Cost Model

- An application system cost model based on domain engineering and application engineering was proposed by Gaffney and Cruickshank.
- The cost of an application system is expressed as the sum of two component costs:
  - the investment in domain engineering apportioned over N application systems.
  - the cost of application engineering to develop a specific system.
- Therefore, the cost of an application system, C_s is equal to the prorated cost of domain engineering plus the cost of application engineering.
- Let the cost of application engineering be the cost of the new code plus the cost of the reused code in the new application system, and let R denote the fraction of code that is reused code.

Application System Cost Model

- Now, we have
  - $Cs = C_{dp} + C_a$
  - $\rightarrow Cs = Cd/N + Cn + Cr$
- Where:
  - $C_{dp} = C_d/N$
  - $C_a = C_n + C_r$
  - $C_s =$ the total cost of the application system
  - $C_d =$ the total cost of domain engineering
  - $C_{dp} =$ the prorated portion of C_d shared by each of the N application systems
  - $C_a =$ the cost of an application system
  - $C_n =$ the cost of the new code in the application system
  - $C_r =$ the cost of the reused code in the application system

Application System Cost Model

Each of the costs, C_d, C_n, and C_r, is taken to be the product of a unit cost (LM/KSLOC) and an amount of code (KSLOC), where LM/KSLOC stands for labor-months/1000 source lines of code. Hence,

$$C_d = C_{de} * S_t, \; C_n = C_{vn} * S_n,$$

and

$$C_r = C_{vr} * S_r$$

The equation for reuse cost is:

$$C_s = C_{us} S_s = \frac{C_{de} S_t}{N} + C_{vn} S_n + C_{vr} S_r$$

where
C_{us} = unit cost of the application system
C_{de} = unit cost of domain engineering
C_{vn} = unit cost of new code developed for this application system
C_{vr} = unit cost of reusing code in this application system
S_t = expected value of the unduplicated size of the reuse library, measured in source statements
S_n = amount of new code in terms of source statements developed for this application system
S_r = amount of reused code incorporated into this application system in source statement
S_s = total size of the application system in source statement

Application System Cost Model

Let $S_n/S_s = l - R$ and $S_r/S_s = R$, where R is the proportion of reuse. The reuse cost equation can be rewritten as:

$$C_{us} = \frac{C_{de}}{N}\frac{S_t}{S_s} + C_{vn}(1-R) + C_{vr}R$$

Now let $S_t/S_s = K$, the library relative capacity. Thus, the basic reuse cost equation is:

$$C_{us} = \frac{C_{de}}{N}K + C_{vn} - (C_{vn} - C_{vr})R$$

The basic reuse cost equation assumes a single reuse of S_r units (SLOC, KSLOC) in each of the "N application" systems. Thus, this expression is applicable to systematic reuse of units of code relatively dense in functionality.

Business Model

- Poulin and Caruso developed a model at IBM to improve measurement and reporting software reuse. Their results are based on a set of data points as follows:
- **Shipped source instruction (SSI):** SSI is the total count of executable code lines in the source files of a product.
- **Changed source instruction (CSI):** CSI is the total count of executable code lines that are new, added, or modified in a new release of a product.
- **Reused source instruction (RSI):** RSI is the total source instructions shipped but not developed or maintained by the reporting organization.
- **Source instruction reused by others (SIRBO):** SIRBO is the total lines of source instructions of an organization reused by others. It is calculated as follows:
  - **SIRBO** = (Source instructions per part) $\times$ (The number of organizations using the part)

Buisness Model

- **Software development cost (Cost per LOC):** This metric concerns the development of new software, and it is calculated in two steps:
  - Let S denote the total cost of the organization, including overhead; and divide S by the total outputs of the organization in number of lines of code (LOC).
- **Software development error rate (Error rate):** It is a historical average number of errors uncovered in the products. To estimate the cost of avoiding maintenance, a historical average value is used.
- **Cost per error:** To quantify the advantage of better quality reusable assets, the historical mean cost of maintaining components with traditional development methods is used as a base line. Now, the cost per error metric is calculated in two steps:
  - let S denote the sum of all costs; and divide S by the number of errors repaired.

Business Model

- The discussed metrics are combined to form three derived metrics:
  - reuse percent
  - reuse cost avoidance
  - reuse value added

Reuse Percent

Reuse percent of a product $= \frac{RSI}{RSI+SSI} \times 100\%$

Ruse percent of a product release $= \frac{RSI}{RSI+CSI} \times 100\%$

Reuse Cost Avoidance

- The purpose of this metric is to measure reduced total product costs as a result of reuse.
- One must retrieve and evaluate the reusable assets to choose the appropriate ones to be integrated into the system being developed.
- For example, the cost of integrating a reusable software element is 20% of the cost of developing the same element anew.
- The financial benefit due to adopting reuse in the development phase of a project is calculated as follows:

$$Development\,cost\,avoidance = RSI \times (1 - 0.2) \times (new\,code\,cost)$$

- In addition, saving in maintenance cost attributed to reuse is much more than those during software development, because of the fewer defects in reused components. The saving is:

$$Service\,cost\,avoidance = RSI \times (error\,rate) \times (cost\,per\,error)$$

- The total reuse cost avoidance is calculated as the sum of cost avoidance in the development and maintenance activites, which is:

$$Reuse\,cost\,avoidance = Development\,cost\,avoidance + Service\,cost\,avoidance$$

Reuse Value Added

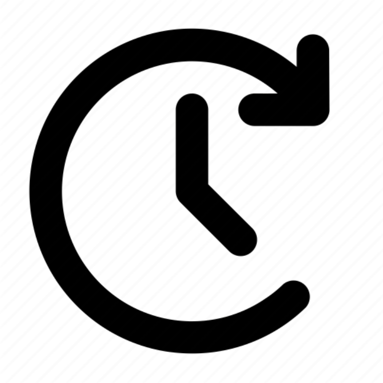
- The main idea behind RVA is to provide a metric to reward an organization that reuses software components and helps other organizations by developing reusable components.
- Reuse value added is derived from SSI, RSI, and $SIRBO$:

$$Reusevalueadded = \frac{(SSI + RSI) + SIRBO}{SSI}$$

- Organizations with no involvement in reuse have an $RVA = 1$.
- An $RVA = 2$ indicates the organization is twice as effective as it would be without reuse.

For Next Time

- Review EVO Chapter 9.3 - 9.6
- Read SA Chapter 1
- Watch Lecture 25



Are there any questions?

ROAR