What is software reuse?

• Software reuse is the process of creating software systems from existing software systems.
• Reuse is like a savings account. Before we collect any interest, we have to make a deposit, and the more we put in, the greater the dividend.
Software reuse is the process whereby an organization defines a set of systematic operating procedures to specify, produce, classify, retrieve, and adapt software artifacts for the purpose of using them in its development activities.
Why Reuse?

- Reuse is not a goal in itself
- Reuse is driven by business goals:
  - Increased productivity
  - Reduced development effort
  - Reduced development time
  - Reduced development cost
  - Increased quality
  - better interoperability
  - Easier maintenance
What to reuse

- Requirements
- Architectures
- Design
- Code
- Data
- Test data
- Documentation
MANAGEMENT ISSUES

Roadblocks to reuse

Hardware Analogy

Some Suggestion for establishing an Approach to reuse
Roadblock to REUSE
A Hardware Analogy

- Its totally opposite to software
Some Suggestions for Establishing an Approach to REUSE
Suggestion For Reuse

Scope of Reuse Application Domain

<table>
<thead>
<tr>
<th>Application Specific</th>
<th>Domain Specific</th>
<th>Domain Independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finance</td>
<td>Healthcare</td>
<td>Utilities, ADT, System Services</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

max. part of system:
- Application Specific: 15%
- Domain Specific: 65%
- Domain Independent: 20%
Suggestion For Reuse

• Reuse plan had to be there in an organization.
• Reusability to be made as an integral part of any technical and managerial training.
• Encourage the tools and methods which enhance the Reuse.
• Management must let it be known that it actively encourages the reuse of software.
• Reuse should not be just treated as a “Business” but as a Plan.
REUSE PROCESS

Artifact MEANS

Relic

Object work of art

Piece
REUSE ARTIFACTS

- Plan
- Cost Estimation
- Architecture
- Requirements Model and Specification
- Designs
- Source Code
- User and Technical Documentation
- Data
- Test cases
A Process Model
“Domain Engineering is about finding commonalities among system to identify components that can be applied to many systems, to identify program families that are positioned to take fullest advantage of those components.”
Guide for identifying reusable S/W Components

• Is component functionality required on future implementations?
• How common is the component's function within the Domain?
• Is there duplication of the component’s function within the domain?
• Is the component hardware-dependent?
Guide for identifying reusable S/W Components

• Does the hardware remain unchanged between implementations?
• Can the hardware specifics be removed to another component?
• Can a non reusable component be parameterized to get a reusable component?
• Is the designed optimized for the next implementation?
• Is reuse through modification feasible?
• How valid is the component decomposition for reuse?
Characterization Functions

- It specify whether a potentially reusable artifact is in fact applicable in a particular situation.

Its being judged by following
Characterization Functions

• A set of Domain characteristics for a reusable artifact can be represented as \{D_{pi}\} where each item, D_{pi}, in the set represents a specific domain characteristics.

• The value D_{pi} represent an scale that is the indication of relevance characteristics for the artifact (Relic).
Scale indicating Artifact characteristics

1: not relevant to whether reuse is appropriate
2: relevant only under unusual circumstances
3: relevant; the artifact can be modified so that it can be used, despite differences
4: clearly relevant, and if the new software does not have this characteristics, reuse will be inefficient; reuse may still be possible
5: clearly relevant, and if the new software does not have this characteristics, reuse will be ineffective; reuse is not recommended
Defined as “A distinct construct within a structural model” it’s a pattern based Domain Engineering approach which work under the assumption that every application domain has repeating pattern (data, behavior and function)
3 Characteristics of Structural Points

• A structured point has a limited number of instances.
• It should be easily understood.
• Information hiding have to be there to reduce the system complexity.
Building Reusable Components

- There is no magic behind REUSE.
- Abstraction
- Hiding
- Functional Independence
- Refinement
- Structured Programming
- Object Oriented Method
Analysis and Design for REUSE

It needs the following

• Standard data
• Standard Interface protocols
• Program templates
Construction Methods

- Frame Technology.
- Frame Hierarchy.
• In addition to assessing whether the cost of adaptation for reuse is justified, the s/w team also assesses whether achieving required functionality and performance can be done cost-effectively.
Components-Based Development

- Data Exchange Model
- Automation
- Structured Storage
- Underlying Object model
  - OpenDoc
  - OMG/CORBA
  - OLE 2.0
HOW to Classifying and Retrieving Components?
Describing Reusable Components

- A reusable S/W software component can be described in many ways, but ideally its defined by 3 C
  - Concept :- Abstract Idea
  - Content :- How Concept is Realized
  - Context :- Place Reusable Compo. In Domain
Enumerated Classification

- Components are described by defining a hierarchical structure in which classes and varying levels of subclasses of software components are defined.

  - E.g. Menu Based (Hierarchical)
Faceted Classification

It make the classification based on Domain area. The feature are called FACETS, which describe the function performed by component.

Here the the Automated Function called as Thesaurus function are used to search the desired technical component
Attribute-value Classification

• A set of attribute are defined for all components in a domain area.
• Values are then assigned to these attributes in the same way as Faceted classification.

• Attribute-value is same as Facet classification with following exception:-
  1. no limit on the number of attribute that can be used
  2. attribute don’t have priority
  3. the thesaurus function are not used.
Reuse Environment should have

- Component Database
- Library management
- A S/W Component Retrieval System
- CASE tools
Thank You
Software Reuse Activities

- 1. Requirement identification
- 2. Asset library search
- 3. Asset retrieval
- 4. Asset customization
- 5. Asset integration
External & Internal Reuse

External reuse:
• The use of software obtained from another organization or application.

Internal reuse:
• Software developed and used repeatedly by the same group of people on the same application.
Reusability = usability + usefulness

Usability = degree to which an asset is ‘easy’ to use (independent of functionality)

Usefulness = ‘frequency’ of suitability for use (independent of packaging)

Usefulness includes economic considerations
What makes SW reuse difficult?

• Limited reuse potential
• S/W is very information rich
• Limited economic benefit
• Reuse is not for free!
• Difficult to organize & manage
• Requires change in existing procedures
• Does our library have a book on HW reuse?
The Domain Analysis Process

• Select specific functions/objects
• Abstract functions/objects
• Define taxonomy
• Identify common features
• Identify specific relationship
• Abstract relationship
• Derive a functional model
• Define a domain language