Introduction

- Software Reuse - the process of creating software systems from existing software rather than building software systems from scratch

- Requirements for software reuse techniques [Krueger 1992, ACM Computing Surveys]
  - it must reduce the cognitive distance between the initial concept of a system and its final executable implementation
  - it must be easier to reuse the artifacts than it is to develop the software from scratch
  - you must know what an artifact does
  - you must be able to “find it” faster than you could “build it”

DSSA and their repositories

- Domain Specific Software Architectures (DSSA)
  - A process and infrastructure that support the development of a Domain Model, Reference Architecture for a family of applications within a particular problem domain
  - Goal: to support the generation of application within a particular domain (aka a product-line)
DSSA Approach

- **Domain engineering**
  - Generate an underlying DSSA support mechanism for a specific family of application systems
  - Generate domain model based on the domain requirements
  - Generate domain reference architecture using domain model
  - Domain reference architecture - the base software architecture for the family of application system contained in the domain

- **Application engineering**
  - Use the mechanism to generate new application systems
  - Use application system specification and constraints to refine and/or extend the reference architecture in order to generate the appropriate architectural instantiation to be used

DSSA Reuse Characteristics

- **Thorough domain understanding**
- **Domain model** and **reference architecture** clearly define what is common in the domain
- The reuse of reference architecture drives reuse of common domain design
- Reference architecture provides an integrating general framework for the reusable components
- **Domain model, reference architecture, repository** are continuously evolving
- Domain specific repository reduces asset management problem

DSSA Selection of Reusable Components

- **Based on the elements of the reference architecture that are preserved**

  More than one option may exist to be use for specific need: choosing component depends on the existent need and the actual information available in the repository for each of the component

  - External Network - component1 is faster and component2 has special security considerations
Faceted Approach to Component Classification

- Software artifacts are classified according to various facets. Several characteristics of the artifacts must be considered in order to facilitate the comparison between different artifacts [Prieto-Diaz and Freeman 1987]

- **Functionality**: describes the function component is intended to perform
  - function (operation performed), object (type of data object on which the operation is performed) and medium (larger data structure in which the data object is located)

- **Environment**: describes the context for which the component was designed
  - system type (type of subsystem for which the component was designed), functional area (application dependent activities) and setting (application domain)

Extension to the Faceted Approach

- Include some few other facets of artifacts
  - execution time
  - memory space
  - level of certification

- the relevance of these facets on a particular selection is totally domain dependent

Matching Algorithm

[Prieto-Diaz and Freeman 1987]

1. Search the library
   - if identical match is found then
     - use it
   - else
     - make a set of similar components
     - for each component in the set
     - compute degree of match
     - rank the set based on the degree of match
     - select the best match from the ranked set
     - modify the selected component and use it

Extended Matching Algorithm

- search the library
  - if identical matches were found then
    - make a set of the identical matches
  - else
    - make a set of similar components
    - request the user to prioritize the various facets
    - for each component in the set
    - compute degree of match based on facets priorities
    - rank the set based on the degree of match
    - present the ranking result of the strongest candidate to the user
    - based on user judgement, select the preferred candidate
    - modify the selected component (if needed) and use it
Summary

- A basic selection criterion for reusable components is its function and other necessary characteristics, such as reliability, usage environment required, timing, and space concerns.

- Since different domains have different needs, they may benefit from differing sets of facets as well as varying priorities on these.

- DSSA’s make use of domain specific repositories, a natural way to improve its reuse process result is the inclusion of the facets that are relevant to the domain, along with some means of prioritizing those.

- While performing the domain modeling, one can determine the set of facets that are relevant in that context, and then use them to populate and query the repository.

- Optimal reuse by having domain specific repositories.