Software Reuse

Haiyan Qiao
ECE Dept.
University of Arizona

What is software reuse? Why?

- Software reuse is the process of using existing software artifacts rather than building them from scratch.
- The primary purpose is to reduce the time and effort required to build software systems. The quality is also enhanced.

History of Software

- First Generation: 1950’s To have a flawless program a programmer needed to have a very detailed knowledge of the computer where he or she worked on. A small mistake caused the computer to crash.
- Second Generation: These generation made use of symbols and are called assemblers. But an assembler still works on a very low level with the machine. For each processor a different assembler was written.
- Third Generation: At the end of the 1950's the 'natural language' interpreters and compilers were made.
- Forth Generation: A 4GL is an aid witch the end user or programmer can use to build an application without using a third generation programming language. Therefore knowledge of a programming language is strictly spoken not needed. In the 1990's the expectations of a 4GL language are too high. And the use of it only will be picked up by Oracle and SUN that have enough power to pull it through.

Why is software reuse difficult?

- Software reuse involves Selection, specialization, and integration of artifacts.
- It implies a higher level of abstraction that describes the artifacts in terms of what they do instead of how they do it.
- Useful abstractions for large, complex, reusable software artifacts will be complex.

Difficulties in practice

- For a effective software reuse technique, 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.
- It will be found faster than to be built.
- Open Area: high-level abstractions for software artifacts.

Storage and retrieval of reusable assets

- Retrieval depends on matching a candidate asset against a user query, the representation of both the query and the asset is an important consideration.
- Queries can be represented in different ways:
  - Natural language and templates
  - List of key words
  - Sample input/output
  - Input/output signature or functional description
  - Design or program patterns (for structural methods)
- Two distinct goals of asset retrieval: exact retrieval, approximate retrieval

Attributes of a software library

- Nature of asset:
  - source code, exe code, requirements specification, design descsription, test data, documentation, proof
- Scope of library: within a proejct, a organization or larger scale
- Query representation
  - Functional specification, signature specification, keyword list, design pattern, behavior sample
- Asset representation
  - Functional specification, signature specification, source code, executable code, requirements documentation, keywords
- Storage structure
  - Flat structure, hypertext links, refinement ordering, ordering by genericity

Attributes of a software library

- Navigation schema
  - Exhaustive linear scan, navigating hypertext links, navigating refinement relations.
- Retrieval Goal
  - Correctness, functional proximity, structural proximity
- Relevance criterion
  - Correctness, signature matching, minimizing functional distance, minimizing structural distance
- Matching criterion
  - Correctness formula, signature identity, signature refinement, equality/subsumption of keywords, natural language analysis, pattern recognition.

Storage/Retrieval Methods

- Information Retrieval Methods
  - A specialized form of information storage and retrieval.
  - Restriction: scope, retrieval goal
- Descriptive Methods
  - Match a keyword-based query against assets that are presented by (structured) list of descriptive keywords.
  - Restriction: scope, gap between relevance and matching criterion

Storage/Retrieval Methods

- Operational semantics methods
  - Software has discriminating feature: executable. Selection by excitability.
  - Queries take the form of an interface specification: a sample of input data.
- Denotational semantics methods
  - Function matching.
  - Restriction: signature matching doesn't ensure correctness. Relevance criterion?

Storage/Retrieval Methods

- Topological methods
  - Identify the library assets that come closest to providing the features described in the query
  - Functional distance, structural distance
  - Query representation: not uniform
- Structural methods
  - Select candidate on basis of structure, instead of function properties.
  - The intent is to use retrieved components after modification.

Related Work

- APU – Automated Programmerfor Unix
  - It uses top-down decomposition of problems, employing a hierarchical planner and a layered knowledge base rules, derivational analogy.
- AUTOBAYES - a fully automatic program synthesis system for statistical data analysis domain. Input/output
- Specification level test
  - Alloy: a first-order relational language. –TestEra
  - UMLTest tool
  - Z specification

References

- Software Reuse, Charles W. Krueger, ACM Computing Surveys, Vol.24, No.2, June 1992
- A Survey of Software Reuse Libraries, A.Mili, R.Mili, R.T. Mittermeir, Annals of Software Engineering 5(1998) 349-414
- Synthesis of Unix Programs using Derivational Analogy, Sanjay Bhansali, Mehdi T. Harandi, Journal of Machine Learning, Vol10, 7-55, 1993
- Hierarchical Case-Based Reasoning Integrating Case-Based and Decompositional Problem-Solving Techniques for Plant-Control Software Design, Barry Smyth, Mark T. Keane, Padraig Cunningham, IEEE Transactions o Knowledge and Data Engineering, Vol 13, No 5, September/October 2001

Reference cont'

- Automated Deduction -A Basis for Applications, Volume III Applications, Chapter 5 Program Synthesis, Chapter 6 Termination Analysis for Functional Programs, Wolfgang Bibel and Peter H. Schmitt, Kluwer Academic Publicashers, 1998
- The proceedings of IEEE international conference on ASE (formerly the Knowledge Based Software Engineering Conference) On-line Bibliography
  - http://ase.informatik.uni-essen.de/olbib/index.html
- Automated Software Engineering Group, NASA Ames Research Center
  - http://ase.arc.nasa.gov/
- Z specification language
  - http://www.rbjones.com/rbjpub/cs/csfm03.htm