Today's Menu

Last Seminar:
- Course Orientation
- Key Terminology

This Seminar:
- SW Reuse: Status and Future
- Areas of Interest

Next Seminar (Wed.):
- Domain Engineering
- DARE

Comments on Review

→ No questions to yourself if you're leading

→ Summary is too long
  - One-paragraph in total
  - Take-away message by the author(s)

→ Questions
  - Do need a separate section

→ MSU mail server was down

→ Exercise this time => no deduction what-so-ever

The TSE'05 Paper

→ Guest Editors' Introduction
  - Collection of extended ICSR-8 best papers

→ Take-away message (goal)
  - What's achieved & what's not in SW Reuse
  - Themes = sub-areas = areas of interest = outline of this course
  - What're they?

→ How did the authors achieve the goal?
  - Free-form survey of 4 questions
  - 7 responses
  - The authors are experts themselves

Process: Proactive, Reactive, Extractive

Cost

One-of-a-kind Development (no reuse)

Proactive Reuse

Number of products

Exercise this time => no deduction what-so-ever
Terminology Clash

> Domain Engineering = (Software) Product Line Engineering

Section 6.2, FAST has 3 processes: domain qualification, domain engineering, application engineering

- Legacy code
  - Domain expertise
- Domain analysis
- Design
- Domain representation
- Domain terminology
- Reference requirements
- Reference architecture
- Reusable components
- Family asset repository
- Requirements
- Traceability
- Components
- Application engineering
  - New requirements
  - Application requirements
  - Application design
  - Application coding
  - Application testing


---

Standalone “Domain Eng.”

Reuse is planned, enabled, enforced!


---

Process vs Technique

> What: entity relation diagram (ERD)

- Notations (syntax), semantics

> How to perform (technical) activity/activities

- Nouns -> Entities
- Verbs -> Relations
- Domain knowledge -> Attributes, Cardinalities, etc.

---

Some Other Questions

- Success stories? (SEI maintains a SPL Hall of Fame)
- What’re the major disadvantages of SW Reuse?
- What if reusing bugs?
- Does SW Reuse have any effect on testing and prototyping?
- How to compute the ROI (in order to initiate a reuse program, e.g., proactively building a SPL)?
- Isn’t FLOSS (free/libre/open source SW, e.g., Eclipse) = reuse?
- Do legal problems (e.g., copyright) inhibit reuse?
- How to reuse in ubiquitous or trustworthy computing (at runtime)?
- Is there any research required with combining the study of SW Reuse and my area?
Comments on Leading Discussion

→ Do your homework
  % Read the background paper(s)
  % Share with us your "Ah-ha" moments
  % Tell us sth that we haven’t known yet
    > Related knowledge
    > Your experience & insights into the topic

→ Know your audience
  % Try your best to answer our questions

→ I’ll evaluate your performance
  % Content: subject matter; knowledge
  % Format: interactive; rehearsal
  % Timing: 30-40 minutes

Software Reuse Truisms

→ Truism
  % A statement that’s obviously true and says nothing new or interesting
  % Simple statements on SW Reuse that have been difficult to satisfy in practice

#1: For a SW Reuse technique to be effective, it must reduce the cognitive distance between the initial concept of a system and its final executable implementation.

#2: For a SW Reuse technique to be effective, it must be easier to reuse the artifacts than it is to develop the software from scratch.

#3: To select an artifact for reuse, you must know what it does.

#4: To reuse a software artifact effectively, you must be able to "find it" faster than you could "build it."

16 Questions about SW Reuse

→ Goal: Confirm or refute 16 common beliefs
  % What’re the factors that will affect SW Reuse?
  % Target audience: practitioners

→ How: Survey
  % 113 people from 29 organizations (US & Europe)
  % Not randomly drawn; Not form a large random sample of the SE community; Fairly representative (position, experience, educational background, domain, organization size, etc.)
  % Statistical methods used: Spearman correlation, boxplots

Results (selected)

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Does programming language affect reuse?</td>
<td>No</td>
</tr>
<tr>
<td>3. Do CASE tools promote reuse?</td>
<td>No</td>
</tr>
<tr>
<td>4. Do developers prefer to build from scratch or to reuse?</td>
<td>Prefer to reuse</td>
</tr>
<tr>
<td>6. Does reuse education influence reuse?</td>
<td>Yes</td>
</tr>
<tr>
<td>7. Does SE experience influence reuse?</td>
<td>No</td>
</tr>
<tr>
<td>10. Do legal problems inhibit reuse?</td>
<td>No</td>
</tr>
</tbody>
</table>
Summary

→ Status & Future
  % 9 themes: Business & Finance, Measurement & Experimentation, Componentry, Domain Engineering, Programming Languages, Libraries, Architectures, Generative Methods, Reliability & Safety
→ 4 Truisms
  % Obvious but difficult to satisfy in practice
→ 16 Questions
  % What matters & what's not?
→ To-do:
  % Read “DARE” paper & send review by next Tuesday
  % Get familiar with the DARE tool (link from course website)