

Software Reuse

M.R.V. Chaudron

www.win.tue.nl/~mchaudro

Dept. of Mathematics and Computing Science

Eindhoven University of Technology

Next week: prof. Crnkovic

Look up his page on internet and look at the work he has done in the area of real-time components.

Reuse

Reuse is like a savings account. Before you collect any interest, you have to make a deposit, and the more you put in, the greater the dividend.

attributed to Ted Biggerstaff, 1983 ITT Reuse workshop

What is software reuse?

Software reuse is the process whereby an organisation 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.

Mili et.al. 2002

Why Reuse?

Reuse is not a goal in itself (neither are CBSE, SPI, ...)

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?

Asset

- Requirements
 - Typically are the result of a labour intensive process involving analysts and domain experts. They codify important domain knowledge.
- Architectures
 - Description of the principal solutions for achieving a set of functional and non-functional requirements. They capture important design decisions and their rationale.

What to reuse?

- Design
 - Detailed specifications for subsystems; e.g. design-patterns
- Code
 - Machine processable function. Embodies knowledge of encoding solution in implementation language. Also has non-functional properties.
 - Many possible forms:
 - Executable, source, macro's, template's, libraries, ...
- Data
 - Standards, formats, ...

What to reuse?

- Test data
 - Reuse test scenario's e.g. after a modification/extension.
Test scenario's capture knowledge about typical faults.
- Documentation
 - Explanation that accompanies some development document
(req/arch/design/implement)

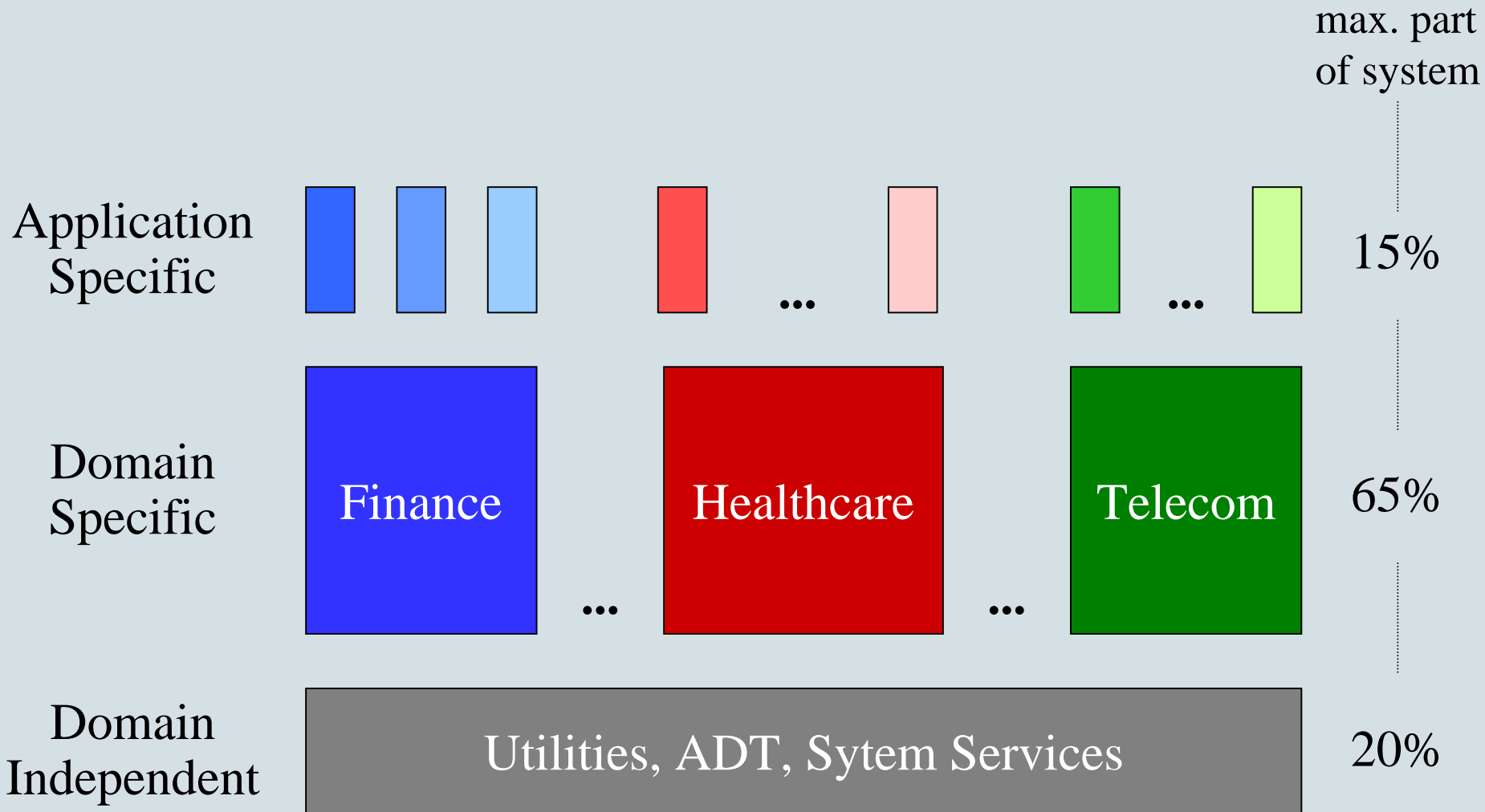
Not Considered Reuse

- Use of OS, Tools (e.g. Compilers)
- New versions of a system
- COTS
 - spreadsheet, database, ...
 - (domain independent) libraries
- Generated code
- Modified assets (!)
- Copy & Paste (code scavenging)

Software Reuse Activities

1. Requirement identification
2. Asset library search
3. Asset retrieval
4. Asset customization
5. Asset integration

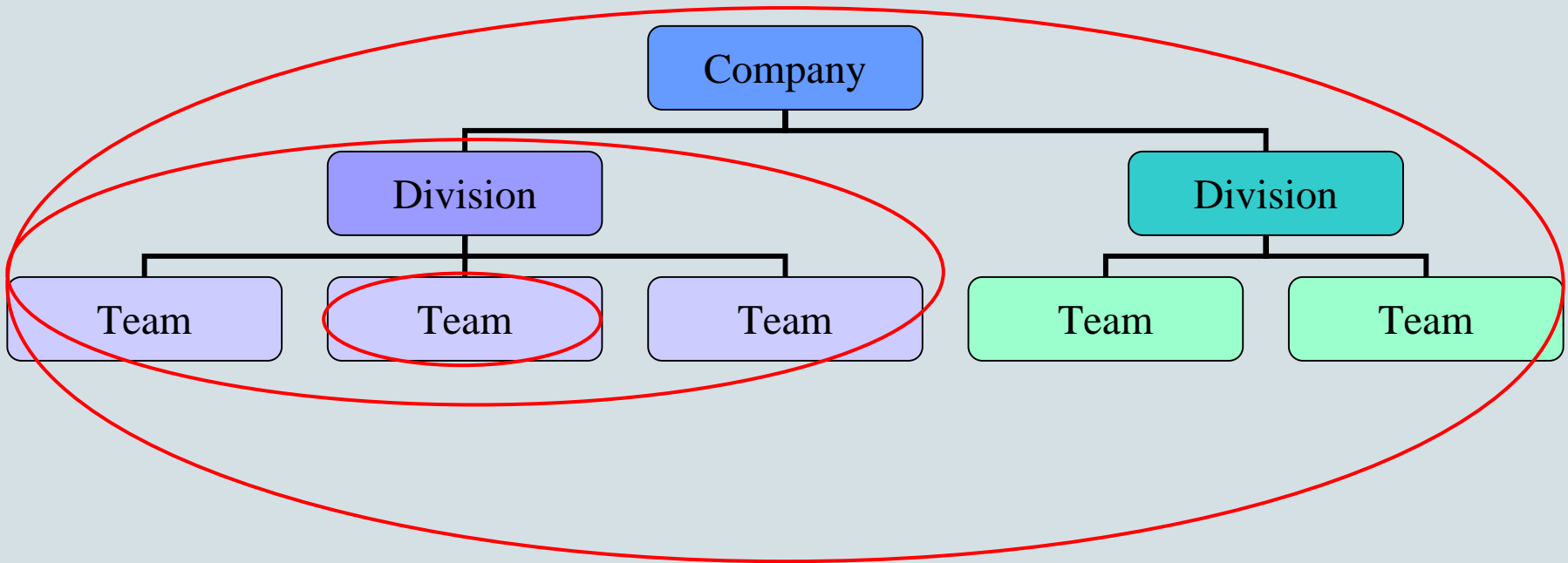
Scope of Reuse Application Domain



Scope of Reuse

- Domain independent
 - ADT, GUI, Math-lib. typically no more than 20% of any application
- Domain dependent
 - aircraft, finance, medical, ... contributes the most to reuse reported up to 85%
- Application specific / custom
 - at least 15% of any application

Scope of Reuse in an Organisation



Organizational structure determines where reuse can take place

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)

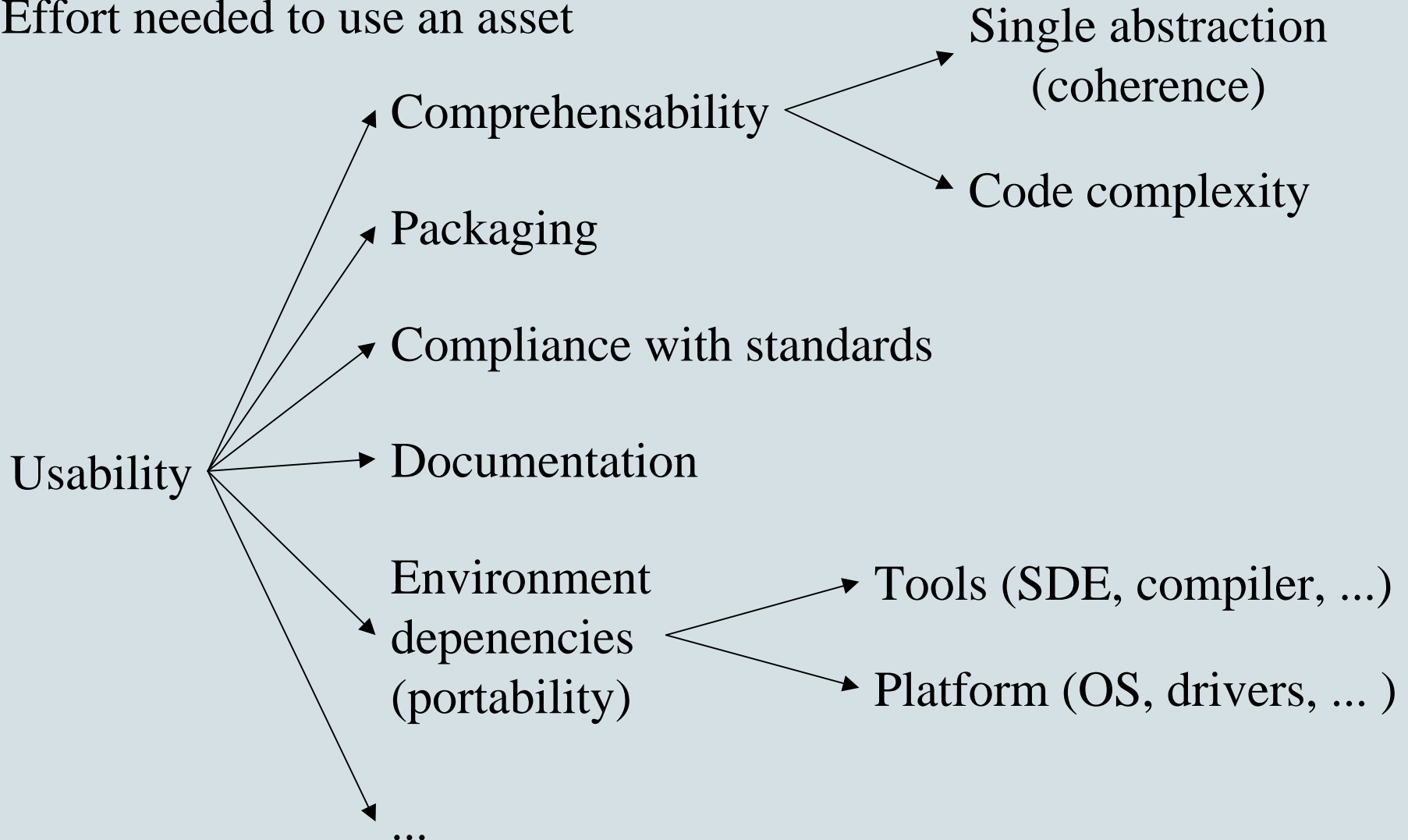
Usability is largely technical

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

Usefulness includes economic considerations

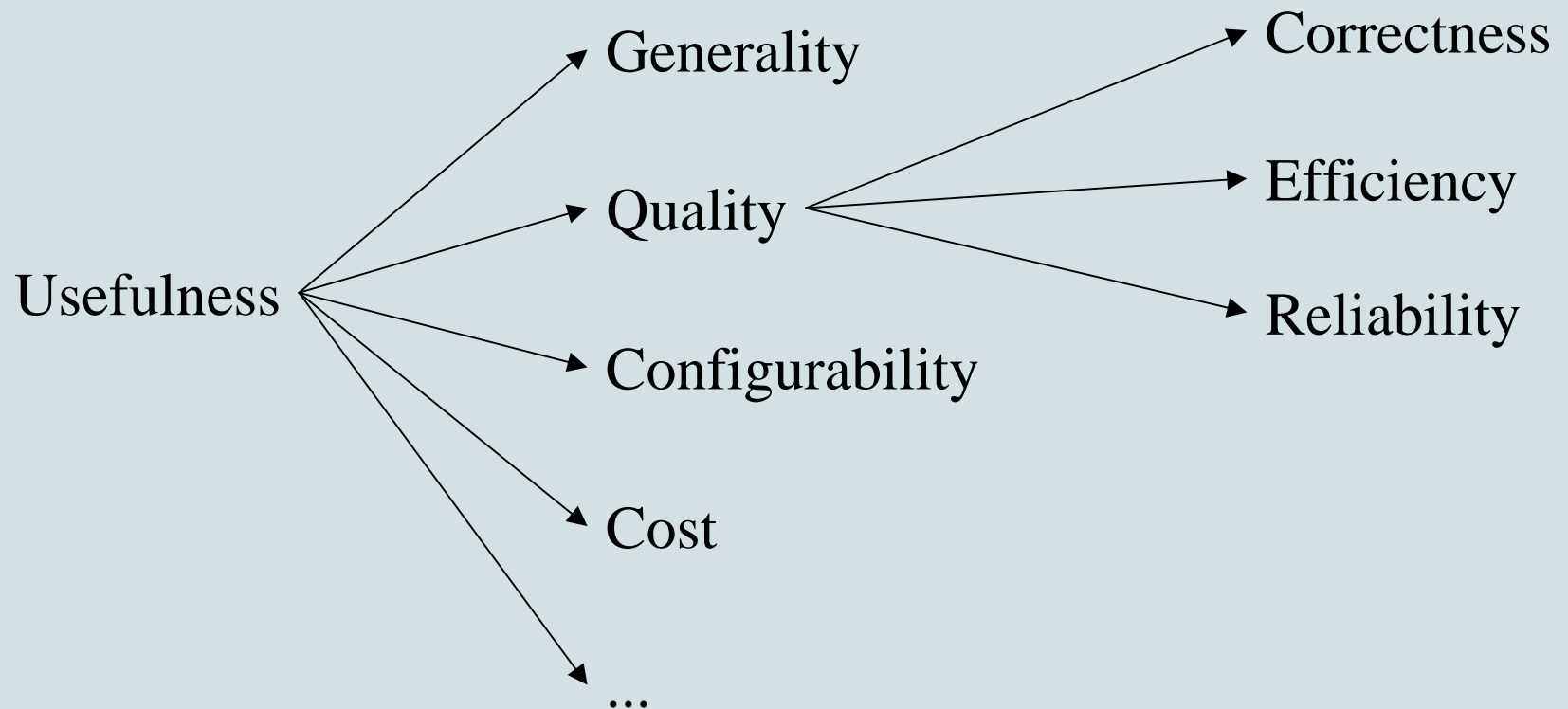
Usability

Effort needed to use an asset



Usefulness

How well does it fit in a context?



Factors of Asset Usability

Ease of:

- locating & retrieving
- understanding
- customization
- integration

} depend on meta-information
e.g. documentation

Black-box reuse	White-box reuse
Assets are integrated into target system <i>without modification</i>	Assets <i>are modified</i> before integration into target system
	- Requires understanding of the inner-working of the component
	- Requires re-assessment of qualitative properties of the component: test anew.
	- Loss of producer ‘guarantees’

Greater benefit,
More difficult to realize

Smaller benefit,
Easier to realize

Success factors in Reuse Management

- Focus on a narrow domain that has considerable commonality
 - Gather components based on domain analysis rather than ad/hoc
- A large volume of software must be developed
- The reuse program must be planned over a long period of time in order to amortize cost of new procedures and to populate the reuse repository

Success factors in Reuse Management

- Apply strict quality criteria for accepting assets in the reuse library
- Training of staff:
 - How to do reuse,
 - Familiarity with available assets
- Management responsibilities:
 - Collect reuse metrics in order to manage reuse
 - Introduce reuse in a incremental manner rather than as big-bang
 - Senior management commitment is required

What makes SW reuse difficult?

Does our library have a book on HW reuse?

- Limited reuse potential
 - Sw is very information rich
- Limited economic benefit
 - Reuse is not for free!
- Difficult to organize & manage
 - Requires change in existing procedures

Comparing Reuse and CBSE

	<i>Reuse</i>	<i>CBSE</i>
<i>subject</i>	‘asset’ – any SE artefact	‘component’ – independent unit of deployment
<i>main technical challenges</i>	asset retrieval	composition - 3 rd party - dynamic
<i>main organisational focus</i>	intra-organisation	inter-organisation

Concluding Remarks

- Reuse is like a savings account
- Relative Extra Cost of Writing for Reuse (RCWR) is 50%
- Relative Cost of Reuse (RCR) is 20%
- Organizational Structure determines where reuse takes place
- Business decisions drive reuse
 - set targets for time-to-market, production cost, ...
 - use metrics to measure and manage progress

References

- Reuse-Based Software Engineering: Techniques, Organization and Controls, H. Mili, A. Mili, S. Yacoub, E. Addy, Wiley & Sons, 2002
 - Measuring Software Reuse: Principles, Practices and Economic Models, J.S. Poulin, Addison-Wesley, 1997
- also see : http://home.stny.rr.com/jeffreypoulin/html/reucalc_basic.html

Questions?

Technological infrastructure for reuse

- Configuration Management
- Quality Assurance
 - Enforce coding standards
- Validation, Testing, Certification

Testing: Find paper by Weyuker on testing components

Configuration management

Factors that complicate CM in Reuse/CBD settings in comparison with once-off settings are:

- Multiple versions of components need to be maintained due to (backward) compatibility issues of different products
- Any change to a component must consider all products that use this component
 - Often, each change leads to a new version, rather than the modification of an existing asset