

## Software Quality Assurance Software Configuration Management

### Software Quality Assurance (SQA)

- ☛ a collection of activities "during" sw development
- ☛ focus on "increasing the quality" of sw (recall BCFH)
- ☛ often conducted by an independent group in the organization: often with the final veto over the release of a sw product

### What is Software Quality?

- ☛ conformance to explicitly stated functional & performance reqs.

Software Requirements Specification



Software Product

"Software Quality Factors" (*broader than performance requirements*)

-ilities:

-ities:

## What is Software Quality?

- conformance to explicitly documented development standards  
(*build the software the right way*)

Software Requirements Specification



Software Product

"Standards":

ANSI/IEEE (recall ATM example)

ISO

DODMIL 2167, Air Force, Navy

Bell Core (communication)

Department of Commerce, etc.

- conformance to implicit characteristics expected of all professionally developed sw (*expectation of a reasonable person*)

***Make customer happy!***

Lawrence Chung

## Metrification of Software Quality

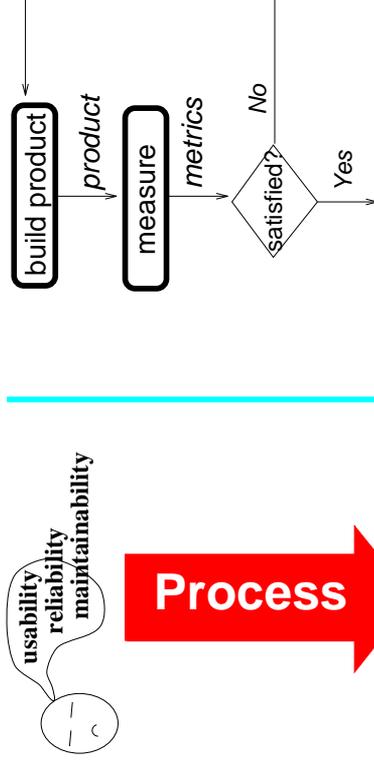
- an attempt to provide a quantitative assessment of sw quality
- usual metrification process:
  - determine a set of desirable attributes (i.e., *ilities*)
  - determine relative importance/weight of such attributes
  - evaluate the quality (rating) of each of the attributes
  - compute weighted rating for each
  - sum up all the weighted ratings

-ilities	relative weight	rating	weighted rating
usability	.3	6	1.8
reliability	.6	5	3.0
maintainability	.1	7	0.7
<b>Overall Quality</b>			<b>5.5/10</b>

- an inexact science at this point
- however, aids in understanding the factors that affect sw quality  
a first-cut approximation  
very poor quality factor

Lawrence Chung

## Build Software Quality



usability  
reliability  
maintainability

**Process**

**Product**

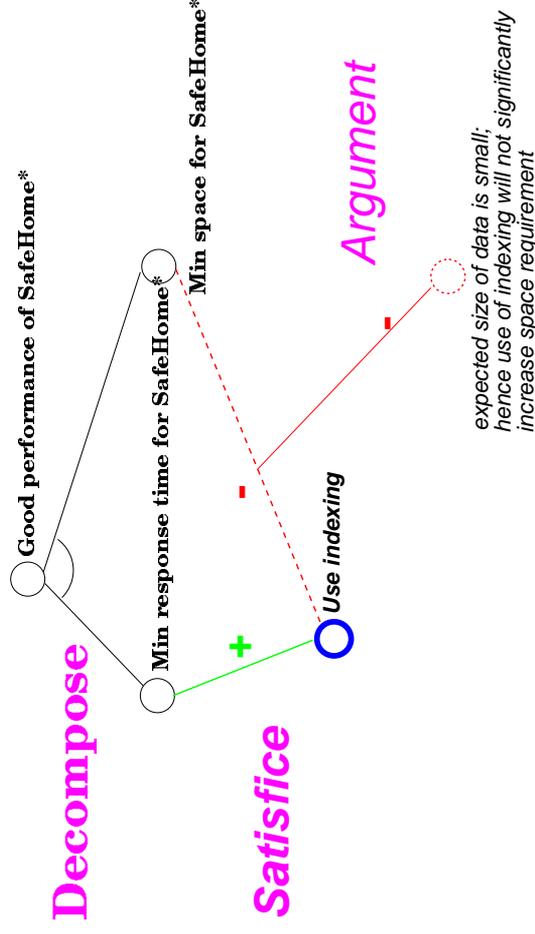
## Alas! Infinite design space

... but are you still trying to find the coin?

- ☛ clarify
- ☛ consider design alternatives
- ☛ analyze tradeoffs considering domain characteristics
- ☛ record design rationale

Lawrence Chung

## Process-oriented approach



Lawrence Chung

## SQA Activities

### ☛ Application of technical methods:

analysis, selection & recommendation of CASE tools

L-CASE (e.g., Compiler, Debugger, Pretty Print, ...)

U-CASE (e.g., DFD/SADT/IDEF/OO/DD, SC, ...)

I-CASE (e.g., IEF/Composer, Teamwork, System Architect, ...)

Requirements (what, who, when, ...) & selection criteria (supportability)

### ☛ Enforcement of standards

What kind of standards are out there?

Which one to use? (e.g., International, North American, European)

Are standards fixed?

Certification needed?

### ☛ Control of change:

no change -> dead/no improvement

changing & conflicting reqs during and after

formal management of changes to the sw and documentation

### ☛ Sw measure & reporting mechanisms:

ongoing assessment of sw quality (SRS -> .. -> release)

tracking changes in quality as system evolves (e.g., ver2 << ver1?)

warning mgmt if quality appears to be degrading

### ☛ Formal technical reviews

### ☛ Formal technical reviews

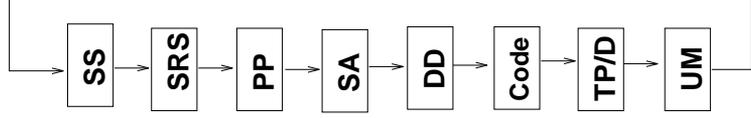
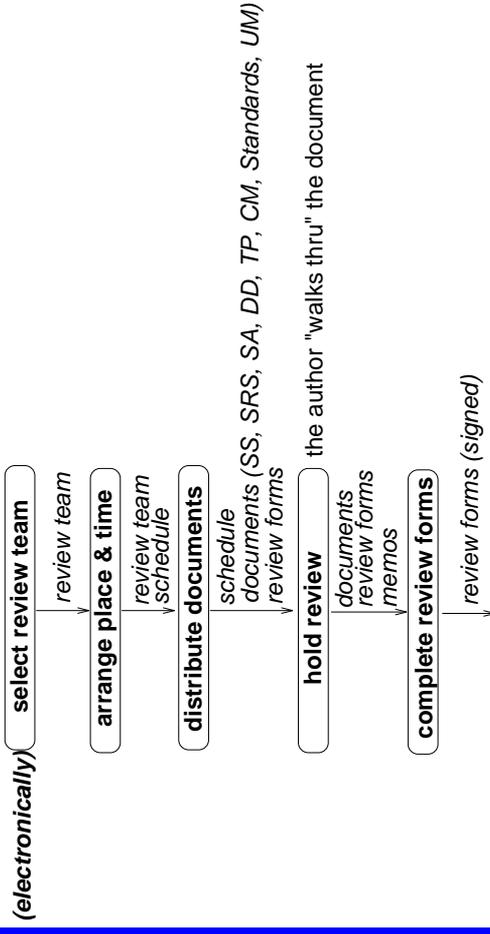
organization is similar to Code Inspection (moderator, inspectors, author)

but usually faster paced (> 150loc) & less intense (> 3lines)

often primarily to detect errors in a sw design (-> SRS -> SS)

on product, design (low-level, arch.), SRS, SS

### ☛ Review Process Architecture



## SQA Activities

### ⌘ *Purpose of formal technical reviews:*

1. accept the product (SS/SRS/Arch...) without further modification
2. accept the product provisionally; minor errors must be corrected, but no further review
3. reject the product due to serious errors;  
Once errors are corrected, another review will be performed

### ⌘ *Pressman's review guidelines:*

- review the product not the producer
- limit debate and rebuttal
- identify problem area but don't attempt to solve every problem
- develop a checklist to guide the review process
- ... and other things you've been doing all along
  - limit size of review team
  - require advance preparation ...
- train reviewers to review; periodically review the review process

Lawrence Chung

## Software Configuration Management (SCM)

⌘ A sw configuration collectively refers to all info. produced as part of the SE process



### ⌘ The First Law of System Engineering:

*No matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the lifecycle*

### ⌘ Purpose of SCM:

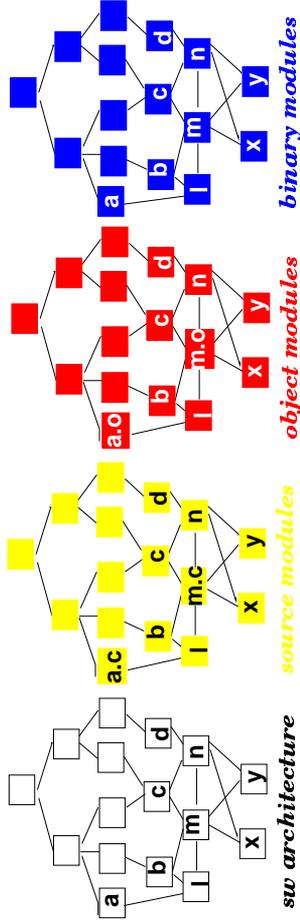
*Manage the structure of a sw system over its lifetime*  
*Bring order to the chaos of a continually evolving sw system*

### ⌘ SCM includes:

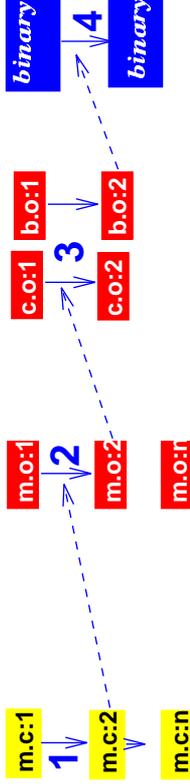
- ⌘ System Modeling
- ⌘ System Composition
- ⌘ Version Control
- ⌘ Change Control
- ⌘ Software Release

Lawrence Chung

## System Modeling



assume a sw system is composed of an arbitrary collection of modules, each with a series of versions

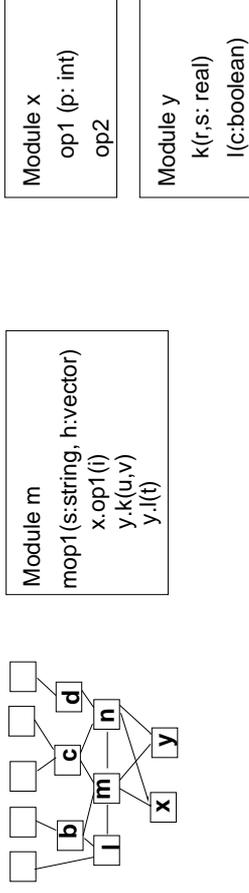


an interface is the link between the server module that provides a service and the client module that uses the service

⊗ **Recall: data flow** ⊗ **Recall: operation/service**

## System Modeling

then, a **system model** is a complete and detailed description of the client/server relationships in a sw system at a given point



a sw system is "consistently composed" if for every client/server relationship, the client & the server agree on the interface between them

import/export (private/public) restrictions

# of parameters

parameter types

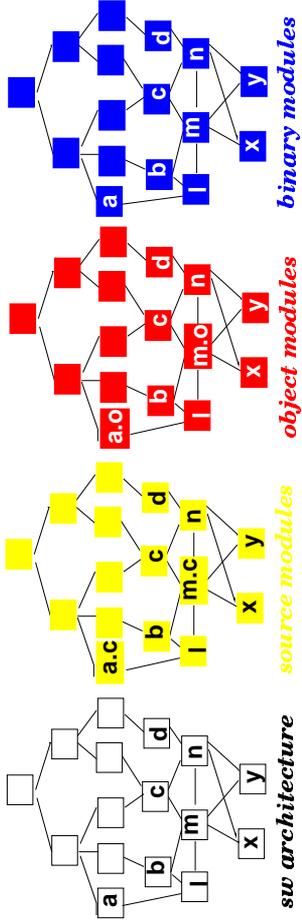
return type

What if algorithms change?

for every client/server relationship, the system model must specify the version of {the interface, the server, the client}

names

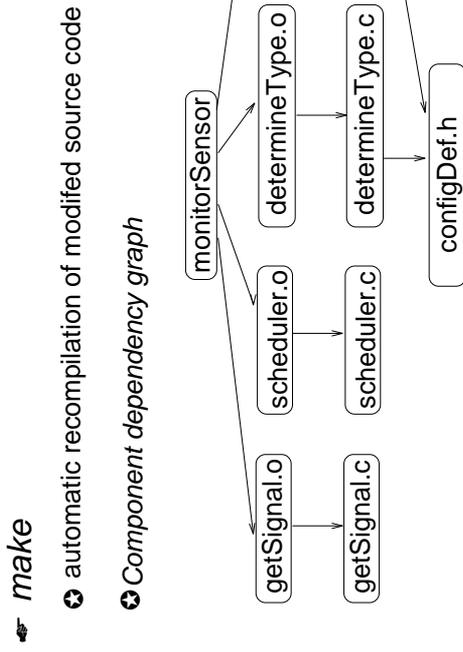
## System Composition Process



- ⊗ Input: System Model and sw source/object/binaries
- ⊗ Output: a correctly composed system
- ⊗ Assume: modules are separately compiled to produce binary modules
  - A formal "interface" is the only compile time link between client & server modules*
- ⊗ Use the system model to identify the correct binary version for each module
  - preferably in terms of sw architecture versions*
- ⊗ For each client/server relationship, check that the client & the server were compiled with the same version
  - compiler checks: all imported modules (servers) exist*
  - all operations invoked by the client exist*
  - parameters (& types) match*

Lawrence Chung

## System Composition Process



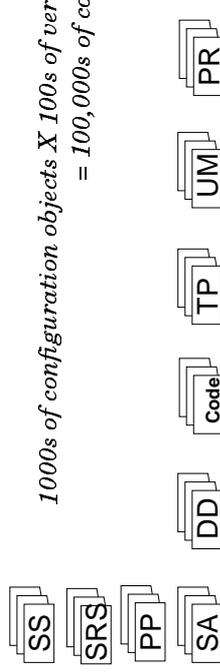
- ⊗ drawbacks:
  - unnecessary recompilation with comments files instead of abstractions
  - can be complicated to write and understand
  - integration with VCS not adequate
- ⊗ Module Interconnection Languages are used to express system models containing all of the rules for constructing composite/derived items.

Lawrence Chung

## Version Control

☛ a sw sys development involves the creation & evolution of 1000s of configuration objects

☛ every change to a config. object logically creates a new version of the object => 100s of versions of an object over the lifetime of a system


  
 1000s of configuration objects X 100s of versions / conf.-obj =
   
 = 100,000s of config-obj-versions

☛ any given instance of a sw sys is composed of specific versions of every object in the system

☛ a version control system combines procedures & tools to manage different versions of configuration objects  
*(e.g., archiving, automatic logging, differential comparison, etc.)*

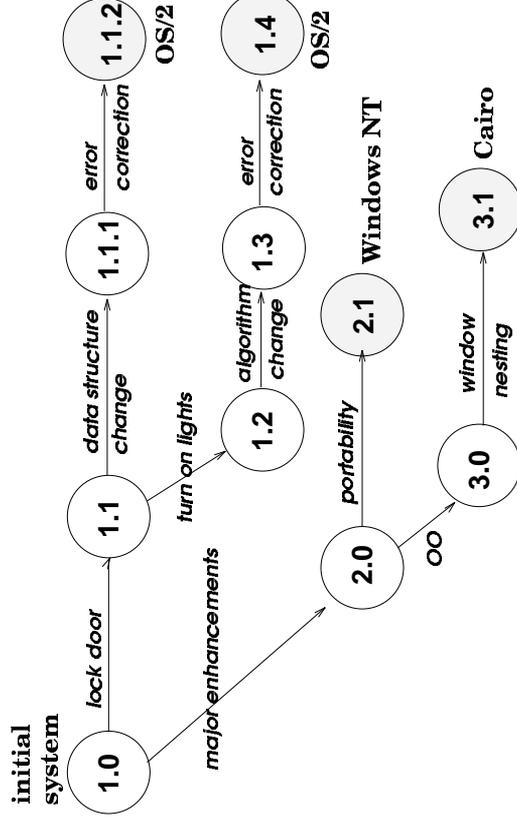
**Do you know why you are working on your SRS now?  
 Are there 10 teams working on the same object?**

Lawrence Chung

## Version Control

☛ an "evolution graph" represents diff. versions of a config. object /system

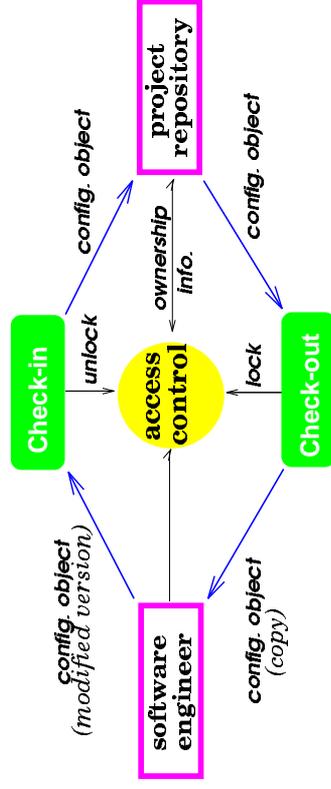
- ☛ typical naming convention: major.minor.variant
- relative to the characteristics of sw system**
- ☛ major versions typically correspond to releases for the sw
- ☛ minor versions for significant change (e.g., days' work)



Lawrence Chung

## Change Control

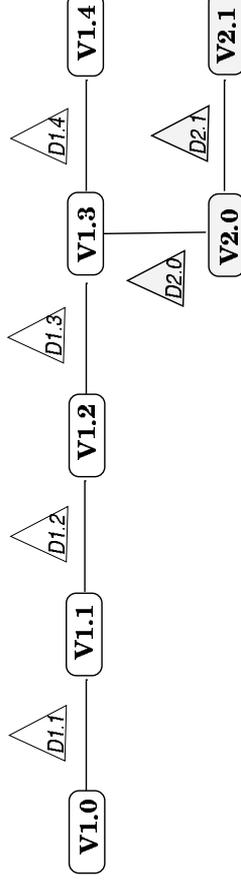
- ☛ change must be "managed":  
(-> *an empty library, purchase of 100 copies of the same book, etc.*)
- ☛ typically a "change committee": senior developers & managers
  - ☒ determines desirability of a change (e.g., control via PDA)
  - ☒ detect conflicting/overlapping changes (e.g., no control via mobile phone)
  - ☒ estimate the cost and impact of changes (e.g., SS, SRS/Proto, SA, DD, Code, TP/TD, UM, ...)
  - ☒ schedule changes relative to software releases
  - ☒ tracks the change process making sure change applications
- ☛ access and synchronization control
  - ☒ access control ensures the sw engineer is authorized to check out
  - ☒ synchron. control locks the object at check-out & unlocks at check-in



Lawrence Chung

## Version Management Tools

- ☛ Consider parallel development releases



- ☛ large number of versions -> storage space explosion

☒ delta = diff (old version, new version)

☒ {version\_i} = version\_master + {delta\_i}

	RCS	SCCS
version_master	version_last	version_first
format	ASCII text	ASCII text
user-supplied info	yes	no

Lawrence Chung