Software Configuration Management

André van der Hoek
Institute for Software Research
University of California, Irvine
andre@ics.uci.edu

Configuration Management

- "Configuration management (CM) is a discipline whose goal is to <u>control changes</u> to large software through the functions of: component identification, change tracking, version selection and baselining, software manufacture, and managing simultaneous updates (team work)."

Walter Tichy, SCM-1, 1988

CM Spectrum of Functionality

<u>Components</u>
- •Versions
- •Configurations
- •Baselines
- •Project contexts

<u>Structure</u>
- •System model
- •Interfaces
- •Consistency
- •Selection

<u>Construction</u>
- •Building
- •Snapshots
- •Regeneration
- •Optimization

<u>Controlling</u>
- •Access control
- •Change requests
- •Bug tracking
- •Partitioning

<u>Accounting</u>
- •Statistics
- •Status
- •Reports

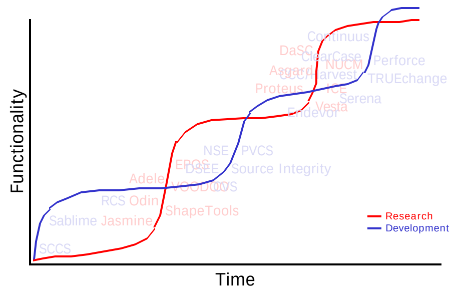
<u>Auditing</u>
- •History
- •Traceability
- •Logging

<u>Process</u>
- •Lifecycle support
- •Task mgmt.
- •Communication
- •Documentation

<u>Team</u>
- •Workspaces
- •Merging
- •Families

Susan Dart, SCM-3, 1991

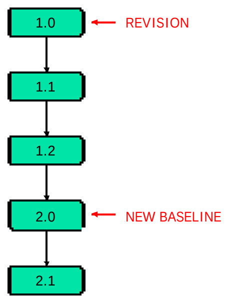
Three Generations of CM Systems



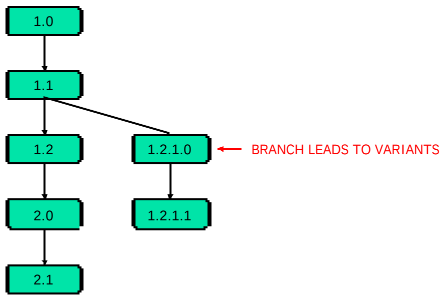
First Generation

- Focused on:
  - Archiving individual elements
  - Strictly avoiding conflicts
  - Optimizing builds
- Characterized by:
  - Simple, separate tools
  - Development orientation
- Canonical examples:
  - RCS
  - Make

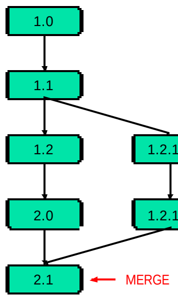
Archiving Individual Elements



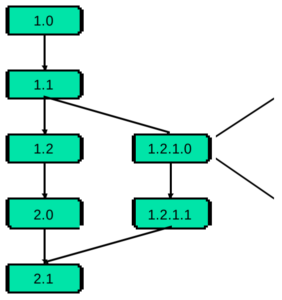
Archiving Individual Elements



Archiving Individual Elements



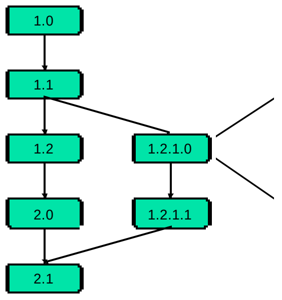
Archiving Individual Elements



Author = “André v/d Hoek”
Date = 01/12/2001
Time = 7:52am
Comment = “Trying new stuff”

ATTRIBUTES (TAGS)

Avoiding Conflicts



Author = “André v/d Hoek”
Date = 01/12/2001
Time = 7:52am
Comment = “Trying new stuff”
Lock = “andre@ics.uci.edu”

LOCK

Optimizing the Build Process

```
accesslist.o: accesslist.h accesslist.c
    gcc –o accesslist.o accesslist.c

andre.o: andre.c
    gcc –c andre.o andre.c

clientprotocol.o: clientprotocol.c clientprotocol. h
    gcc –c clientprotocol.o clientprotocol.c

main.exe: andre.o accesslist.o clientprotocol. o
    gcc andre.o accesslist.o
    mv a.out main.exe
    chmod a+x main.exe
```

Optimizing the Build Process

```
-r--r--r--    1 andre   faculty    2651   Aug  4  1999   Makefile
drwxr-xr-x  2 andre   faculty    4096   Aug  4  1999   RCS/
-r--r--r--    1 andre   faculty    5688   Aug  4  1999   accesslist.c
-r--r--r--    1 andre   faculty    1608   Aug  5  1999   accesslist.h
-rw-r--r--    1 andre   faculty   22840   Aug  4  1999   accesslist.o
-rw-r--r--    1 andre   faculty   10061   Aug  4  1999   andre.c
-rw-r--r--    1 andre   faculty   21944   Aug  4  1999   andre.o
-rw-r--r--    1 andre   faculty   14224   Aug  5  1999   clientprotocol.c
-rw-r--r--    1 andre   faculty    3491   Aug  6  1999   clientprotocol.h
-rw-r--r--    1 andre   faculty   48272   Aug  4  1999   clientprotocol.o
```

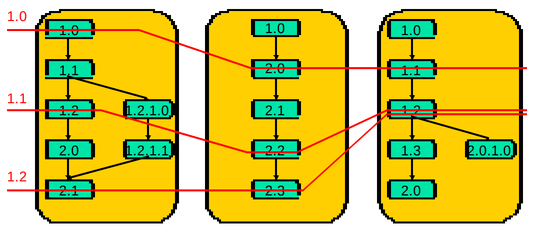
First Generation

Components •Versions •Configurations •Baselines •Project contexts	Structure •System model •Interfaces •Consistency •Selection	Construction •Building •Snapshots •Regeneration •Optimization	Controlling •Access control •Change requests •Bug tracking •Partitioning
Accounting •Statistics •Status •Reports	Auditing •History •Traceability •Logging	Process •Lifecycle support •Task mgmt. •Communication •Documentation	Team •Workspaces •Merging •Families

Second Generation

- Focused on:
  - Archiving compound elements
  - Different version models
- Characterized by:
  - Integrated versioning & build tools
  - Development orientation
- Canonical examples:
  - DSSE
  - Adele

Archiving Compound Elements



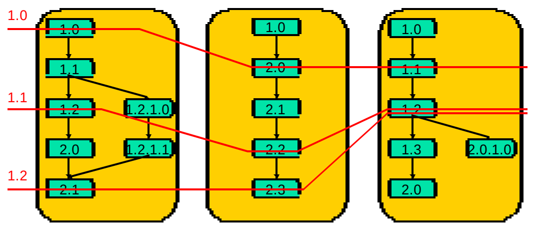
Different Version Models

- State-based extensional
  - Version tree
- State-based intensional
  - Conditional compilation
- Change-based extensional
  - Change packages
- Change-based intensional
  - Change sets

Conditional Compilation

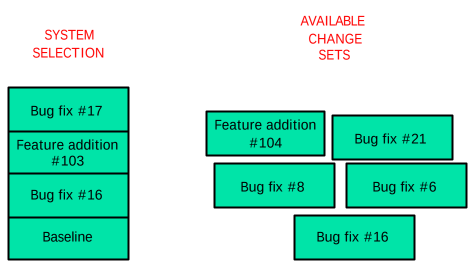
```
...
#ifdef UNIX
     #include <stdio.h>
#endif
#ifdef GRAPHICS
     #include <graphics.h>
     #ifdef SMARTGRAPHICS
          #include <smart.>
     #endif
#endif
```

Change Packages



1.0: BASELINE
1.1: BUG FIX for reporting error #16
1.2: FEATURE ADDITION for graphical user interface

Change Sets



Second Generation

Components •Versions •Configurations •Baselines •Project contexts	Structure •System model •Interfaces •Consistency •Selection	Construction •Building •Snapshots •Regeneration •Optimization	Controlling •Access control •Change requests •Bug tracking •Partitioning
Accounting •Statistics •Status •Reports	Auditing •History •Traceability •Logging	Process •Lifecycle support •Task mgmt. •Communication •Documentation	Team •Workspaces •Merging •Families

Third Generation

- Focused on:
  - Providing process support
  - Being all-encompassing
- Characterized by:
  - Large, complex tools
  - Management orientation
- Canonical examples:
  - ClearCase + ClearGuide
  - Continuus

Third Generation

Components •Versions •Configurations •Baselines •Project contexts	Structure •System model •Interfaces •Consistency •Selection	Construction •Building •Snapshots •Regeneration •Optimization	Controlling •Access control •Change requests •Bug tracking •Partitioning
Accounting •Statistics •Status •Reports	Auditing •History •Traceability •Logging	Process •Lifecycle support •Task mgmt. •Communication •Documentation	Team •Workspaces •Merging •Families

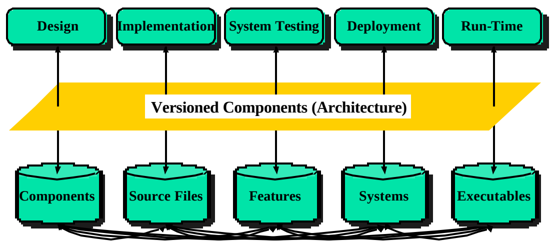
Fourth Generation

?

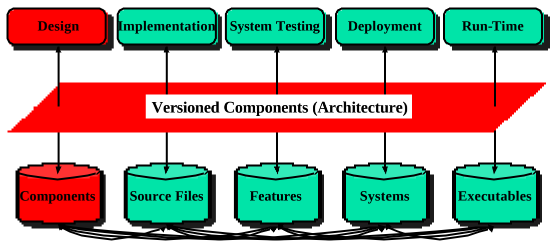
Today's Rapidly Changing Landscape

- Construction by component assembly
- Integrators and suppliers
- Multiple organizations and locations
- Decentralized control
- Customized products
  - tailored to the individual customer
- Continuous change management
  - management after development

Long-Term Vision



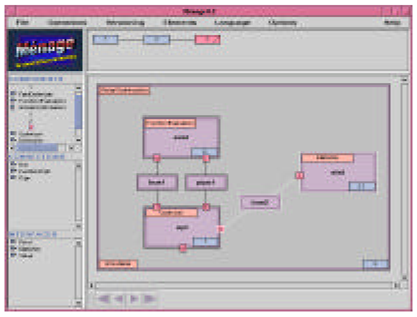
Long-Term Vision



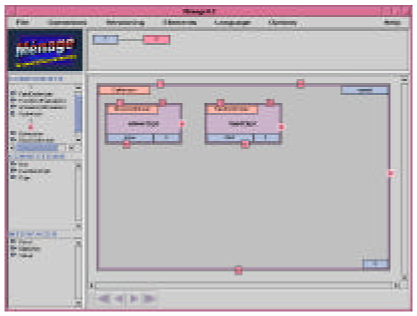
Versioned Components

- Required modeling capabilities
  - Evolution
    - Multiple revisions, temporary branches, baselines
  - Variability
    - Property-based alternatives, interface compatibility
  - Optionality
    - Property-based inclusion
- Results in modeling a product family/product line

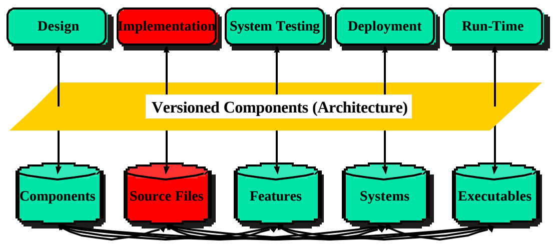
Ménage



Ménage

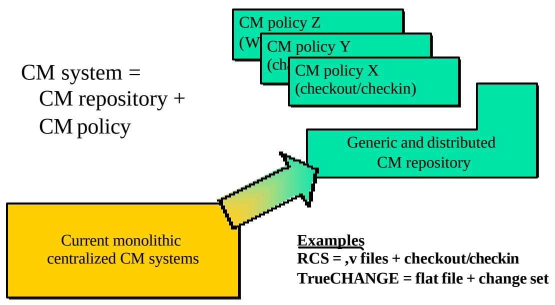


Long-Term Vision



Separation of Repository from Policy

CM system =
CM repository +
CM policy

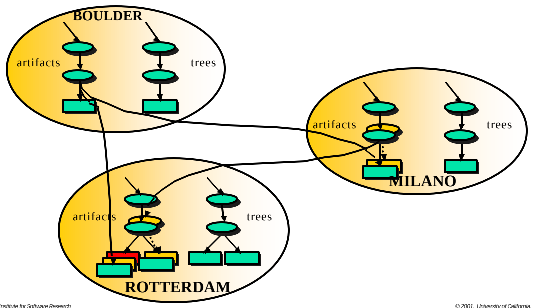

CM policy Z
(W
CM policy Y
(ch
CM policy X
(checkout/checkin)
Generic and distributed CM repository
Current monolithic centralized CM systems

Examples
RCS = ,v files + checkout/checkin
TrueCHANGE = flat file + change set

Peer-to-Peer Repositories Policy

- Pattern
  - checkout/checkin
- Manages compound artifacts
- Each artifact can be stored in a different location
  - cross-repository membership

Repository Design



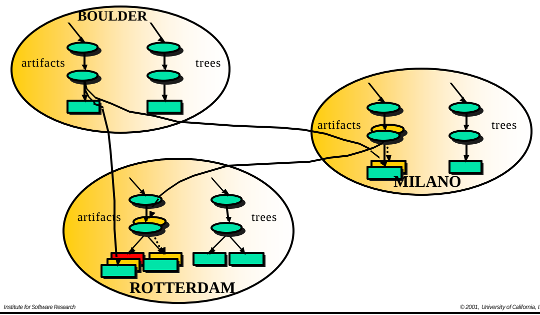
Core Policy Design

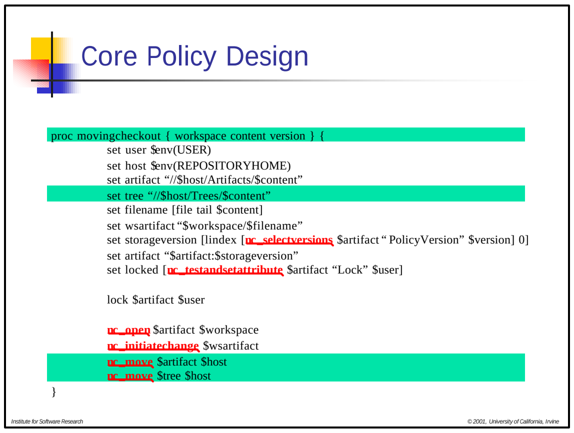
```
proc createfederation { myhost collection itshost theartifact } {
        set user $env(USER)
        set workspace "/tmp/workws"
        set filename [file tail $collection]
        set artifact "//myhost/Artifacts/$collection"
        set wsartifact "$workspace/$filename"

        lock $artifact $user

        nc_open $artifact $workspace
        nc_initiatechange $wsartifact
        nc_add //$itshost/theartifact
        nc_commitchange $wsartifact
        nc_close $wsartifact
        nc_removeattribute $artifact "Lock"
}
```

Movement upon Checkout Policy

- Pattern
  - peer-to-peer repositories
- Artifacts move from physical repository to physical repository
  - move is triggered by checkout

Repository Design

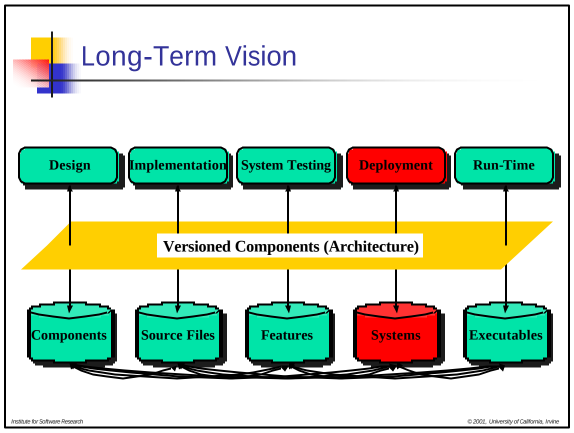




Core Policy Design

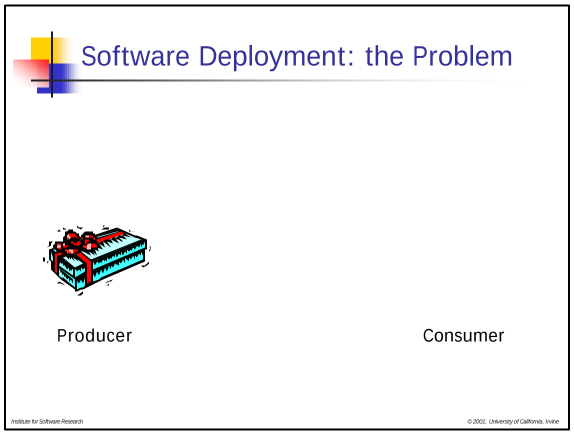
```
proc movingcheckout { workspace content version } {
            set user $env(USER)
            set host $env(REPOSITORYHOME)
            set artifact "//$host/Artifacts/$content"
            set tree "//$host/Trees/$content"
            set filename [file tail $content]
            set wsartifact "$workspace/$filename"
            set storageversion [lindex [pc_selectversions $artifact " PolicyVersion" $version] 0]
            set artifact "$artifact:$storageversion"
            set locked [pc_testandsetattribute $artifact "Lock" $user]

            lock $artifact $user

            pc_open $artifact $workspace
            pc_initiatechange $wsartifact
            pc_move $artifact $host
            pc_move $tree $host
}
```

Institute for Software Research © 2001, University of California, Irvine



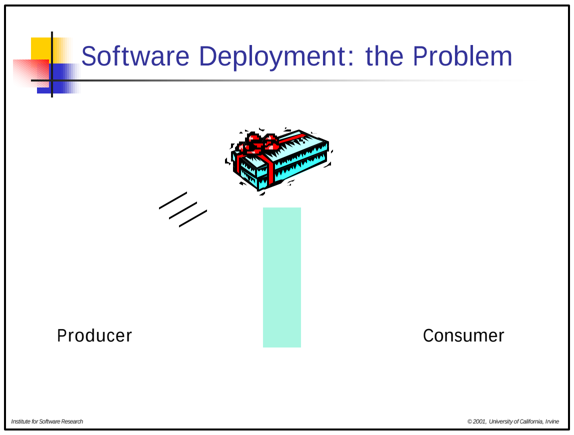
Long-Term Vision

Institute for Software Research © 2001, University of California, Irvine



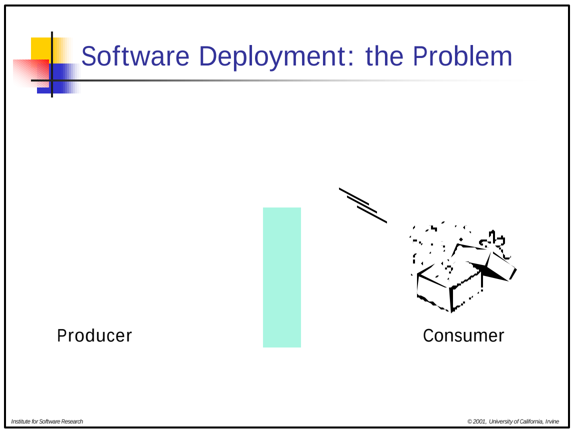
Software Deployment: the Problem

Institute for Software Research © 2001, University of California, Irvine



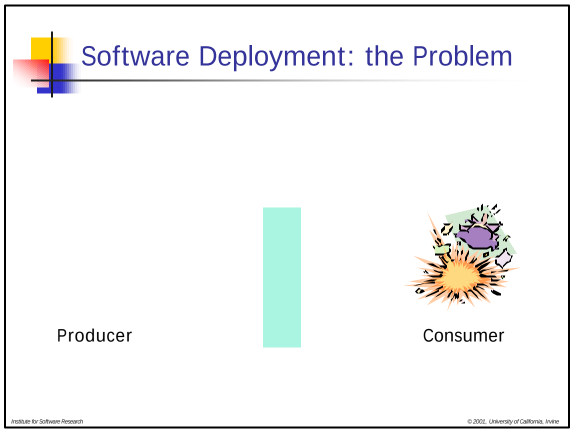
Software Deployment: the Problem

Institute for Software Research © 2001, University of California, Irvine



Software Deployment: the Problem

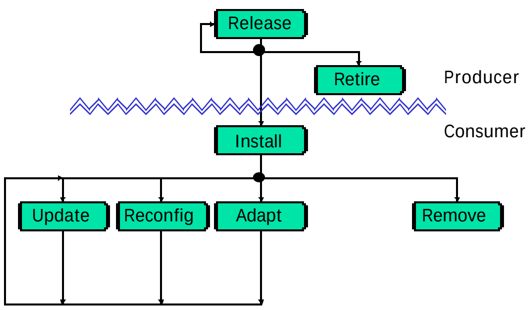
Institute for Software Research © 2001, University of California, Irvine



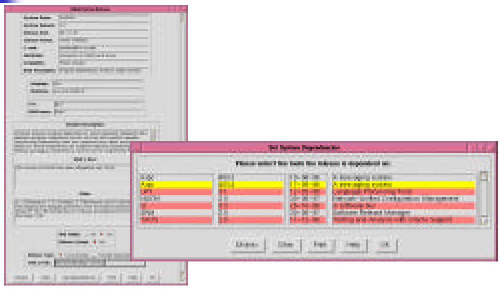
Software Deployment: the Problem

Institute for Software Research © 2001, University of California, Irvine

Software Deployment Life Cycle

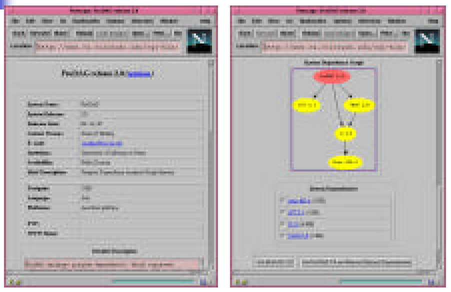


SRM



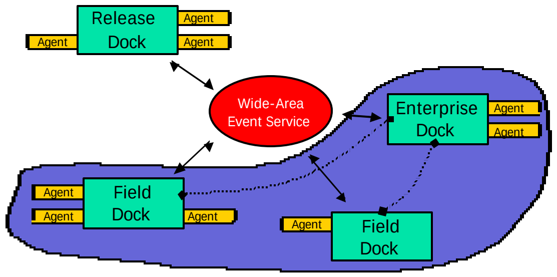
Releasing a complete system of systems.

SRM



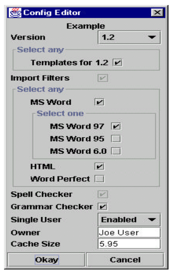
Retrieving a complete system of systems.

Software Dock



Agent-based deployment.

Software Dock



Config Editor

Example

Version 1.2

Select any
- Templates for 1.2 ✓

Import Filters ✓

Select any
- MS Word ✓
  - Select one
    - MS Word 97 ✓
    - MS Word 95 ☐
    - MS Word 6.0 ☐
- HTML ✓
- Word Perfect ☐

Spell Checker ✓

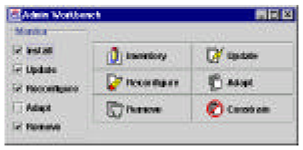
Grammar Checker ✓

Single User: Enabled

Owner: Joe User

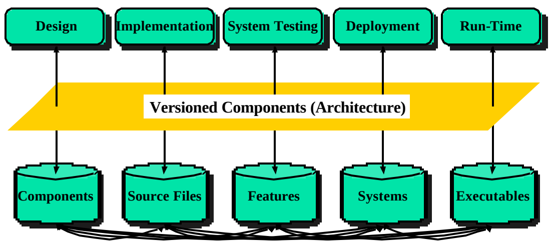
Cache Size: 5.95

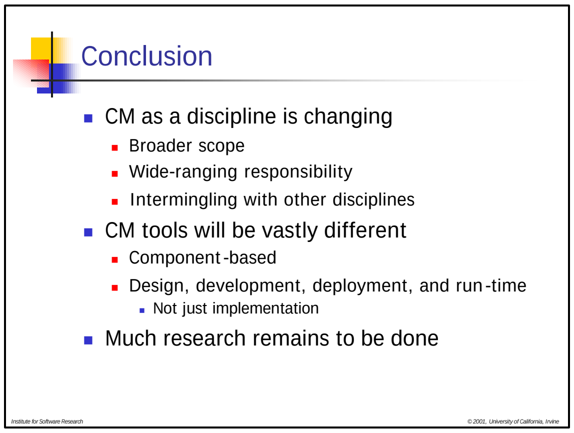
Okay Cancel



Enterprise, distributed control.

Long-Term Vision





Conclusion

- CM as a discipline is changing
  - Broader scope
  - Wide-ranging responsibility
  - Intermingling with other disciplines
- CM tools will be vastly different
  - Component-based
  - Design, development, deployment, and run-time
    - Not just implementation
- Much research remains to be done

Institute for Software Research