

Software Configuration Management



Lecture: Change & Defect Management

**René Krikhaar
Niels Veerman**

Question and Remarks



- ◆ Chapter 5 Software Release Methodology?
- ◆ Final Assignment?
- ◆ Contact with company?

Lecture 4: Branching Strategies



- ◆ Why do we need software branching ?
- ◆ Which steering factors are important in development control?

Lecture 5: Release Management



- ◆ What is Release Management?
- ◆ What is Installation Management?

Paper Presentation



- ◆ A Case Study of the Release Management of a Health Care Information System
 - ◆ Wessel Heringa
 - ◆ Peter van der Meer
 - ◆ Daan Zonneveld

Lecture Objectives



- ◆ Learn about Defect Management
- ◆ Learn about Change Management
- ◆ Learn about the implementation of Change/Defect Management

Defect



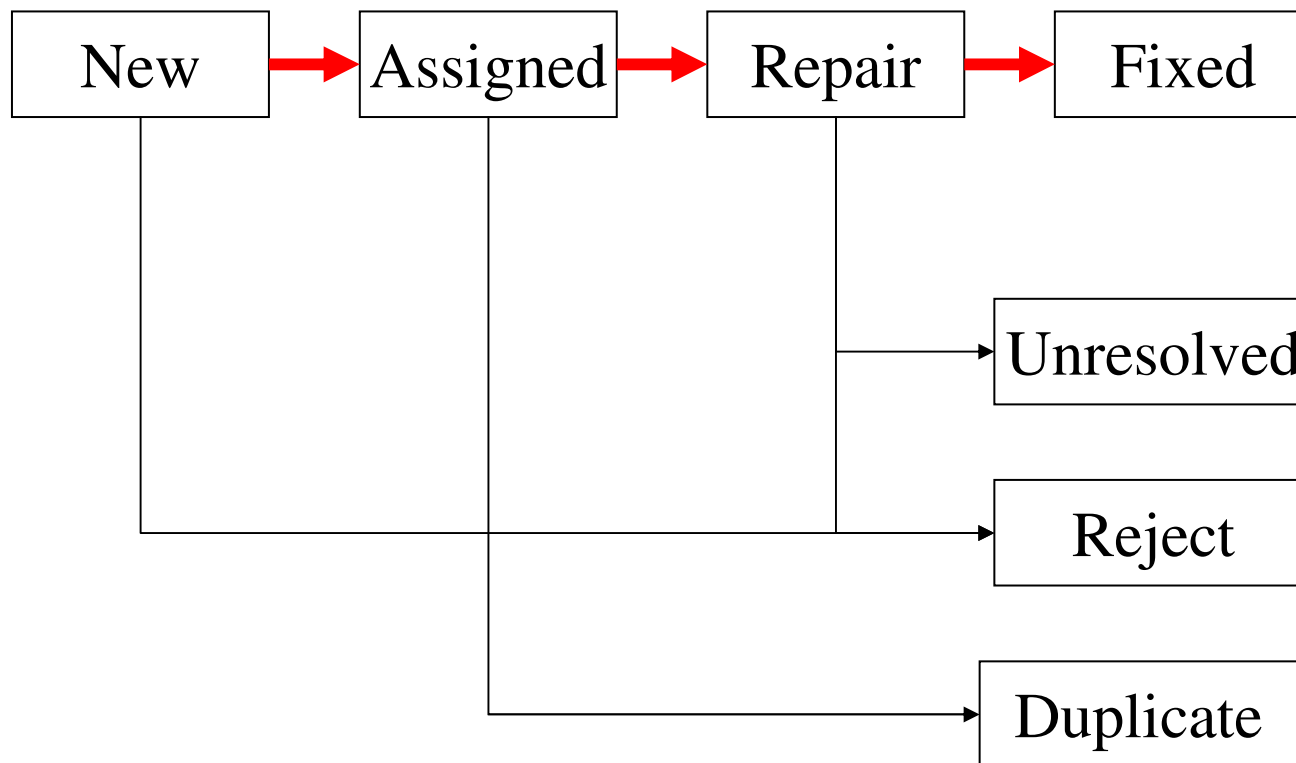
- ◆ Any nonconformance of a characteristic with specified requirements
 - ◆ Problem Report, PR
- ◆ Content? What is needed to describe a Defect?

Defect content



- ◆ Description
- ◆ Configuration Item
- ◆ Release / Baseline / Version
- ◆ Date the Defect Raised
- ◆ Consequences
- ◆ Log files
- ◆ Risks
- ◆ Severity
- ◆ Priority
- ◆ State

Defect Flow



Defect Tracking



- ◆ Which defects are resolved in this version?
- ◆ Did we fix Defect 12345?
- ◆ Is Defect 21345 resolved in platform X?
- ◆ Can I merge fix of defect 12345 in my branch?

Change Request



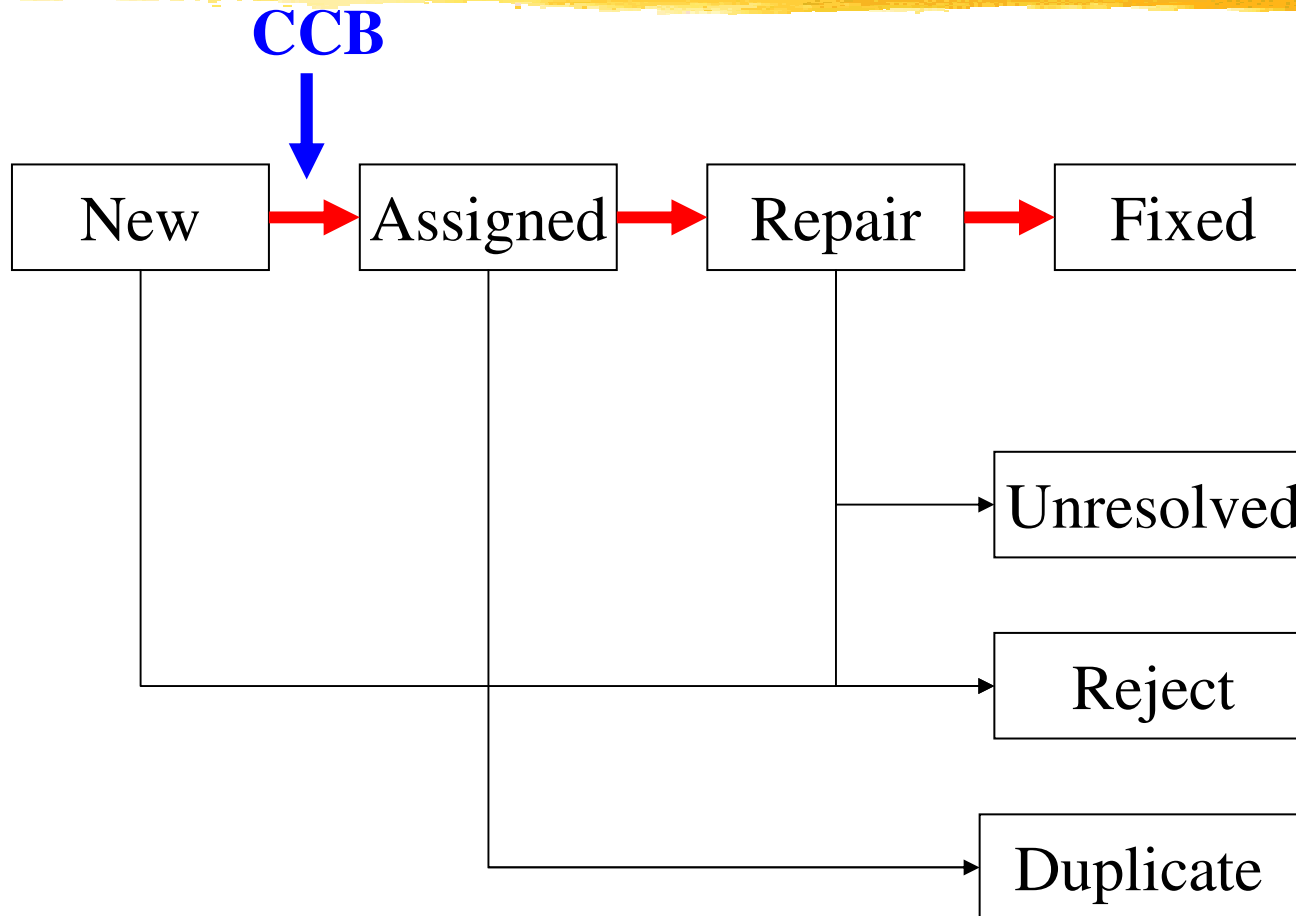
- ◆ What is Change Request?
- ◆ What is difference with Defect?
- ◆ A formal application for modification to any configuraiton item
- ◆ What is the impact on development?

Change Request Flow



- ◆ Change Request \leq Requesting Person
- ◆ Decision \leq Change Control Board
- ◆ Activity / Task \leq Project Leader
- ◆ Change Set \leq Developer (doc's, source, ...)
- ◆ Verification \leq Tester
- ◆ Releasing \leq Configuration Manager
- ◆ Validation \leq Requesting Person

Change Request Flow



Integration Change and Version Management



- ◆ CR / PR may result some actions to resolve it
- ◆ Resolution results in changes of files (doc's, source, test scripts, etc.) (by someone)
- ◆ New version of these files is committed; some reference to the CR/PR

More tight integration



- ◆ Task (Synergy), Activity (UCM), Ticket (TRAC), ...
 - ◆ Only changes possible **in** a TAT
 - ◆ Project Leader assigns TAT's
 - ◆ Connection Project Management and Change Management
 - ◆ Results in a Change Set
- ◆ Milestone (e.g. Release) = set of TAT's

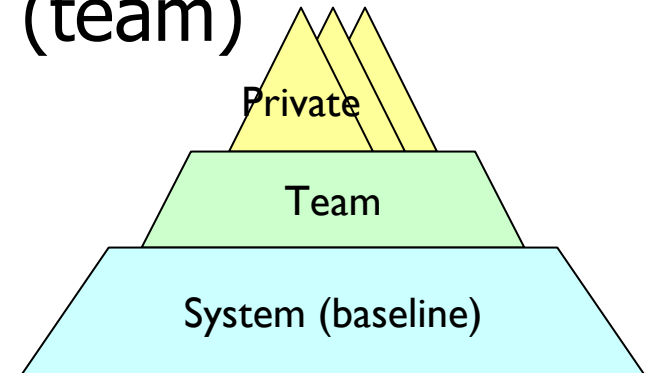
Change Set



- ◆ Set of CI's being modified for resolving PR or implementing CR
- ◆ In Theory:
 - ◆ Any (random) set of Change Sets is a new release providing the described features and fixed defects (given starting point: baseline)

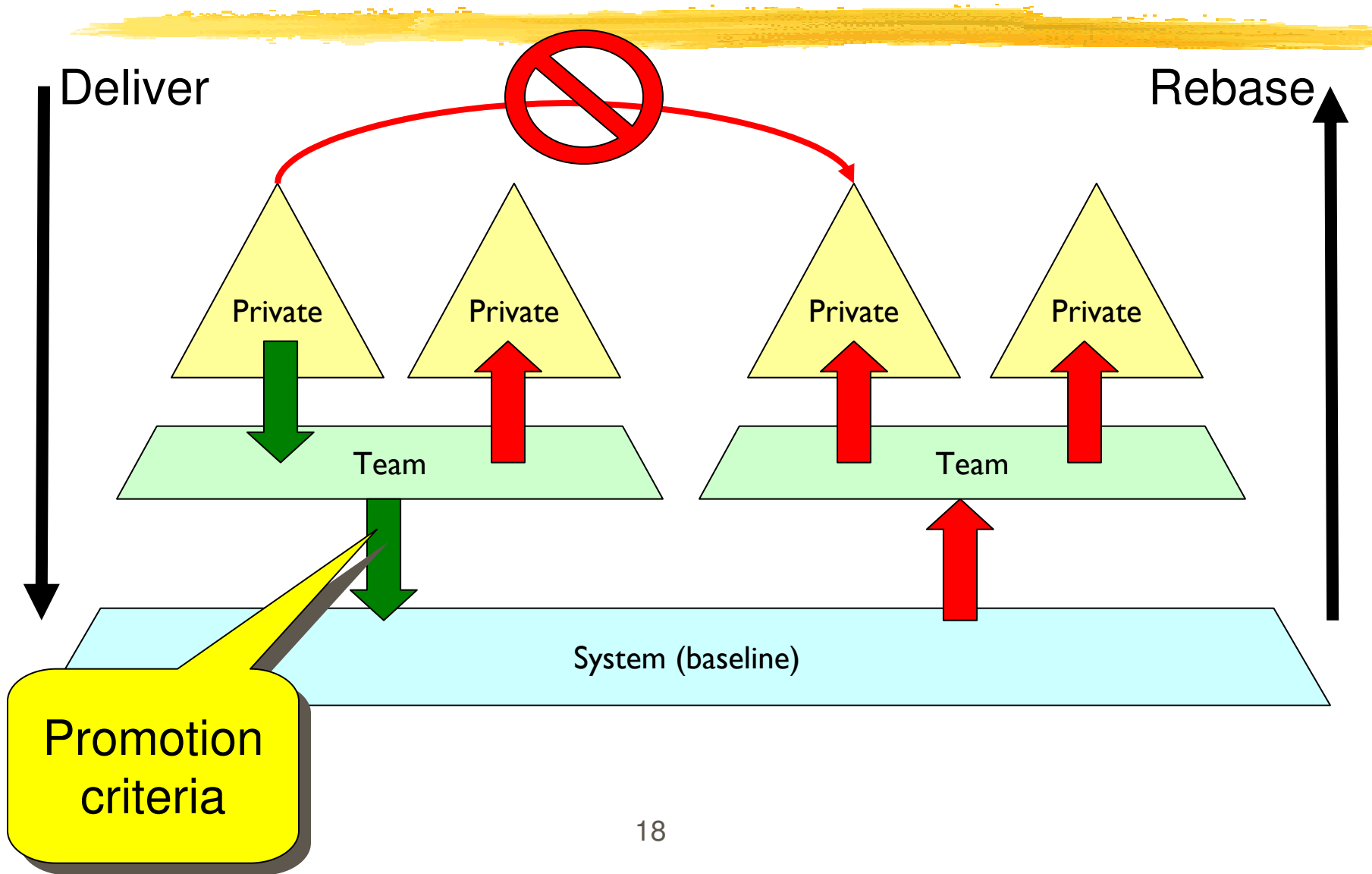
UCM -1-

- ◆ Configuration is a composition of
 - ◆ Workspace-specific changes (private)
 - ◆ Accepted integrated changes (team)
 - ◆ System baseline (public)

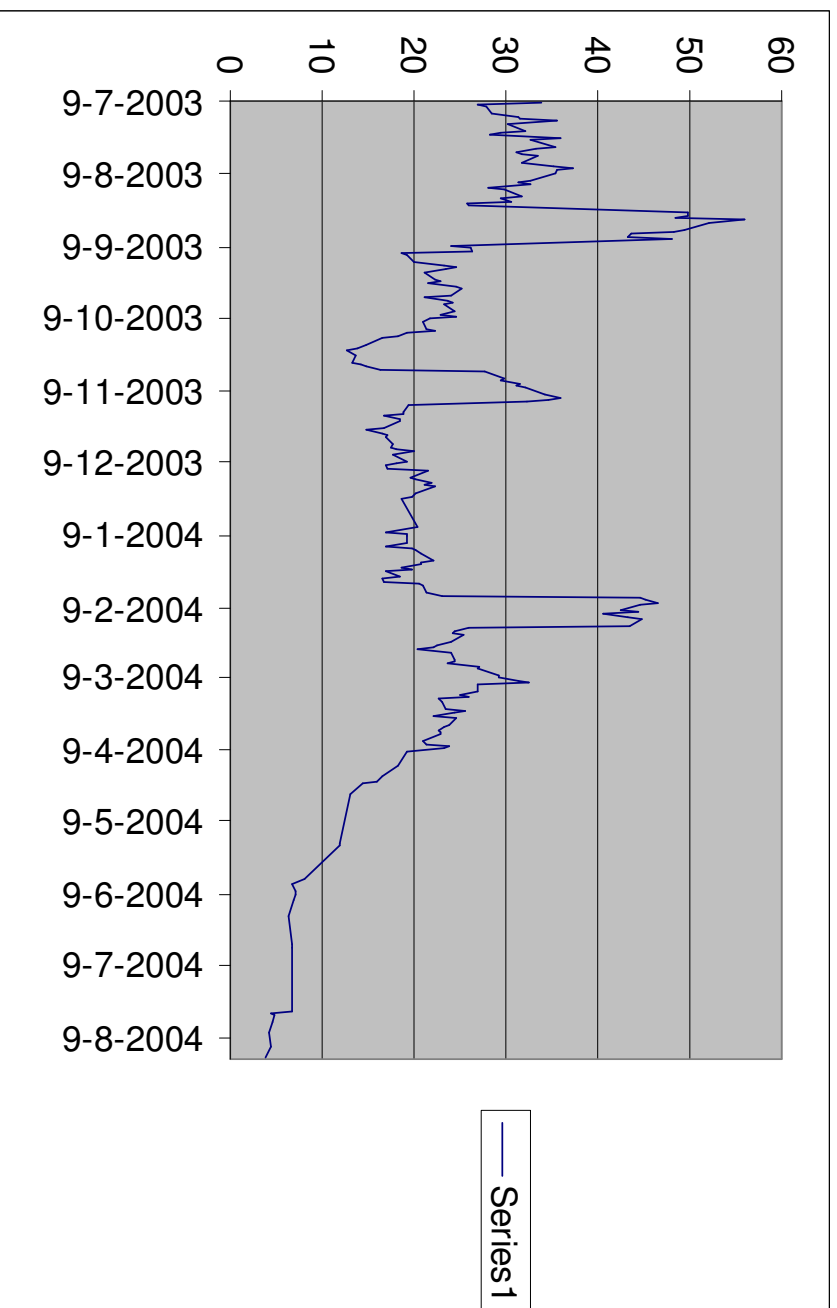


- ◆ **Baseline**
 - ◆ Reproducible configuration with a known quality

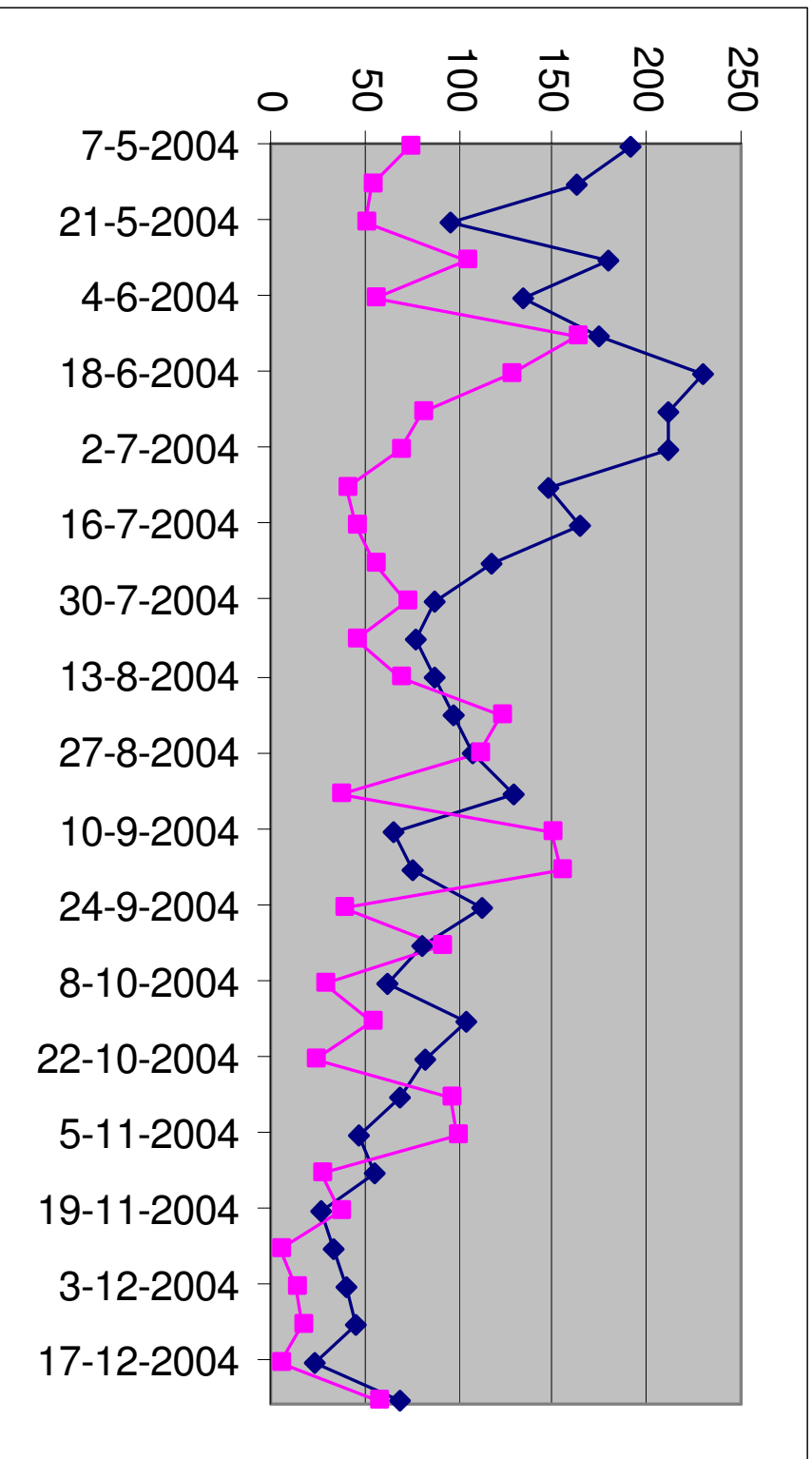
UCM -2-



Changgesets per day



New and Verified Defects



Conclusions



- ◆ Defects have to be stored and managed
- ◆ Change Requests can be handled similar to defects but scope and impact is larger
- ◆ Metrics provide insight in project status
- ◆ Implementation of Defect Management consists of
 - ◆ Clear description of content
 - ◆ Flow in which a defect exists
 - ◆ Actors make transitions in the flow

Functions in SCM



- ◆ Version Management
 - ◆ Versions and Code-lines
- ◆ Configuration Selection
 - ◆ Labeling
- ◆ Concurrent Development
 - ◆ Branching & Merging
- ◆ Distributed Development
 - ◆ Europe and India
- ◆ Build Management
 - ◆ Compile, Link
- ◆ Release Management
 - ◆ Packaging
- ◆ Workspace Management
 - ◆ Control & Merging
- ◆ Change Management
 - ◆ Changes & Defects Process
- ◆ Status Accounting
 - ◆ PR list & Document List
- ◆ Status Auditing
 - ◆ User Req. Satisfaction
- ◆ PDM Integration
 - ◆ Hardware Development

Homework



- ◆ **Next Lecture: 22 May 2007**

- ◆ Guest Lecture of OCE of Xavier Brankaert
- ◆ Configuration Management in relation to Software / Platform Development and Integration @ Océ

- ◆ **Paper Court Trial Game**

- ◆ ArchEvol: Versioning Architectural-Implementation Relationships
 - ◆ Elefelious G. Belay
 - ◆ Johan Piet
 - ◆ Jorge Lopez Figuerola

Paper Court Trial Game



- ◆ Paper Court Trial Game
 - ◆ 1 Person presents highlights of paper (15 min)
 - ◆ 1 Person defends paper (3 Arguments)
 - ◆ 1 Person attacks paper (3 Arguments)
 - ◆ 1 Discussion Leader
 - ◆ 1 Person makes minutes of discussion => WEBSITE
 - ◆ Mail to nveerman@cs.vu.nl **before** Monday
 - ◆ Class room is Jury
- ◆ Presentation/Game is part of examination
- ◆ Jury (you all) prepares by reading the paper