Chapter 17
Software Configuration Management

Object-**O**riented
Software **C**onstruction

Armin B. Cremers, Tobias Rho, Daniel Speicher & Holger Mügge
(based on Bruegge & Dutoit, Kniesel)

Outline of the Lecture

- Software Configuration Management (SCM)
  - Motivation: Why software configuration management?
  - Activities and roles in software configuration management
- Terminology and Methodology
  - What are Configuration Items, Baselines, etc. ?
  - What goes under version control?
- SCM Approaches

Why Software Configuration Management ?

- The problem:
  - Multiple people have to work on software that is changing
  - More than one version of the software has to be supported
    - Released systems
    - Custom configured systems (different functionality)
    - System(s) under development
    - … for different machines and operating systems

➔ *Need for coordination*

- Software Configuration Management
  - manages evolving software systems
  - controls the costs involved in making changes to a system
  - disciplines and techniques of initializing, evaluating and controlling change to software products during and after the software engineering process

Software-Development: Difficulties and Problems

Is requirement #462 impl. In current release ?

How many level 1 bugs still exist?

Analyst

Project Manager

Bug 98

Bug 179

Bug 251

Bug 348

Hopefully I didn't miss a file.

Why does this build not work?

Does bug #784 still exist?

Build 3

Build 2

Build 1

Developer

Integrator

Tester

SCM Activities

- Configuration item identification
  - modeling of the system as a set of evolving components
- Promotion management
  - is the creation of versions for other developers
- Release management
  - is the creation of versions for the clients and users
- Branch management
  - is the management of concurrent development
- Variant management
  - is the management of versions intended to coexist
- Change management
  - is the handling, approval and tracking of change requests

SCM Roles

- Configuration Manager
  - Identifies configuration items
  - Defines procedures for creating promotions and releases
- Change control board member
  - Approves or rejects change requests
- Developer
  - Creates promotions triggered by change requests or normal development activities
  - Checks in changes and resolves conflicts
- Auditor
  - Selects/evaluates promotions to be released
  - Ensures consistency and completeness of releases

Terminology and Methodology

- What are
  - Configuration Items
  - Baselines
  - Versions, Revisions and Releases

➔ The usage of the terminology presented here is not strict but varies for different configuration management systems.

Terminology: Configuration Item (CI)

- An aggregation of hardware and / or software
- Treated as a single entity in the configuration management process

- Software configuration items are not only program code segments but all type of documents according to development, e.g.
  - all type of code files
  - drivers for tests
  - analysis or design documents
  - user or developer manuals
  - system configurations (e.g. version of compiler used)

- In some systems, not only software but also hardware configuration items (CPUs, bus speed frequencies) exist!
  (Integration with Product Data Management PDM)

Finding Configuration Items (CIs)

- Large projects typically produce thousands of entities (files, documents, ...) which must be uniquely identified.

- But not every entity needs to be configured all the time. Issues:
  - What: Selection of CIs (What should be managed?)
  - When: When do you start to place an entity under configuration control?

➔ Starting too early introduces too much bureaucracy

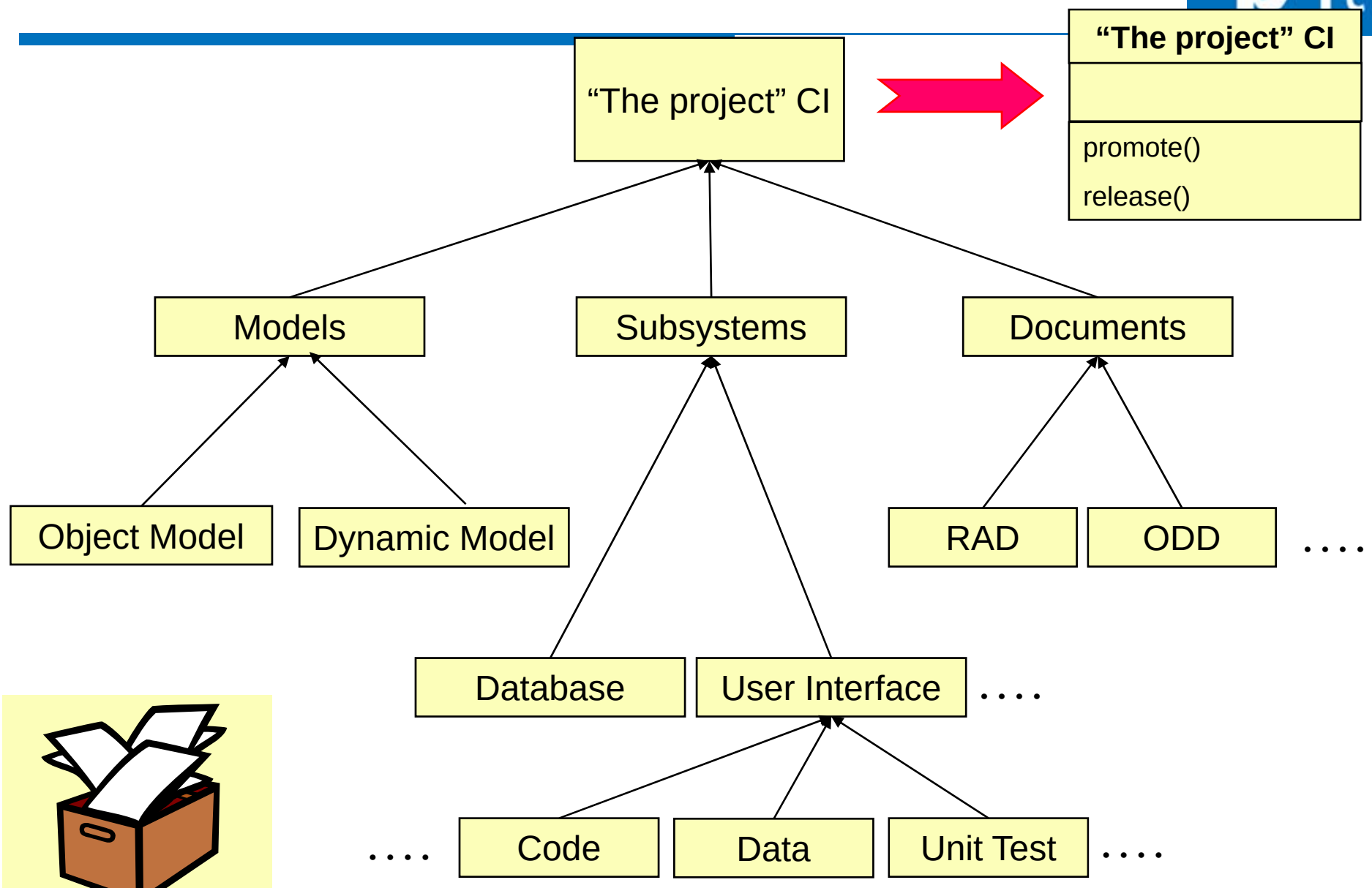
➔ Starting too late introduces chaos

Finding Configuration Items (continued)

- Very similar to object modeling
- Items follows subsystem decomposition
- Use techniques similar to object modeling for finding CIs

- An entity naming scheme should be defined so that related documents have related names.

Configuration Identification is similar to Object Identification



Terminology: Baseline

- *Selected project state*
- *A specification or product*
  - *formally reviewed and agreed to by responsible management*
  - *serves as the basis for further development*
  - *can be changed only through formal change control procedures*

Examples:

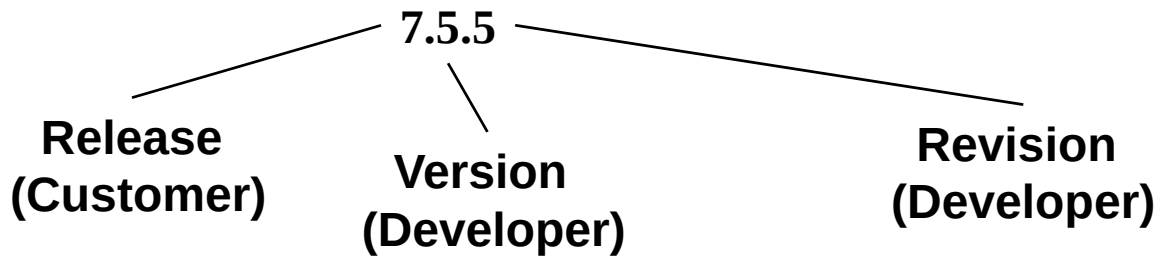
Baseline A: The API of a program is completely defined; the bodies of the methods are empty.

Baseline B: All data access methods are implemented and tested; programming of the GUI can start.

Baseline C: GUI is implemented, test-phase can start.

More on Baselines

- As systems are developed, a series of baselines is developed, usually after a review (analysis review, design review, code review, system testing, client acceptance, ...)
  - *Developmental baseline* (RAD, SDD, Integration Test, ...)
    - ➔ Goal: Coordinate engineering activities.
  - *Functional baseline* (first prototype, alpha release, beta release)
    - ➔ Goal: Get first customer experiences with functional system.
  - *Product baseline* (product)
    - ➔ Goal: Coordinate sales and customer support.
- Many naming schemes for baselines exist (1.0, 6.01a, ...)
- 3 digit scheme:



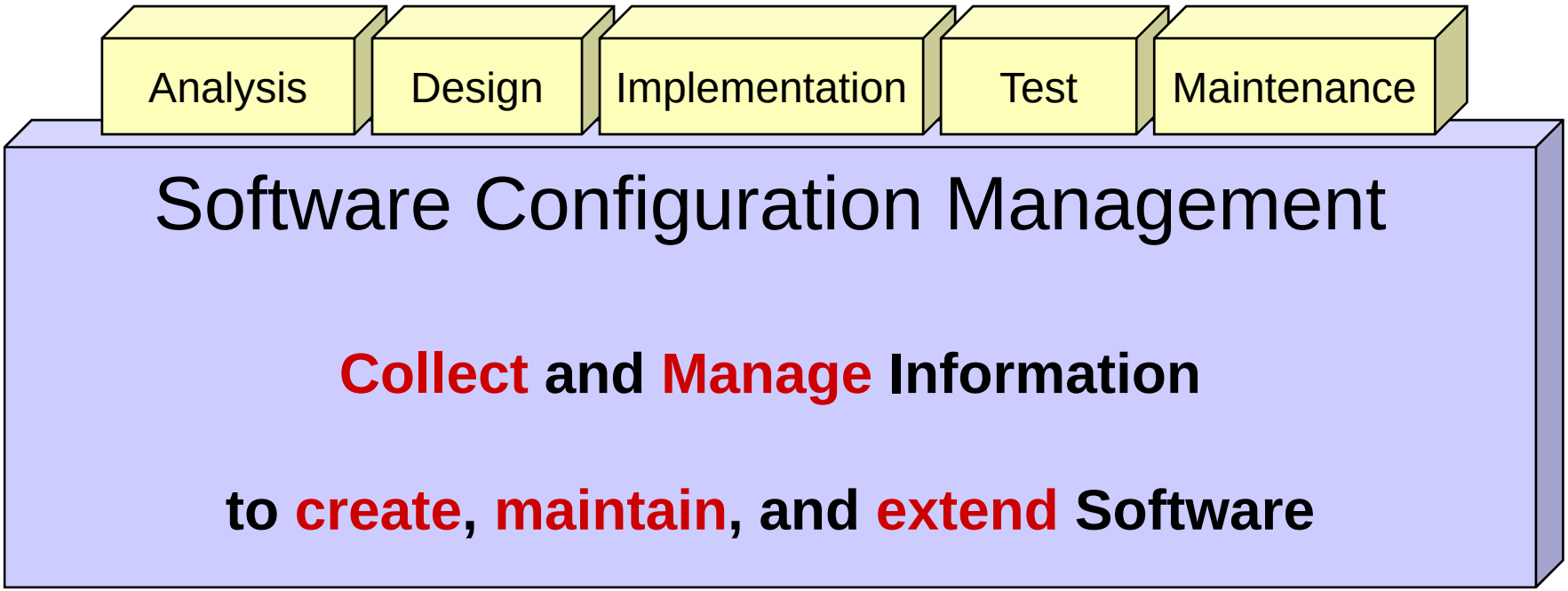
Version vs. Revision vs. Release

- Version:
  - An *initial* release or re-release of a configuration item
  - Associated with a *complete compilation* or recompilation of the item.
  - Different versions have different functionality.

- Revision:
  - *Change* to a version that corrects only errors in the design/code
  - Does not affect the documented functionality.

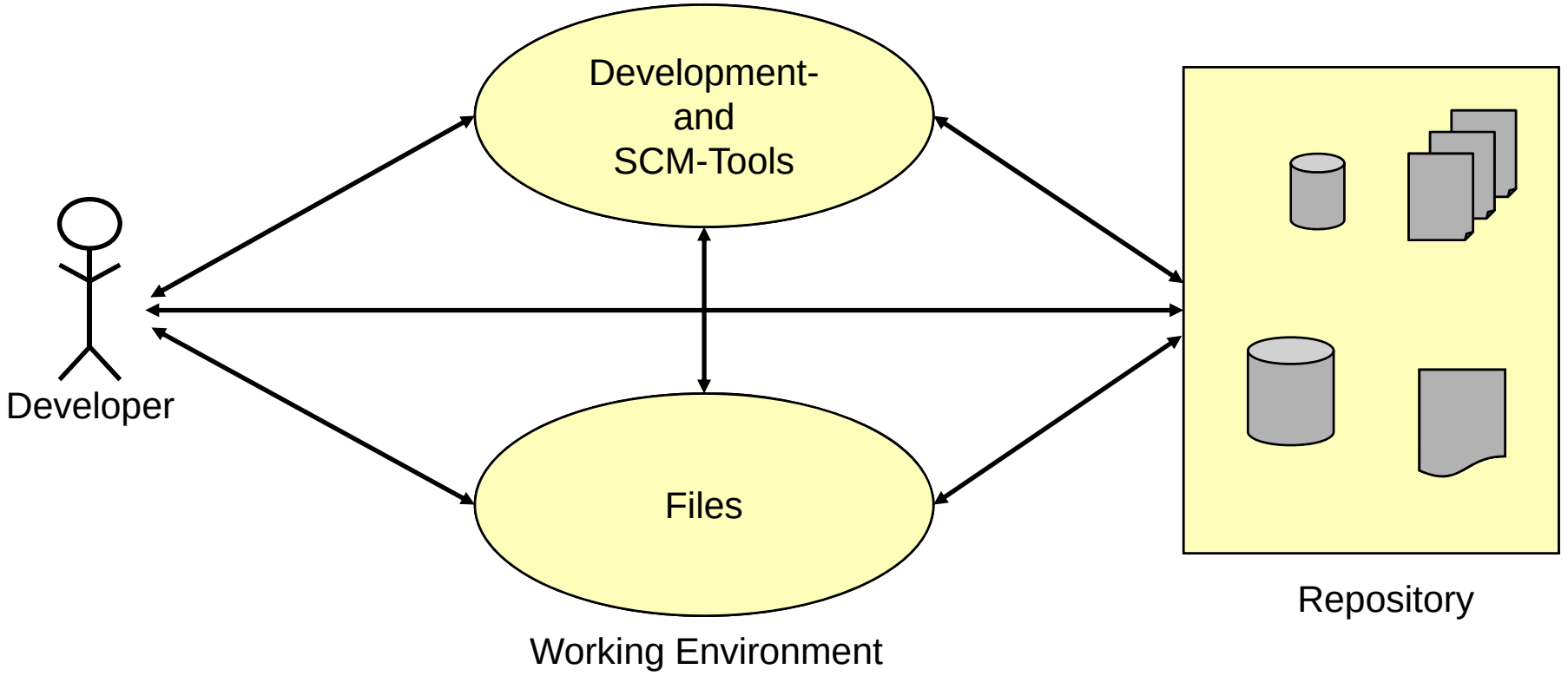
- Release:
  - The *formal distribution* of an approved version or revision.

SCM: The Fundament of the Development Process

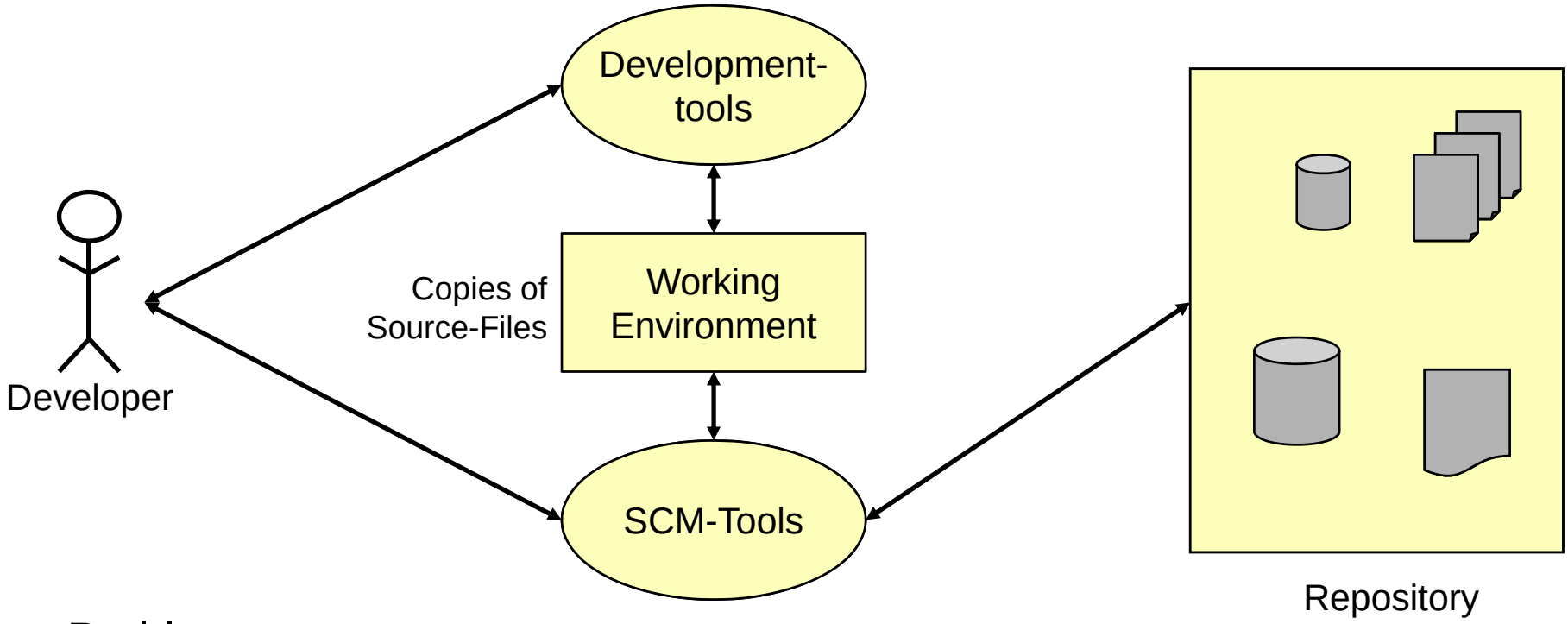


- Source code
- Binaries
- Documentation
- HTML/XML files
- Bitmaps & JPEGs
- cgi, java scripts
- Requirements
- Models
- Test scripts

Basic SCM-Approaches: General Scenario

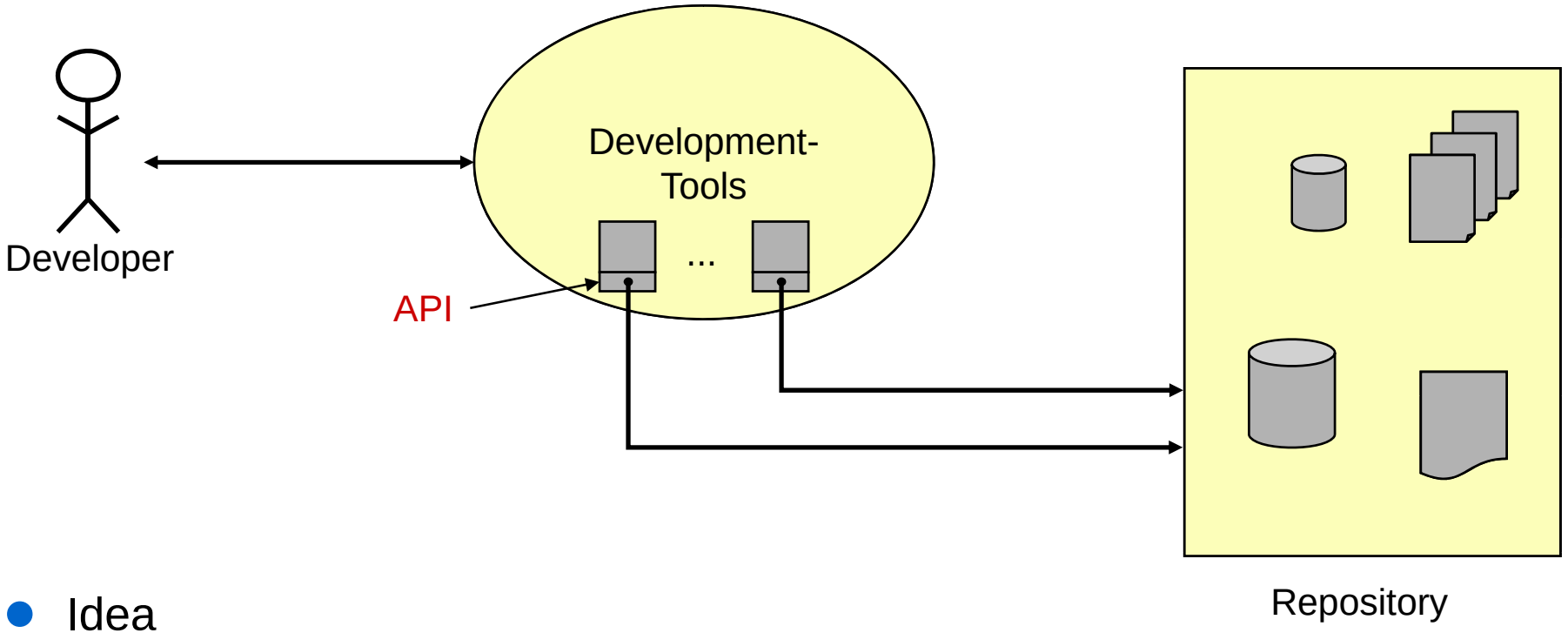


SCM-Approach: "Vault Model"



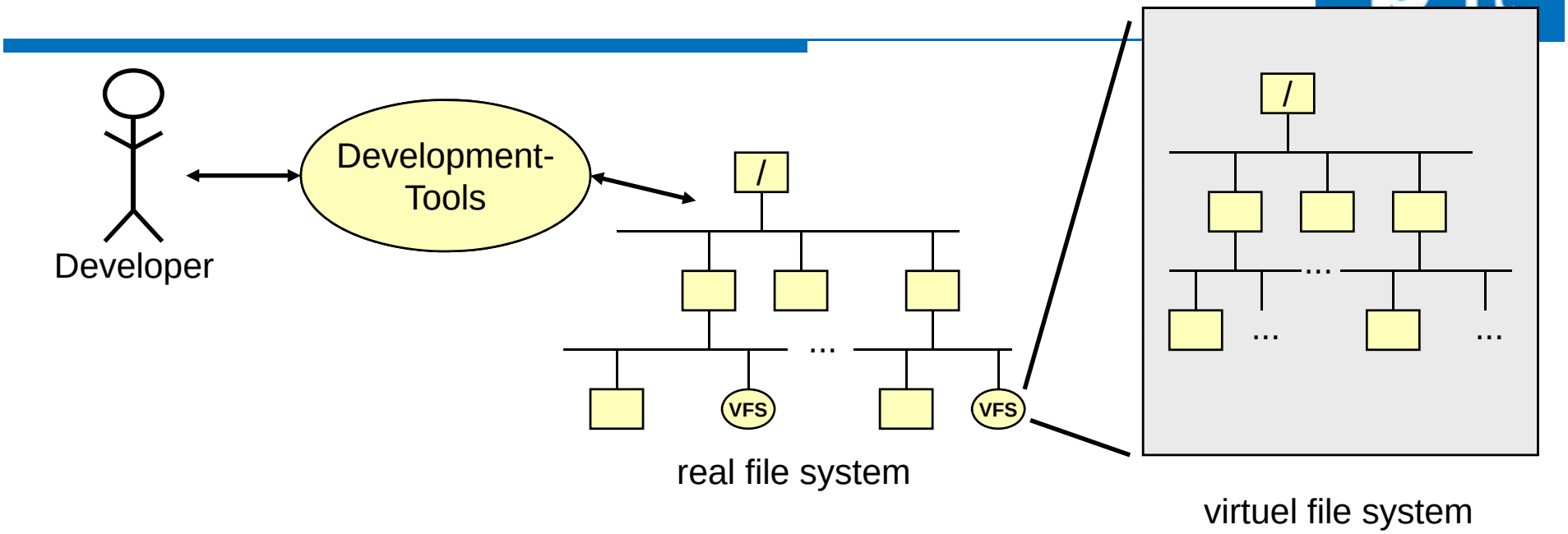
- Problems
  - Development tools can not directly access the repository
  - many private versions outside the repository
  - no central control of private copies by the SCM system (no global locking)
  - possible data loss by concurrent updates (lost changes)

SCM-Approaches: "Standard Repository"



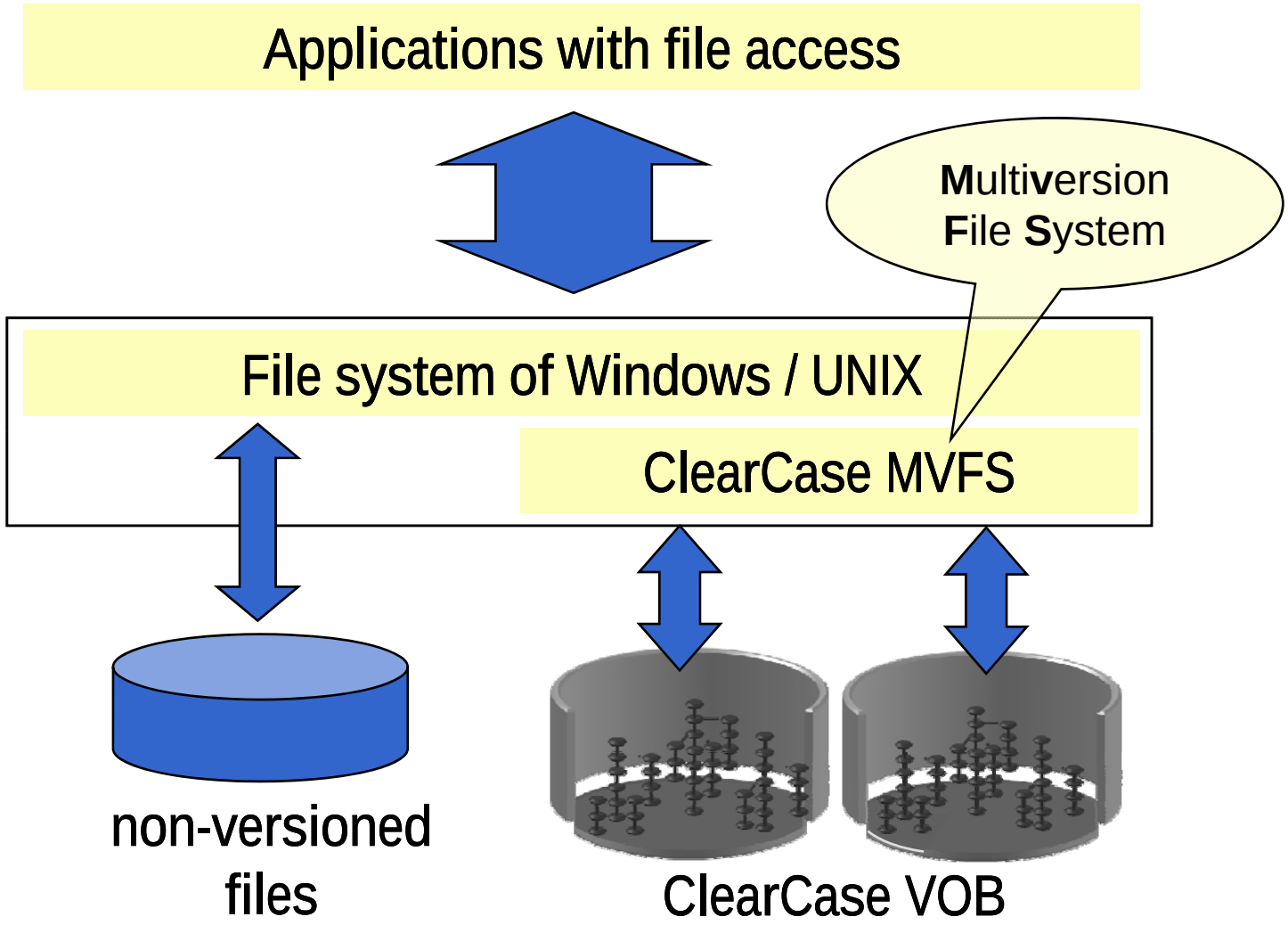
- Idea
  - All development tools directly access the repository via a given interface (API describes a "standard")
- Problems
  - All tools must be adapted to this standard
  - ... every time the standard is modified / extended

SCM-Approach: "Virtual File System" (VFS)

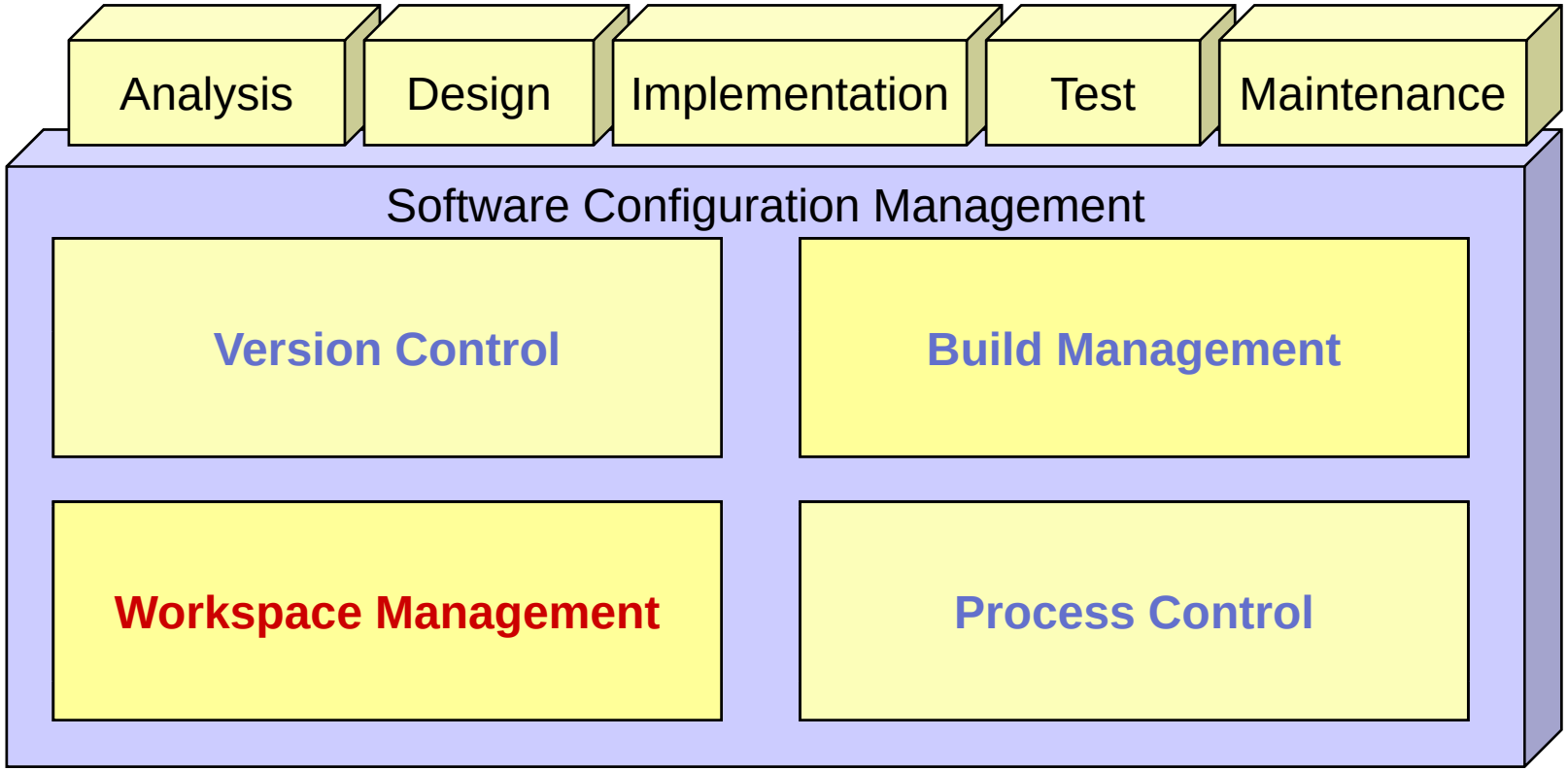


- Idea
  - Intercept all I/O - operations of the operating system (open, read, write) and bypass them to the repository
- Advantages
  - transparence, because the repository is shown as a regular directory tree
    - seamless integration with existing standard software
    - user does not have to change his mode of operation

ClearCase® Architecture



SCM: The Fundament of the Development Process



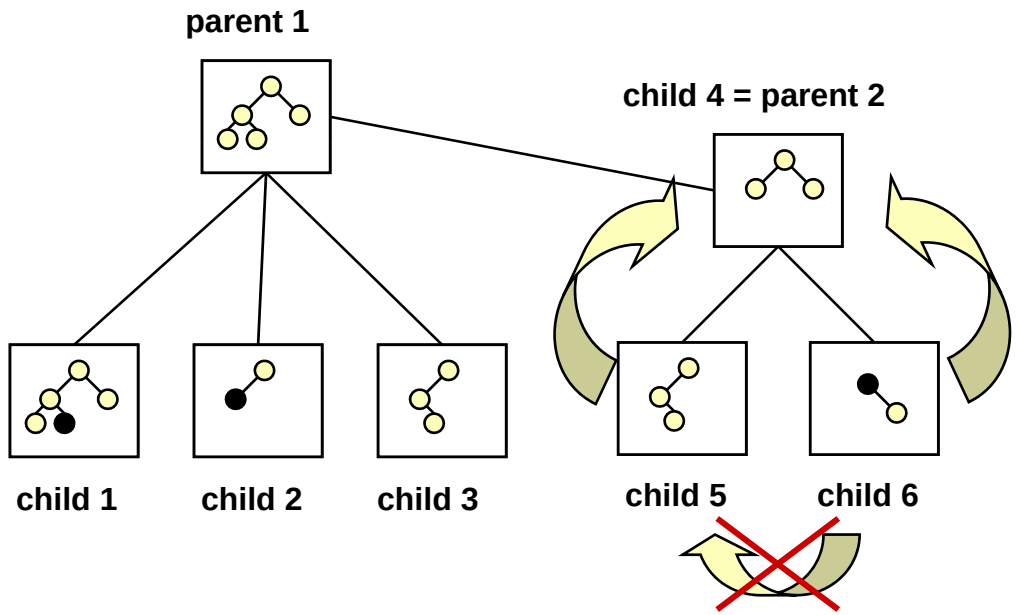
Workspace Management (1)

- Setup and maintenance of a personal working environment
  - set of needed objects
    - ➔ Sources, Binaries, ...
  - ... for a certain task
    - ➔ bug fixing, development of a new functionality or an extension

- Approaches
  - unconstrained:
    - ➔ no well-defined working environment needed
  - hierarchical sub-environments:
    - ➔ existing application areas can be copied and changed locally
  - change sets:
    - ➔ Original version of an object + set of changes of this object
  - rule-based environments:
    - ➔ needed objects are defined via rules

Workspace Management: Hierarchical Sub-Environments

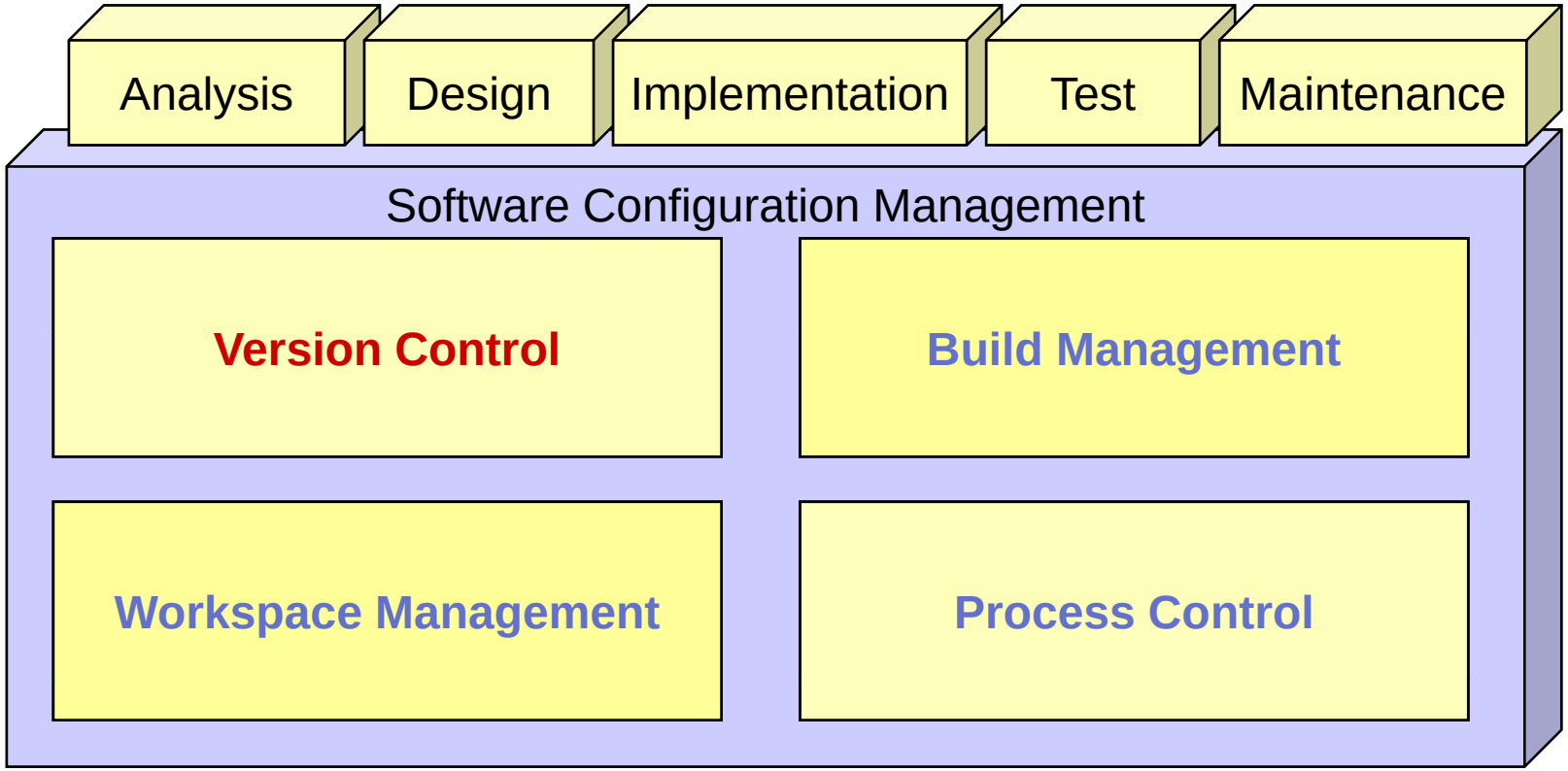
- Principle
  - copy - modify - merge



Workspace Management: Change Sets

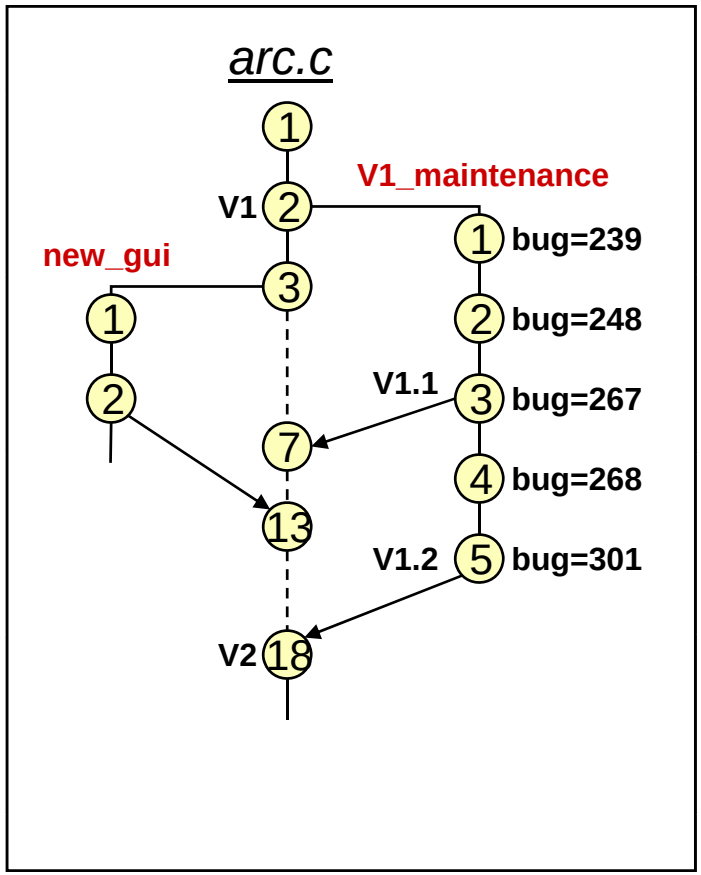
- Grouping of changes that base on each other
  - e.g. bug fixes
  - often used for the organization of on development line
  - at the end of a development process the cumulated changes are released as a new version (patch bundles, service release / pack)

SCM: The Fundament of the Development Process



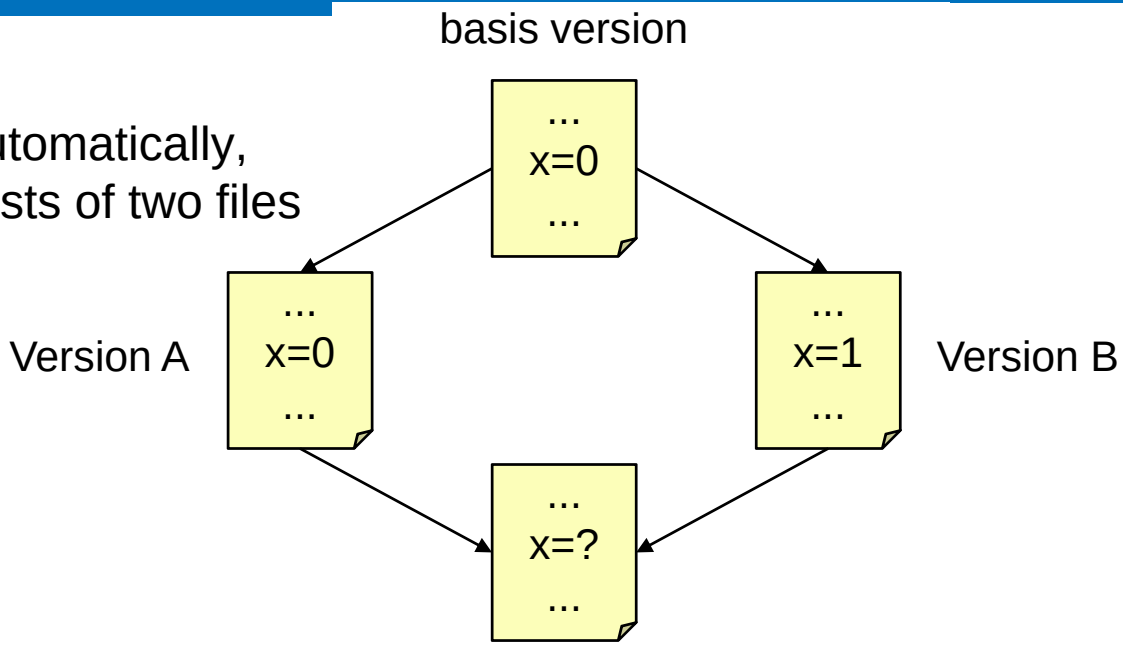
Version Control (1): Branching

- Branching Graph Model
  - Representation of the most important development lines and change sets
  - symbolic names for better comprehension and simpler references
  - every branch has additional administrative annotations
    - symbolic name
    - reference to a branch point
    - comments
    - ...



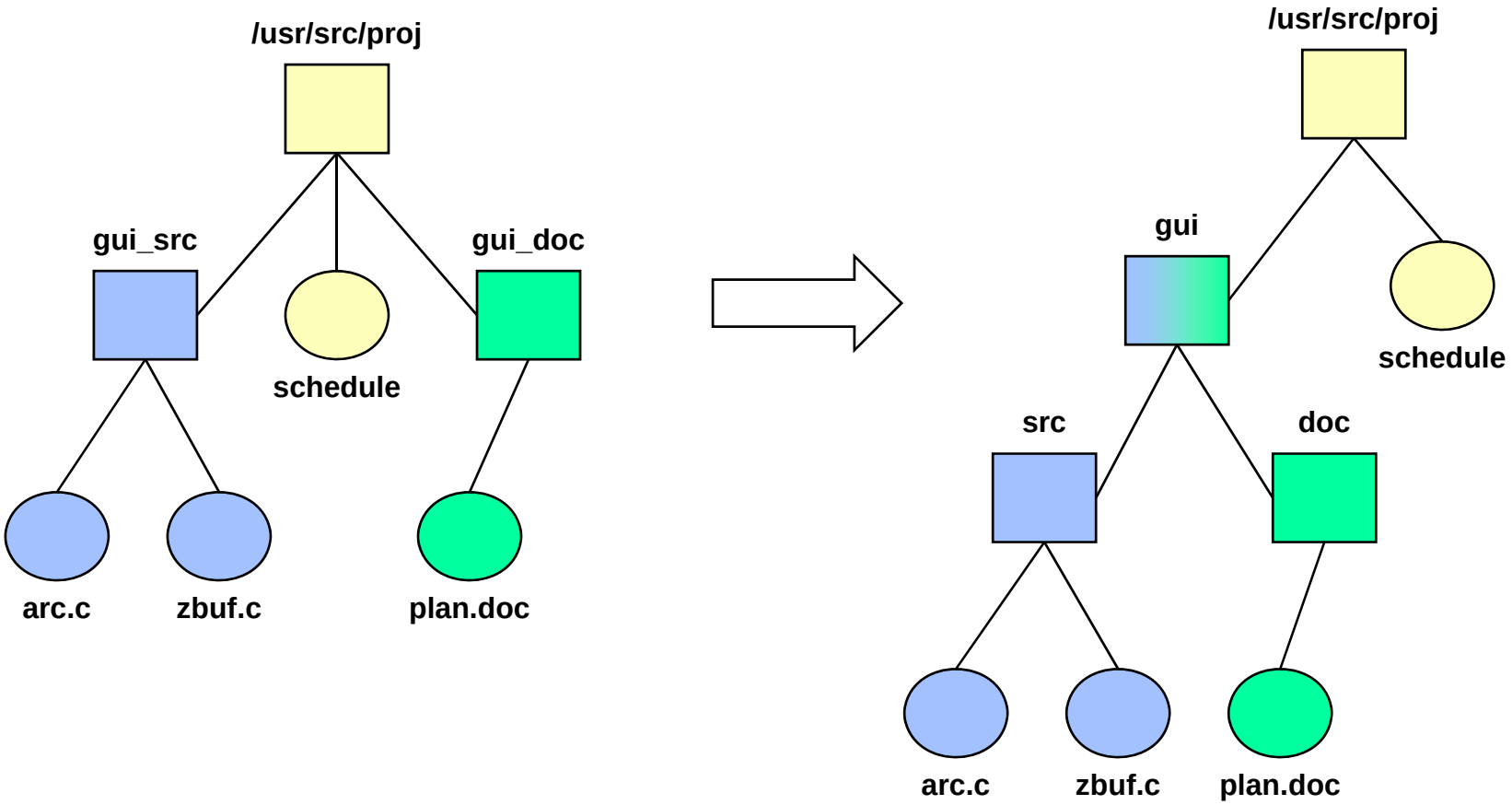
Version Control (2): Merging

- Problem
  - merging happens automatically, if shared acestor exists of two files is known



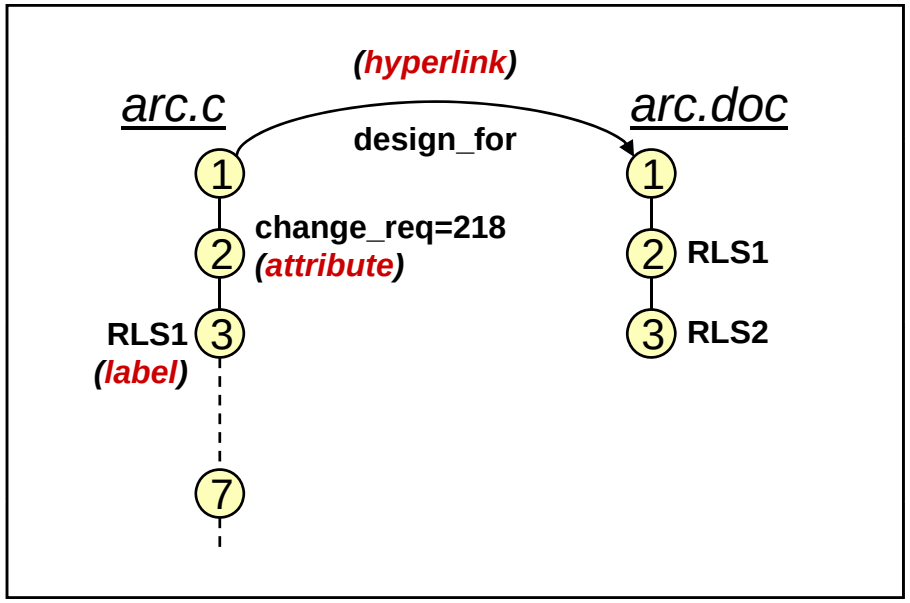
- experimental results
  - in approx. 90% of all cases the merging can is done without user interaction
  - approx. 1% of all merges was erroneous, most of these were detected becaus of syntax errors

Versioning of folders

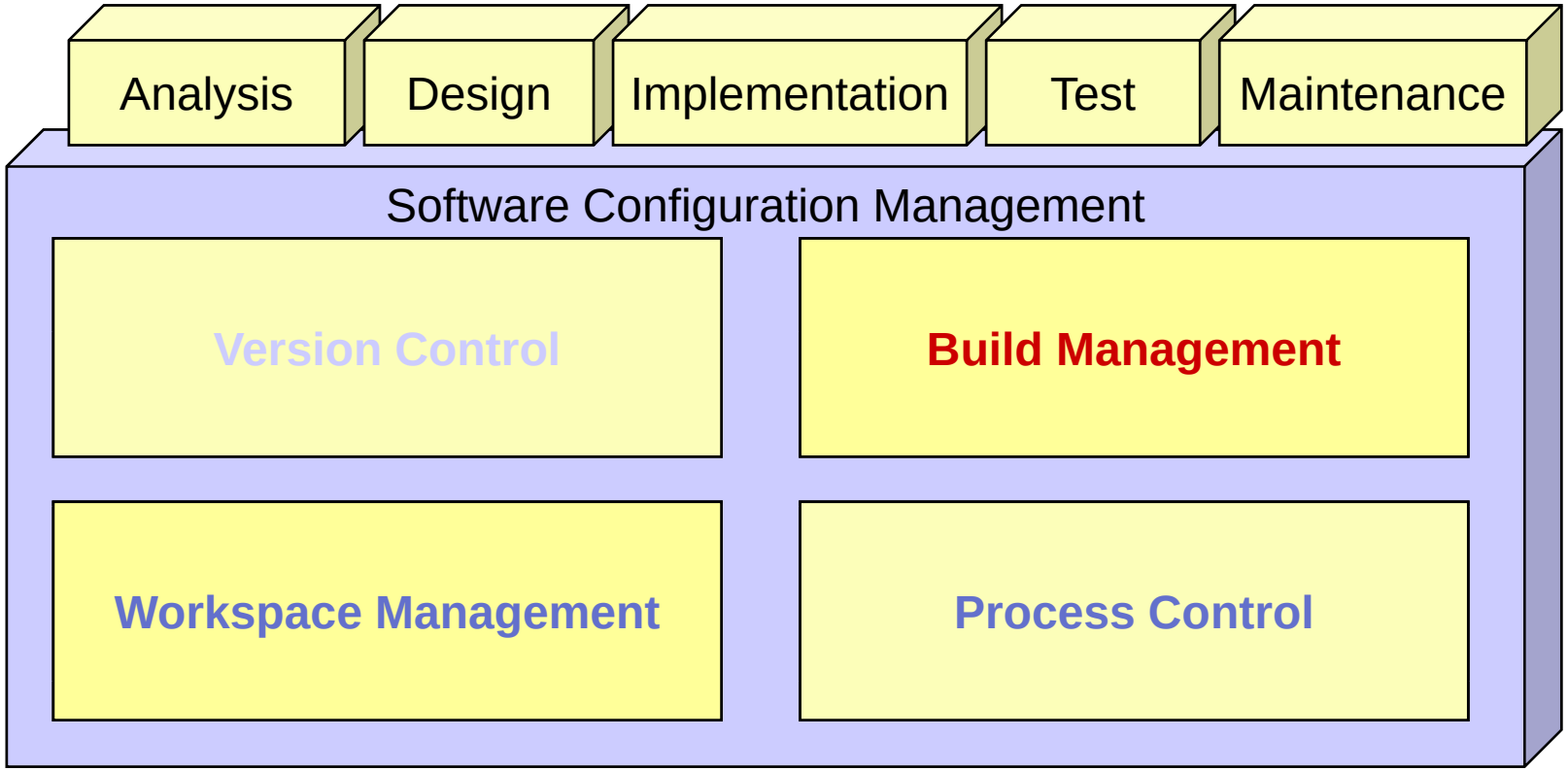


Version Control (3): Possible Extensions

- Labels
  - symbolic names
- Attributes
  - additional comments
- Hyperlinks
  - relationships
- triggers
  - ECA-rules
- Change Sets
- Annotations (on code level)
- Namespaces
- ...



SCM: The Fundament of the Development Process



Build Management (1)

- Management of all information necessary for the reproducible creation of derived objects

- necessary data:
  - correct versions of all source files and other derived objects
  - names and versions of all used tools
  - all parameter adjustments

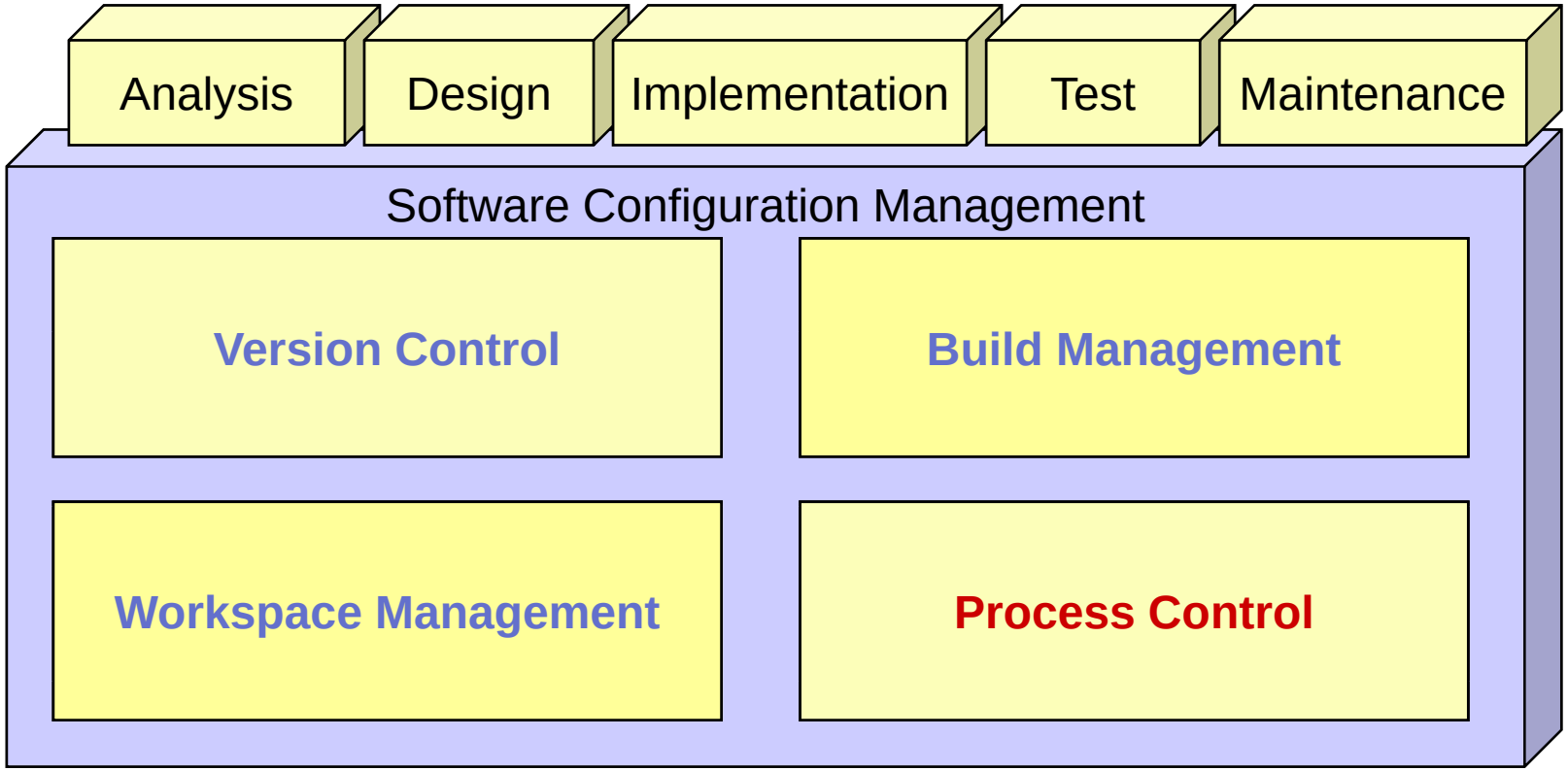
- resulting properties of build management
  - derived objects can be used by different users at the same time (binary sharing)
  - only the essential operations for the creation of a derived object must be carried out (minimal rebuilding)
  - a bill-of-materials list can automatically created which lists all needed components

Build Management (2)

- Problems with conventional tools
  - the "normal" make tool does not support binary sharing
  - insufficient BOM list prevents binary sharing, because it can not be decided if really "everything" related to the object is identical
  - minimal rebuilding requires analysis of dependencies for which specialized solutions (incremental compilers) exist, but no solution which can be generalized to an inhomogeneous build process

- Solution in ClearCase
  - The VFS tracks all I/O accesses by the compilers and other tools and thereby collects and logs all directly and indirectly referenced objects
  - clearmake is responsible for this task
    - ➔ integrated into IDEs (e.g. Eclipse, WebSphere Studio)

SCM: The Fundament of the Development Process



Process Control

- Development process encompasses analysis, design, implementation and maintenance of a software product

- SCM is just a part of this process
  - Identification of problems
  - Isolation of Problems,
  - Definition of procedures
  - Creation of time tables
  - Organization of (automatic) testing
  - measurement of software properties

- tools to automate the development processes: workflow manager
  - action-response concept
  - working on a task may influence or start other tasks
  - Unified Change Management (Rational/IBM)

Summary

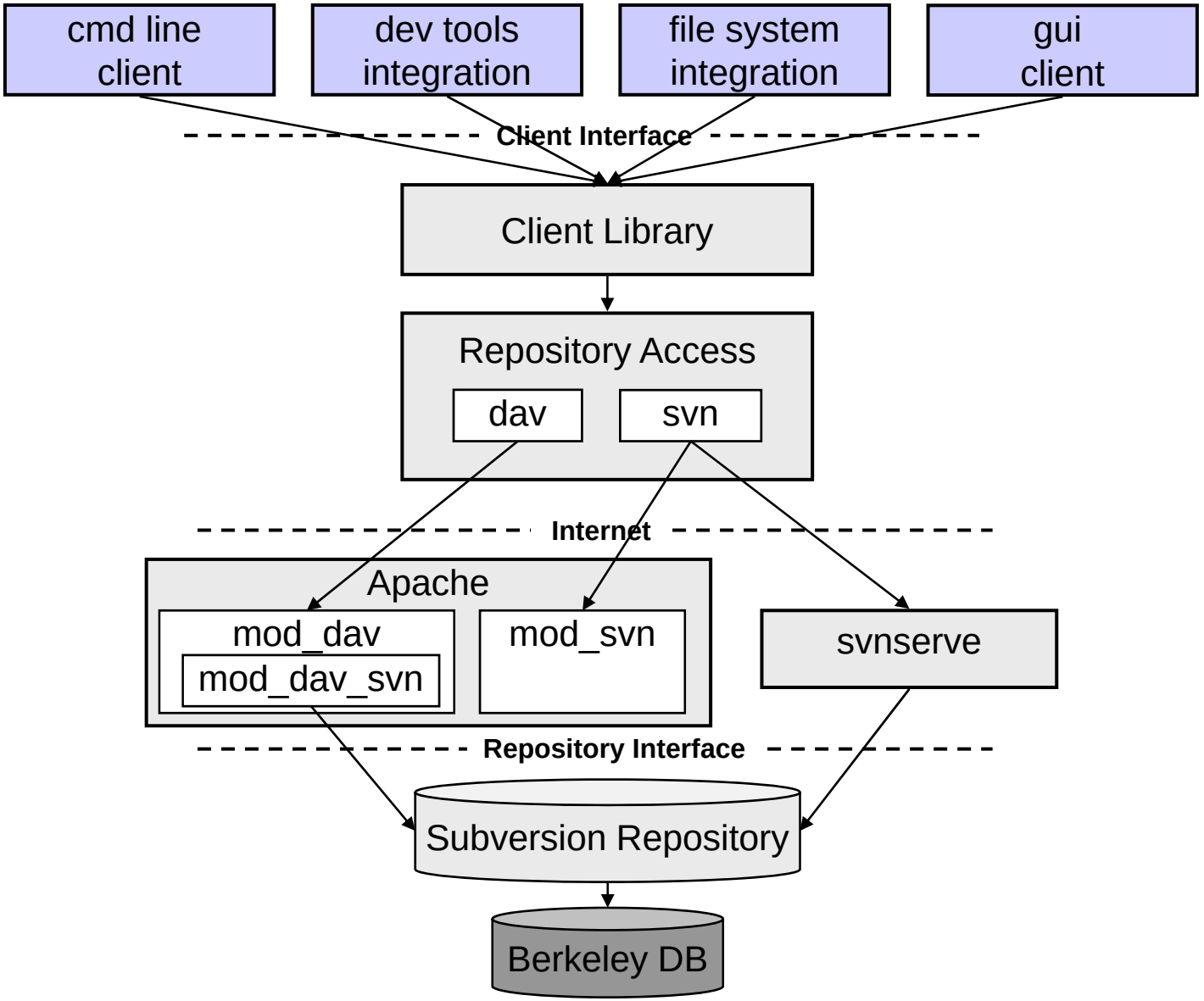
- Software Configuration Management (SCM)
- Configuration Items, Baselines
- What goes under version control?
- Fundament of the Development Process
  - Version Control
  - Workspace Management
  - Build Management
  - Process Control
- SCM Approaches

Spare

Subversion – Alternative for Version Control

- Open-source tool for software versioning
- 'real' rename, move, copy (in contrast to CVS)
- directories, renaming and file metadata is versioned
- atomic commits
- robust network layer (WebDAV DeltaV)
- based on DBMS (BerkleyDB) or file-based repository
- Branches / tags in constant time
- clean text / binary handling
- concept for user and access rights

Subversion Architeture for a VFS



VFS Open-Source Solution
Apache + Subversion + WebDAV

- Clients
  - Linux: davfs2 file system driver
  - Windows: Novell Netdrive
- Branch/Versions
  - directly visible
- Restrictions
  - Synchronization of changes
  - Only support for auto-versioning
  - no view support

ClearCase Summary

- ClearCase Pros
  - Industrial strength
  - Excellent merge tools
  - Good GUI on Windows
  - Proven reliability
  - Scales up well
- ClearCase Cons
  - Heavyweight server and client
  - Very steep learning curve for users
  - All merges are server based
    - ➔ Merges over high latency links are slow
  - Very expensive
  - Weak GUI on Unix platforms
  - Server communication is RPC based so anything over a high latency link is slow
  - High administration overhead

Subversion Summary

- Subversion Pros
  - Widespread IDE support ,editor and file system / tool integrations
    - Eclipse: http://www.polarion.org/index.php?page=overview&project=subversive
    - Windows Explorer: http://tortoisesvn.tigris.org/
  - lightweight server and client
  - Quick learning curve for users (many similarities with cvs)
  - Fast and efficient on the network
    - Flexible architecture
    - Low administration overhead
- Subversion Cons
  - Merge tracking is currently completely manual
  - Weak support for checking out part of a repository (Views)
  - *supports only version control*
  - Transparent work on the repository is limited
  - No sharing of derived objects