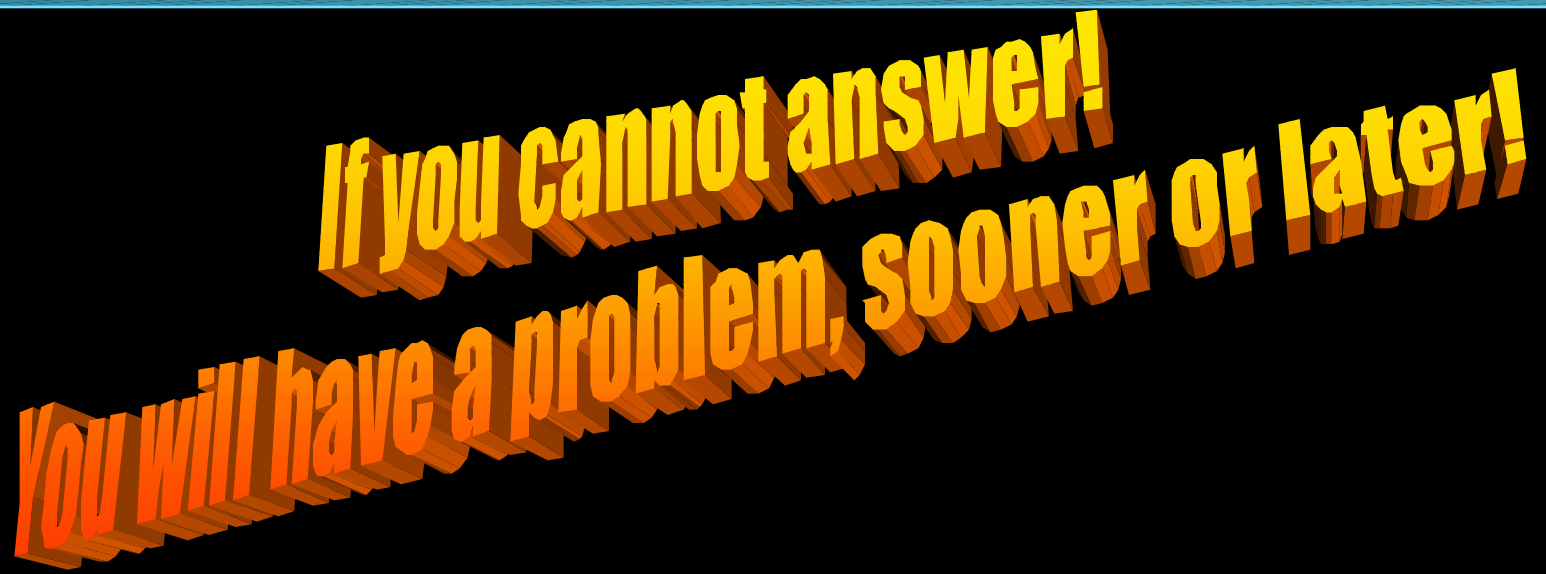
Using XML for Archiving Software-Lifecycle Artifacts in Revision Control Systems

Holger Dörnemann
Technical Lead, Rational Software GmbH

Motivation

- Modern software projects are very complex, in terms of
  - project size (requirements, duration, man power, roles)
  - deliverables to produce
- Can you answer the following questions for your entire project (any project you dealt with):
  - Which requirements met release X.Y.Z?
  - What did you test against the release of October 19xx?
  - What specifications lead to the release labeled with ‘THE_BEST_RELEASE’

Motivation



- Can you answer the following questions for your entire project (any project you dealt with):
  - Which requirements met release X.Y.Z?
  - What did you test against the release of October 19xx?
  - What specifications lead to the release labeled with ‘MANUA_LOA_RELEASE’

Software Engineering Processes

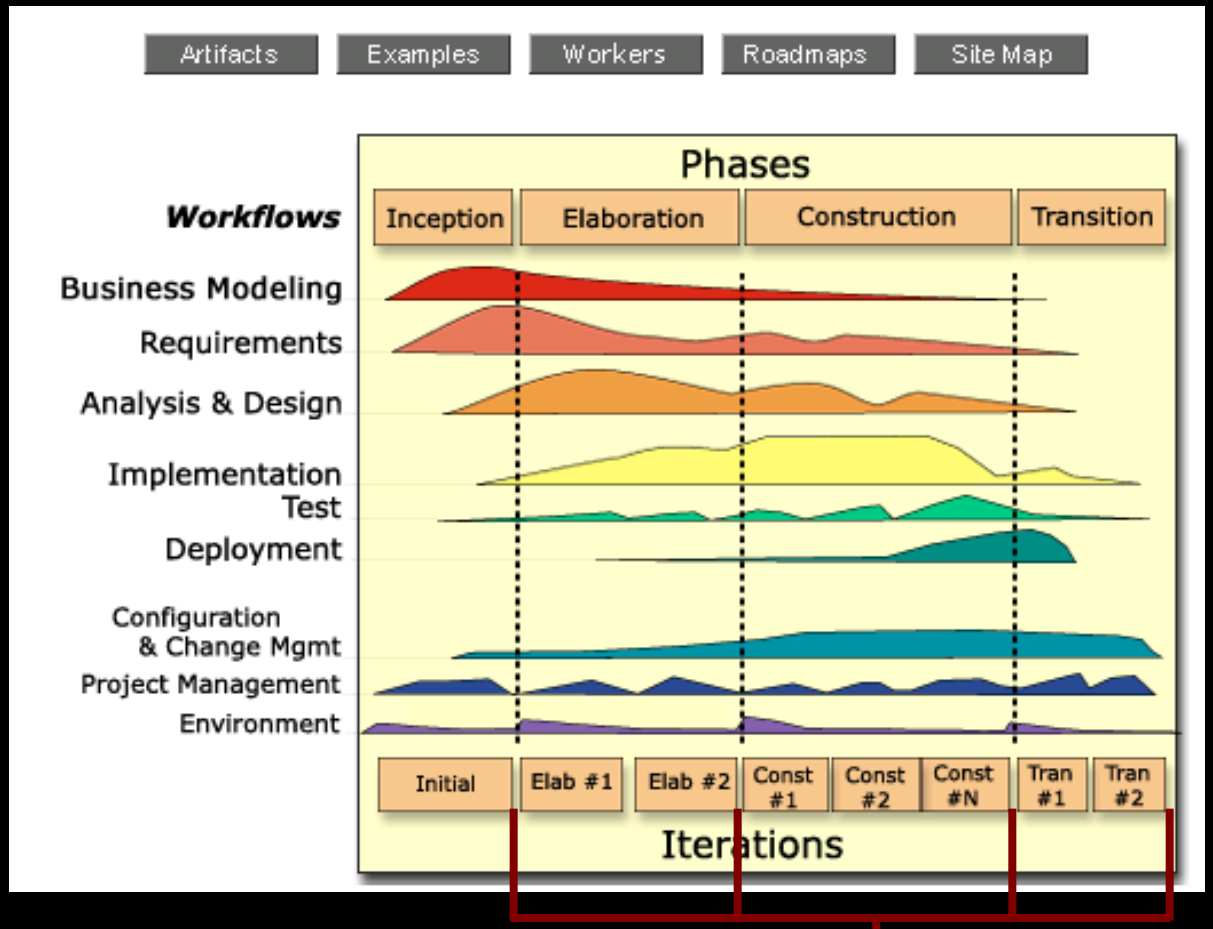
- To deal better with complex software projects many different engineering processes have been introduced
- Common elements of software processes are:
  - roles - people who do something
  - activities - the work they do
  - deliverables - the result of their work
  - workflows - oder in which things have to be done
- The Rational Unified Process defines
  - WORKERS, ACTIVITIES, ARTIFACTS and WORKFLOWS

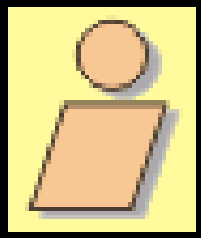
Software Engineering Processes

- To deal better with complex software projects many different engineering processes have been introduced

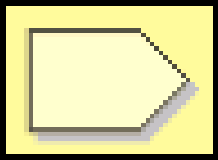
- The Rational Unified Process defines
  - WORKERS, ACTIVITIES, ARTIFACTS and WORKFLOWS

Rational Unified Process





Worker



Activity

Artifacts are:

- Documents
- Models
- Database data
- Assets in general
- Application!

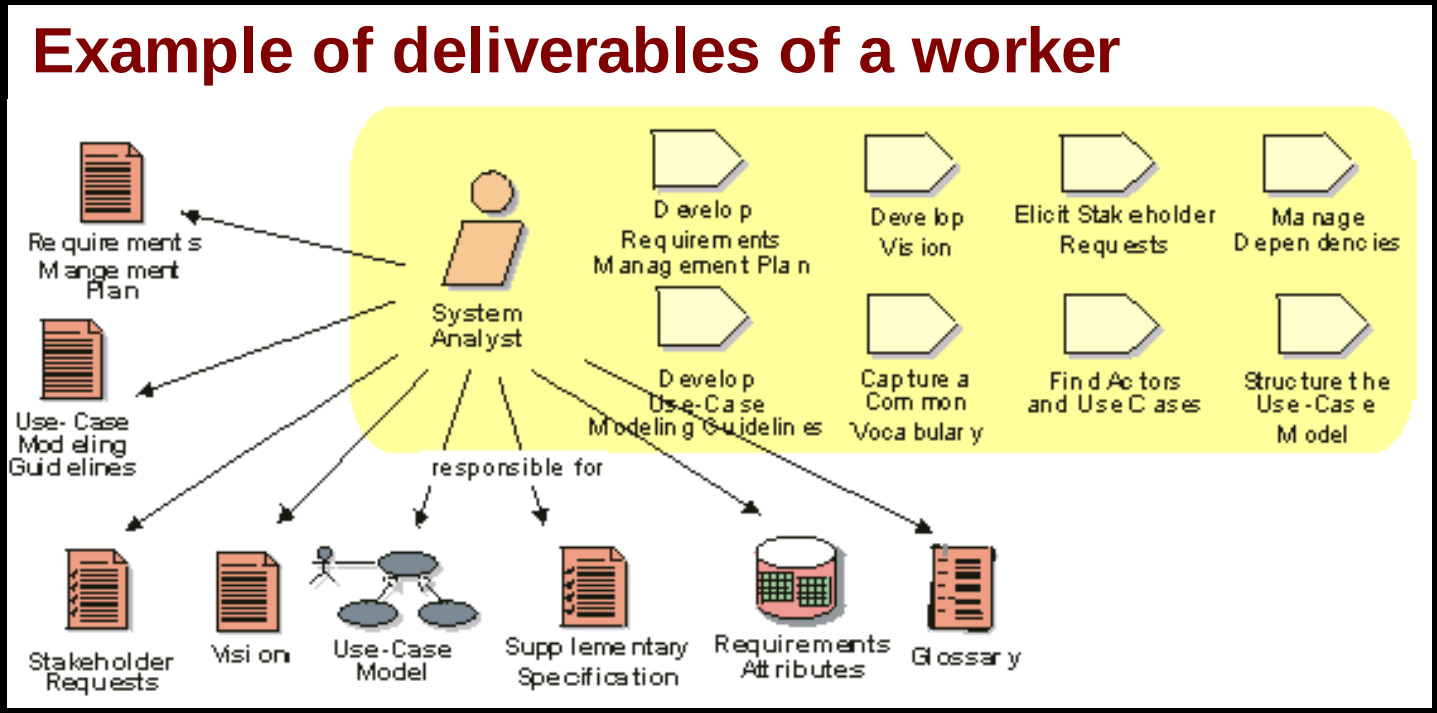
Artifacts

Situation of Today's Projects

- According to the implemented process, projects must
  - deliver certain artifacts (10 - ??)
  - use different tools to cope with activities
  - write code!
  - make sure that the code is maintained in a configuration management or revision control system
- Most projects struggle with keeping code and artifacts (specifications, models etc.) in sync!

Situation of Today's Projects

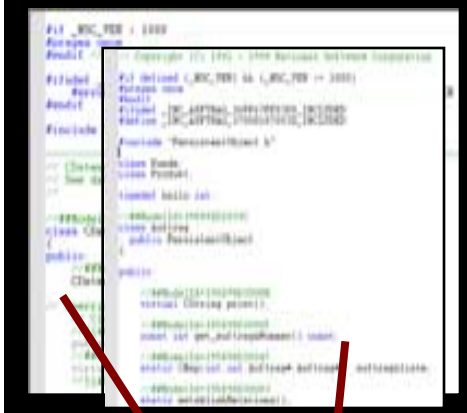
Example of deliverables of a worker

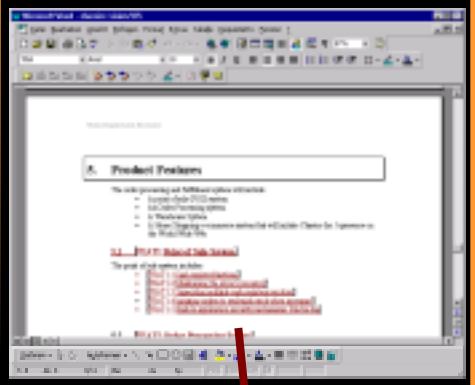


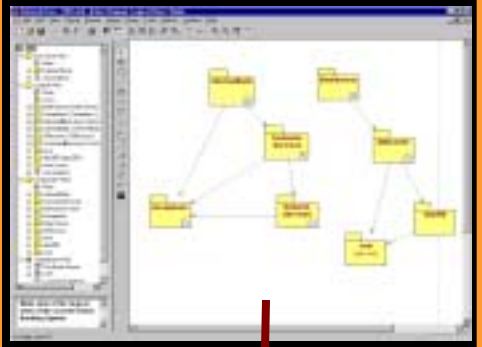
- Most projects struggle with keeping code and artifacts (specifications, models etc.) in sync!

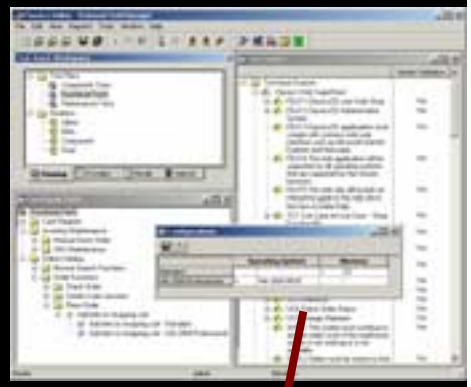
Why Dealing with Code AND Artifacts is so hard

Source	Requirements	Models (e.g. UML)	Test Assets









Rel 2.0	Rel 2.0	Rel 2.0	Rel 2.0
Source Database	**Requirements-Database**	**Model-Repository**	**Test-Repository**
Config.Mgmt. System	Oracle, SQL-Server, Repositories	Oracle, SQL-Server, Repositories	Oracle, SQL-Server, Repositories

XML WORLD EURO EDITION

Rational the e-development company

The Overall Problem

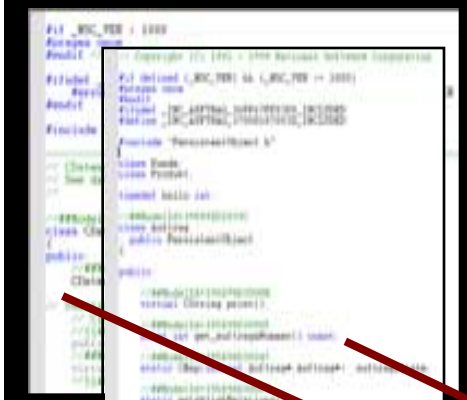
- Different places where software lifecycle artifacts are versioned
- Great source for inonsistencies and failures
- Large administration effort
- Possible solution: Use a single repository
  e.g. a configuration management system

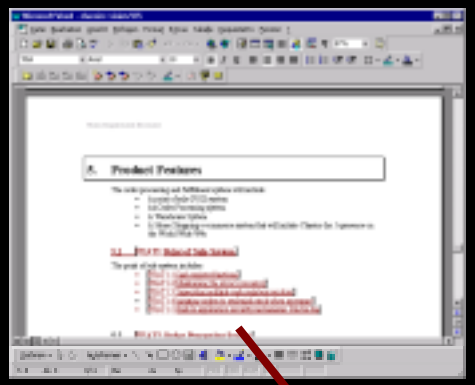
Typical Problems with Tools

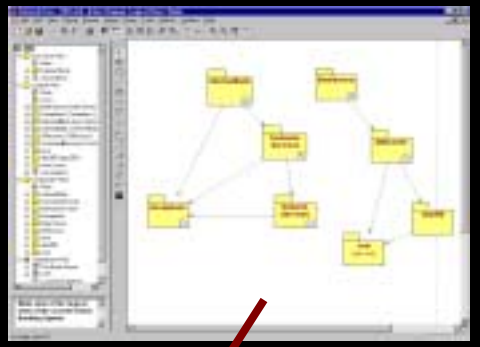
- **Artifacts of repository-based tools are hard to handle:**
  - **database / repository must not be put under revision control**
  - **data are not available in a standarized format**
  - **output only possible via tool API** (if one exists)
- **Other artifact problems cover**
  - **binary data formats (hard to diff / merge)**
  - **general unavailability of diff / merge tools**

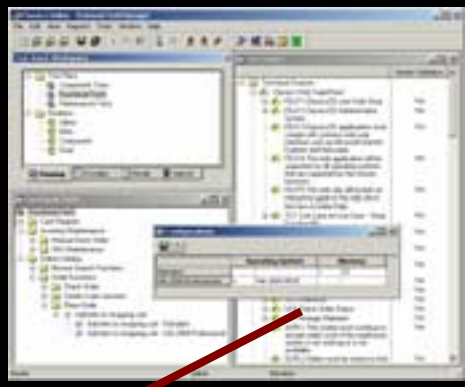
How to overcome this difficult situation?

Source	Requirements	Models (e.g. UML)	Test Assets









Best solution:

One repository for all assets of the software lifecycle

Rel 2.0

Configuration Management System

Best solution:

Use one standard format

X M L

Today's Challenges with this Solution

- Configuration Management Systems are build to handle file elements
- Other tools normally handle information on a different scale of granularity ("things" in files / repositories)
- To solve:
  - is granularity a matter of architecture or
  - is it a matter of performance or
  - is it both?

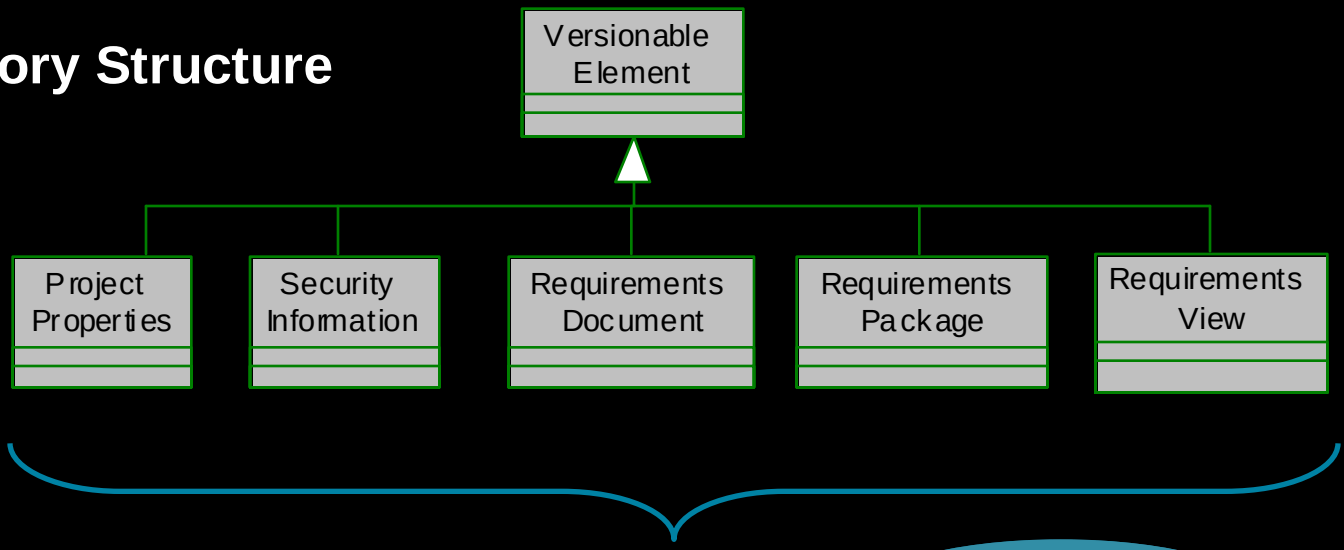
A First Step Towards a Unified Solution

- Make at least milestone lifecycle artifacts identifiable with software release and/or components
- Advantages:
  - Artifacts like requirements, test assets are associated with a distinct source-code base
  - Pre-condition for branching different variants of the same release
  - Quality assurance made practical
  - Can have change records

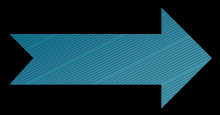
How to Break Down Elements?

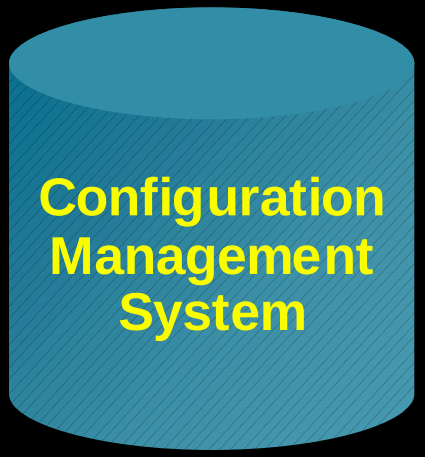
- **Example: Requirements Management Artifacts**

Repository Structure



Project.XML
Permissions.XML
DocumentList.XML
ViewList.XML
ReqFEAT.XML
etc.

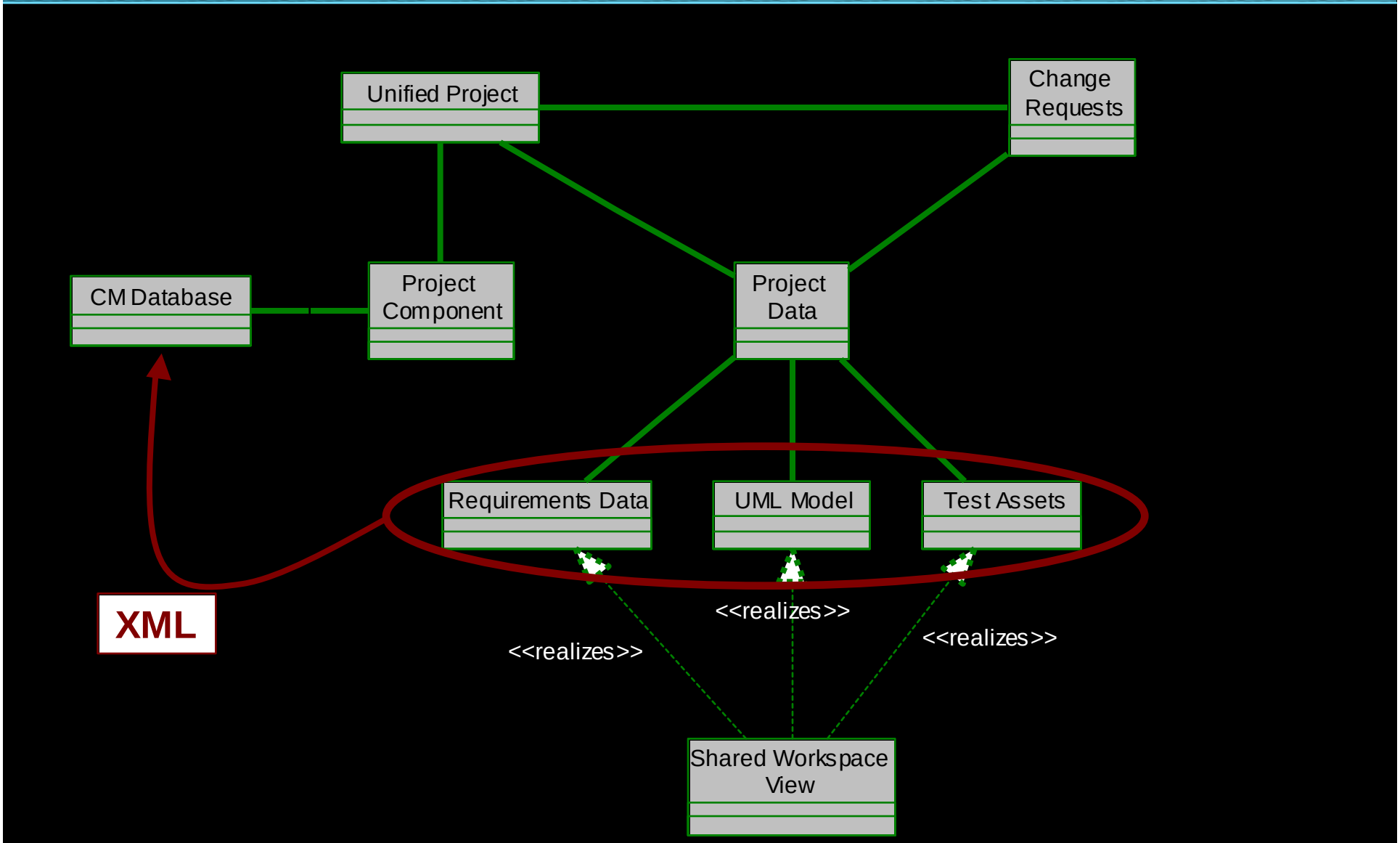




Obstacles for Requirements Management

- Granularity still not far enough:
  - All requirements of one type are in one XML-file
  - Would need one XML-file per requirement
- All XML-files must be placed under version control:
  - Large file traffic expected
  - Need for shared workspaces
  - How to achieve that everyone is working on the latest available data?

Project's Solution for Today



Summary (1)

- **Projects have a need for a common versioning of software lifecycle artifacts + source code**
- **Almost impossible today due to tool restrictions**
  (databases, repositories, source code DB)
- **We have a need for:**
  - **standard data format, like XML**
  - **common repository (e. g. config management system)**

Summary (2)

- Challenges to solve:
  - granularity of information (performance?!)
  - handling (workspaces, actual versions)
- Achievable advantages:
  - effective branching of variants
  - change records possible
- Today's solution:
  - archive and version milestone artifacts

Questions?

Rational®
the e-development company™

Holger Dörnemann (doernemann@rational.com)

Rational Software GmbH
Niederlassung Düsseldorf
Wanheimer Strasse 43
D-40472 Düsseldorf
+49 - 211 - 417928-16