Software Artefact Traceability Management and Recovery

Andrea De Lucia
University of Salerno

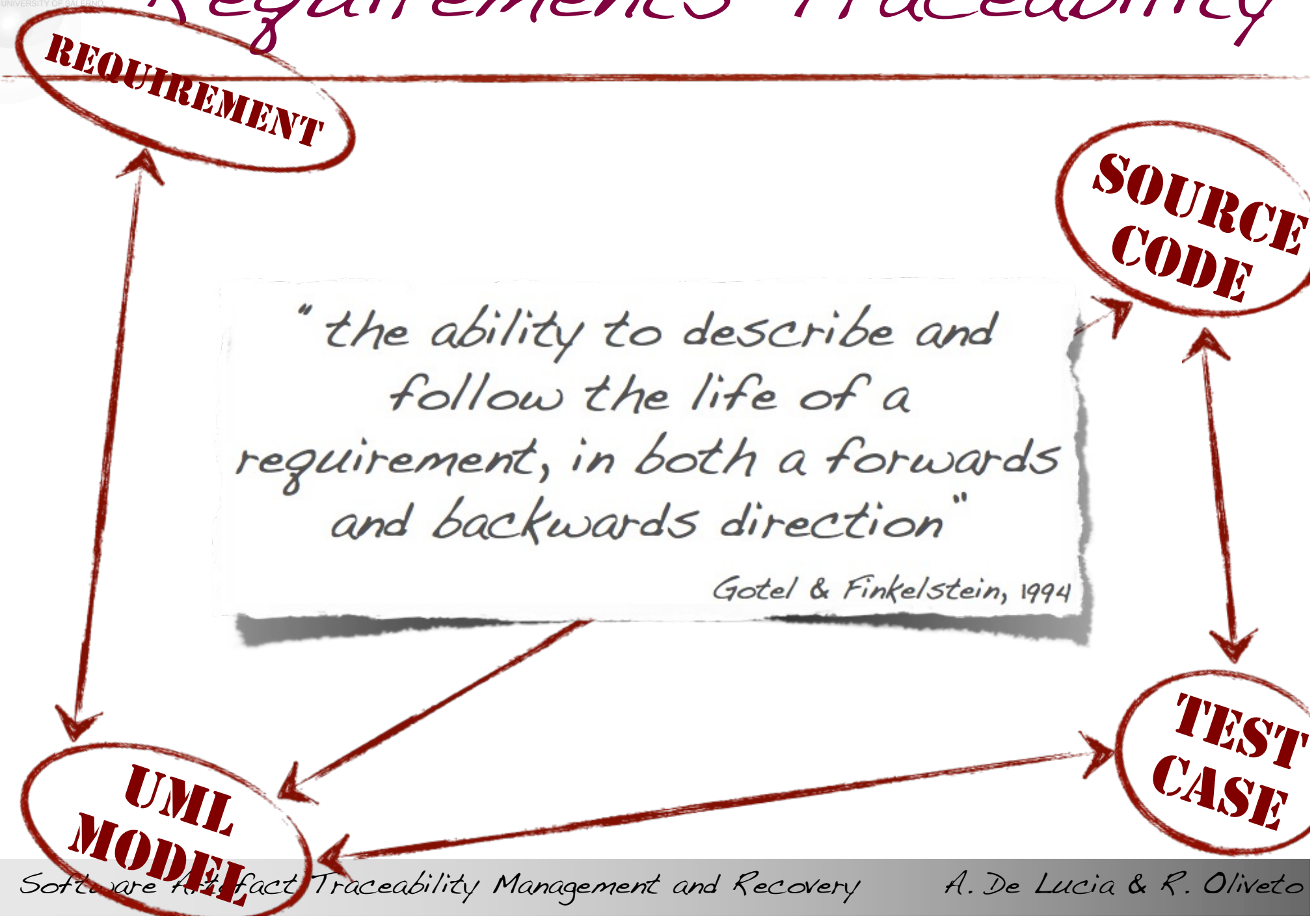
Rocco Oliveto
University of Molise

SE@SA Lab - Software Engineering Lab

Outline

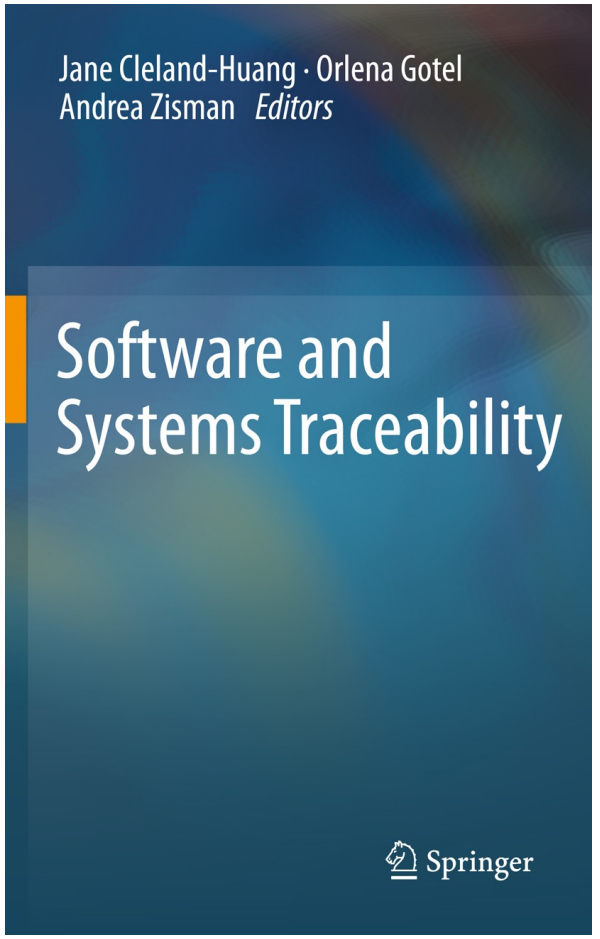
- ✓ Part A: Traceability Management
  - Basic Terminology for Traceability
  - Traceability Activities
    - Planning and Managing Traceability
    - Creating Traces
    - Maintaining Traces
    - Using Traces
  - Traceability Challenges
- ✓ Part B: IR-based Traceability Recovery
  - …

Requirements Traceability



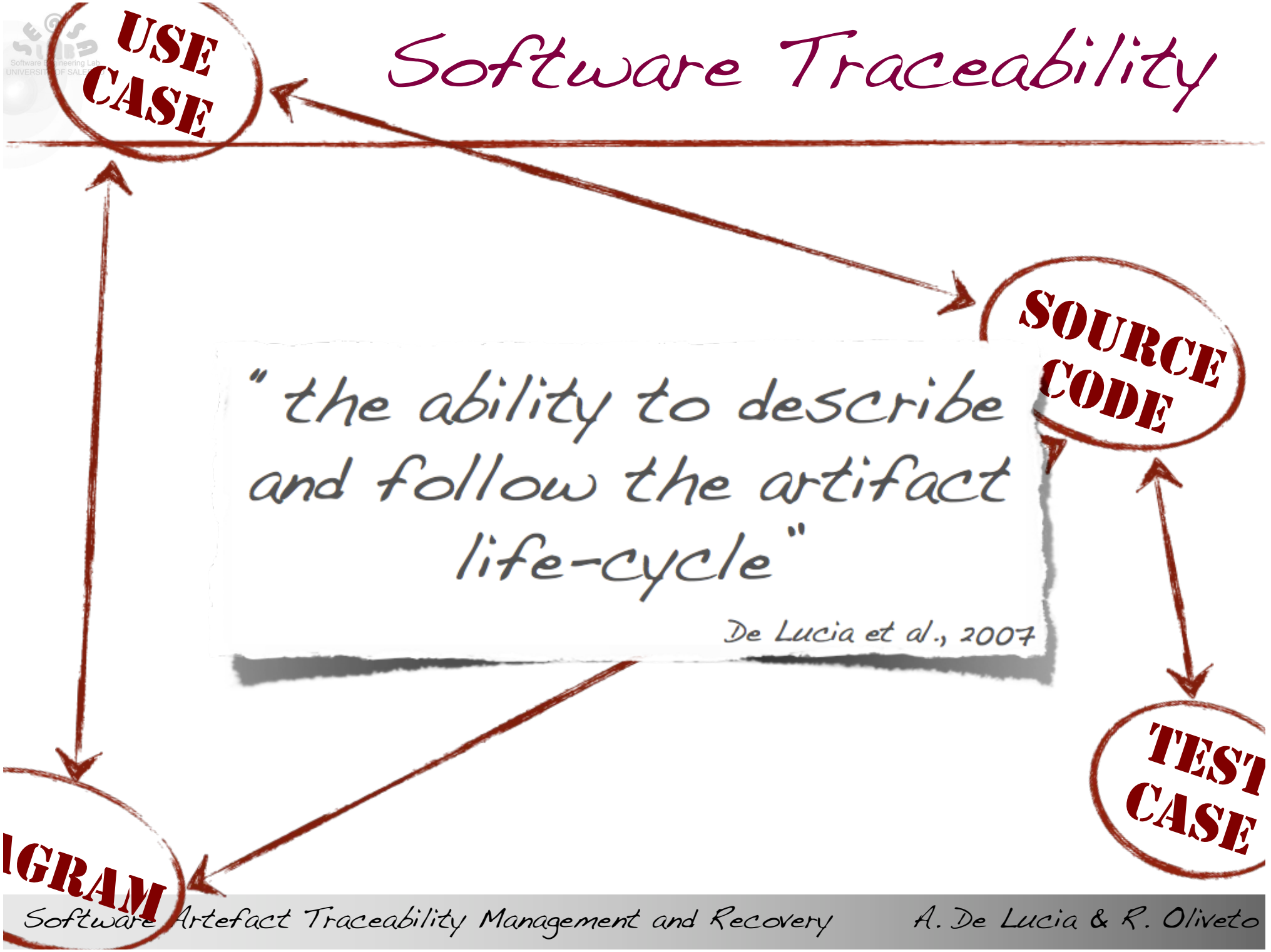
Traceability: beyond requirements

- ✓ Requirements Traceability
- ✓ Software Traceability
- ✓ System Traceability



ISBN: 978-1-4471-2238-8

Software Traceability



The Need for Traceability

You can't manage what you can't trace

Watkins & Neal, 1994

Traceability & Standards

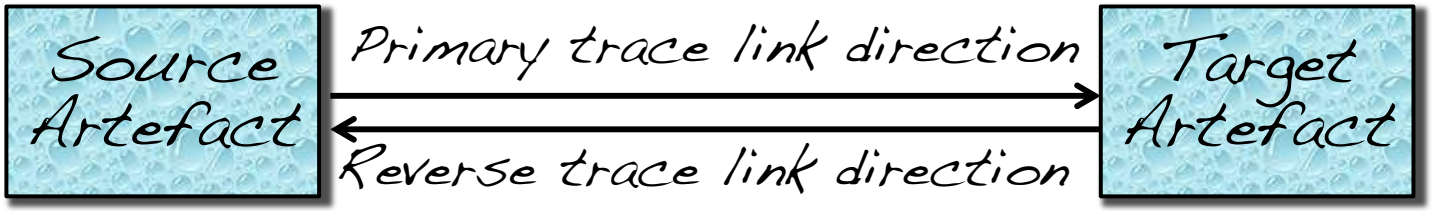
- ✓ Traceability discussed at the NATO Conference on 1968
- ✓ Traceability as a recommended or legally required activity (since the ’80)
  - IEEE (Std. 830-1998)
  - SEI CMMI
  - ISO-IEC & IEEE (Std 12207-2008)
  - US Department of Defense
  - U.S. Federal Aviation Administration
  - U.S. Food and Drug Administration

Research on Traceability

- ✓ Proliferating in the ’90
- ✓ Specific Events
  - Traceability in Emerging Forms of Software Engineering (TEFSE) workshop series
  - Workshops funded by NASA (2006) and NSF (2007)
- ✓ International Center of Excellence for Software Traceability (CoEST)
  - Established in 2005
  - Tracy project (funded by NSF)
  - TraceLab tool
  - Towards a Traceability Body of Knowledge (TBOK)

Basic Terminology

- ✓ Trace (noun) — Gotel et al., 2012a
  - A triplet of elements comprising
    - a source artifact,
    - a target artifact, and
    - a trace link associating the two artifacts



- ✓ Trace (verb)
  - the act of following a trace link from a source artifact to a target artifact (primary trace link direction) or vice-versa (reverse trace link direction)

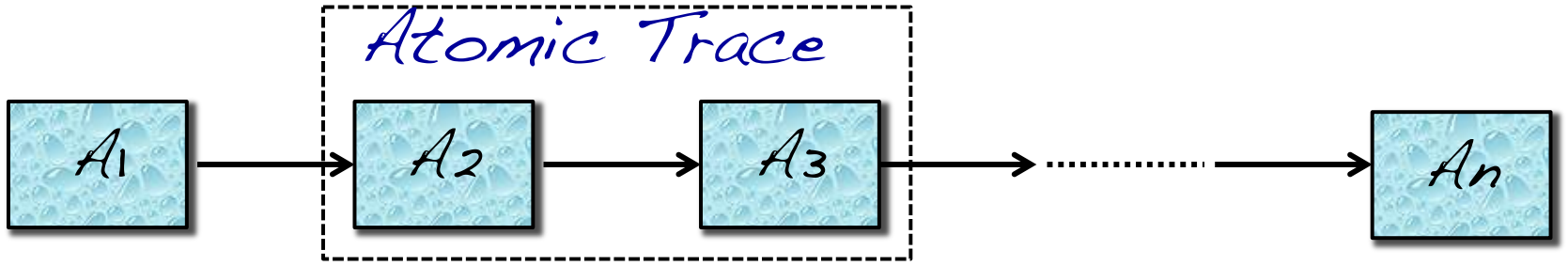
Trace Attribute and Trace Relation

- ✓ Trace Artefact
  - A traceable unit of data
    - Different artefact types and different granularity levels for a trace
    - Requirement, use case, test case, class diagram, class, operation, …
- ✓ Trace Attribute
  - Additional information (i.e., meta-data) that characterizes properties of the trace or of its individual trace elements
  - Attributes of a trace link:
    - Date, type, semantics, (primary/reverse) direction
- ✓ Trace relation
  - All the trace links created between two sets of trace artifact types
  - Commonly recorded in a traceability matrix

Chained Trace

A trace (noun sense) comprising multiple atomic traces, such that a target artifact for one atomic trace becomes the source artifact for the next atomic trace

Chained Trace



Traceability is … Trace "ability"

- ✓ The potential for traces to be established and used
  - An attribute of an artefact or of a collection of artefacts
- ✓ Where there is traceability, tracing can be undertaken and the specified artifacts should be traceable
  - Tracing: the activity of either establishing or using traces
  - Can be manual, automated, or semi-automated
    - Most of the approaches are semi-automated (see later)

Horizontal vs Vertical Traceability

- ✓ Horizontal traceability
  - The potential for horizontal tracing
  - Tracing artifacts at the same level of abstraction
  - Includes traces between versions of the same artifact
- ✓ Vertical traceability
  - The potential for vertical tracing
  - Tracing artifacts at different levels of abstraction (e.g., requirements-to-code)

Horizontal vs Vertical Traceability - Different definitions

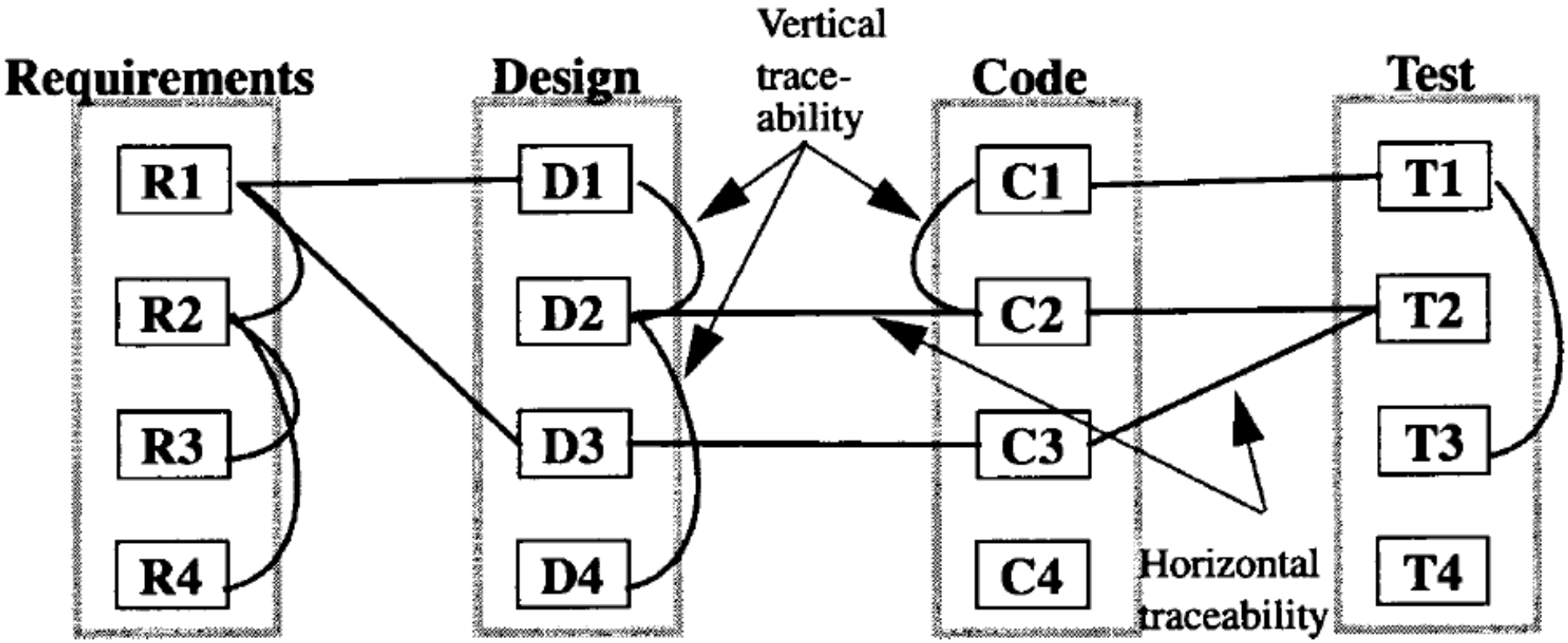


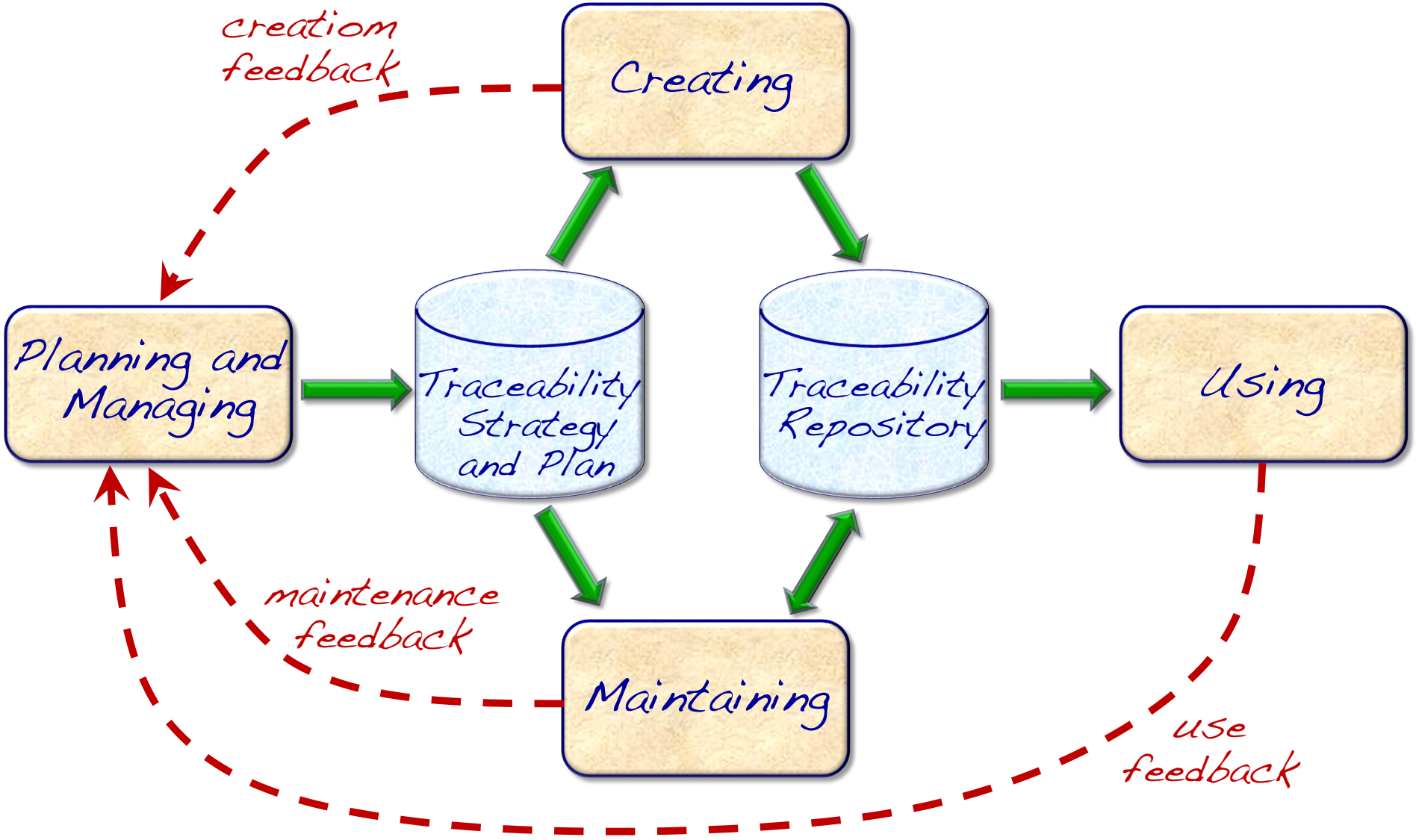
Figure taken from Lindvall and Sandahl (1998)

Original classification by Pfleeger and Bohner (1990)

Backward and Forward Traceability

- ✓ Forward traceability
  - The potential for forward tracing
  - e.g., in vertical traceability tracing higher-level artifacts onto lower-level artifacts
- ✓ Backward traceability
  - The potential for backward tracing
  - e.g., in vertical traceability tracing lower-level artifacts onto higher-level artifacts
- ✓ Exploiting the primary or reverse direction of a trace link

Traceability Activities



Traceability Strategy

- ✓ Effective traceability is not by chance
  - Determining stakeholders and system requirements for traceability
  - Designing a suitable traceability solution
    - Traceability Information (Reference) Model
    - Traceability Process
  - Providing the control to keep requirements and solutions relevant and effective during the life of a project

Traceability Information Model

- ✓ An abstract expression of the intended traceability for a project
- ✓ A graph defining trace granularity
  - Trace artifact types
  - Trace link types
  - Trace relationships on a project

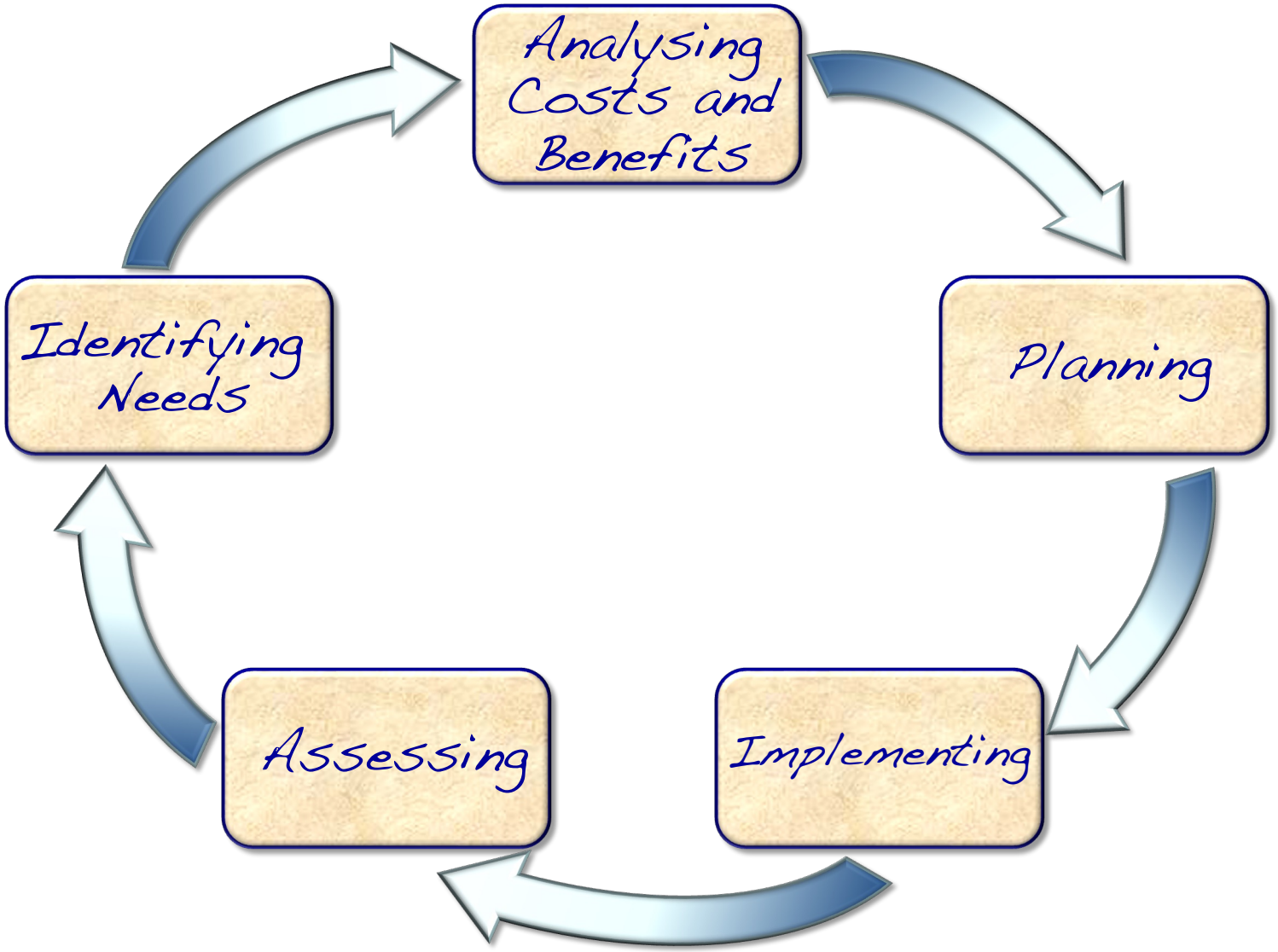
Ramesh and Jarke, 2001
Lindvall and Sandahl, 1996

- ✓ Addressing anticipated:
  - Traceability queries
  - Traceability-enabled activities and tasks

Traceability Information Model

- ✓ Additional information
  - Cardinality of the trace artifacts associated to a trace link
  - Primary trace link direction
  - Purpose of a trace link (link semantics)
  - Location of the trace artifacts
  - Tracer responsible for creating and maintaining the trace link

Planning and Managing Traceability



Cost-Benefit Analysis

- ✓ Cost is a key reason why many projects neglect or abandon traceability efforts
- ✓ Capturing, storing, and maintaining all possible traces is prohibitively time-consuming
  - Collecting only needed traces
  - Identifying available money, people, infrastructure, and tools
  - Ranking requirements for selective traceability based on
    - Predicted volatility
    - Predicted risk
    - Required reliability
    - Importance for stakeholders

Ingram and Riddle, 2012

Estimating Costs for Traceability

- ✓ Only few studies about this …
  - Some focusing on the time required to create trace links
    - Cleland-Huang et al., 2004
  - … or on different levels of granularity
    - Egyed et al., 2007
  - A more general study states that tracing effort absorbs 5% of the total project costs, as part of quality assurance activities
    - Heindl and Biffl, 2005
- ✓ Factors affecting costs
  - System size
  - Granularity of traces
  - Volatility of requirements
  - Project duration
  - Availability of automated tools
  - …

Traceability Management Tools

- ✓ Dedicated Requirements Management Tools
  - CaliberRM, DOORS, IBM's RequisitePro, …
- ✓ Lifecycle Tools
  - UML tools, bug/issue/project tracking tools, IDEs, …
- ✓ General-purpose Tools and Proprietary Development
  - text editors, graphic editors, spreadsheet tools, databases, and wikis …

Gotel and Mäder, 2012

ADAMS

- ✓ Developed using J2EE technologies, AJAX, and MySQL (first release in 2003)
- ✓ Support for
  - Project Management
  - Quality Management
  - Artifact Management
  - Traceability Management
  - Collaborative Development

De Lucia et al. 2010

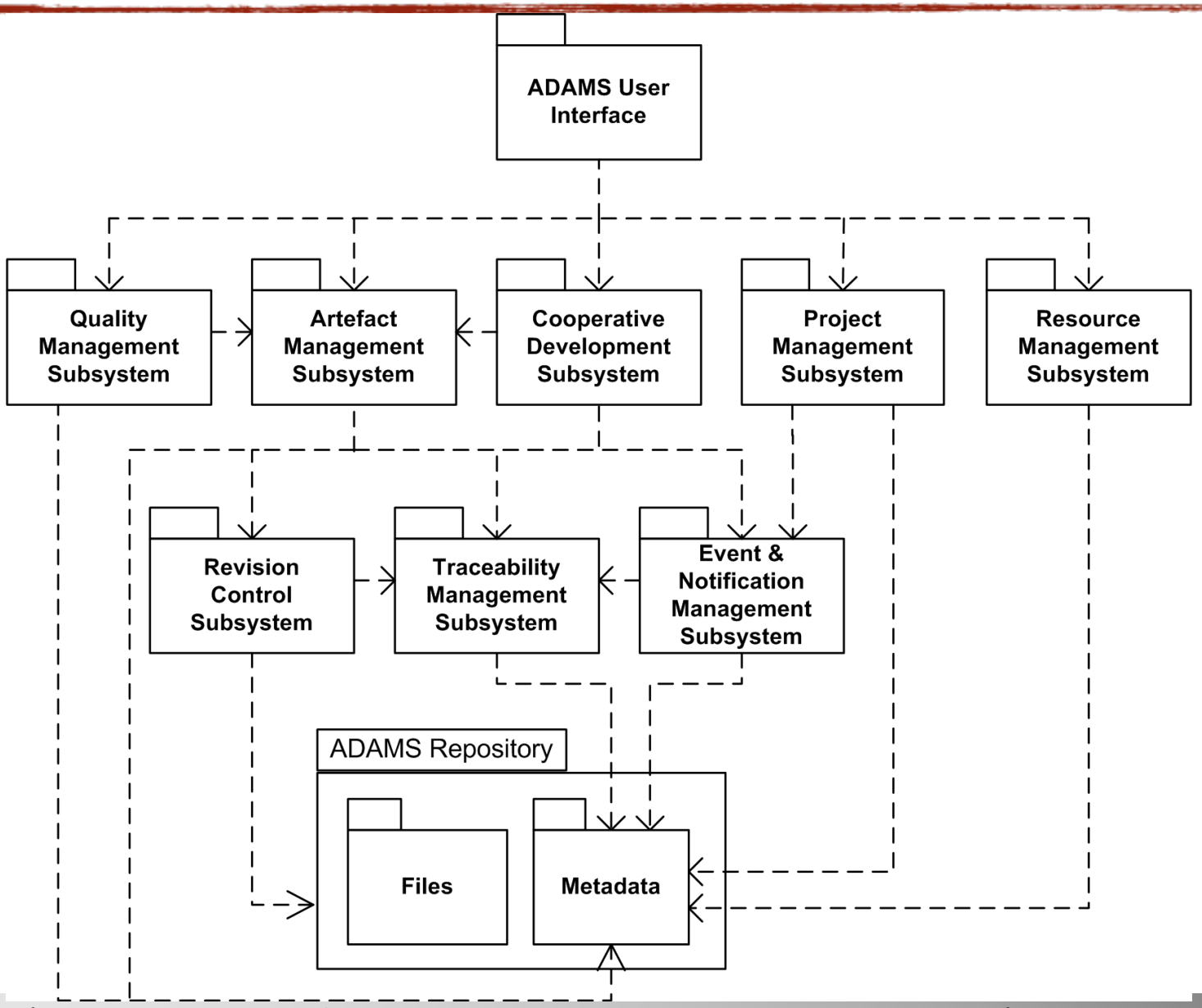
Fine-grained Artefact Management

- ✓ Software artefacts have a well-defined structure
  - Documents
  - UML Diagrams
  - Source code
- ✓ Fine-grained Versioning Management enables structured (composite) artefact versioning
  - … product-oriented WBS

Fine-grained Artefact Management

- ✓ Advantages
  - Reuse
  - Consistency
  - Traceability management
  - Concurrent development
  - Conflict avoidance and reduction of branches
  - Accurate responsibility definition ad management
  - Context Awareness

ADAMS Architecture



Project Management

AD.A.M.S.
ADvanced Artefact Management System

June 8th, 2012

User: Andrea De Lucia

- Logout
- General
  - ToDo List
    - All Projects
    - My Projects
    - My Artefacts
    - My Inspections
- Manual

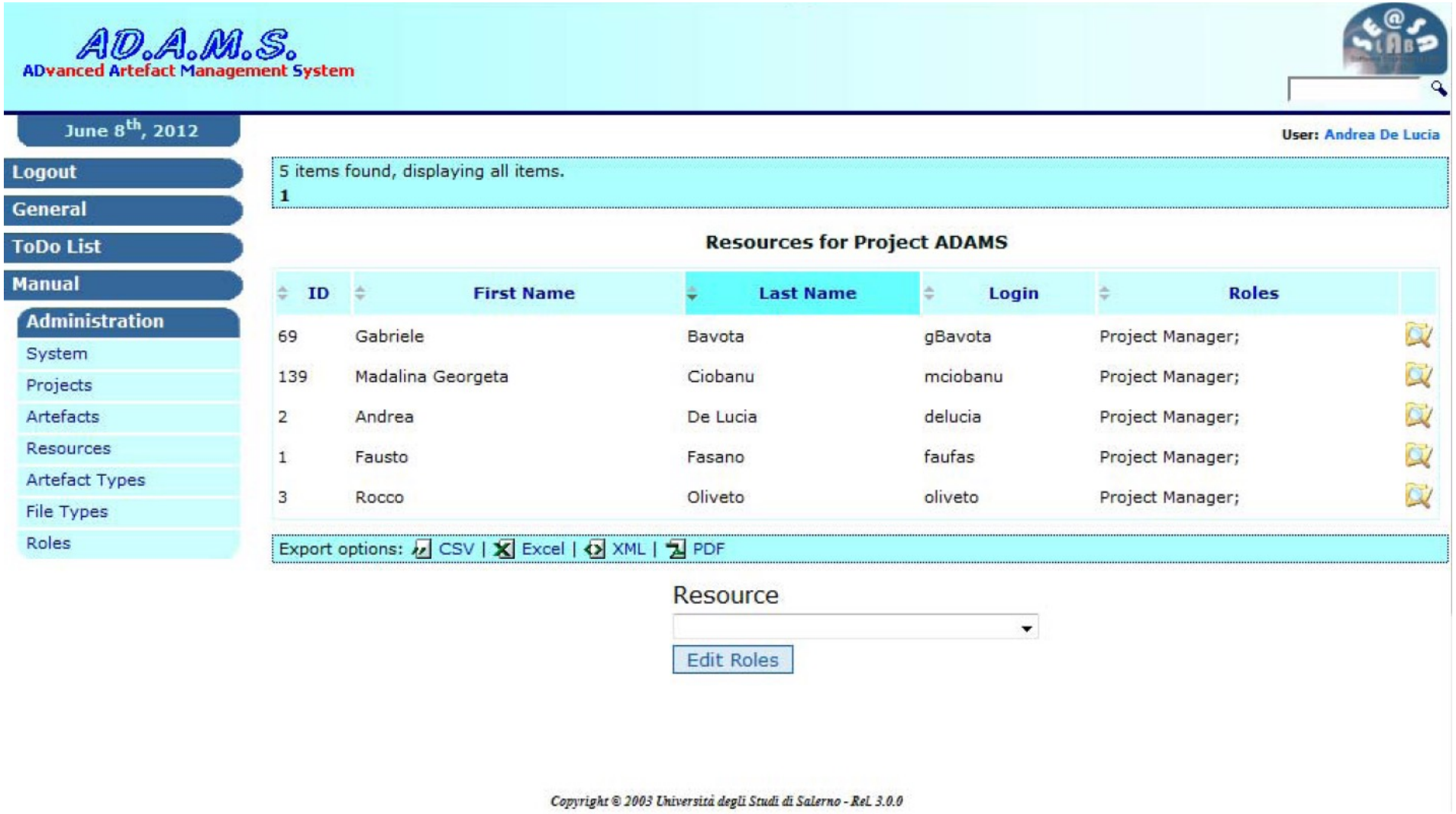
29 items found, displaying 1 to 10.
[First/Prev] **1**, 2, 3 [Next/Last]

List of Projects assigned to Andrea De Lucia

Name	Description	Creation Date	Start Date	End Date	Role	Status
C Analyzer - evolution	Evoluzione di un analizzatore C	2012-03-28	2012-03-28	2012-07-27	Project Manager	ACTIVE
Caras	Sistema di gestione polizze assicurative automobilistiche	2011-10-20	2011-10-20	2012-02-29	Project Manager	ACTIVE
Centro Sportivo Polifunzionale - Migrazione	Migrazione di un software per la gestione di un centro sportivo	2012-03-16	2012-03-16	2012-07-27	Project Manager	ACTIVE
ElectroBiblion	Gestione di una biblioteca	2011-10-13	2011-10-20	2012-02-29	Project Manager	ACTIVE
Emme3	Sistema di supporto alla didattica universitaria	2011-10-20	2011-10-20	2012-02-29	Project Manager	ACTIVE
Exemplar Projects	Esempi di progetti con buon livello di documentazione	2011-11-18	2011-11-18	2012-09-28	Project Manager	ACTIVE
Farmatek - Migrazione	Migrazione di un software per la gestione di una farmacia	2012-03-16	2012-03-16	2012-07-27	Project Manager	ACTIVE
IMIS - Migrazione	Migrazione di un Sistema Informativo per la gestione di un Master	2012-04-20	2012-04-20	2012-07-27	Project Manager	ACTIVE
iTunz	Store on-line per l'acquisto di materiale multimediale	2011-10-20	2011-10-20	2012-02-29	Project Manager	ACTIVE
MySchool	Gestione della comunicazione scuola-famiglia	2011-10-13	2011-10-20	2012-02-29	Project Manager	ACTIVE

Workable Late Overdue

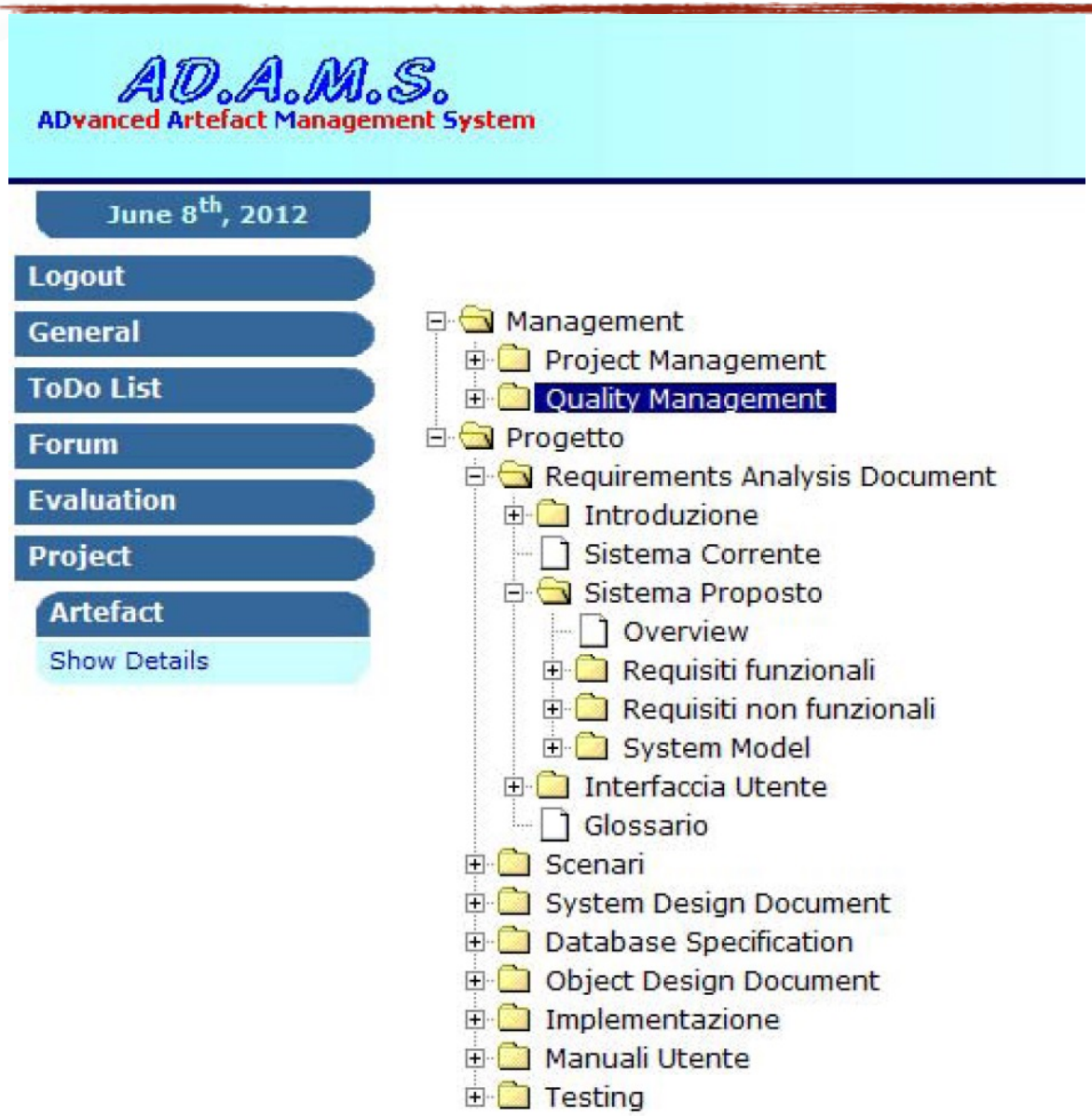
Resource Allocation



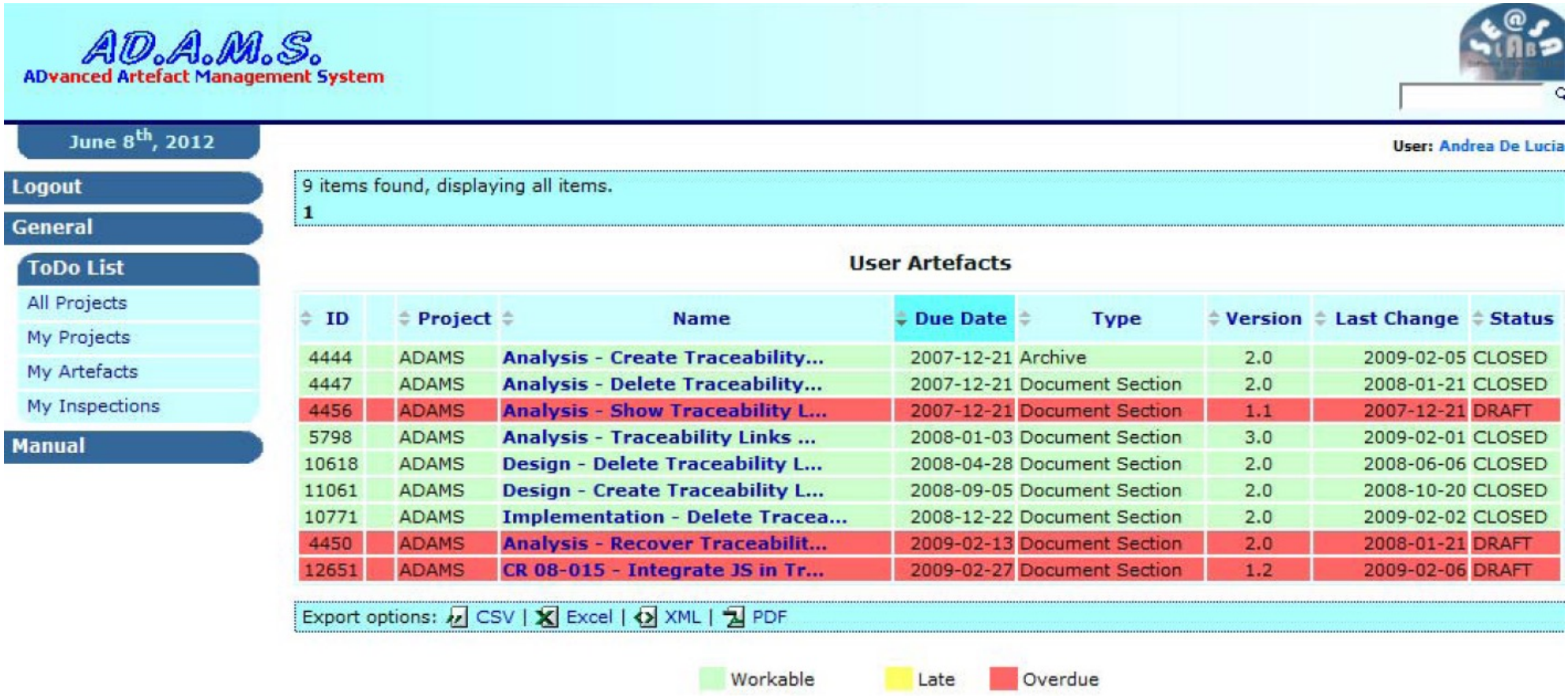
Work Breakdown Structure

- ✓ Product-Oriented WBS
  - Hierarchical decomposition of artefacts
  - Defining deadlines and allocating resources to (sub-)artefacts
  - Hierarchical (fine-grained) versioning …
  - Automatic re-composition …
- ✓ Artefact life-cycle
  - Artefact draft versions (promotions)
  - Artefacts reviewed and baselined

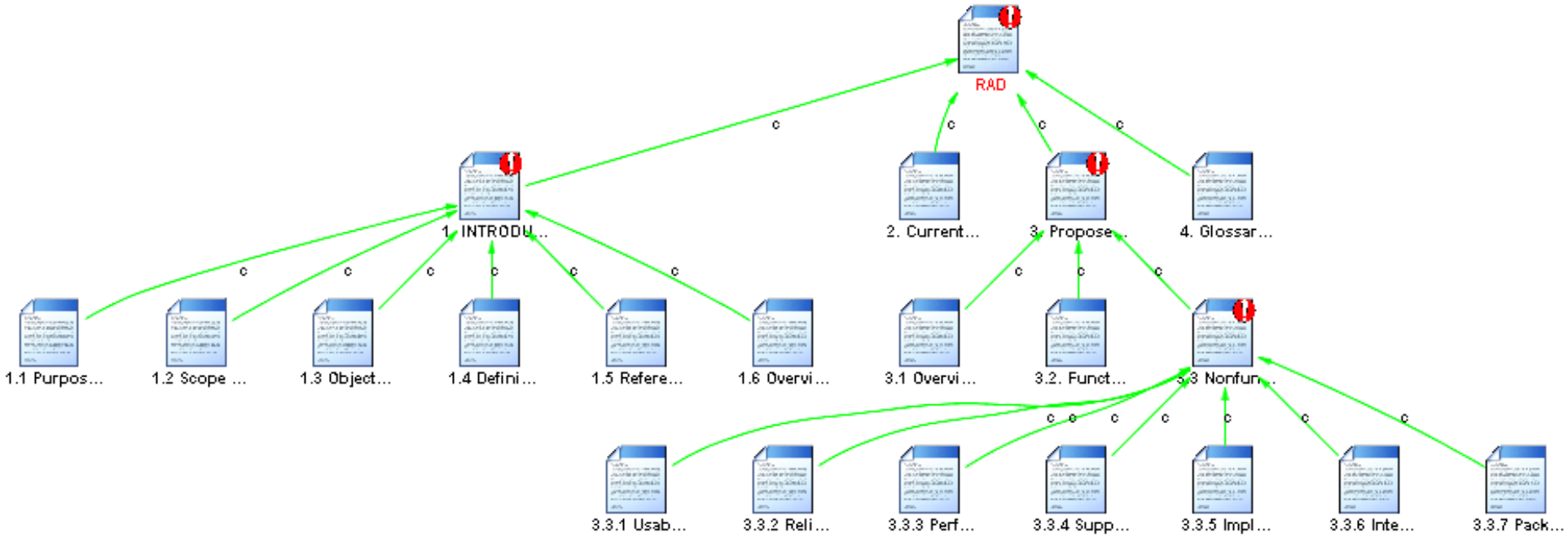
Hierarchical Artefact View



Flat Artefact View



Graph-based Artefact View



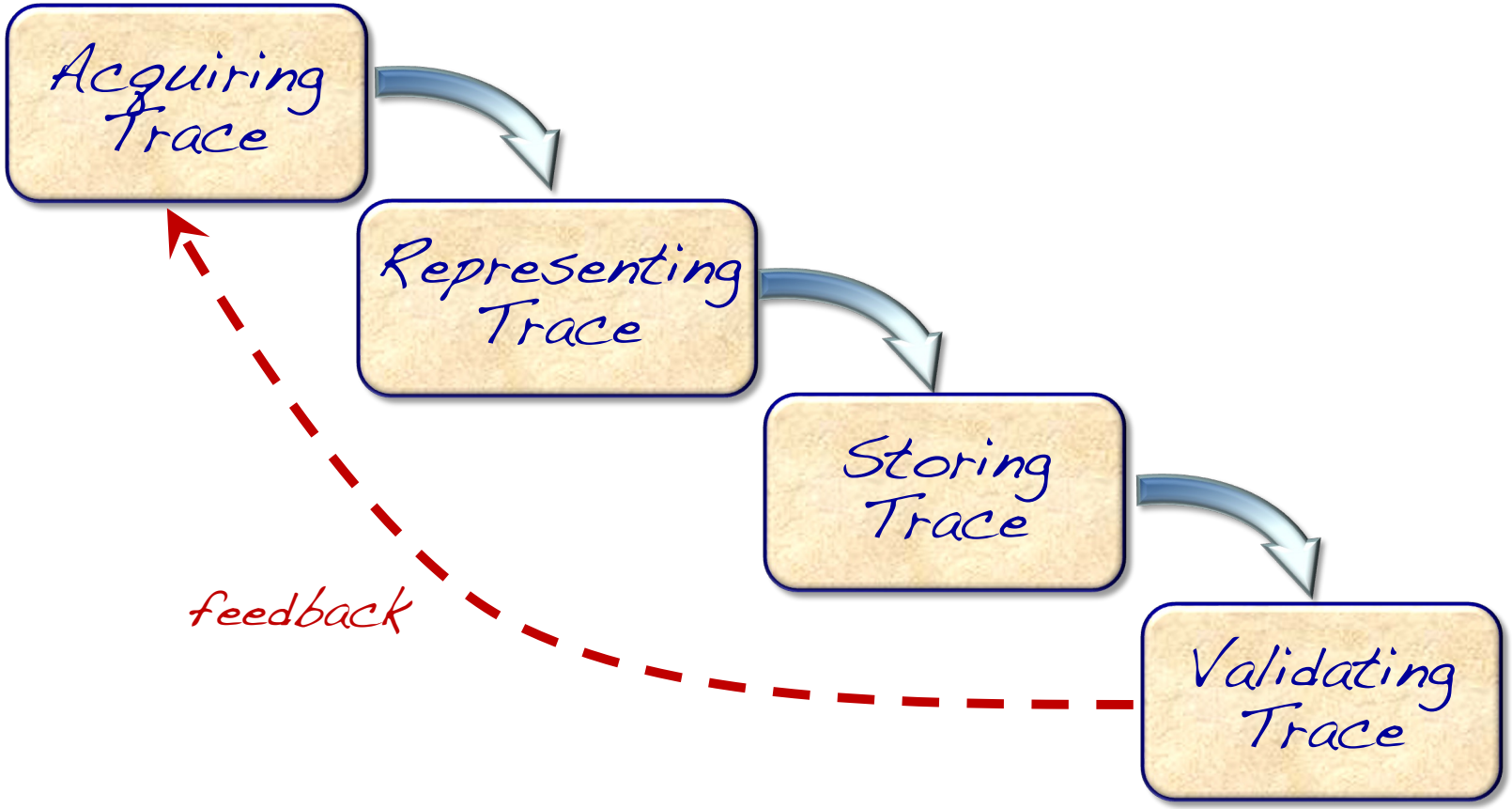
Traceability in ADAMS

- ✓ Artefacts and links have a type
- ✓ Tracing all types of artefacts
  - According to the Traceability Information Model
  - See later artefact creation in ADAMS …
- ✓ Trace links are versioned
- ✓ Visualizing the links (see later)
  - Flat format
  - Graph-based view
- ✓ Event-Based Traceability (see later)

Assessing Traceability

- ✓ Traceability quality: a measurable property of the overall traceability at a particular point in time on a project
  - Correctness
  - Accuracy
  - Precision
  - Completeness
  - Consistency
  - Timeliness
  - Usefulness
  - …

Creating Traces



Capture vs Recovery

- ✓ Trace Capture
  - Creating links concurrently with the creation of artifacts
  - Typically during development
- ✓ Trace Recovery
  - Creating links after the artifacts have been created and manipulated
  - Typically during maintenance and evolution
  - Lost or out-of-date links

Structural vs Knowledge based

De Lucia et al. 2008

- ✓ Structural Traceability
  - Artefacts expressed in a formal language
    - Source code and models expressed in XML or UML
  - Traces derived by formal analysis techniques (e.g. source code parsing) and rules
- ✓ Knowledge Based Traceability
  - Informal information extracted from artefacts, processes, and repositories
  - Traces derived by heuristics + knowledge of the application/solution domain
  - Requires more human interaction for validation

Supporting Trace Creation (1/3)

- ✓ Event Based — Cleland-Huang et al. 2003
  - Applicability: (almost) any software artefact
  - Defining event types (e.g. changes)
  - Tracking events and suggesting trace links
- ✓ Rule Based — Zisman, 2012
  - Applicability: source code and higher level artifacts expresses in a structured form (e.g., UML, XML)
  - Includes static code analysis
  - Often combined with event based

Supporting Trace Creation (2/3)

- ✓ Dynamic analysis — Egyed and Grunbacher, 2002
  - Requires the availability of scenarios (test cases)
  - Applicability: executable components
- ✓ Mining version history — Gall et al., 1998
  - Analysis of co-changes, logical coupling
  - Useful in traceability recovery
  - Applicability: versioned artefacts (mainly source code)

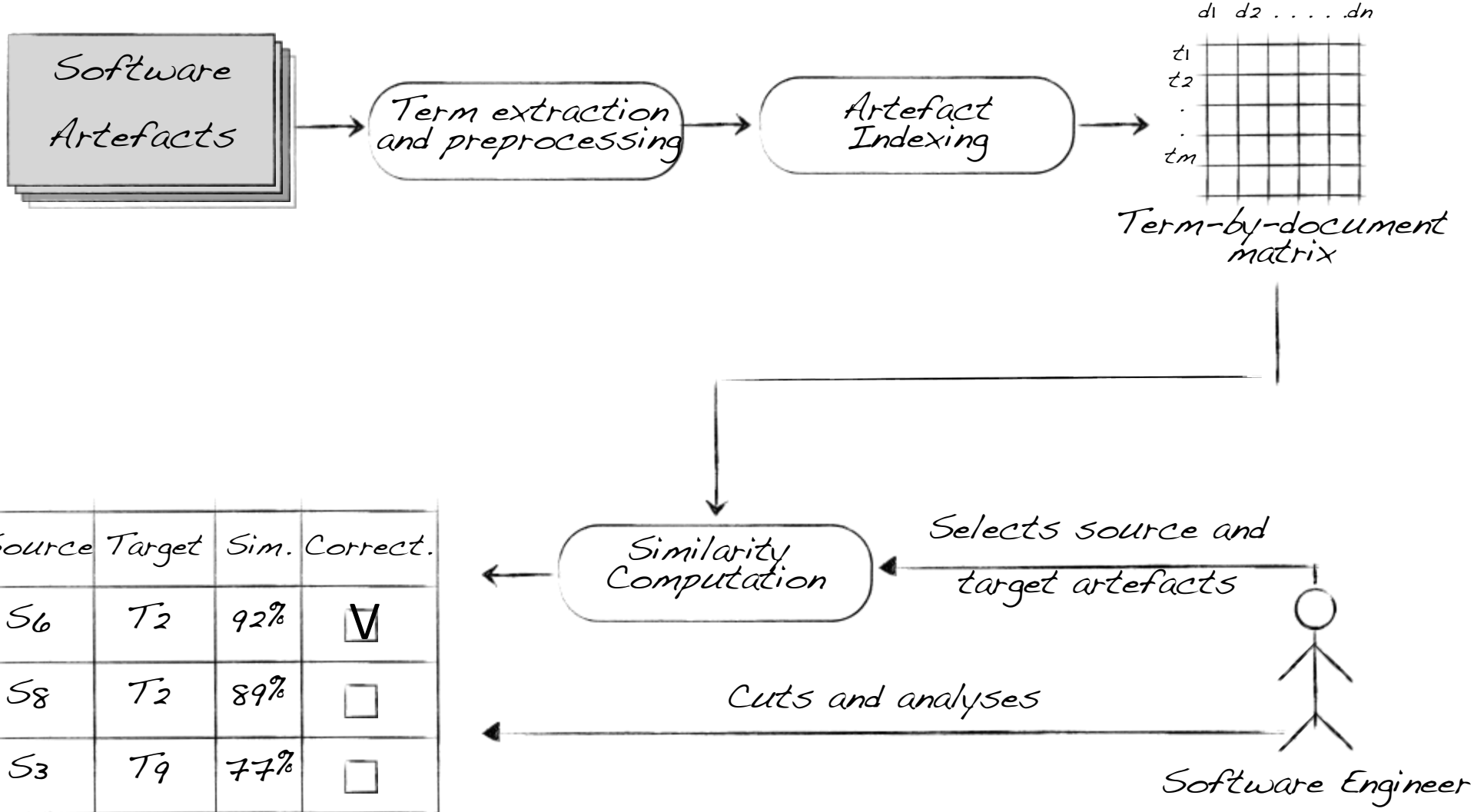
Supporting Trace Creation (3/3)

- ✓ Text analysis
  - Applicability: (almost) any artifact type
  - Rationale: Software artifacts contain text
  - Goal: tracing artifacts with high similarity
  - Name tracing: regular expressions exploiting naming conventions, string edit distance *Antoniol et al., 2000*
  - Text Retrieval or Information Retrieval …

    De Lucia et al., 2012

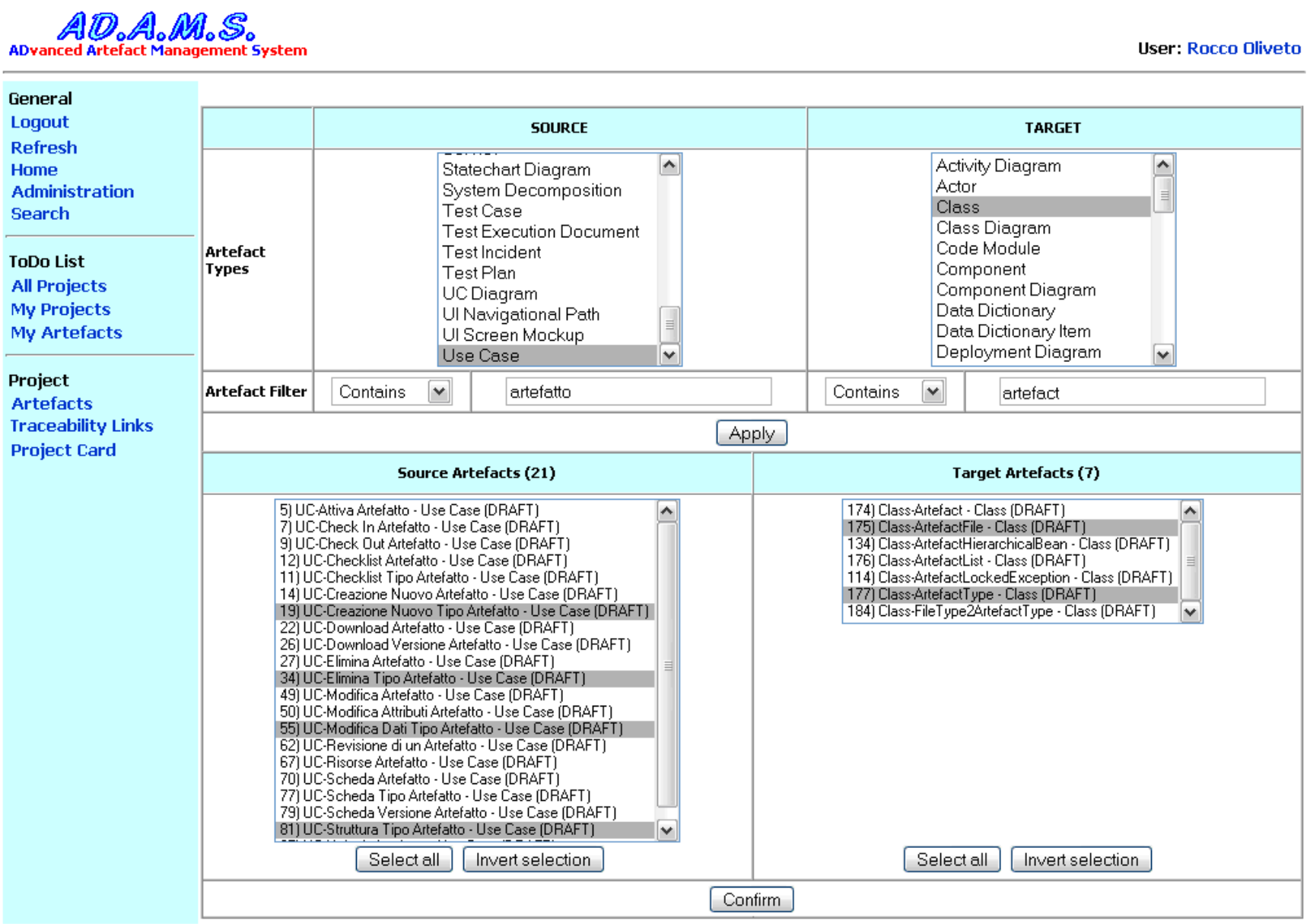
- ✓ Combining different approaches …

IR-based traceability



Source	Target	Sim.	Correct.
S6	T2	92%	☑
S8	T2	89%	☐
S3	T9	77%	☐
...	...	...	...

Traceability Creation in ADAMS



Manual Traceability

AD.A.M.S.
ADvanced Artefact Management System

User: Rocco Oliveto

General
Logout
Refresh
Home
Administration
Search

ToDo List
All Projects
My Projects
My Artefacts

Project
Artefacts
Project Card

Traceability Links
Confirm actions

Untraced Links (6 links)

ID	Source Artefact	ID	Target Artefact	Trace Link
19	UC-Creazione Nuovo Tipo Artefatto (Use Case)	177	Class-ArtefactType (Class)	☑
34	UC-Elimina Tipo Artefatto (Use Case)	175	Class-ArtefactFile (Class)	☐
55	UC-Modifica Dati Tipo Artefatto (Use Case)	175	Class-ArtefactFile (Class)	☐
55	UC-Modifica Dati Tipo Artefatto (Use Case)	177	Class-ArtefactType (Class)	☑
81	UC-Struttura Tipo Artefatto (Use Case)	175	Class-ArtefactFile (Class)	☑
81	UC-Struttura Tipo Artefatto (Use Case)	177	Class-ArtefactType (Class)	☑

Traced Links (2 links)

ID	Source Artefact	ID	Target Artefact	Remove Link
19	UC-Creazione Nuovo Tipo Artefatto (Use Case)	175	Class-ArtefactFile (Class)	☐
34	UC-Elimina Tipo Artefatto (Use Case)	177	Class-ArtefactType (Class)	☐

Classified as False Positives (0 links)

ID	Source Artefact	ID	Target Artefact	Trace Link
EMPTY				

IR-based Traceability

AD.A.M.S.
ADvanced Artefact Management System

User: Fausto Fasano

General
Logout
Refresh
Home
Administration
Search

ToDo List
All Projects
My Projects
My Artefacts

Project
Artefacts
Traceability Links
Project Card

Link analysis
New analysis

Recovering Functional Requirement onto Use case traceability links

Threshold: < 90 % > 

Previous iteration statistics:

Threshold	95%
Suggested links	3
Traced Links	2
False positives	1

Suggested Links

ID	Source Artefact	ID	Target Artefact	Similarity measure	Action: Trace link	Action: Classify as False Positive
169	Doctor management (Requirement)	212	Delete doctor (Use Case)	94.71	☐	☐
168	Medicine management (Requirement)	197	Modify medicine (Use Case)	94.27	☐	☐
167	Patient management (Requirement)	209	Show patient (Use Case)	92.66	☐	☐
168	Medicine management (Requirement)	199	Show medicine (Use Case)	92.66	☐	☐
169	Doctor management (Requirement)	213	Show doctors (Use Case)	91.73	☐	☐
169	Doctor management (Requirement)	207	Modify patient(Use Case)	90.62	☐	☐

False Positives

ID	Source Artefact	ID	Target Artefact	Similarity measure	Action: Trace link
167	Patient management (Requirement)	211	Modify doctor (Use Case)	95.41	☐

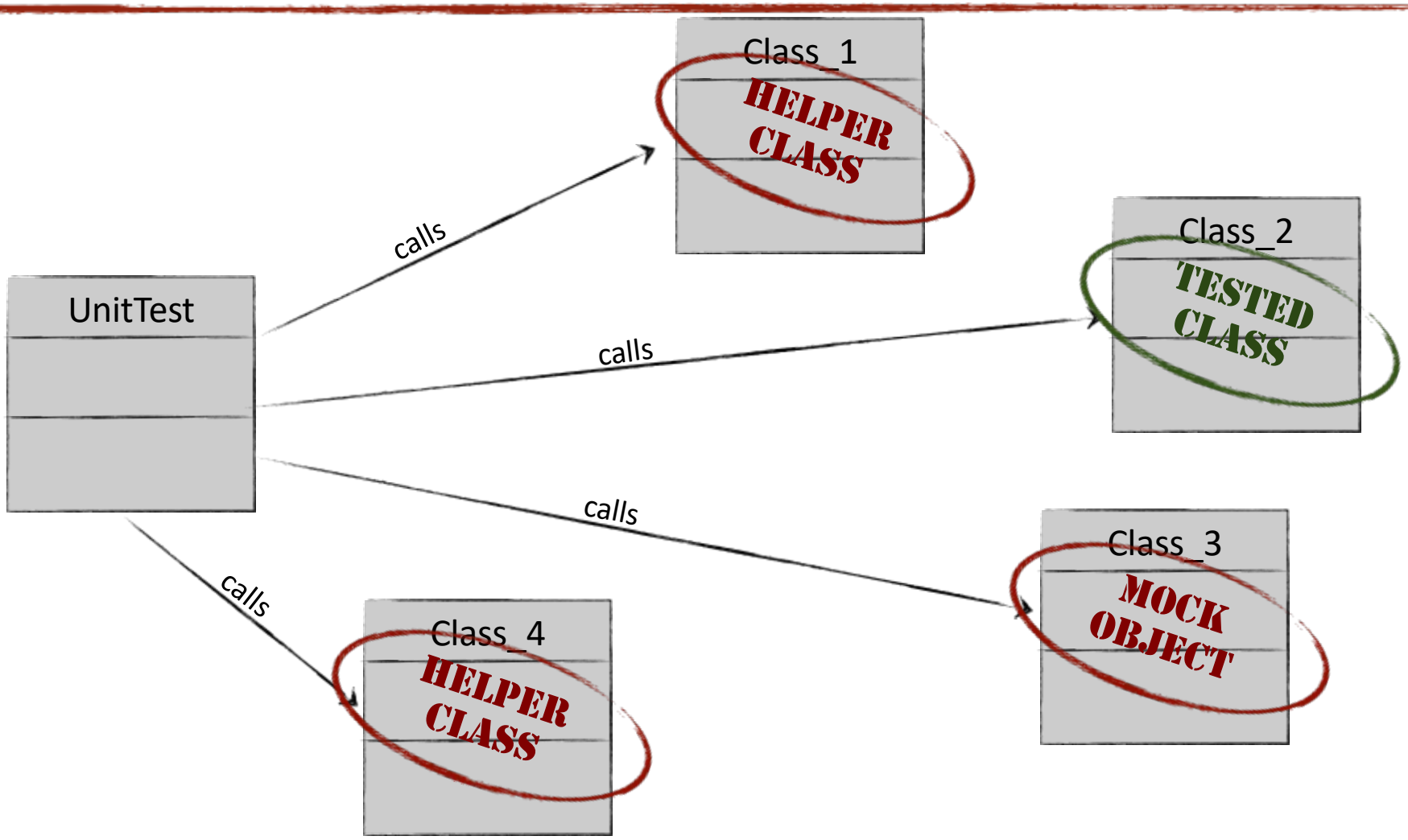
Submit actions

Test-to-code traceability creation

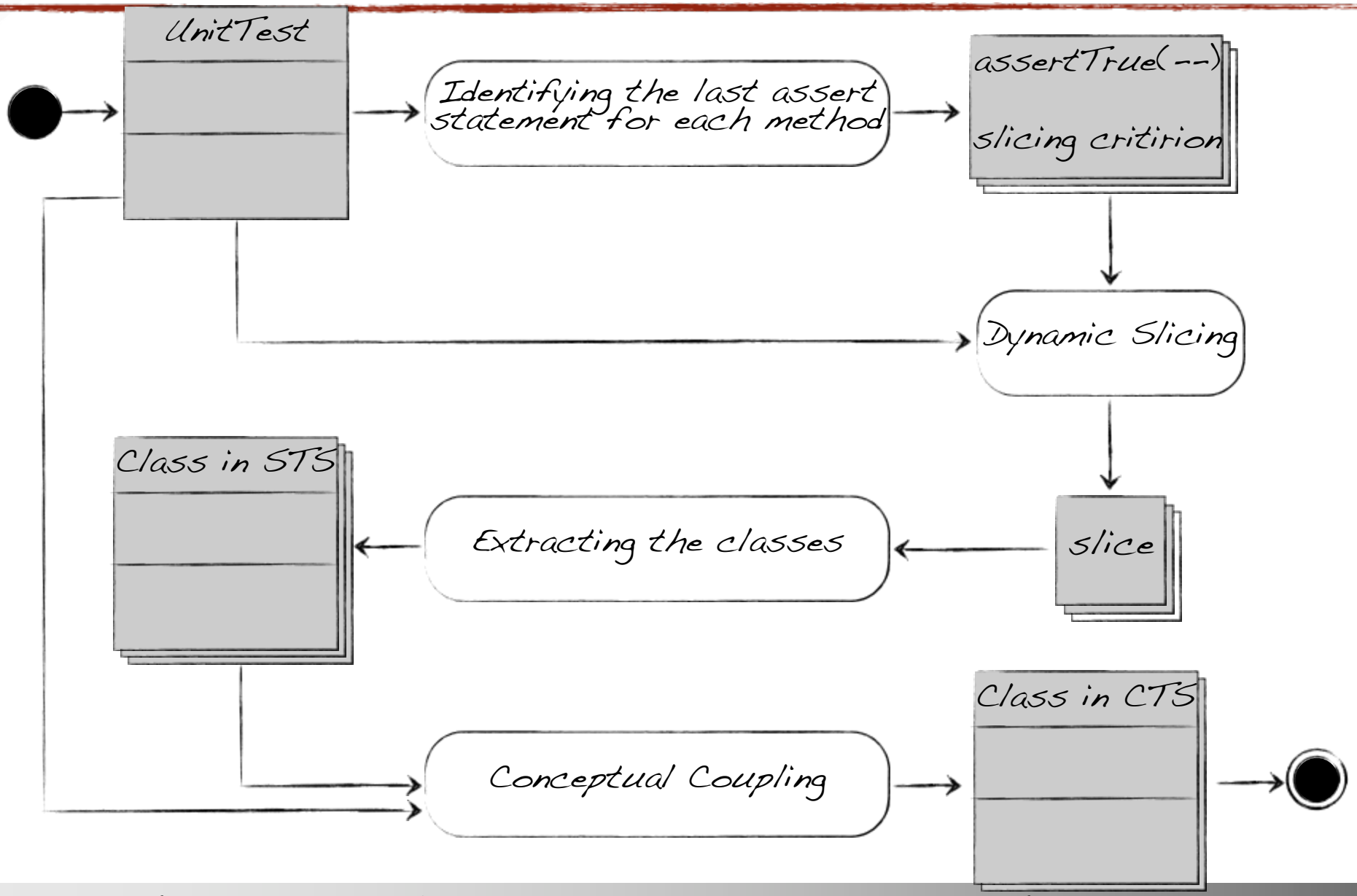
- ✓ Creating traceability between Junit tests and tested classes
- ✓ Heuristics and approaches
  - Naming Conventions
  - Last call before assert
  - Data Flow Analysis
  - SCOTCH (Slicing and COupling based Test to Code traceability Hunter)

Qusef et al., 2011

Test-to-code traceability challenges



SCOTCH


UnitTest
Identifying the last assert statement for each method
assertTrue(--)
slicing critirion
Dynamic Slicing
slice
Extracting the classes
Class in STS
Conceptual Coupling
Class in CTS

Example: Unit Test class

```
public class RemoveElementsTest extends TestCase {

    public void testRemoveElement () {
       ConnectedGraphFactory factory = ConnectedGraphFactory.newFactory();
       ConnectedGraph graph = factory.newGraph();
       NodeElement node1 = graph.getStartNode();
       NodeElement node2 = graph.newNode();

       ... ... ...

       EdgeElement edge1, edge2, edge3, edge4, edge5;
       edge1 = graph.newEdge(node1, node2);
       edge2 = graph.newEdge(node2, node3);

       ... ... ...

       try {
              removedElement = graph.removeElement(edge2);
              fail("DisconnectedGraphException expected");
       } catch (DisconnectedGraphException e) {
              assertTrue(true);
       }
    }

}
```

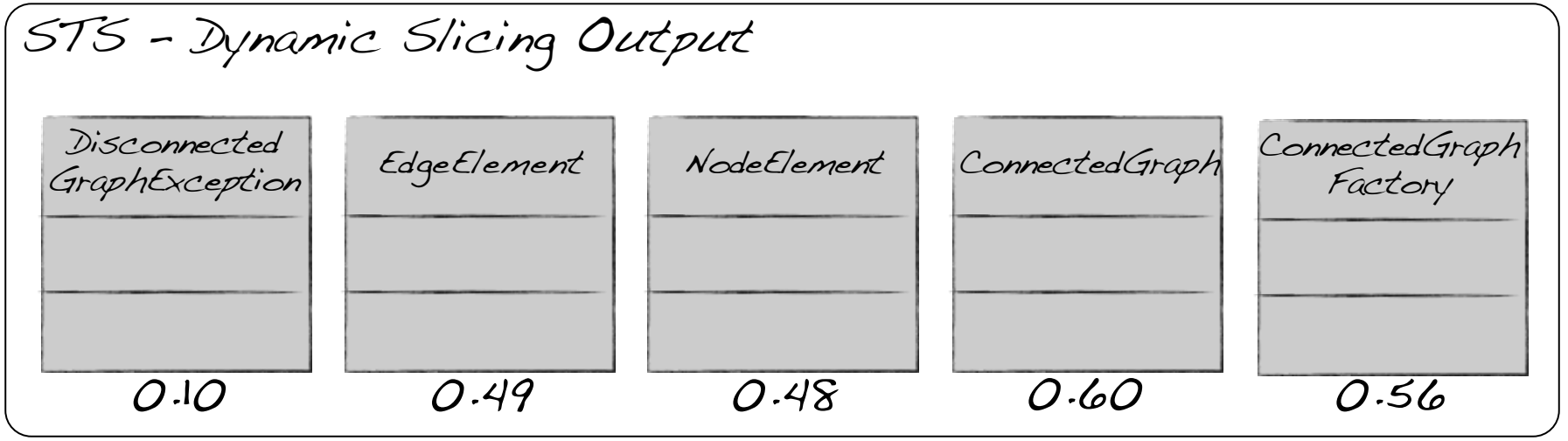
Example: last assert statement

```
public class RemoveElementsTest extends TestCase {

    public void testRemoveElement () {
       ConnectedGraphFactory factory = ConnectedGraphFactory.newFactory();
       ConnectedGraph graph = factory.newGraph();
       NodeElement node1 = graph.getStartNode();
       NodeElement node2 = graph.newNode();

       ... ... ...

       EdgeElement edge1, edge2, edge3, edge4, edge5;
       edge1 = graph.newEdge(node1, node2);
       edge2 = graph.newEdge(node2, node3);

       ... ... ...

       try {
              removedElement = graph.removeElement(edge2);
              fail("DisconnectedGraphException expected");
       } catch (DisconnectedGraphException e) {
              assertTrue(true);
       }
    }

}
```

Example: classes in STS

```
public class RemoveElementsTest extends TestCase {

    public void testRemoveElement () {
       ConnectedGraphFactory factory = ConnectedGraphFactory.newFactory();
       ConnectedGraph graph = factory.newGraph();
       NodeElement node1 = graph.getStartNode();
       NodeElement node2 = graph.newNode();

       ... ... ...

       EdgeElement edge1, edge2, edge3, edge4, edge5;
       edge1 = graph.newEdge(node1, node2);
       edge2 = graph.newEdge(node2, node3);

       ... ... ...

       try {
              removedElement = graph.removeElement(edge2);
              fail("DisconnectedGraphException expected");
       } catch (DisconnectedGraphException e) {
              assertTrue(true);
       }
    }

}
```

Example



ConnectedGraph	0.60
ConnectedGraphFactory	0.56
EdgeElement	0.49
NodeElement	0.48
DisconnectedGraphException	0.10

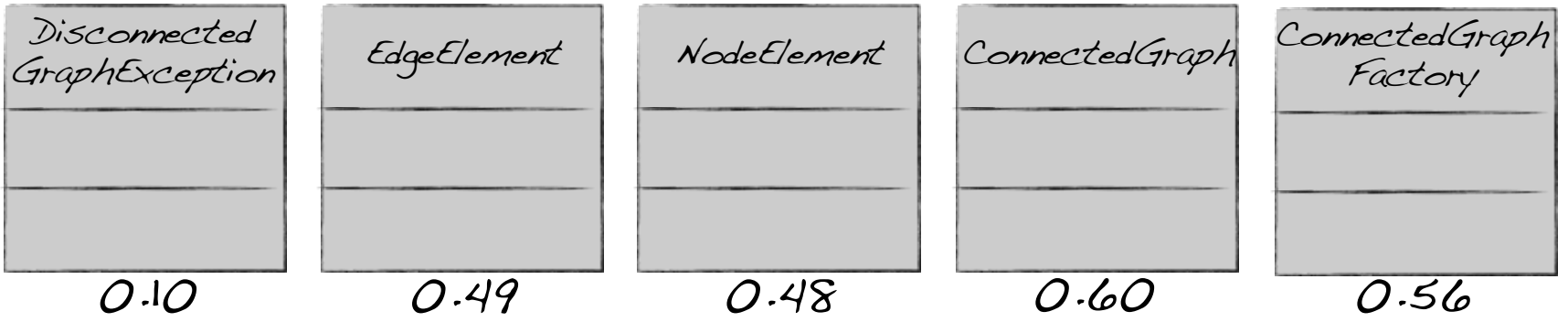
$t = \lambda * max(CCBC)$

$\lambda = 0.95$

$t = 0.95 * 0.60 = 0.57$

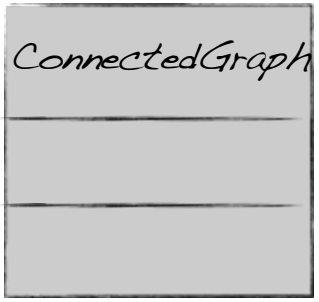
Example

STS - Dynamic Slicing Output

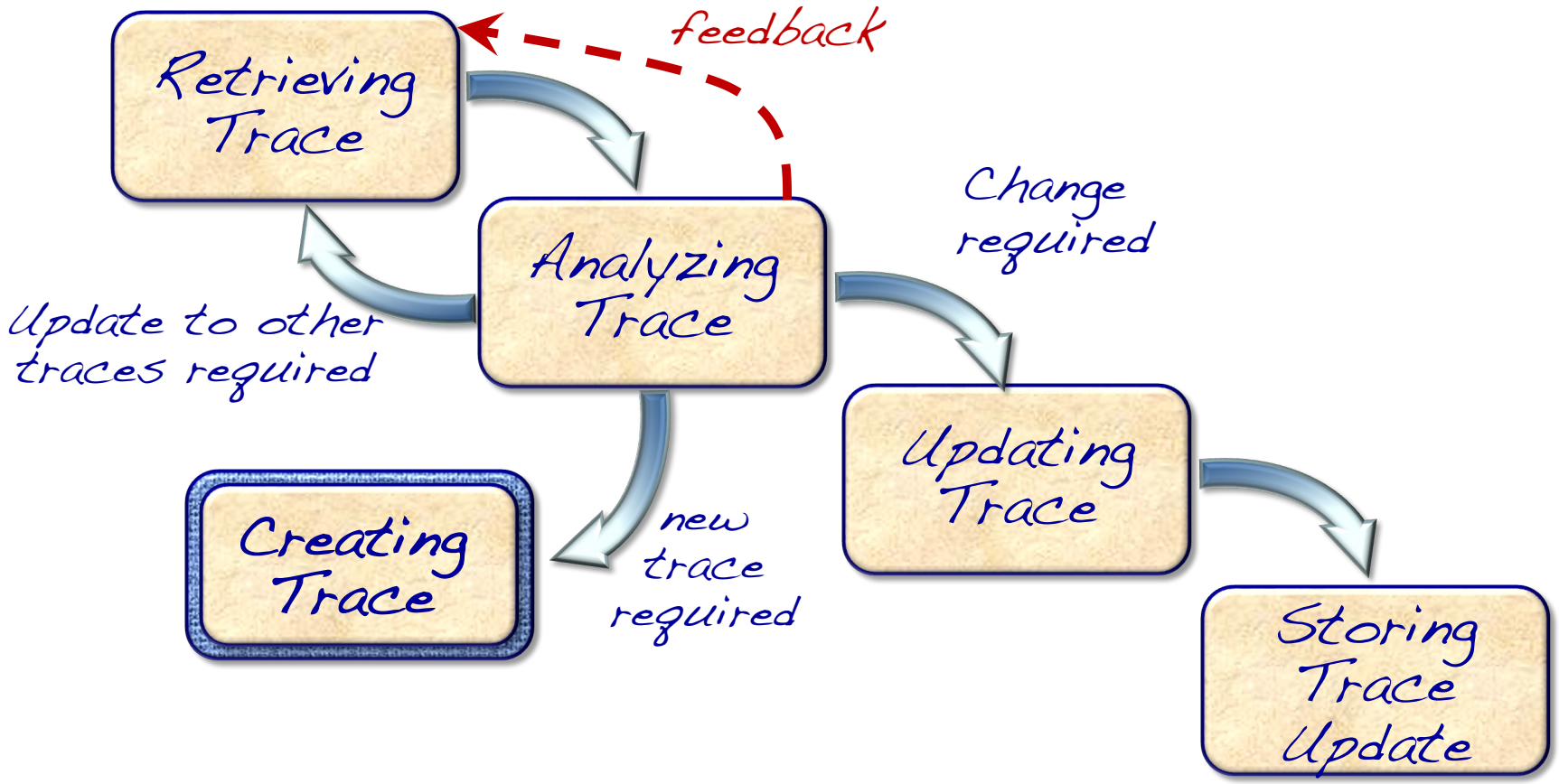


ConnectedGraph	0.60
ConnectedGraphFactory	0.56
EdgeElement	0.49
NodeElement	0.48
DisconnectedGraphException	0.10

CTS - Filtered by CCBC



Maintaining Traces



Supporting Trace Maintenance

- ✓ Event Based
  - Tracking changes (extensions of trace capture methods)
  - Exploiting versioning systems

Nistor, 2005; Nguyen et al., 2007; De Lucia et al., 2010; Heider et al., 2012

- ✓ Rule Based
  - XML or UML models
  - Traceability maintenance rules (extensions of trace capture methods)

Maletic et al., 2005; Mäder and Gotel, 2012; Seibel et al., 2012

- ✓ Text analysis and Information Retrieval …

De Lucia et al., 2007

Support for trace evolution in IR-based traceability

- ✓ Newly suggested links for newly added artefacts
- ✓ When artefacts change similarity of traced artefacts change too
  - New trace links can be suggested
  - Warning links can arise
    - Pairs of traced artefacts with a similarity below a specified "quality threshold"
  - Possible actions
    - The trace link should be removed
    - The consistency of the lexicon of the traced artefacts should be improved

Support for trace evolution in ADAMS

Functional Requirement onto Use case warning links

Threshold: < 70 % >

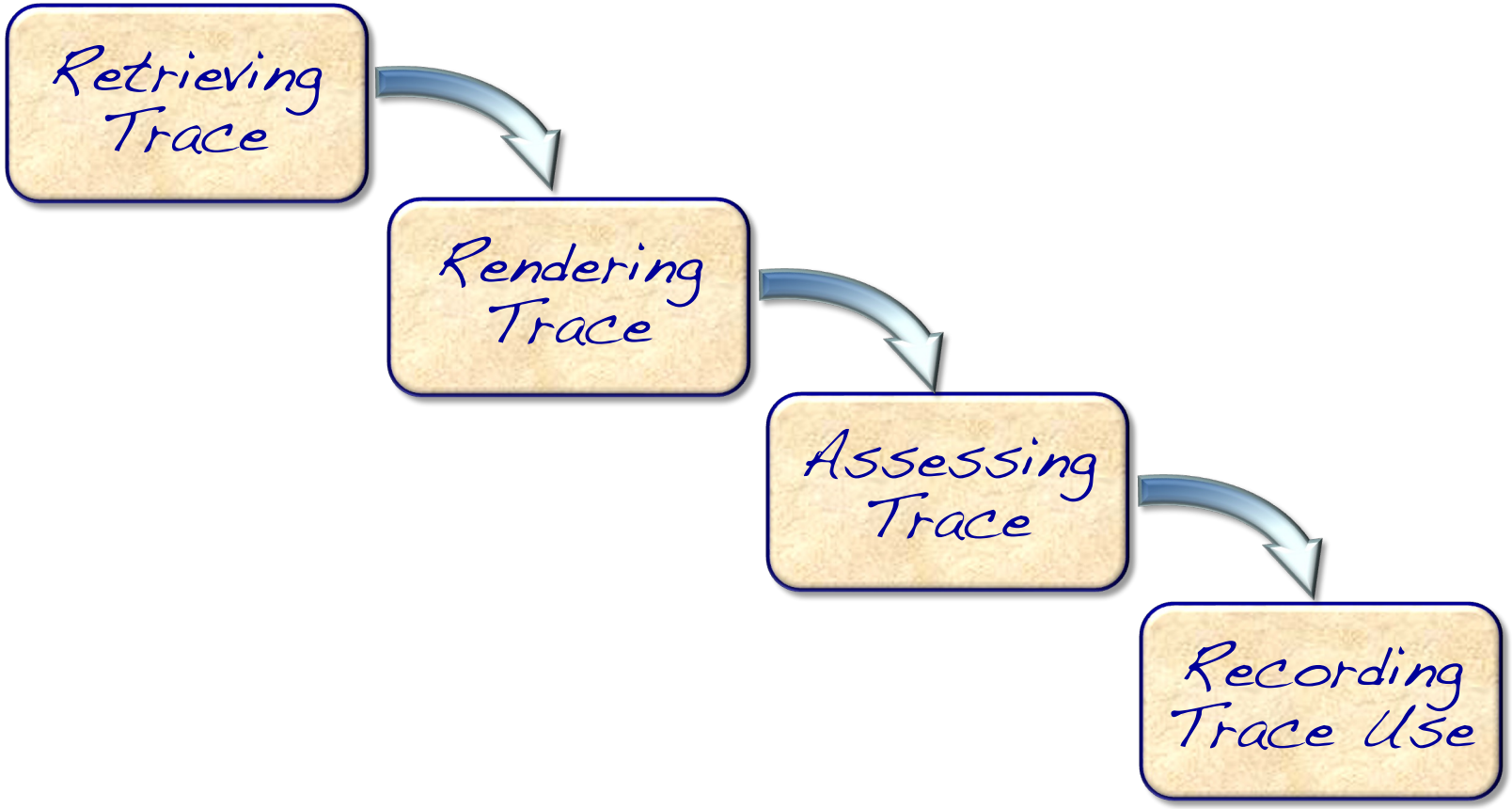
New Warning Links								
ID	Source Artefact	ID	Target Artefact	Similarity measure		Action		
				Previous	Current	Accept	Remove link	Send feedback
167	Patient management (Requirement)	206	Insert patient (Use Case)	74.11	69.11	☐	☐	☐
169	Doctor management (Requirement)	200	Dispense medicine (Use Case)	---	64.59	☐	☐	☐

Iterative Warning Links								
ID	Source Artefact	ID	Target Artefact	Similarity measure		Action		
				Previous	Current	Accept	Remove link	Send feedback
167	Patient management (Requirement)	193	Reserve visit (Use Case)	59.65	58.25	☐	☐	☐

Accepted Warning Links							
ID	Source Artefact	ID	Target Artefact	Similarity measure		Action	
				Previous	Current	Remove link	Send feedback
169	Doctor management (Requirement)	214	Allocate doctor (Use Case)	64.11	68.43	☐	☐

Submit actions

Using Traces



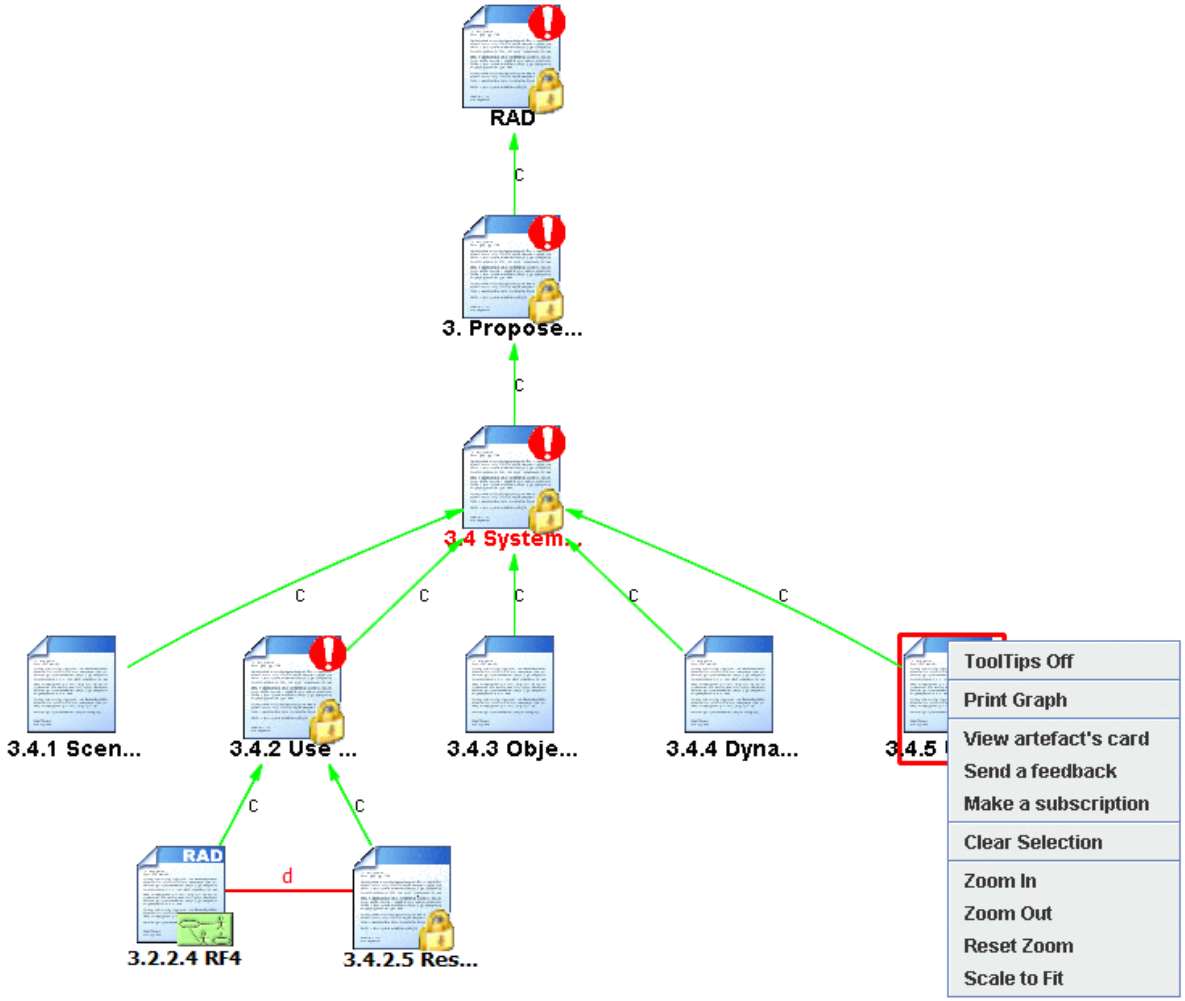
Examples of traceability uses

- ✓ Short term traceability uses
  - Requirements completeness analysis
  - Requirements trade-off analysis
  - Requirements-to-acceptance test mapping
  - Test Coverage Analysis
- ✓ Long term traceability uses
  - Impact Analysis
  - Program Comprehension
  - Software Reuse
  - Change Management

Traceability Visualisation

- ✓ Scalability problems
  - Too many artefacts
  - Too many links
  - Too many different types of information carried out
- ✓ Appropriate visualisation depends on the task
- ✓ Need for user studies …
- ✓ Only preliminary work on this topic …
  - Marcus et al. 2005

Traceability Visualisation in ADAMS



Impact Analysis

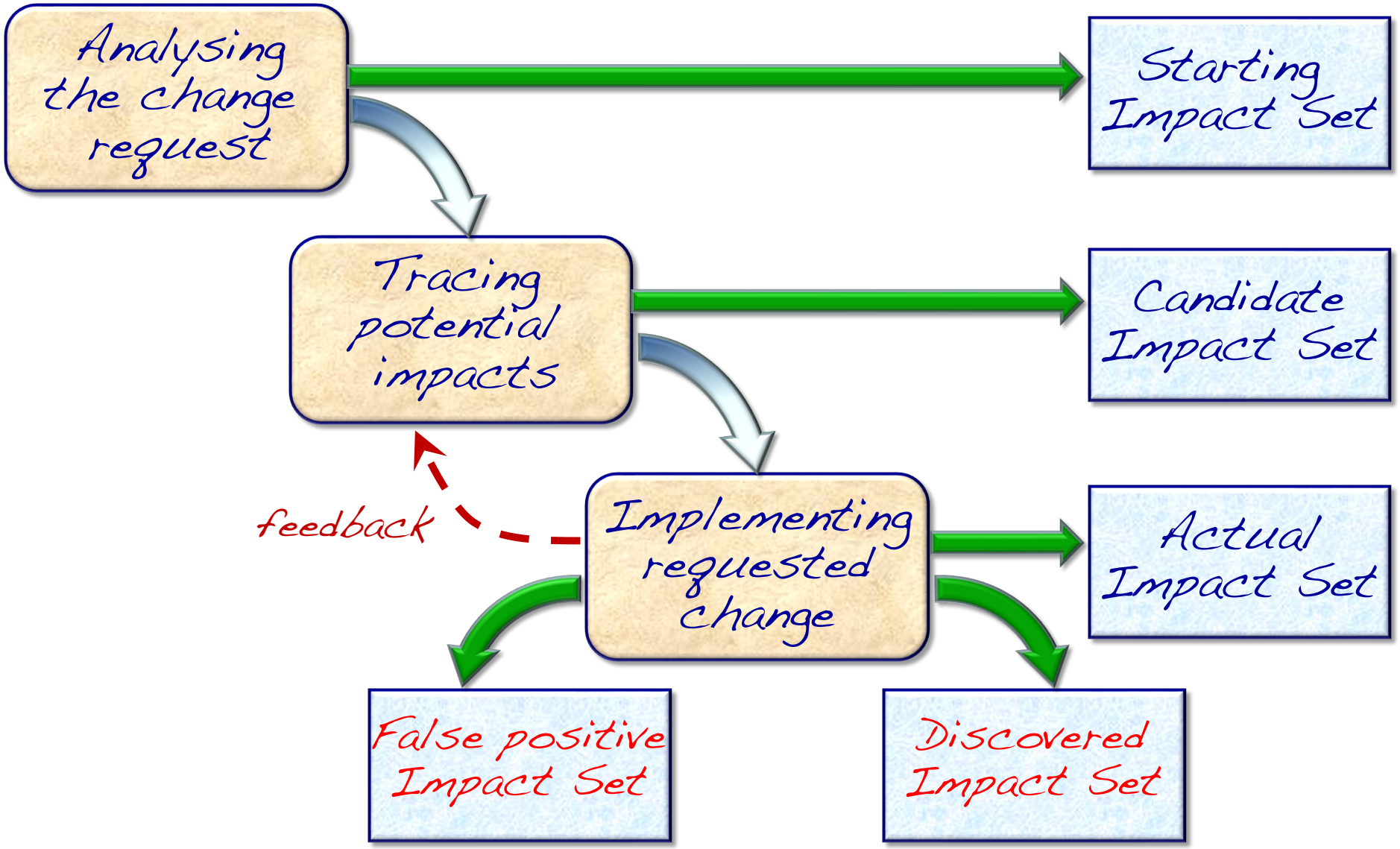
- ✓ Analysing parts of the system that will be affected by the change and examine them for possible further impacts
- ✓ Looking for ripple-effects …
- ✓ A change has not only impact on the source code, but also on the other related software artefacts, such as requirement, design, and test artefacts

Traceability and Impact Analysis

- ✓ Impact analysis can be efficiently supported through traceability information
- ✓ Once a maintainer has identified the high-level document (e.g., requirement, use case) related to the feature to be changed, traceability helps to locate the pieces of design, code and whatever need to be maintained

De Lucia et al. 2008

Impact Analysis Process



Goal of Impact Analysis Process

- ✓ Estimating a CIS as close as possible to the AIS
- ✓ This obviously depends on how accurate is the tracing activity …
  - Direct and indirect impacts …
  - Reachability analysis …
- ✓ … it also depends on how accurate is the identification of the SIS
  - Concept assignment …

Having traces is not enough

- ✓ Underestimating the SIS
  - Affected components that are not reachable from the identified SIS are not obvious or easy to detect
  - Discovered Impact Set …
- ✓ Ineffective pruning strategy during reachability analysis
  - Not all reachable components are actually affected
  - False Positive Impact Set …

Reducing false positives

- ✓ Considering the distance between artefacts
- ✓ Associating traces with specific semantics or probabilities indicating the likelihood of an indirect impact
- ✓ Defining and using propagation rules to navigate the traceability graph
- ✓ Incremental analysis of the traceability graph

Event-Based Traceability

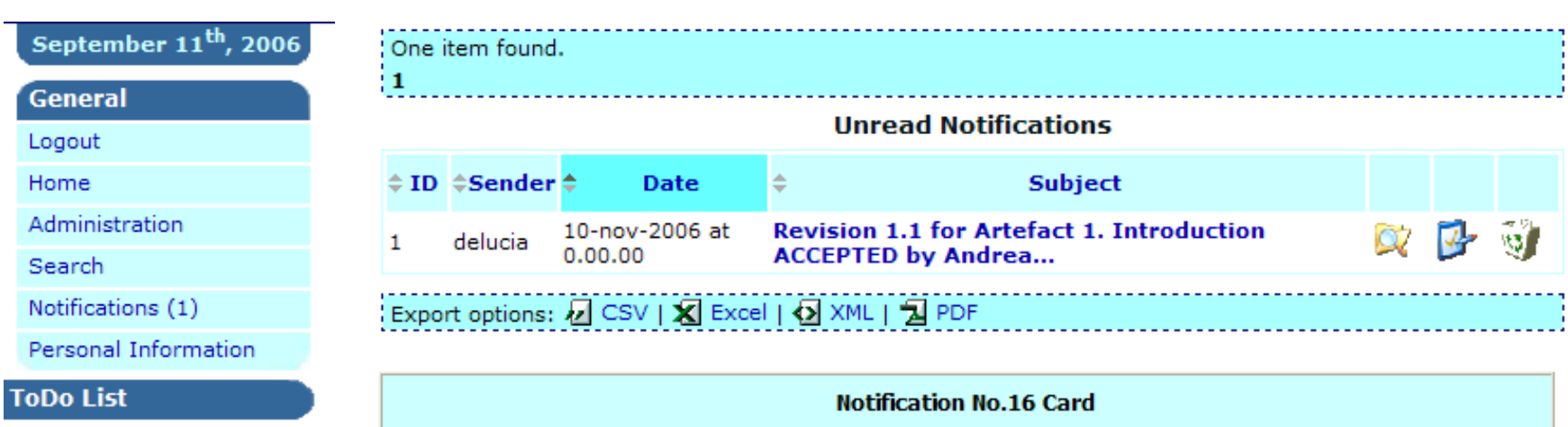
- ✓ A traceability method based upon event notification
  - Requirements and other software artefacts linked through publish-subscribe relationships
  - Requirements and other change instigators take on the role of publishers, while dependent artifacts act as subscribers
  - When requirements are changed, events are published to an event server and related notifications sent to all dependent subscribers

Cleland-Huang et al. 2003

Notifications in ADAMS

- ✓ Event-based notifications
  - Direct Notifications
  - Indirect Notifications (EBT)
- ✓ Intentional messages
  - Informal Comments (on artefacts)
  - Formal Feedbacks (on artefacts)

Notifications in ADAMS



Event-based direct notifications

- ✓ Subscribing events for artefacts of interest
- ✓ Event types are organised by levels of importance
  - Subscribing for a level implies receiving notifications for higher level events
- ✓ Event subscription levels are adaptive
  - Raised to an upper level when notifications concerning the subscribed level are ignored for a while

Event Subscription

Subscription 1

Artefact Name	Employee Functionalities
Event Level	○ No Event ○ Artefact Card modified, Artefact deleted ○ New Feedback ○ Baseline accepted, Baseline rejected ○ New Version submitted for Revision ◉ New Draft Version created (Meaningfull Changes) ○ New Draft Version created (Text Corrections) ○ New Complete Version created for a Branch ○ New Draft Version created for a Branch ○ Artefact unlocked, Artefact activated ○ Artefact Checked-Out, New Branch created ○ New Comment added/modified, Comment removed ○ New Resource allocated, Resource allocation removed ○ All Events
Indirect Notify (Distance)	1

Note: A level event subscription implies the subscription of all the levels above it!

Modify Delete

EBT in ADAMS

- ✓ Resources allocated on an artefact A will be notified when a “high impact” change is made to an artefact B on which A depends
- ✓ Notifications are propagated through the traceability graph
- ✓ Specifying the distance for event propagation on the traceability graph

Impact Specification at check-in

Select the new Status for Artefact Employee Functionalities

◉	DRAFT	○	LOW IMPACT (e.g. Text Corrections, Minor Release)
		◉	HIGH IMPACT (e.g. Major Release)
○	REVISION		

COMMENTS

Added Extend Relation

Add More Files [ ] Add

File 1 C:\Documents and Se Sfoglia...

Next >

Traceability challenges (1/5)

- ✓ The Grand Challenge of Traceability
  - Center of Excellence for Software Traceability
  - 8 challenges for a vision of traceability in 2035
- ✓ Traceability is:
  1. Purposed
  2. Cost-Effective
  3. Configurable
  4. Trusted
  5. Scalable
  6. Portable
  7. Valued
  8. Ubiquitous

Gotel et al., 2012b

Traceability challenges (2/5)

1. Purposed
   - Traceability is fit-for-purpose and supports stakeholder needs
   - *Major research theme*: To define and instrument prototypical traceability profiles and patterns
2. Cost-effective
   - The return from using traceability is adequate in relation to the outlay of establishing it
   - *Major research theme*: To develop cost-benefit models for analysing stakeholder requirements for traceability and associated solution options at a fine-grained level of detail
3. Configurable
   - Traceability is established as specified, moment-to-moment, and accommodates changing stakeholder needs
   - *Major research theme*: To use dynamic, heterogeneous and semantically rich traceability information models (or similar specifications of the intended traceability) to guide the definition and provision of traceability

Traceability challenges (3/5)

4. Trusted
   - All stakeholders have full confidence in the traceability, as it is created and maintained in the face of inconsistency, omissions, and changes
   - All stakeholders can and do depend upon the traceability provided
   - *Major research theme*: To perform systematic quality assessment and assurance of the traceability
5. Scalable
   - Varying types of artefacts can be traced, at variable levels of granularity and in quantity, as traceability extends through-life and across organisational and business boundaries
   - *Major research theme*: To provide for levels of abstraction and granularity in traceability techniques, methods and tools, facilitated by improved trace visualisations, to handle very large datasets and the longevity of these data

Traceability challenges (4/5)

5. Portable
   - Traceability is exchanged, merged, and reused across projects, organisations, domains, product lines, and supporting tools
   - *Major research theme*: To agree upon universal policies, standards, and a unified representation or language for expressing traceability concepts
6. Valued
   - Traceability is a strategic priority and valued by all
   - Every stakeholder has a role to play and actively discharges his or her responsibilities
   - *Major research theme*: To raise awareness of the value of traceability, to gain buy-in to education and training, and to get commitment to implementation.

Traceability challenges (5/5)

8. Ubiquitous
   - Traceability is always there, without ever having to think about getting it there, as it is built into the engineering process
   - Traceability has effectively "disappeared without a trace"
   - *Long term research theme*: To provide automation such that traceability is encompassed within broader software and systems engineering processes, and is integral to all tool support

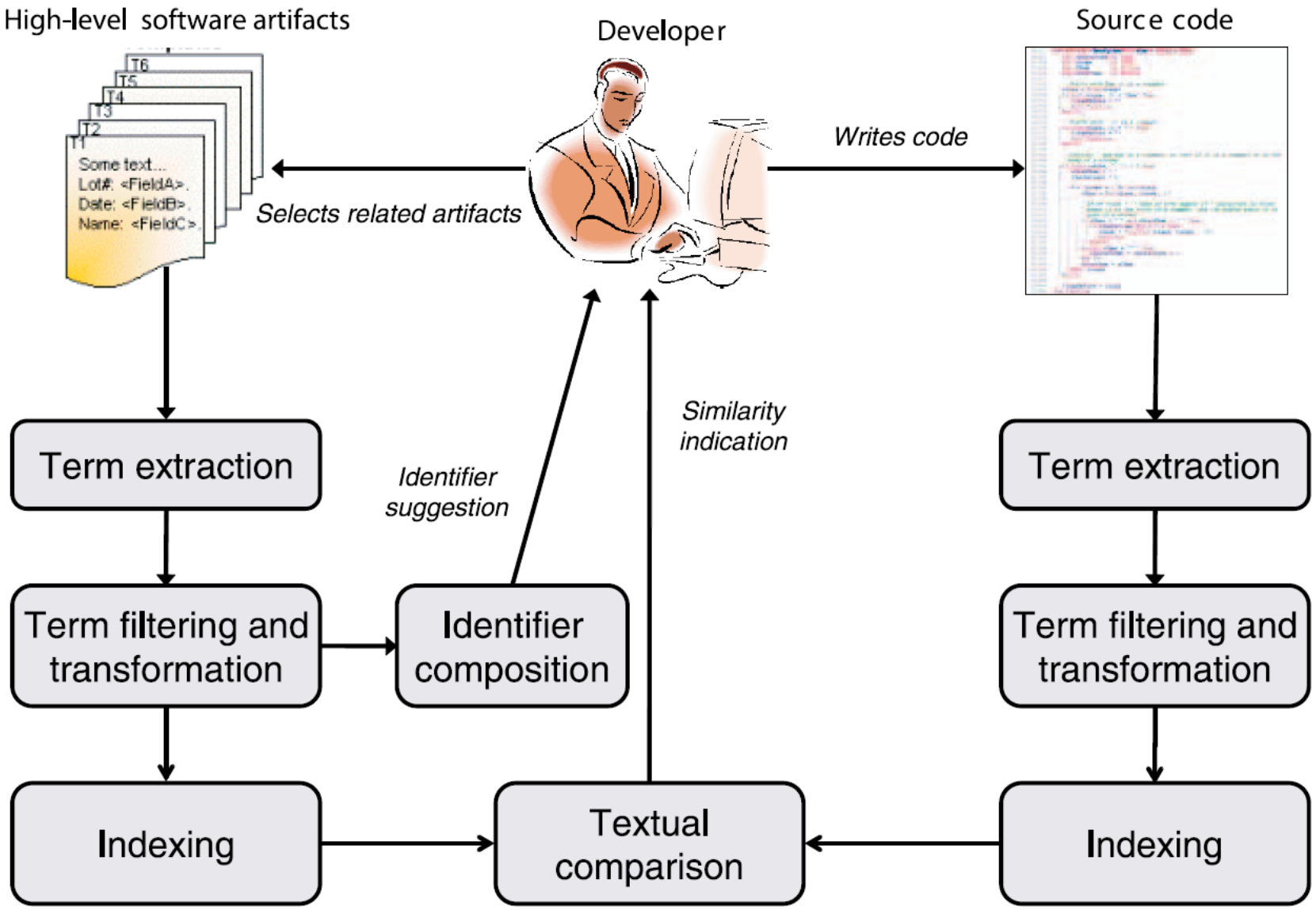
The grand challenge of traceability

Improving Traceability: An Example

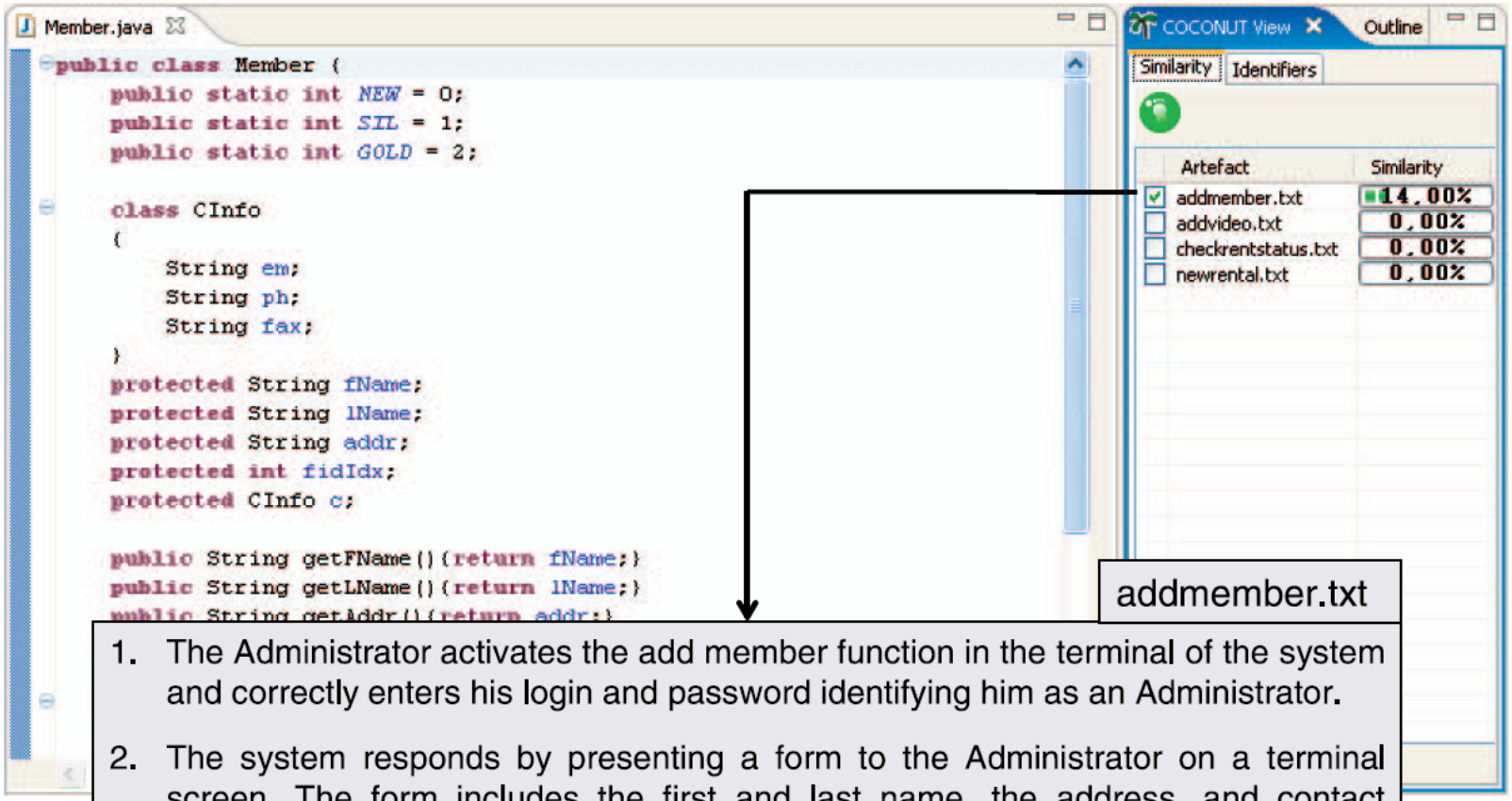
- ✓ Improving documentation-to-code traceability …
  - … by improving source code lexicon
- ✓ COCONUT (Code COmprehension NUrTurant)
  - Showing text similarity between high level artifacts and source code
  - Suggesting identifiers from high level artifacts

De Lucia et al., 2011

COCONUT Method



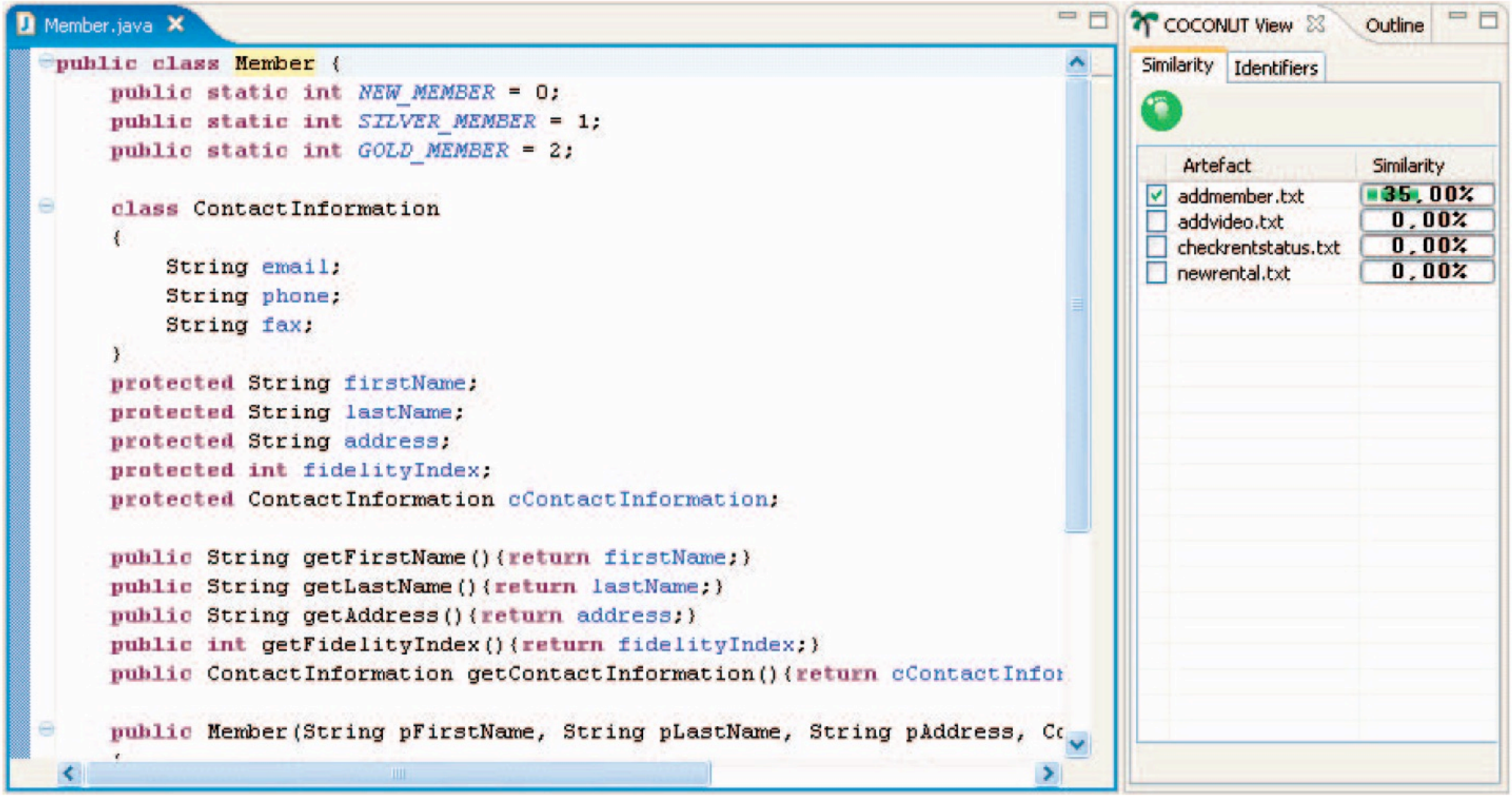
COCONUT Eclipse plug-in



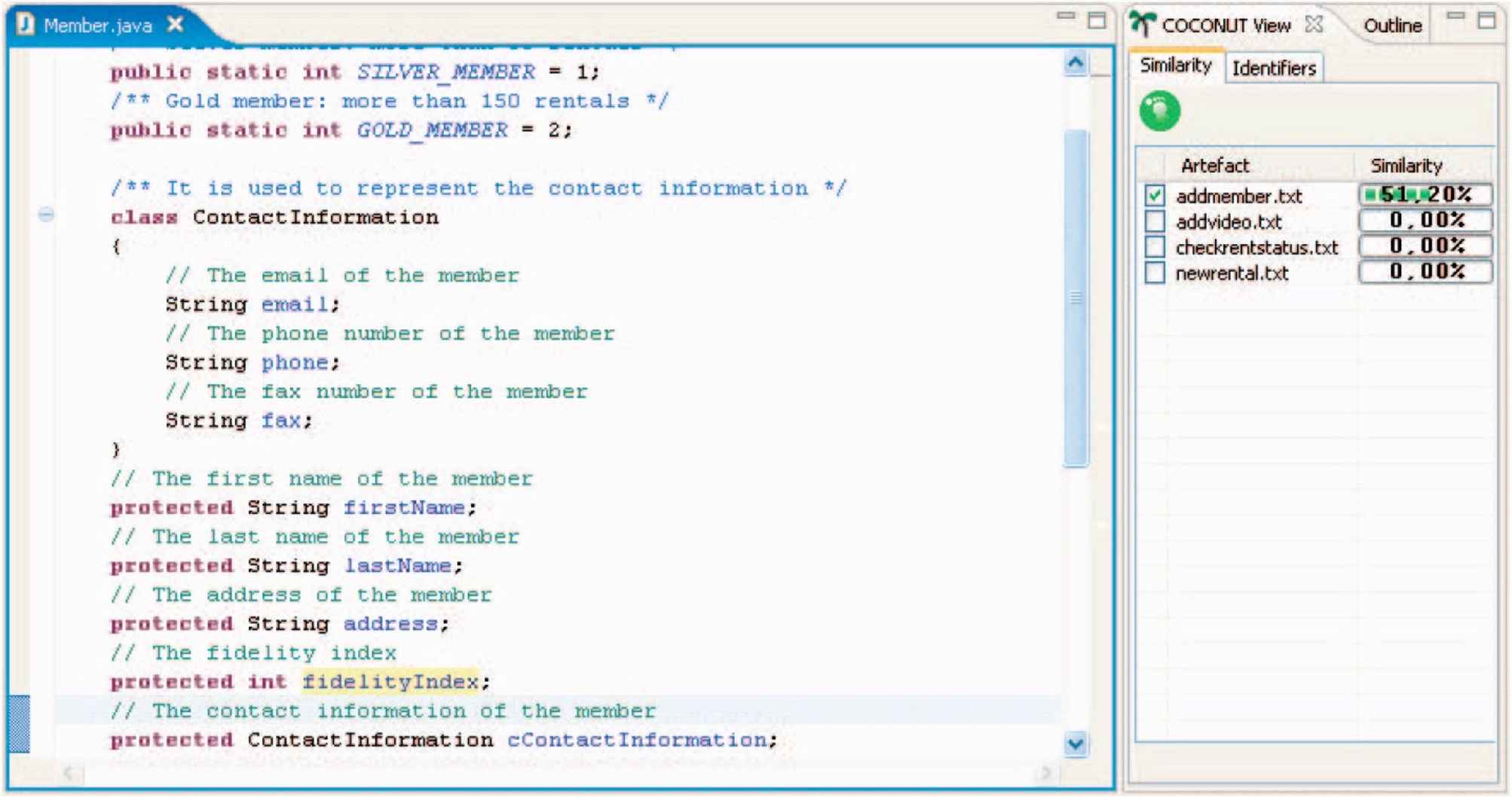
addmember.txt

1. The Administrator activates the add member function in the terminal of the system and correctly enters his login and password identifying him as an Administrator.
2. The system responds by presenting a form to the Administrator on a terminal screen. The form includes the first and last name, the address, and contact information (phone, email and fax) of the customer, as well as the fidelity index. The fidelity index can be: New Member, Silver Member, and Gold Member. After 50 rentals the member is considered as Silver Member, while after 150 rentals the member becomes a Gold Member. The system also displays the membership fee to be paid.
3. The Administrator fills the form and then confirms all the requested form information is correct.

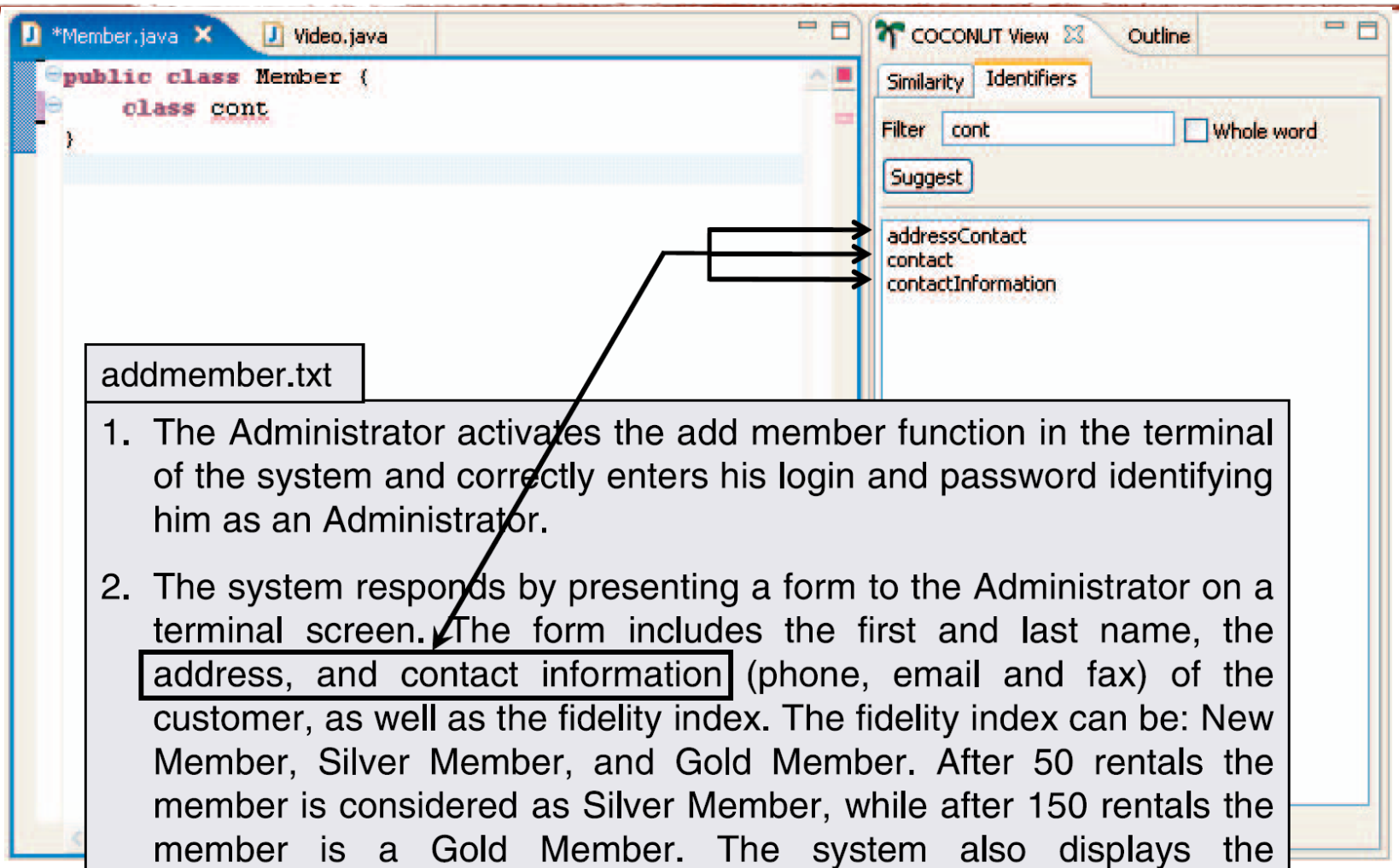
Improving identifiers



Improving comments



Suggesting identifiers



addmember.txt

1. The Administrator activates the add member function in the terminal of the system and correctly enters his login and password identifying him as an Administrator.
2. The system responds by presenting a form to the Administrator on a terminal screen. The form includes the first and last name, the address, and contact information (phone, email and fax) of the customer, as well as the fidelity index. The fidelity index can be: New Member, Silver Member, and Gold Member. After 50 rentals the member is considered as Silver Member, while after 150 rentals the member is a Gold Member. The system also displays the membership fee to be paid.
3. The Administrator fills the form and then confirms all the requested form information is correct.

References (1/4)

- ✓ Antoniol G., Caprile B., Potrich A., Tonella P. (2000). “Design-code traceability for object-oriented systems”. *Annals of Software Engineering*, 9(1-4):35–58.
- ✓ Cleland-Huang J., Change C.K., Christensen M. (2003). “Event-based traceability for managing evolutionary change”. *IEEE TSE*, 29(9):796–810.
- ✓ Cleland-Huang J., Zemont G., Lukasik W. (2004). “A heterogeneous solution for improving the return on investment of requirements traceability”. *Procs of 12th IEEE International Conference on Requirements Engineering*, pp. 230–239.
- ✓ De Lucia, A. Fasano F., Oliveto R., Tortora G. (2007). “Recovering traceability links in software Artefact management systems using information retrieval methods”. *ACM TOSEM*, 16(4).
- ✓ De Lucia, A. Fasano F., Oliveto R. (2008). “Traceability Management for Impact Analysis”. *Frontiers of Software Maintenance*, IEEE, pp. 21-30
- ✓ De Lucia, A. Fasano F., Oliveto R., Tortora G. (2010). “Fine-grained Management of Software Artefacts: The ADAMS System”. *SPE*, 40(11):1007–1034.
- ✓ De Lucia A., Di Penta M., Oliveto R. (2011). “Improving Source Code Lexicon via Traceability and Information Retrieval”. *IEEE TSE*, 37(2): 205-227.
- ✓ De Lucia A., Marcus M., Oliveto R., Poshyvanyk D. (2012). "Information Retrieval Methods for Automated Traceability Recovery". in: J. Cleland-Huang, O. Gotel, and A. Zisman (eds), *Software and Systems Traceability*, Springer-Verlag, pp. 71-98.

References (2/4)

- ✓ Egyed A., Grünbacher P. (2002). “Automating requirements traceability: Beyond the record & replay paradigm”. *Procs of 17th IEEE international conference on Automated Software Engineering*, pp. 163–171.
- ✓ Egyed A., Grünbacher P., Heindl M., Biffl S. (2007). “Value-based requirements traceability: Lessons learned”. *Procs of 15th IEEE International Requirements Engineering Conference*, pp. 115–118.
- ✓ Gall H., Hajek K., Jazayeri M. (1998). “Detection of logical coupling based on product release history”. *Procs of 14th IEEE International Conference on Software Maintenance*, pp. 190–198.
- ✓ Gotel, O., Finkelstein, A. (1994). “An analysis of the requirements traceability problem”. *Procs of 1st IEEE Intl. Conference on Requirements Engineering*, pp. 94–101.
- ✓ Gotel, O. et al. (2012a). “Traceability Fundamentals”. in: J. Cleland-Huang, O. Gotel, and A. Zisman (eds), *Software and Systems Traceability*, Springer-Verlag, 2012, pp. 3-22.
- ✓ Gotel, O. et al. (2012b). “The Grand Challenge of Traceability (v1.0)”. in: J. Cleland-Huang, O. Gotel, and A. Zisman (eds), *Software and Systems Traceability*, Springer-Verlag, 2012, pp. 343-412.
- ✓ Gotel O., Mäder P. (2012). “Acquiring Tool Support for Traceability”. in: J. Cleland-Huang, O. Gotel, and A. Zisman (eds), *Software and Systems Traceability*, Springer-Verlag, 2012, pp. 43-70.

References (3/4)

- ✓ Heider W., Grünbacher P., Rabiser R., Lehofer M. (2012). “Evolution-Driven Trace Acquisition in Eclipse-Based Product Line Workspaces”. in: J. Cleland-Huang, O. Gotel, and A. Zisman (eds), *Software and Systems Traceability*, Springer-Verlag, 2012, pp. 195-214.
- ✓ Ingram C., Riddle S. (2012). “Cost-Benefits of Traceability”. in: J. Cleland-Huang, O. Gotel, and A. Zisman (eds), *Software and Systems Traceability*, Springer-Verlag, 2012, pp. 23-42.
- ✓ Lindvall M., Sandahl K. (1996). “Practical implications of traceability”. *SPE*, 26(10):1161–1180.
- ✓ Lindvall M., Sandahl K. (1998). Traceability aspects of impact analysis in object-oriented systems”. *JSME*, 10(1):37–57.
- ✓ Mäder P., Gotel O. (2012). “Ready-to-Use Traceability on Evolving Projectseability”. in: J. Cleland-Huang, O. Gotel, and A. Zisman (eds), *Software and Systems Traceability*, Springer-Verlag, 2012, pp. 173-194.
- ✓ Maletic J. I., Collard M.L., Simoes B. (2005). “An XML based approach to support the evolution of model-to-model traceability links”. *Procs of 3rd ACM Intl Workshop on Traceability in Emerging Forms of Software Engineering*, pp. 67–72.
- ✓ Marcus, A., Xie X., Poshyvanyk D. (2005). “When and how to visualize traceability links?”. *Procs of 3rd Intl Workshop on Traceability in Emerging Forms of Software Engineering*, pp. 56–61.

References (4/4)

- ✓ Nistor E. C., Erenkrantza J. R., Hendrickson S.A., van der Hoek A. (2005). "ArchEvol: Versioning architectural implementation relationships". *Procs of 12th Intl Workshop on Software Configuration Management*, pp 99–111.
- ✓ Nguyen T. N., Munson E. V., Boyland J. T., Thao J. T. (2007). "Infrastructures for development of object-oriented configuration management services*". Procs of 27th International Conference on Software Engineering*, pp 215–224.
- ✓ Pfleeger S. L., Bohner S. A. (1990). "A framework for software maintenance metrics". *Procs of 6th IEEE Conference on Software Maintenance*, pp. 320–327.
- ✓ Qusef A., Bavota G., Oliveto R., De Lucia A., Binkley D. (2011). "SCOTCH: Test-to-Code Traceability using Slicing and Conceptual Coupling". *Procs of 27th IEEE International Conference on Software Maintenance*, pp. 63-72
- ✓ Ramesh B., Jarke M. (2001). "Towards reference models for requirements traceability". *IEEE TSE*, 27(1):58–93.
- ✓ Seibel A., Hebig R., Giese H. (2012). "Traceability in Model-Driven Engineering: Efficient
- ✓ and Scalable Traceability Maintenance". in: J. Cleland-Huang, O. Gotel, and A. Zisman (eds), *Software and Systems Traceability*, Springer-Verlag, 2012, pp. 215-240.
- ✓ Watkins, R., Neal, M. (1994). "Why and how of requirements tracing". *IEEE Software*, 11:104–106.
- ✓ Zisman A. (2012). "Using Rules for Traceability Creation". in: J. Cleland-Huang, O. Gotel, and A. Zisman (eds), *Software and Systems Traceability*, Springer-Verlag, 2012, pp. 147-172.

Questions ?