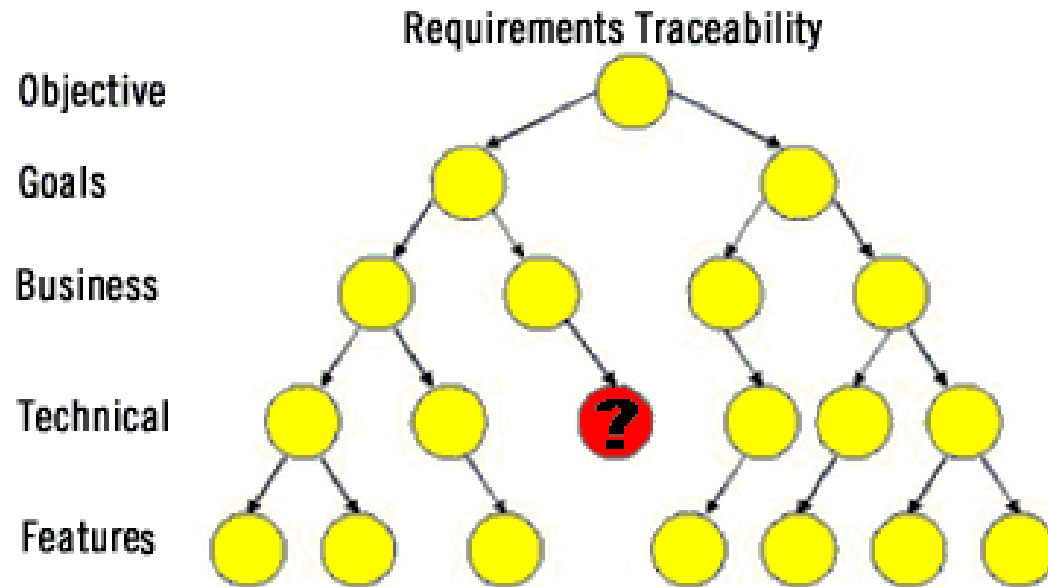


Automating Requirements Traceability



Ilya Ryzhov
Nikita Shah

«Ability to describe and follow the life of requirement by defining relations to other artifacts»

Goals:

- Facilitate communication;
- Support integration of changes;
- Preserve design knowledge;
- Assure quality.

Automating RT using Trace Analyzer

Dependencies between requirements and
source code

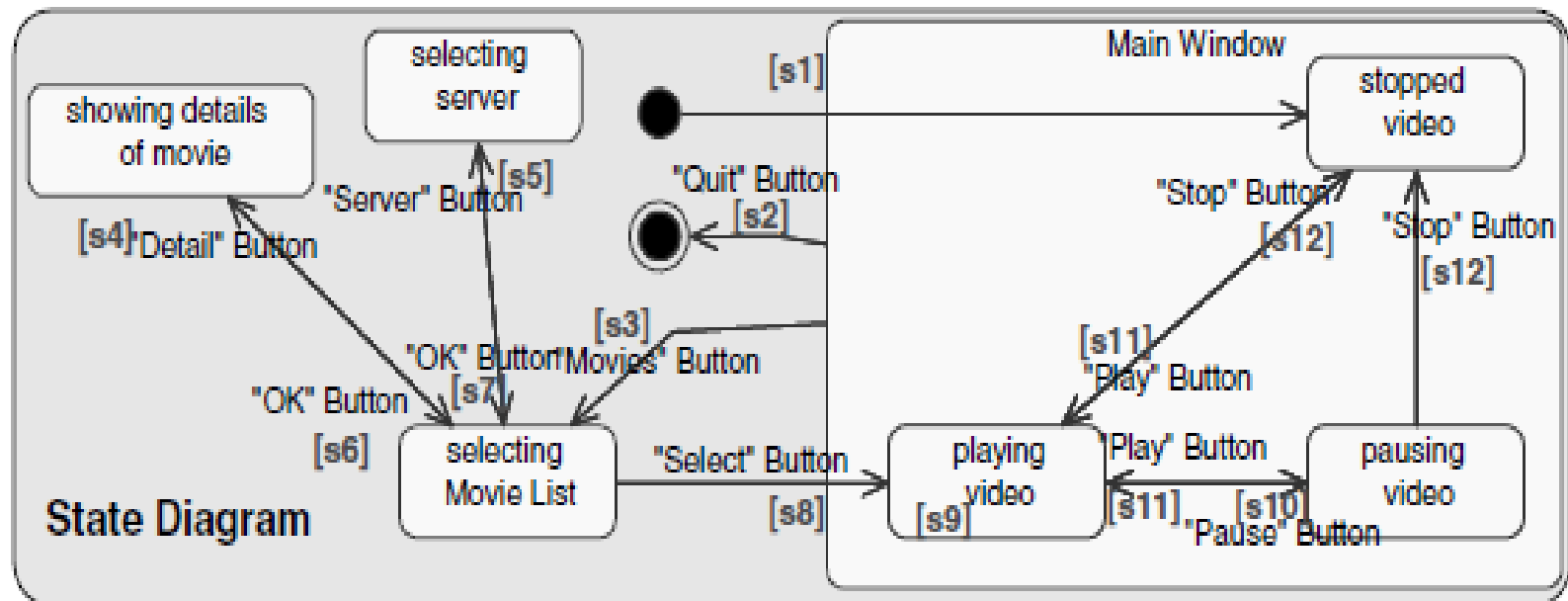


Graph with all overlaps



Graph analysis

Video-on-demand (VOD) Case Study



List of Requirements

r0	Users should be able to display a list of available movies and select one from the list
r1	Play movie immediately after selection from list
r2	Users should be able to display textual information about a selected movie
r3	User should be able to pause a movie
r4	3 seconds max to load movie list
r5	3 seconds max to load textual information about a movie
r6	1 second max to start playing a movie
r7	Provide VCR-like user interface
r8	User should be able to stop a movie
r9	User should be able to start a movie

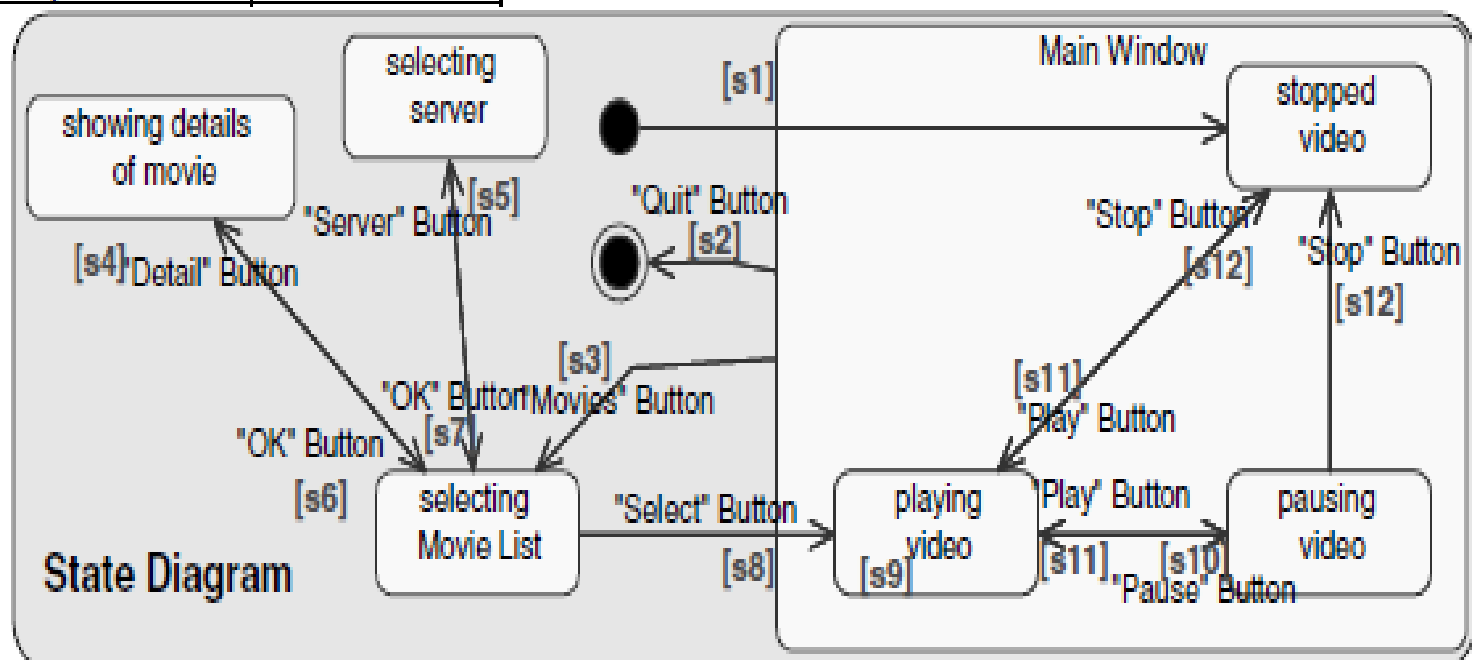
Scenarios and Observed Footprints

<i>Test Scenario</i>	<i>Artifact</i>	<i>Observed Java Classes</i>
1. view movie list	[s3]	[C,J,R,U]
2. view textual movie information	[s4,s6][r2]	[C,E,J,N,R]
3. select/play movie	[s8,s9][r6]	[A,C,D,F,G,I,J,K,N,O,T,R,U]
4. press stop button	[s9,s12][r8]	[A,C,D,F,G,I,K,O,T,U]
5. press play button	[s9,s11][r9]	[A,C,D,F,G,I,K,N,O,T,R,U]
6. change server	[s5,s7]	[C,R,J,S]
7. playing	[s9]	[A,C,D,F,G,I,K,O]
8. get textual movie information	[r5]	[N,R]
9. movie list	[r4]	[R]
10. VCR-like UI	[r7]	[A,C,D,F,G,I,K,N,O,R,T,U]
11. select movie	[r0]	[C,J,N,R,T,U]
12. select/play movie	[r1]	[A,C,D,F,G,I,J,K,N,O,R,T,U]
13. press pause	[s9,s10][r3]	[A,C,D,F,G,I,K,O,U]

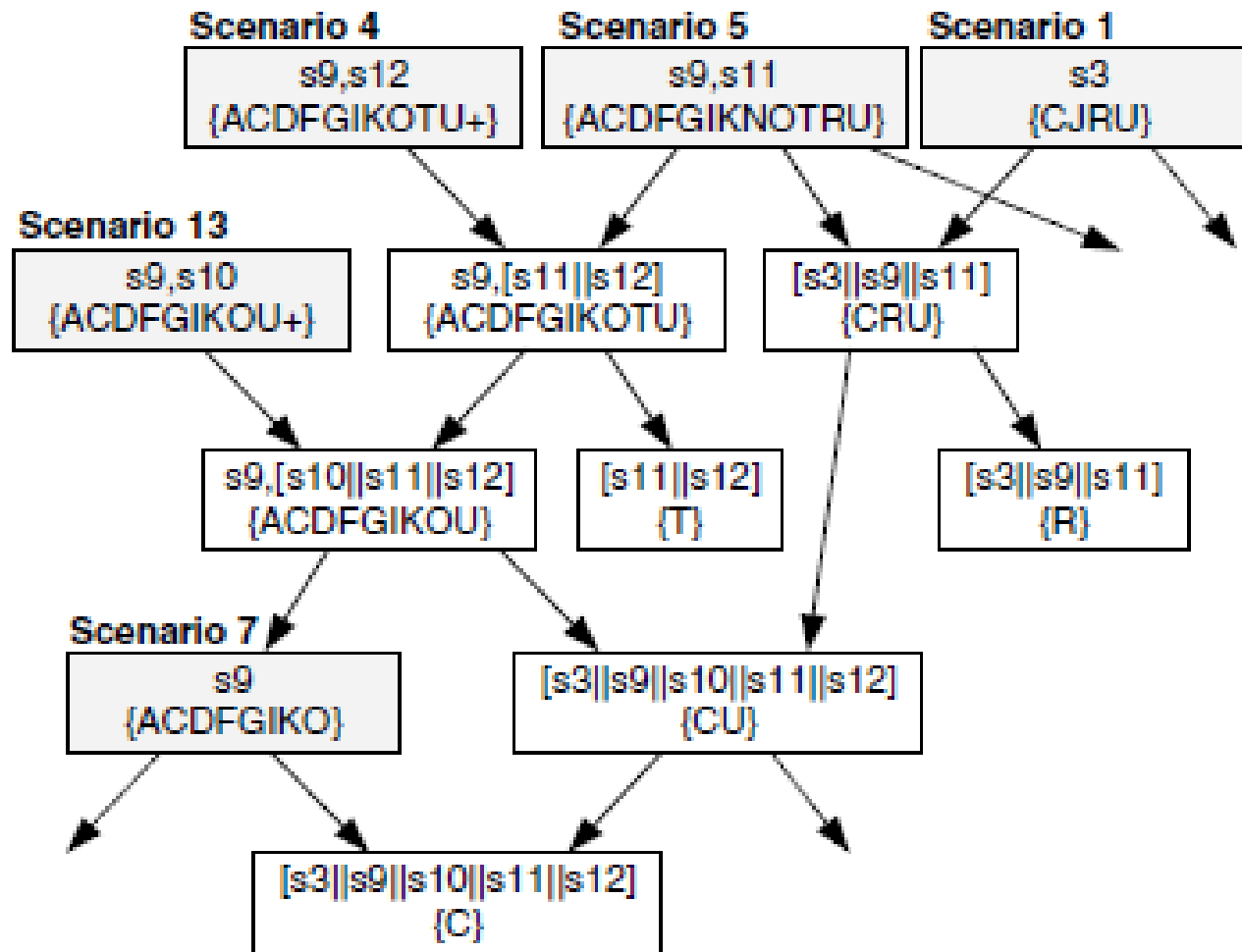
Scenarios and Observed Footprints

Test Scenario	Artifact
1. view movie list	[s3]
2. view textual movie information	[s4,s6][r2]
3. select/play movie	[s8,s9][r6]
4. press stop button	[s9,s12][r8]
5. press play button	[s9,s11][r9]
6. change server	[s5,s7]
7. playing	[s9]
8. get textual movie information	[r5]
9. movie list	[r4]
10. VCR-like UI	[r7]
11. select movie	[r0]
12. select/play movie	[r1]
13. press pause	[s9,s10][r3]

Links between scenarios and state diagram



Footprint Graph



Dependencies between artifacts and source code (Java classes)

[illegible]

Requirement Conflict

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
r0			F							F				F				F		F	F
r1	F		F	F		F	F		F	F	F			F	F			F		F	F
r2			F		F					F				F				F			
r3	F		F	F		F	F		F		F				F						F
r4																		F			
r5														F				F			
r6	F		F	F		F	F		F	F	F			F	F			F		F	F
r7	F		F	F		F	F		F		F			F	F			F		F	F
r8	F		F	F		F	F		F		F				F					F	F
r9	F		F	F		F	F		F		F			F	F			F		F	F
s3			P							P								P			P
s9	F		P	F		F	F		F		F				F						
s10			P																		P
s11			P											P				P		P	P
s12			P																	P	P

r5	3 seconds max to load textual information about a movie
r6	1 second max to start playing a movie

Adding / Changing Requirements

r6	3 second max to start playing a movie
r10	Avoid image degradation caused by temporary network-load fluctuations

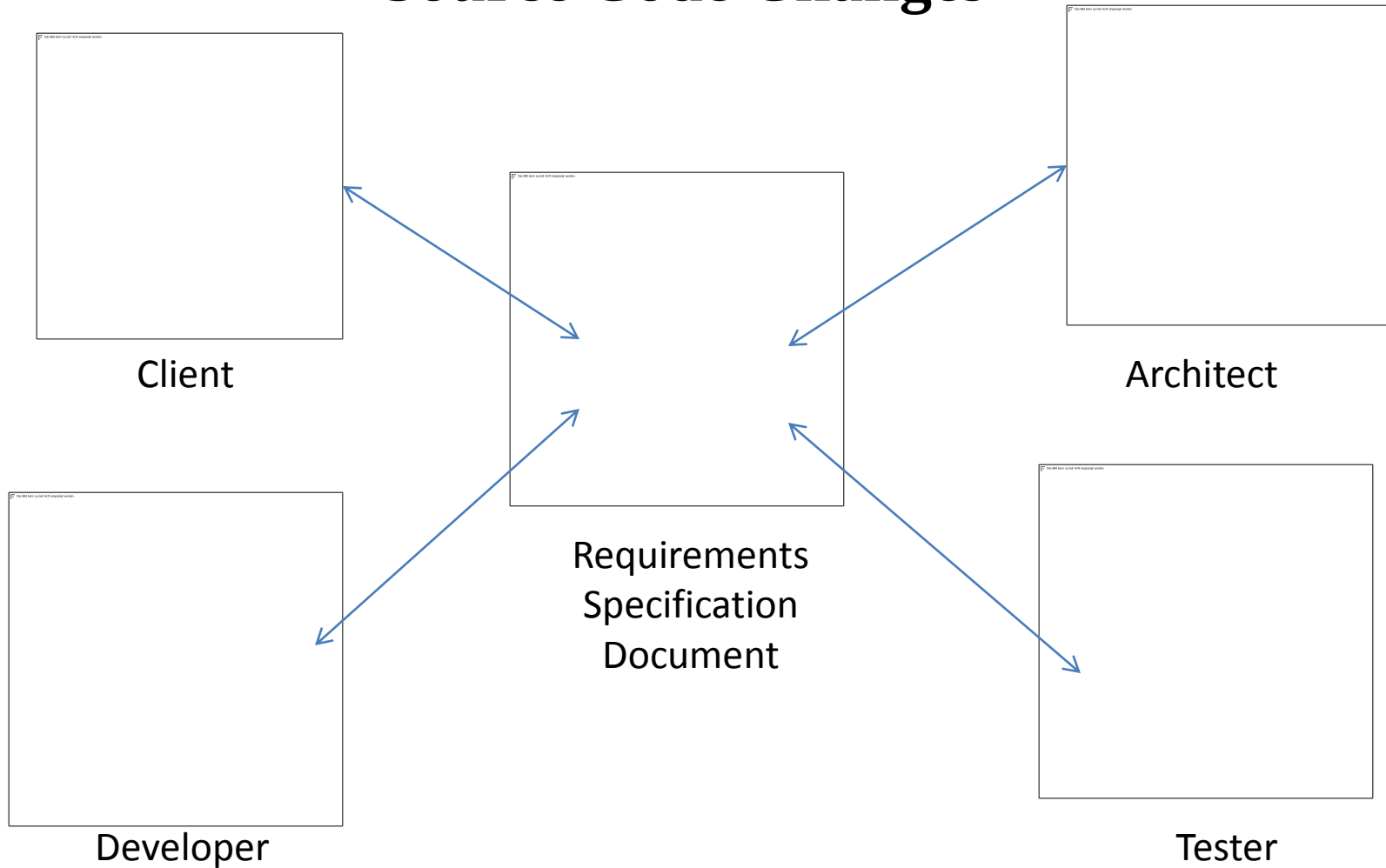
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
r10	F			F			F		F		F				F						

Findings

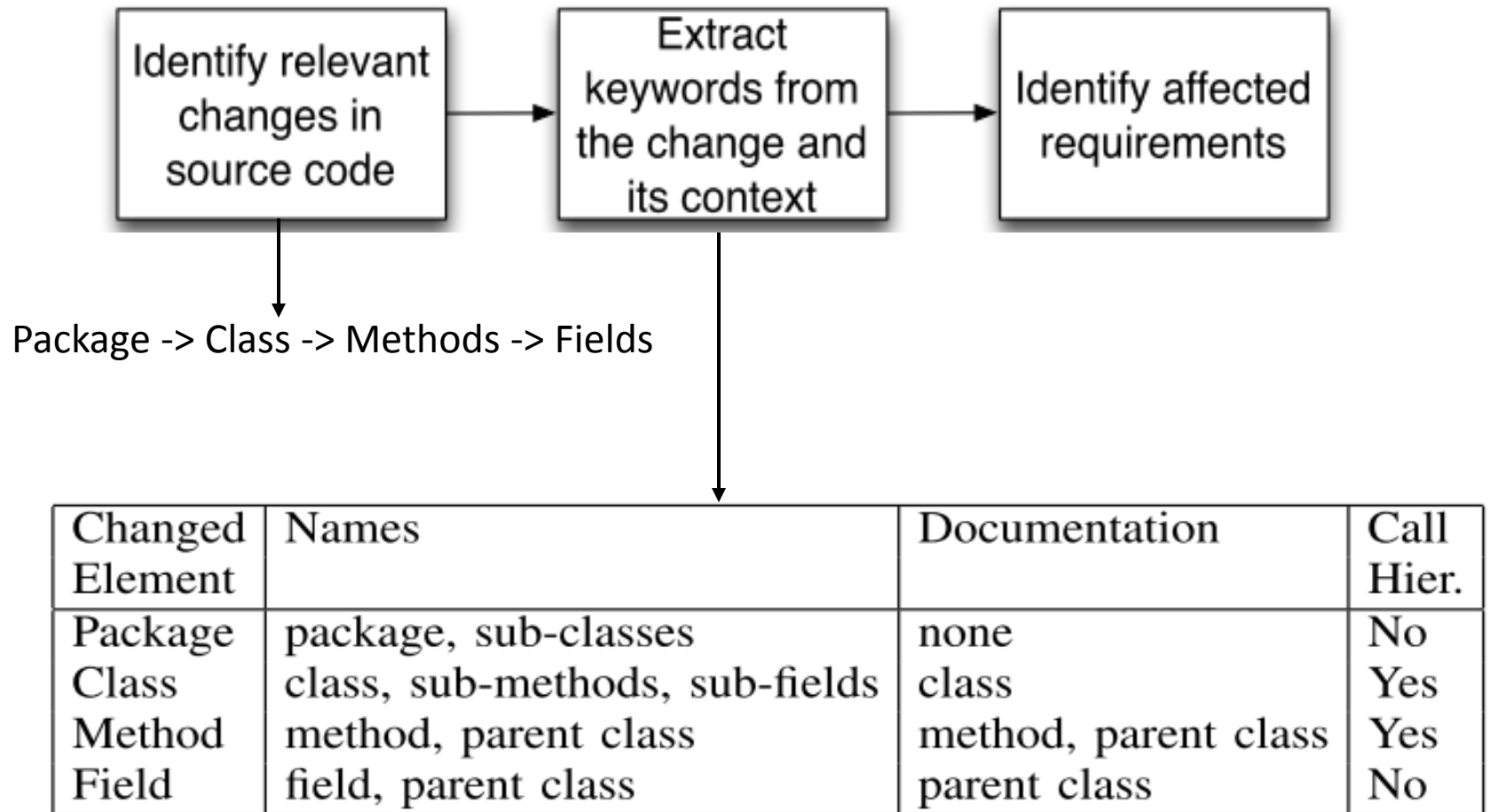
Automated Requirements Traceability:

- Reduces the effort and complexity of eliciting traces;
- More complete traces;
- Creates traces that are difficult to anticipate.

Identifying Outdated Requirements Based on Source Code Changes



Approach for identifying outdated requirements



What heuristics can be used to identify source code changes that likely affect the external behavior of the system?

1. Changes in method bodies are in most cases related to refactoring and/or bug fixes. ❌
2. Additions of new elements (classes; methods; package; fields) are usually related to the addition or extension of features. ✅
3. Additions and removals of elements having similar names are usually rename operations. ❌
4. Changes in methods signature are usually related to refactoring. ❌
5. Changes in private elements can affect the external behavior of the system. ✅
6. Additions of several methods having the same name are usually related to the same feature. ✅

Case Study - iTrust Health care project

Tools:

1. Prototype:

- Compares two versions of code and identifies the requirements-related changes.
- Extracts the keywords related to each of the changes.

2. RETRO (Requirements Tracing On target) :

- It is an automated tool for generating traceability links among textual artifacts

Case Study - iTrust Health care project

Sample UseCase

UC1 Create and Disable Patients Use Case

- 1.1 Preconditions: The iTrust HCP has authenticated himself or herself in the iTrust Medical Records system [UC3].
- 1.2 Main Flow: An HCP creates patients [S1] and disables patients [S2]. The create/disable patients and HCP transaction is logged [UC5].
- 1.3 Sub-flows:
 - [S1] The HCP enters a patient as a new user of iTrust Medical Records system. Only the name and email are is provided. An email with the patient's assigned MID and a secret key (the initial password) is personally provided to the user, with which the user can reset his/her password. The HCP can edit the patient according to data format 6.4 [E1] with all initial values (except patient MID) (...)
 - [S2] The HCP provides the MID of a patient for whom he/she wants to disable [E2]. The HCP provides a deceased date (data format 6.4). An optional diagnosis code is entered as the cause of death.
- 1.4 Alternative Flows:
 - [E1] The system prompts the enterer/editor to correct the format of a required data field because the input of that data field does not match that specified in data format 6.4 for patients.
 - [E2] (...)

Case Study - iTrust Health care project

Study 1 : Evaluate the change identification step of the approach.

Results:

- 33 classes were identified, covering 12 of the 14 requirements-related changes.
- Among the 33 identified classes, 26 actually contained parts of the 14 changes.
- The other 7 classes were simple refactorings.
- Achieved Performance Measures:
 - i. Recall : $12 / 14 = 85.7\%$.
 - ii. Precision : $26 / 33 = 79\%$

Case Study - iTrust Health care project

Study 2 : Evaluate how well the approach can identify affected requirements, i.e. use cases in this case study, based on the extracted keywords.

- Comparison with simple class-based tracing approach.
- 2 problems where change-based approach performed worst.

Use of abbreviations

	Id	Class	Affected Use Cases	Change-based Tracing	Standard Class-Based Tracing
	1	ActivityFeedAction	NEW		
	2	AddRemoteMonitoringDataAction	UC34	<u>1</u>	2
	3	ApptDAO	UC22	29	1
	4	DAOFactory	UC15, UC4	5, 30	21, 34
	5	EditApptAction	UC22	3	3

"Bean" were only called by jsp classes

	13	OverrideReasonBeanValidator	UC15	23	34
	14	PrescriptionBean	UC37, UC15	21, 1	16, 12

Conclusion

It is a new approach for identifying outdated requirements based on analysis of code changes.

Automatically identifying outdated requirements :

- Reduces the efforts and time needed for the maintenance of requirements specifications significantly .
- It helps preserve the knowledge contained in them.
- It might encourage engineers to maintain the requirements after each code release.