

Requirement based Change Impact Analysis during Whole Lifecycle: a Case Study

Yin Li, Juan Li, Ye Yang and Mingshu Li

*Laboratory for Internet Software Technologies
Institute of Software
Chinese Academy of Sciences*

Agenda

- Introduction
- The Model and Application of Requirement Centric Traceability (RCT)
- Quantitative Change Impact Analysis
- Conclusions and Future Work

Agenda

- *Introduction*
- The Model and Application of Requirement Centric Traceability (RCT)
- Quantitative Change Impact Analysis
- Conclusions and Future Work

Introduction (1/3)

- Requirement are Volatile
 - Business change
 - Technology evolution
- Change Impact Analysis (CIA)
 - the activity of identifying the potential consequences
 - estimating what needs to be modified to accomplish a change before it has been made

Introduction (2/3)

- CIA from the view of maintenance
 - Analyzing the change impact on the source code level
 - Helpful to the programmer to catch the influencing details
 - Difficult for high level decision makers to understand the impact and make decision
 - Supposing the change has taken place in code and Ignoring the change in the requirement

Introduction (3/3)

- Our solution
 - Proposing requirement centric traceability to estimate the impact of change in requirement level
 - Using requirement interdependency to model relation among requirements.
 - Using dynamic requirement traceability to model the trace between requirement and artifact.
 - The quantitative impact synthesizes requirement interdependency and traceability matrix.

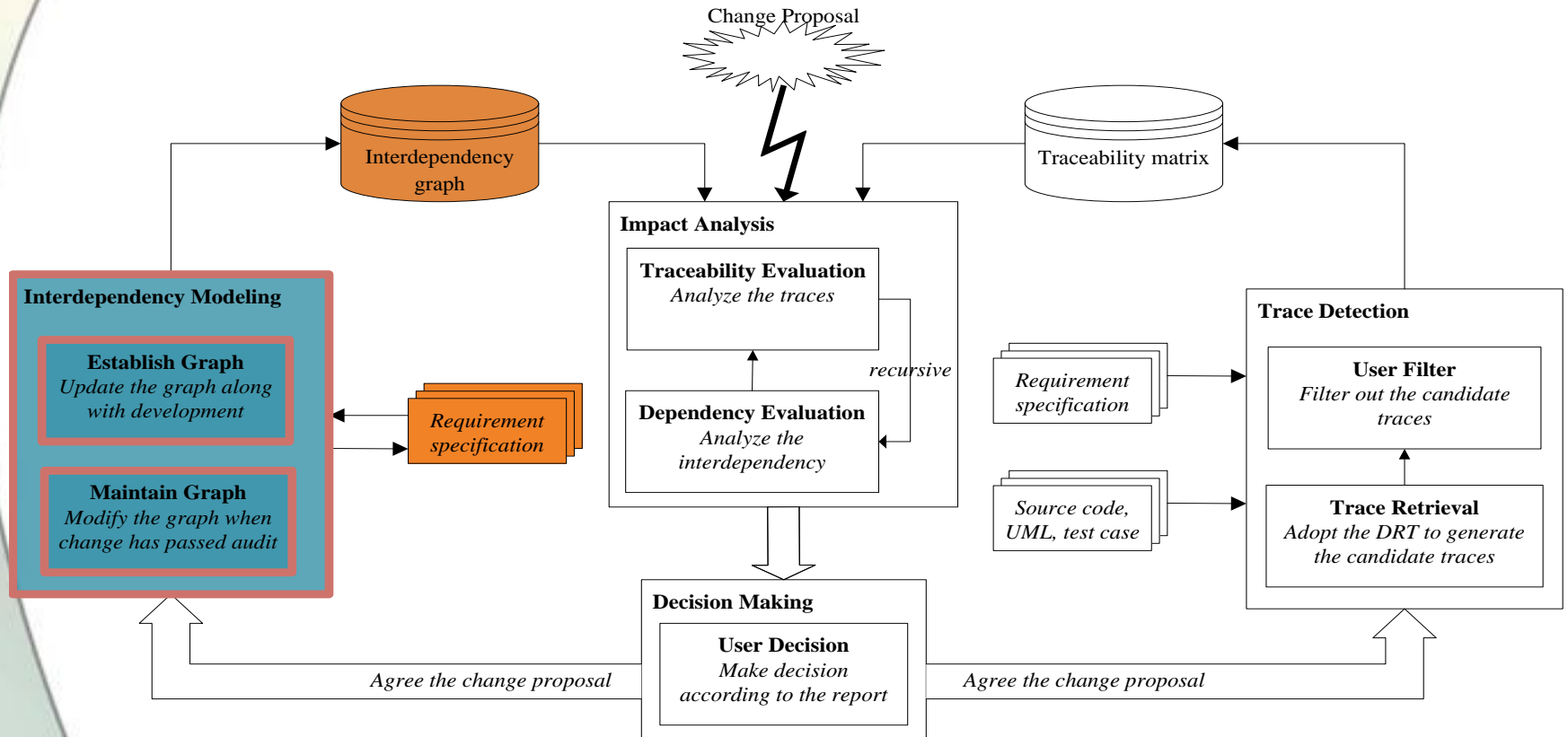
Agenda

- Introduction
- *The Model and Application of Requirement Centric Traceability (RCT)*
- Quantitative Change Impact Analysis
- Conclusions and Future Work

The Process of RCT



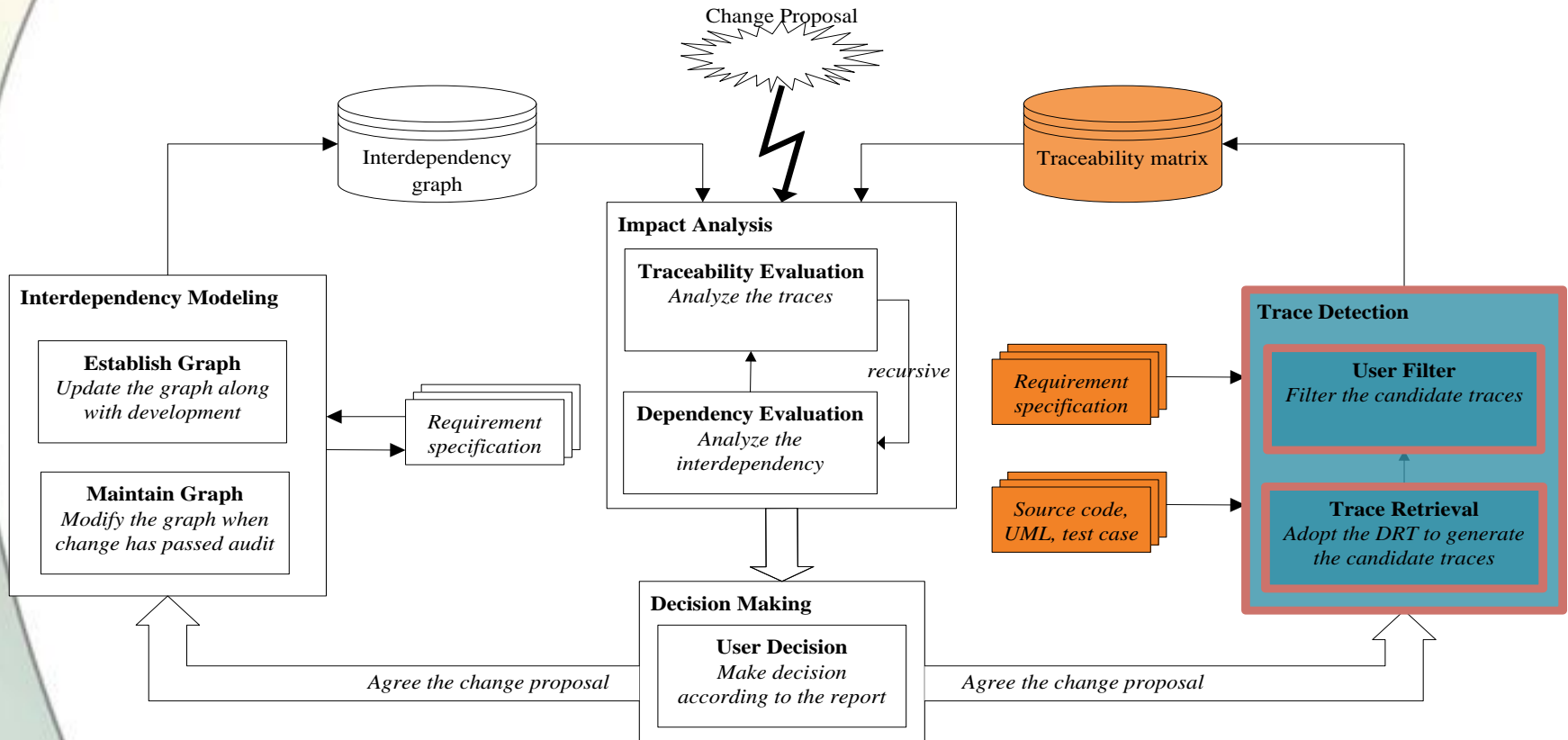
Phase 1: Interdependency Modeling (1/2)



Phase 1 (2/2)

- It's used to construct and maintain the relations among requirements
- Three kinds of change
 - Addition
 - Modification
 - Deletion
- Corresponding interdependency types
 - *Conflict_with* for Addition
 - *Requires* and *Explains* for Modification and Deletion

Phase 2: Trace Detection (1/3)



Phase 2 (2/3)

- Dynamic Requirement Traceability (DRT)
 - Based on Information Retrieval (IR)*
 - Establishing the trace between requirement and artifact like searching the internet using Google
 - The requirement is the query and the artifact is the document.
- Trace identification
 - Candidate traces between requirements and artifacts are identified automatically
 - Based on Vector Space IR model to compute the similarity between requirement and artifact
 - Ranking the searched artifacts according to the similarity
 - Analyst inspects the candidate artifacts

* *Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Pearson Education(1999)*

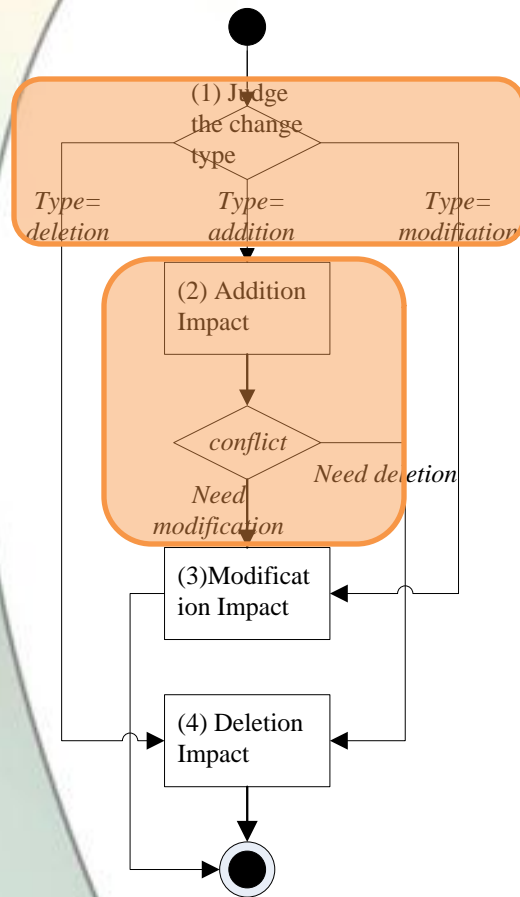
Phase 2 (3/3)

- Establish trace between Chinese requirement and English code
 - Translate the Chinese requirement to English automatically using Google Translate*
 - Match synonyms between requirement and code using WordNet**

* Translation Service provided by Google. http://www.google.com/translate_t

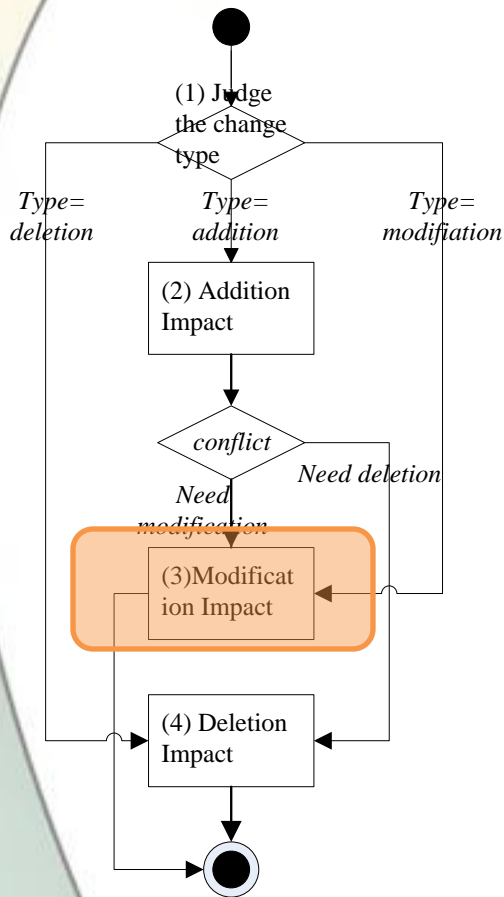
** Cognitive English dictionary. <http://wordnet.princeton.edu/>

Phase 3 Impact Algorithm(2/5)



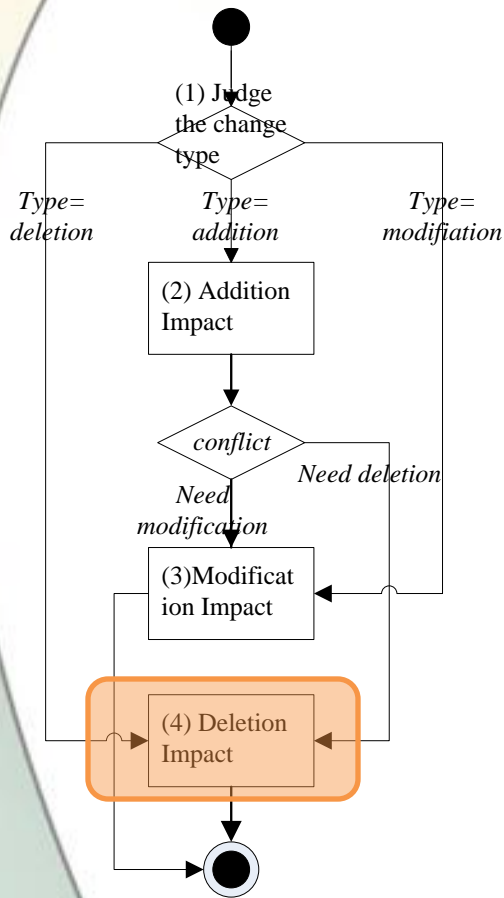
- (1) Judge the change type, if type = *'addition'*, goto (2);
 else if type = *'modification'*, goto (3);
 else if type = *'deletion'*, goto (4);
- (2) If *newReq* conflicts with existed req: *ExistedReqList*
 for each *req* in *ExistedReqList*
 if *req* needs to do some modification, goto (3);
 else if *req* needs to be deleted, goto (4);

Phase 3 Impact Algorithm(3/5)



- (3) For each dependent *dreq* of *modifyReq*
 - get *dreq*'s traces to artifacts into *IA*;
 - revise *IA* to *IA**; //eliminate repeated change impact
 - calculate $number(IA^*) * w_{modify} * df(i)$ into *cost*, // *i* is set to 1 initially
 - add *cost* into *cc*;
 - set *modifyReq* = *dreq*, *i*=*i*+ 1 and goto 3;

Phase 3 Impact Algorithm(4/5)



- (4) For each dependent *dreq* of *deleteReq*
 - get *dreq*'s traces to artifacts into *IA*;
 - revise *IA* to *IA**; //eliminate repeated change impact
 - calculate $number(IA^*) * w_{delete} * df(i)$ into *cost*, // *i* is set to 1 initially
 - add *cost* into *cc*;
 - set *deleteReq* = *dreq*, *i*=*i*+1 and goto 4;

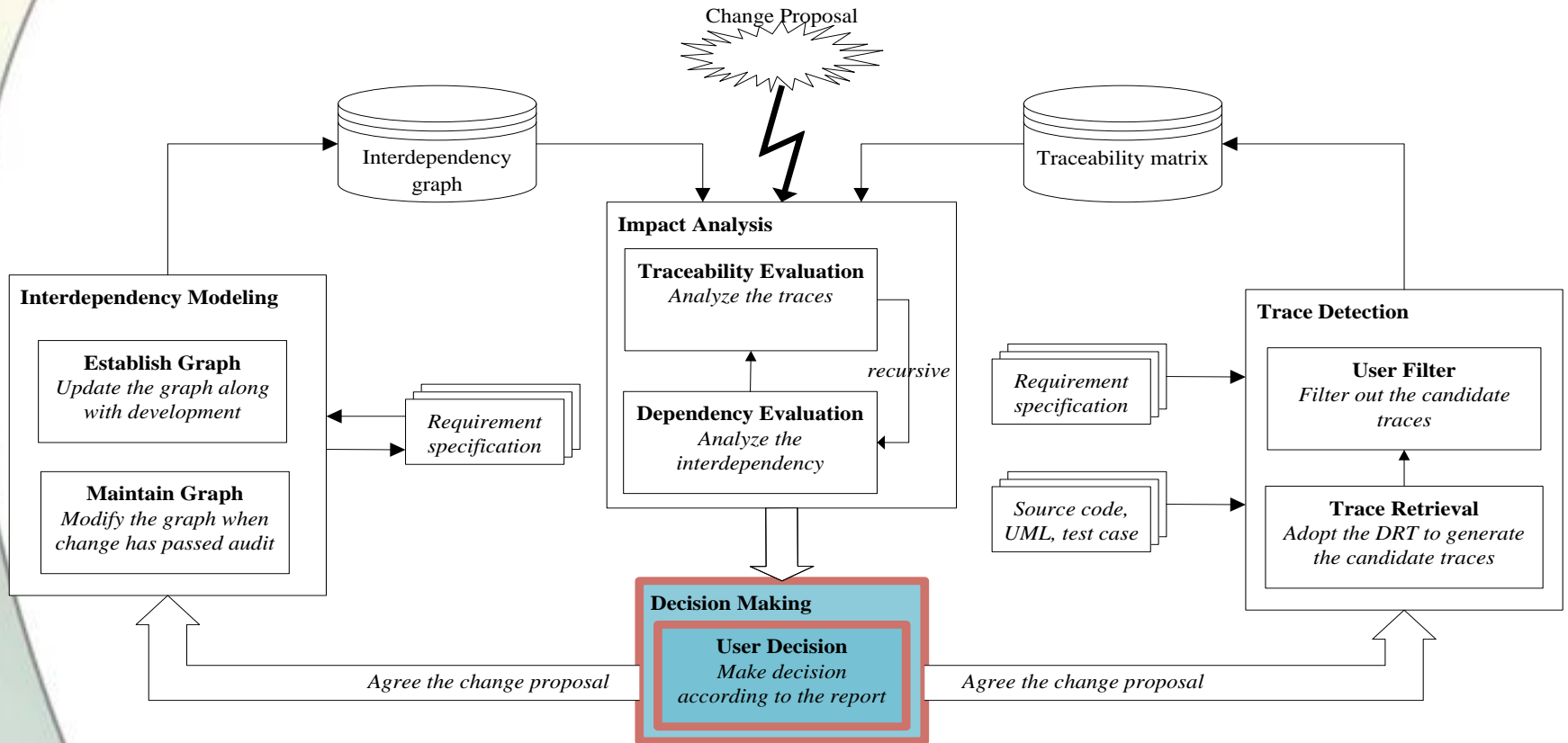
Phase 3 Impact Algorithm(5/5)

- total change impact = sum of the impact of every affected requirement.
- The impact of affected requirement is the product of three parts: the number of current traces, the change type weight and depression factor. Which is

$$\mathbf{number(IA^*)} \times \mathbf{w_{[change\ type]}} \times \mathbf{df(i)}$$

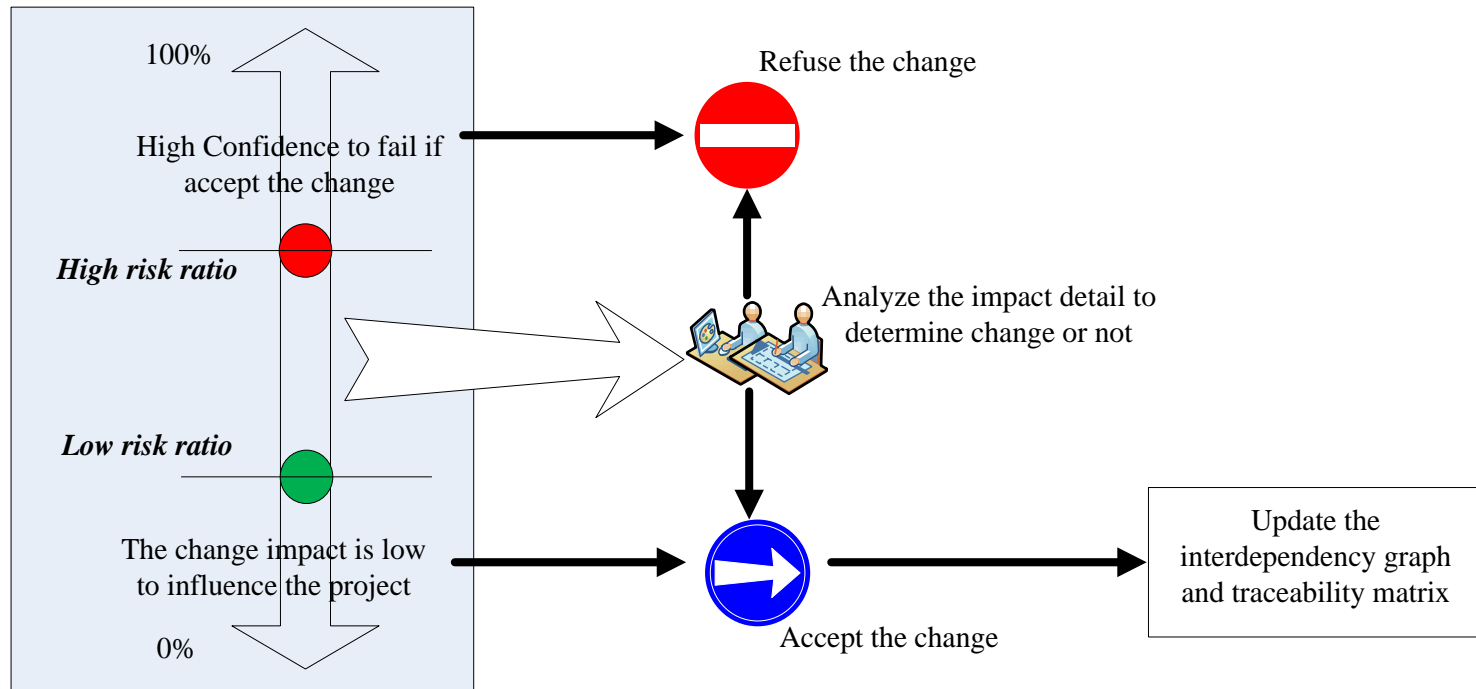
- **$number(IA^*)$** : eliminating the repeatedly calculated impact
- **$w_{[change\ type]}$** : $w_{[modification]}$ is set to 0.2, and $w_{[deletion]}$ is set to 1.
- **$df(i)$** : the depression of impact through interdependency graph, $\frac{1}{2^{i-1}}$

Phase 4: Decision Making (1/2)



Phase 4 (2/2)

- the ratio of the number of affected artifacts to the number of total artifacts



Agenda

- Introduction
- The Model and Application of Requirement Centric Traceability (RCT)
- *Quantitative Change Impact Analysis*
- Conclusions and Future Work

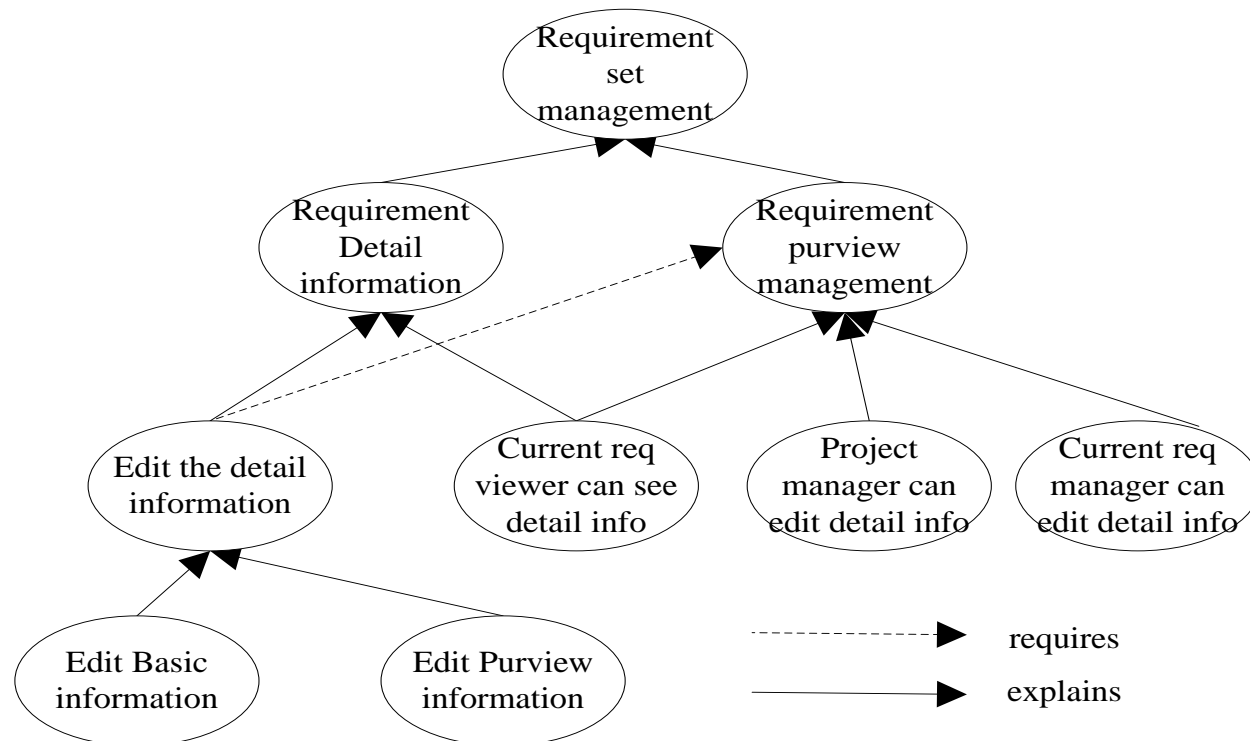
Data Set Used

- Data from a software project which was developing a *Requirement Management System (RM)*
- Project Context

Project Characteristics	Description
Project type	Web based application
Development process	iterative
Development tool	Java/applet/struts/jsp/ajax, mysql, tomcat4
Team size	5
Developer skill	experience
Project duration	30 weeks
Project scale	43 requirements, 70 KLOC, 468 classes
Logical module number	7
Deployment package	pmreq.jar, pmapplet.jar, pmwss.jar
Change proposal time	10
Project status	Successfully completed on time

Construct Requirement Interdependency (2/2)

- A part of interdependency graph in module *Requirement Tree Management*



Establish the Requirement Traceability Automatically (1/3)

- Three metrics for the result generated automatically
 - **Precision** = No. of relevant documents retrieved / No. of documents retrieved
 - **Recall** = No. of relevant documents retrieved / No. of relevant documents
 - **F-measure** =
$$f = \frac{1+b^2}{\frac{b^2}{recall} + \frac{1}{precision}}$$

b represents the relative importance between recall and precision. $b < 1$, $b = 1$ and $b > 1$ mean precision more important, equal important, recall more important respectively. In our case study, we set $b = 3$, which means recall is more important than precision.

Establish the Requirement Traceability Automatically (2/3)

- Dynamic retrieval result on all data set
 - The traces from requirement to the code were generated in three parts, for the implementation was deployed into P1, P2 and P3.
 - The F-measure of three packages reaches its maximum in *chosen set*

Set No.	Java Package	Threshold value	All correct No.	Retrieved No.	Correct retrieved No.	Precision (%)	Recall (%)	F-measure b=3.0
1	P1	0.15	295	1143	218	20.7	80.3	0.624
	P2	0.14	600	2423	563	23.2	93.8	0.72
	P3	0.1	198	735	187	25.4	94.4	0.743
2	P1	0.18	295	946	226	23.9	76.6	0.628
	P2	0.16	600	2123	556	26.2	92.7	0.739
	P3	0.12	198	651	182	27.9	91.9	0.748
<i>Chosen Set</i>	P1	0.2	295	815	218	26.7	73.9	0.629
	P2-full	0.18	600	1889	552	29.3	92	0.757
	P2-part	0.15	600	1363	556	40.8	92.7	0.821
	P3	0.14	198	578	179	30.1	90.4	0.758
4	P1	0.23	295	677	204	30.1	69.1	0.614
	P2	0.2	600	1681	534	31.8	89	0.754
	P3	0.16	198	500	171	34.2	86.3	0.749
5	P1	0.25	295	595	197	33.1	66.8	0.606
	P2	0.22	600	1517	518	34.1	86.3	0.749
	P3	0.18	198	439	159	36.2	80.3	0.715

P1:
pmreq.jar
P2:
pmapplet.jar
P3:
pmwss.jar

Establish the Requirement Traceability Automatically (3/3)

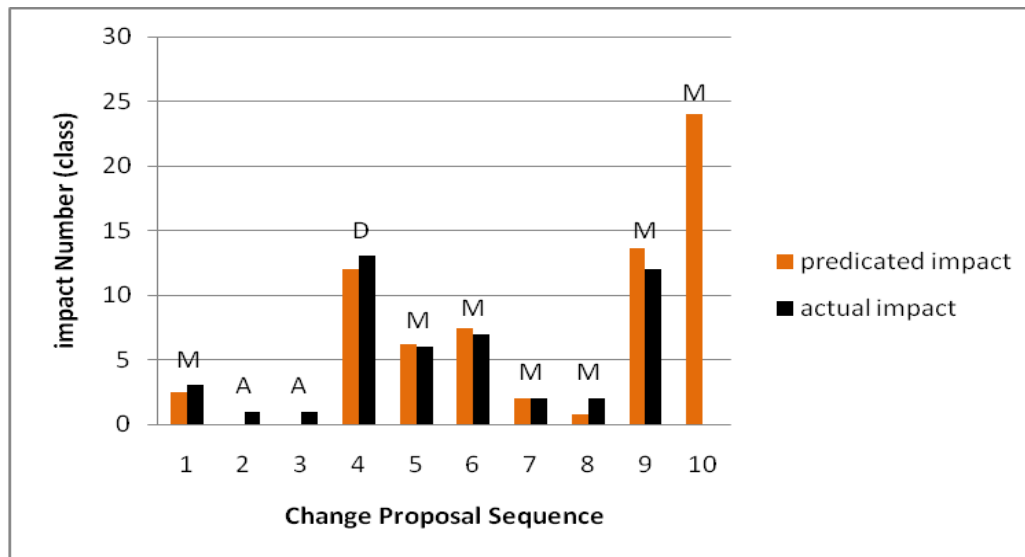
- Three findings
 - For different java package, the precision and recall are not the same. The recall and precision in P2 and P3 are higher than those in P1.
 - The precision in P2-full is 10 percentage points lower than in P2-part.
 - P3 has the highest f-measure
- It's time-saving to identify the traces by Dynamic Requirement Traceability, because the analyst only needs to inspect the 745 potential traces rather than 20,124 potential traces manually.

P1:
pmreq.jar
P2:
pmapplet.jar
P3:
pmwss.jar

Set No.	Java Package	Threshold value	All correct No.	Retrieved No.	Correct retrieved No.	Precision (%)	Recall (%)	F-measure b=3.0
	P1	0.2	295	815	218	26.7	73.9	0.629
<i>Chosen</i>	P2-full	0.18	600	1889	552	29.3	92	0.751
<i>Set</i>	P2-part	0.15	600	1363	556	40.8	92.7	0.821
	P3	0.14	198	578	179	30.1	90.4	0.758

Change Impact Analysis

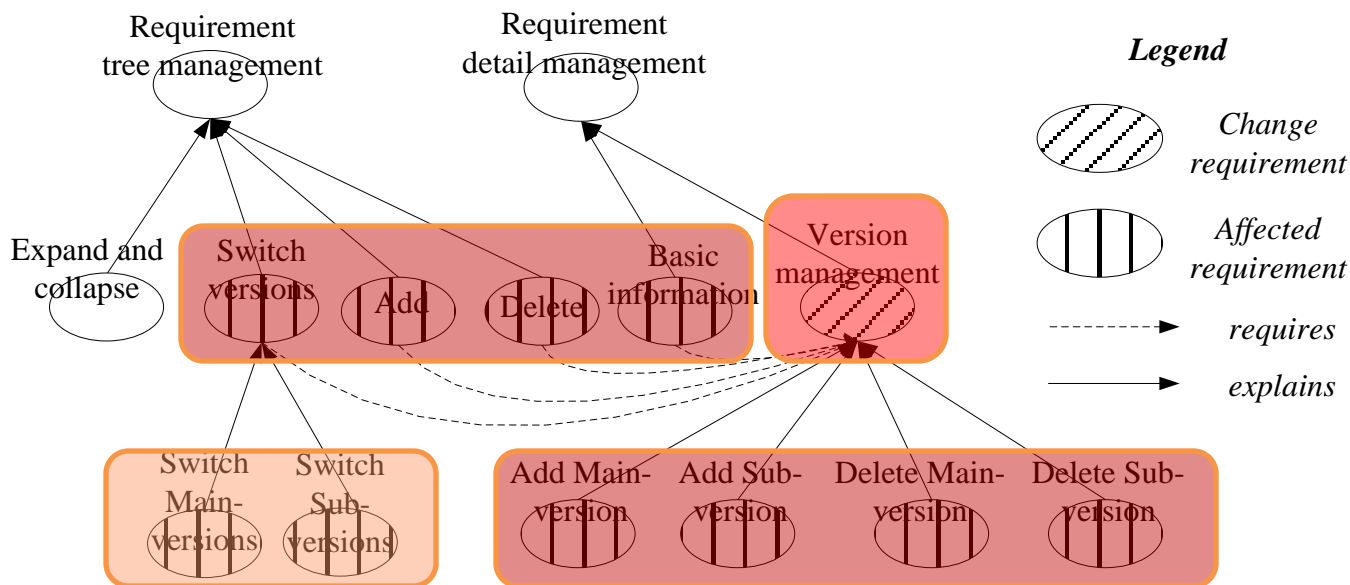
- Change impact during development
 - A., D. and M. represent the change type Addition, Deletion and Modification



- Accepted first 9 change proposals and refused last change proposal

Change Proposal #10: Modify *Version Management* (1/3)

- The impact of modifying *Version Management*



Change Proposal #10: Modify *Version Management (2/3)*

- Dynamic retrieval on changing version mechanism
 - These requirements have no implementation in P3, so the data is collected from P1 and P2

Affected requirement	Correct No.	Precision(%)	Recall(%)
		P2 (threshold=0.15) / P1 (threshold=0.2)	
version management	39/5	35.5/16.1	100/100
add main-version	33/4	45.2/26.7	100/100
add sub-version	30/4	42.9/25	100/100
del main-version	23/4	41.1/21.1	100/100
del sub-version	24/4	40/23.5	100/100
basic information	53/2	61.6/18.4	88.3/87.5
add requirement	32/11	41/53.3	97/72.7
del requirement	11/8	40.7/50	55/62.5
switch versions	30/6	29.4/30	96.8/100
switch main-versions	26/5	37.7/21.7	96.3/100
switch sub-versions	32/4	43.8/21	94.1/100

Change Proposal #10: Modify *Version Management* (3/3)

- The calculated impact of this change is 24 (classes), and the ratio of the impact number to the total class number (468 classes) is 5.1%, lower than high risk value but higher than low risk value.
- The affected classes were analyzed in detail, and we found that these classes are too complex to be modified. So PM decided to delay the modification to the next version

Discussion

- Benefits

- Addressing the change on the requirement specification rather than code, which can help the non-technical people to predict the change cost quantitatively
- The impact to both requirements and artifacts was unified into the number of artifacts
- it's time-saving to apply the IR to automatically establish the traces among the requirements and artifacts, and is highly probable to adopt in practice

Agenda

- Introduction
- The Model and Application of Requirement Centric Traceability (RCT)
- Quantitative Change Impact Analysis
- Related Work
- *Conclusions and Future Work*

Conclusion

- Requirement-Centric Traceability
 - requirement interdependency analysis
 - the three change types
 - dynamic requirement traceability
 - the candidate traces from requirements to artifacts are generated automatically by IR method
 - the actual traces can be gained by user inspection
 - The change impact is synthesized from interdependency graph and traceability matrix
- Application
 - applied the approach to our 5-people and 30-week project

Future Work

- Establishing the interdependency graph automatically
- Improving the accuracy of traces

Thank You!