

Using a Hybrid Method for Formalizing Informal Stakeholder Inputs

Hasan Kitapci

University of Southern California

September 11, 2006

Introduction

- **Requirements related problems have a high impact on projects failure**
 - **Incomplete requirements**
 - **Lack of user involvement**
 - **Unrealistic expectations**
 - **Changing requirements and specifications**
- **Requirements related defects cost more than others when not removed as early as possible in the software development life-cycle**
- **Return on Investment for good requirements is higher than the other software artifacts**
 - **Value-prioritized, feasible, unambiguous**

Avoiding Requirements-Related Failures

- **Make it easier for users to participate in Requirements Specification**
 - Prototypes, brainstorming, negotiation, groupware
 - Helps getting good requirements, clarifying shared vision, managing expectations
 - Produces informal statements

Easy Win Win Results Efficient but Informal

- **Stakeholders converge on elements of requirements in two 2-hour sessions**
 - Interspersed with prototyping
- **Clients generally prefer to stop with informal agreement statements**
 - *“Server should provide file management, user interface functions”*
 - *“Users should be able to upload, share, and download files”*
- **But developers (and clients) need more precise statements of requirements for products to satisfy**
 - *“The system shall have an upload functions for file creation, organization, entry, import, update, validation, access control, query, export, and deletion”*
 - *“The system shall allow sharing users files with configurable permissions”*

Research Question

Informal
Stakeholder
Requirements
Inputs

Formalized
Requirements
Specification

“How to orchestrate gap-bridging techniques to provide sufficiently complete, consistent, unambiguous, and testable software requirements”

Requirements Representation Methods

- **Scaleable techniques to address size and complexity**
- **Techniques helping to identify and remove defects**
- **Four primary categories of formality:**
 2. ***Formal specifications:*** mathematically precise specifications with unambiguous semantics
 3. ***Formatted specifications:*** syntax is often precisely defined, but use imprecise terms and relations
 4. ***Itemized statements:*** natural language specifications with unique identifiers
 5. ***Stories:*** colloquial descriptions of desired capabilities
- **No method dominates the other on all the criteria**
- **The strongest method across all five criteria:
formatted specifications**

Formalizing Approaches to Bridge the Gap

1. Human-initiated Template

- System analyst creates requirements manually

2. Computer-initiated Template

- A computer program produce draft requirements template
- Ensure inspection fixes yield formatted specs quickly

3. Natural Language Processing

- Extract template-relevant meaning
- Domain-specific model checking in mature and focused domains (e.g., telecommunication or aircraft control)

Formalizing Approaches to Bridge the Gap (cont.)

1. Keyword Analysis

- Produce summary information to facilitate specs
- Generate useful subsidiary artifacts (e.g., common terms)

2. Formal Methods Expertise

- Refine the appropriate template elements into formal specs
- Good for specs needing high precision

3. Inspections

- Inspect statements and post-process them into more precise specs
- Find defects early and avoid misinterpretations
- Reduce rework

Comparison Criteria

- **Required human resources**
- **Available technology**
- **Generality** – general applicability in different domain
- **Scalability** – well suited for small and large projects
- **Precision** – provide clear, unambiguous information

Gap-Bridging Approaches

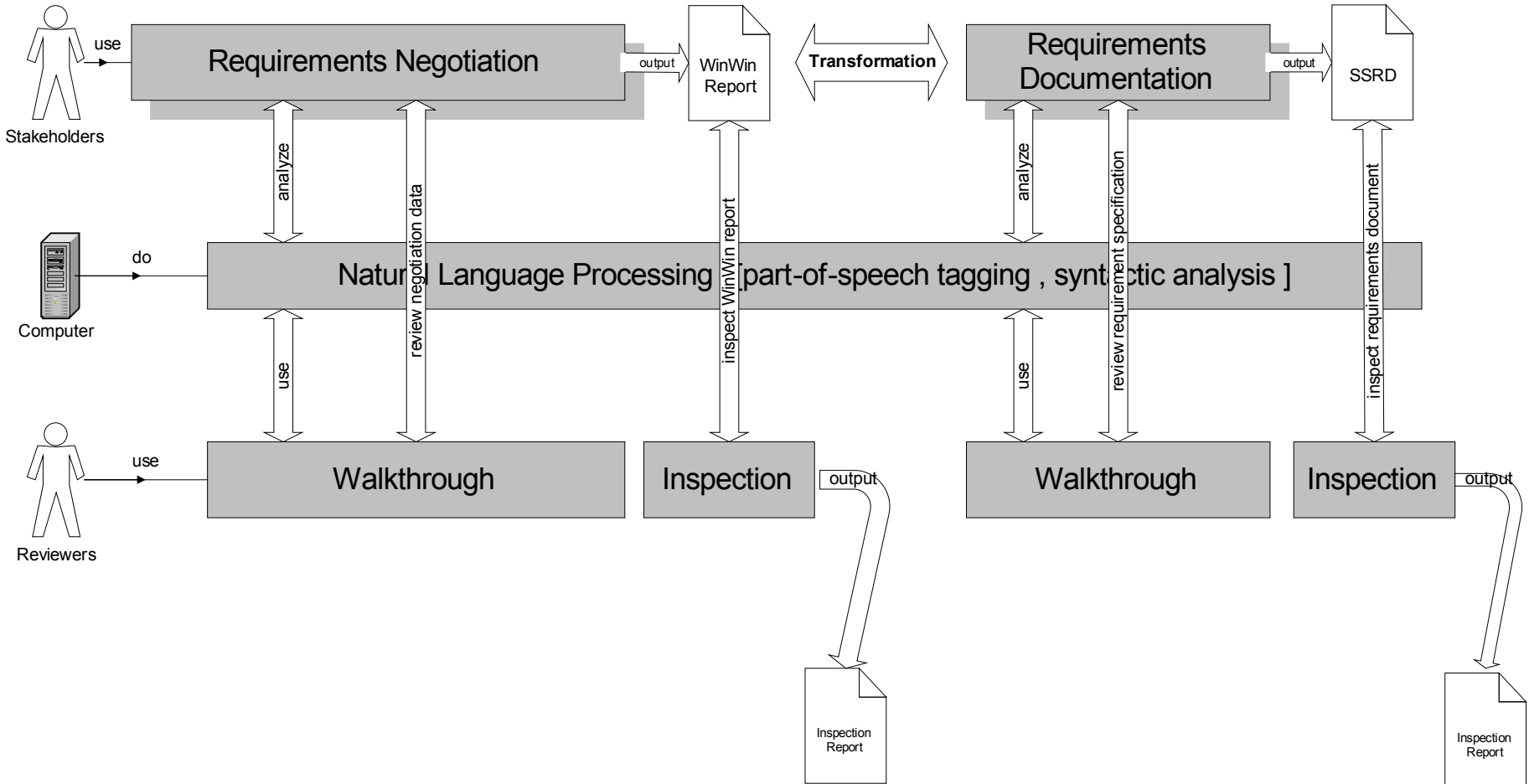
	Required Human Resources	Available Technology	Generality	Scalability	Precision
<i>Human-Initiated Template</i>	**	****	****	**	**
<i>Computer-Initiated Template</i>	** - ****	* - **	* - **	* - **	* - **
<i>Natural Language Processing</i>	** - ****	* - **	* - **	*	* - **
<i>Keyword Analysis</i>	**	****	***	* - ****	* - **
<i>Formal Method Experts</i>	*	**	***	**	****
<i>Inspections</i>	**	****	****	**	***

- **Ratings:** 0-weak, 1-modest, 2-intermediate, 3-strong, 4-very strong
- No approach dominates all of the other approaches
- Inspections and human-initiated template fairly strong
- Hybrid combinations of the approaches attractive

Some Alternative Hybrid Methods

- **Human-initiated template with inspections**
 - Early defect detection
 - Reduce work
- **Human-initiated template with formal method experts**
 - Time-consuming
 - Formal methods not widely used for all kinds of project
- **Computer-initiated template with NLP**
 - Produce template from EasyWinWin output using NLP techniques
 - Reduce size of human resources
- **Computer-initiated template with Formal method expert**
 - Produce template from EasyWinWin output
 - Formalize some type of requirements (e.g., capability requirements)

My Hybrid Method



Mixed-initiative Template

- **Generate report from the Negotiation tool**
 - Use EasyWinWin output to fill requirements templates
 - Map WinWin artifacts into Requirements template attributes
 - Final Output generated as a requirements specification document
- **Modify/Update the requirement templates**
 - Use extracted data to fill attributes in MARS (Measurable, Achievable, Relevant and Specific) or Use Case forms
- **Customizable requirements templates and attributes**

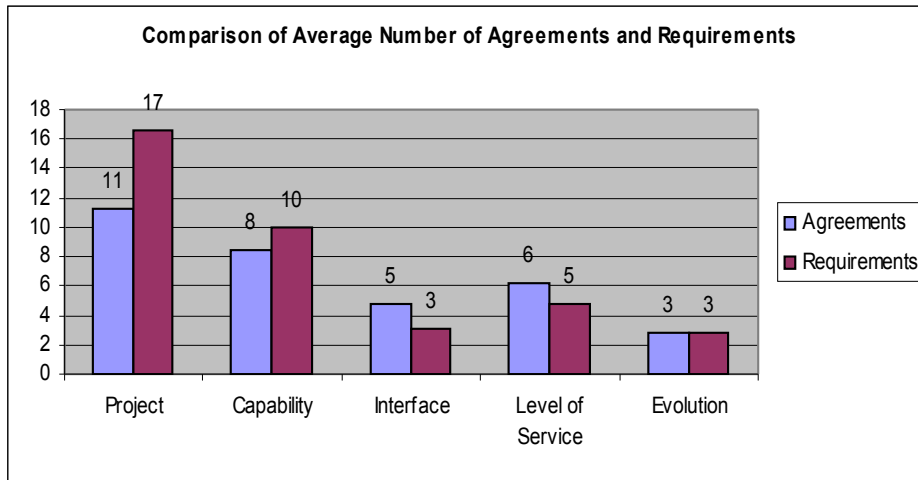
Natural Language Processing & Keyword Analysis

- **Defect identification**
 - Analysis to find NL defects
 - Use indicators to find problematic statement
 - Feedback to stakeholders for further correction
- **Tailorable categories and checking criteria**
- **Extract template-relevant meanings**
 - Extract verbs and nouns, priority, reference, etc.
- **Feedback to stakeholders for further correction**

Inspections

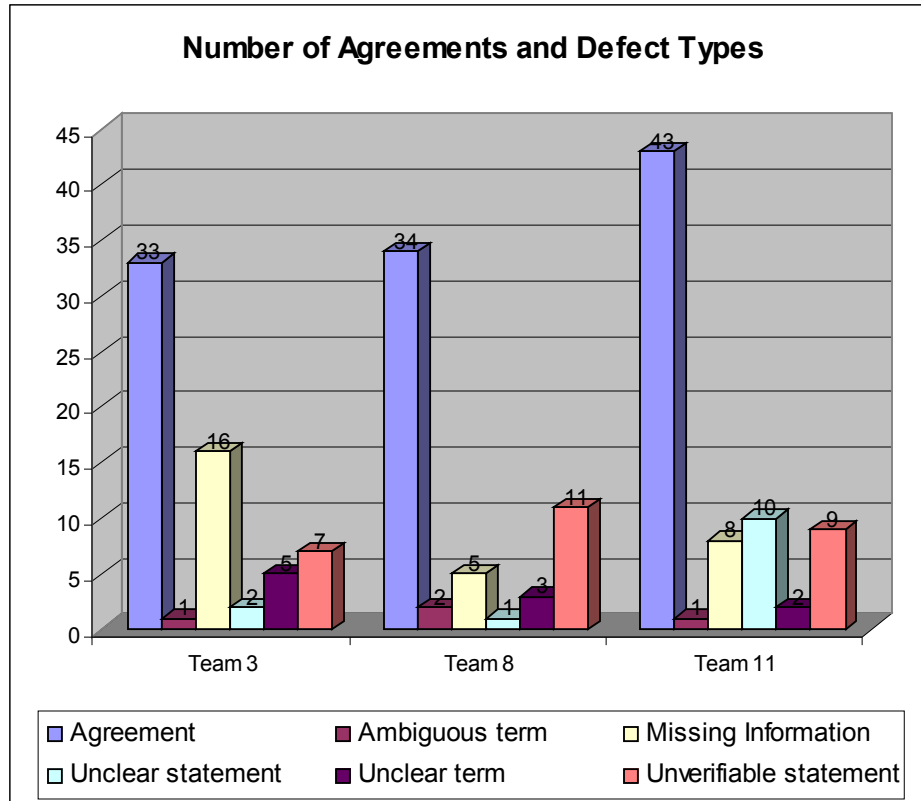
- **Inspect results and post-process them into precise specifications**
 - Use walkthroughs during the process (statement-level checking and requirement-level checking)
 - Use inspections on the outputs (document-level checking)
 - Find ambiguities, conflicting requirements, and other defects as early as possible
 - Avoid misinterpretations of WinWin artifacts

Analysis of Current Shortfalls



- Agreements not turned into requirements
- Requirements without trace to agreements
- Wrong categorization
- Compound agreements
- Multiple topics tagging
- Dependencies valuable during documentation
- Missing tags for WinWin artifacts

Defect Analysis



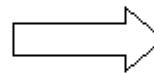
Frequent defect types are missing information and unverifiable statement

- Agreement: *Searchable archive based on predefined criteria*
 - **Unverifiable statement**: indicator “predefined”
 - Found by speech tag: adjective
- Agreement: *User authentication*
 - **Unclear statement**: indicator “not a sentence”
 - Found by speech tag: no subject and verb phrases
- Agreement: *Search by specific fields e.g. title, author*
 - **Missing information**: indicator “no subject”
 - Found by speech tag: start with verb
- Agreement: *More time is required to integrate with Z-bit or keep it as low priority*
 - **Unclear term**: indicator “Z-bit”
 - Found by speech tag: proper noun

Example Improvement: Initial Template Generation

W9 LHF The server should have file management and user management function. (Taxonomy 2.1.2, 2.1.3.1)
A9 The server should have **file management** and **user management** function. [DEVELOPER, CLIENT, ARCHITECH, and USER: 9/26/01] (Taxonomy 2.1.2, 2.1.3.1)(W9)

W10 LHF The users have to be able to upload files, share files, within the group. (Taxonomy 2.1.2.2)
A10 The **users** have to be able to **upload files, share files, within the group**. [DEVELOPER, CLIENT, ARCHITECH, and USER: 9/26/01] (Taxonomy 2.1.2.2)(W10)



Requirement:	RQ-2
Title:	Users can upload/modify/download/delete authorized files
Priority:	Very High
Description:	File management, as a part of the system, will improve information sharing and accessing. This will help users to upload/modify/download/delete files needed in their project.
Input(s):	User's request (click the files button)
Source(s):	User
Output(s):	Upload – the new files Modify – the modified files Download – the receive files in the user's own storage Delete – the file disappears in the project's storage
Destination(s):	Given storage of project group and user's own storage.
Precondition(s):	User know the files wanted to manipulate
Post condition(s):	User manipulates files
Proposed Activity:	File sharing and management as part of the collaboration services
Win Win Agreement(s):	A9, A10
Mainstream Scenario:	1. Upload files needed in their project; 2. Modify files needed in their project; 3. Download files needed in their project; 4. Delete files needed in their project.
Exception Handling Scenario:	If current user has no right to modify the file he is requiring, the system will give an warning information to show that he does not have enough right to do this operation.
Reference:	OCD 4.3 CAP-02

THANK YOU
&
Questions