

# TOOL SUPPORT FOR ESSENTIAL USE CASES TO BETTER CAPTURE SOFTWARE REQUIREMENTS

**<sup>1</sup>MASSILA KAMALRUDIN, <sup>2</sup>PROF JOHN GRUNDY, <sup>3</sup>PROF JOHN HOSKING**

<sup>1</sup>Department of Electrical & Computer Engineering, <sup>3</sup>Department of Computer Science,

University of Auckland, New Zealand

<sup>2</sup>Centre for Complex Software, Systems & Services, Swinburne University of Technology



**THE UNIVERSITY OF AUCKLAND**  
**NEW ZEALAND**



# INTRODUCTION

- Natural language is commonly used to capture software requirements
- Natural language is a human-centric representation for clients and requirements engineers
- The process of capturing requirements and the inherent ambiguities and complexities of natural language leads to major problems including
  - **Inconsistency**
  - **redundancy,**
  - **Incompleteness**
  - **Omissions**

# MOTIVATION

- Modelling requirements
  - Want to represent (semi-)formally the requirements
  - This allows for better checking & analysis than Natural Language alone
- Common requirements models:
  - UML use cases : capture functional requirements mostly informally
    - Strengths → can be shown to end-users, widely used
    - Limitations → time-consuming to build and leads to imprecise analysis
  - KAOS : capture formally
    - Strengths → formal model, can analyse deeply
    - Limitations → challenging for end-users and complex
  - Essential Use Cases (Constantine and Lockwood, 1999): integrate the requirement engineering and interaction design process.
    - Strengths → more formal than UML use cases, can do deeper analysis
    - Limitations :
      - Lack of tool support
      - lack of experience in extracting essential interaction from requirements
      - Lack of integration with other modelling languages

# ESSENTIAL USE CASES (EUC)

“Structured narrative, expressed in a language of the application domain and of users, comprising a simplified, generalized, abstract, technology free and independent description of one task or interaction that is complete, meaningful, and well-defined from the point of view of users in some role or roles in relation to a system and that embodies the purpose or intentions underlying the interaction” (Constantine, 1995).

Specifies a sequence of abstract steps and captures the core part of a requirement.

Shorter and simpler than conventional use cases, and is in the form of a dialogue between the user and system.

Documentation of the interaction without the need to describe the user interface in detail.

Contains User Intentions and System Responsibilities

*\*Responsibility: “what the system must do to support the use case”*

# CAPTURING REQUIREMENTS WITH ESSENTIAL USE CASES (EUCS)

The use case begins when the customer goes to the Customer Log-on page. There, the customer <sup>1</sup>types in his/her name and customer ID on the form and submits it. The system then <sup>2</sup>displays the Tech Support home page with a list of Problem Categories. The customer <sup>3</sup>clicks on installation help within the list, and the system <sup>4</sup>supplies the Incident Report Form. The customer <sup>5</sup>completes and submits the form, and the system <sup>6</sup>presents a suggested resolution.

Essential interaction

User intention	System responsibility
1. Identify self	2. Present help options
3. Select help option	4. Request description
5. Describe problem	6. C

Essential requirement (Abstract interaction)

# PRELIMINARY USER STUDY

Candidate	Answers												Time taken (minutes)
	Identify user		Verify identity		Offer choices		Choose		Dispense cash		Take cash		
1		x		x		x	Y		Y		Y		9
2	Y			x	Y		Y		Y			x	5
3		x		x	y			x	Y			x	10
4		x		x		x	Y		Y			x	7
5		x	Y			x		x	Y			x	10
6		x		x		x	Y		Y			x	7
7	Y		Y		Y		Y		Y		Y		20
8	Y			x		x	Y		Y			x	10
9	Y		Y		Y			x		x		x	10
10.		x		x		x		x	Y			x	25
11.	Y		Y			x		x		x	Y		10
	5	6	4	7	4	7	6	5	9	2	3	8	123
Average time: 123/11=11.2													

Study result of Essential Use Case practice on "Getting Cash" scenario: Correctness and time

- 53% of individual abstract interactions were incorrect
- Only 1 EUC was completely correct
- The **average time taken** to accomplish the EUC development task was **11.2 minutes**.
- The **longest time taken** was ~ **25 minutes** and the shortest ~ 5 minutes
  - significant variability.
  - tended to determine incorrect level of abstraction for their essential interaction
  - time consuming: need to figure out appropriate keyword for abstract interaction

# OUR APPROACH

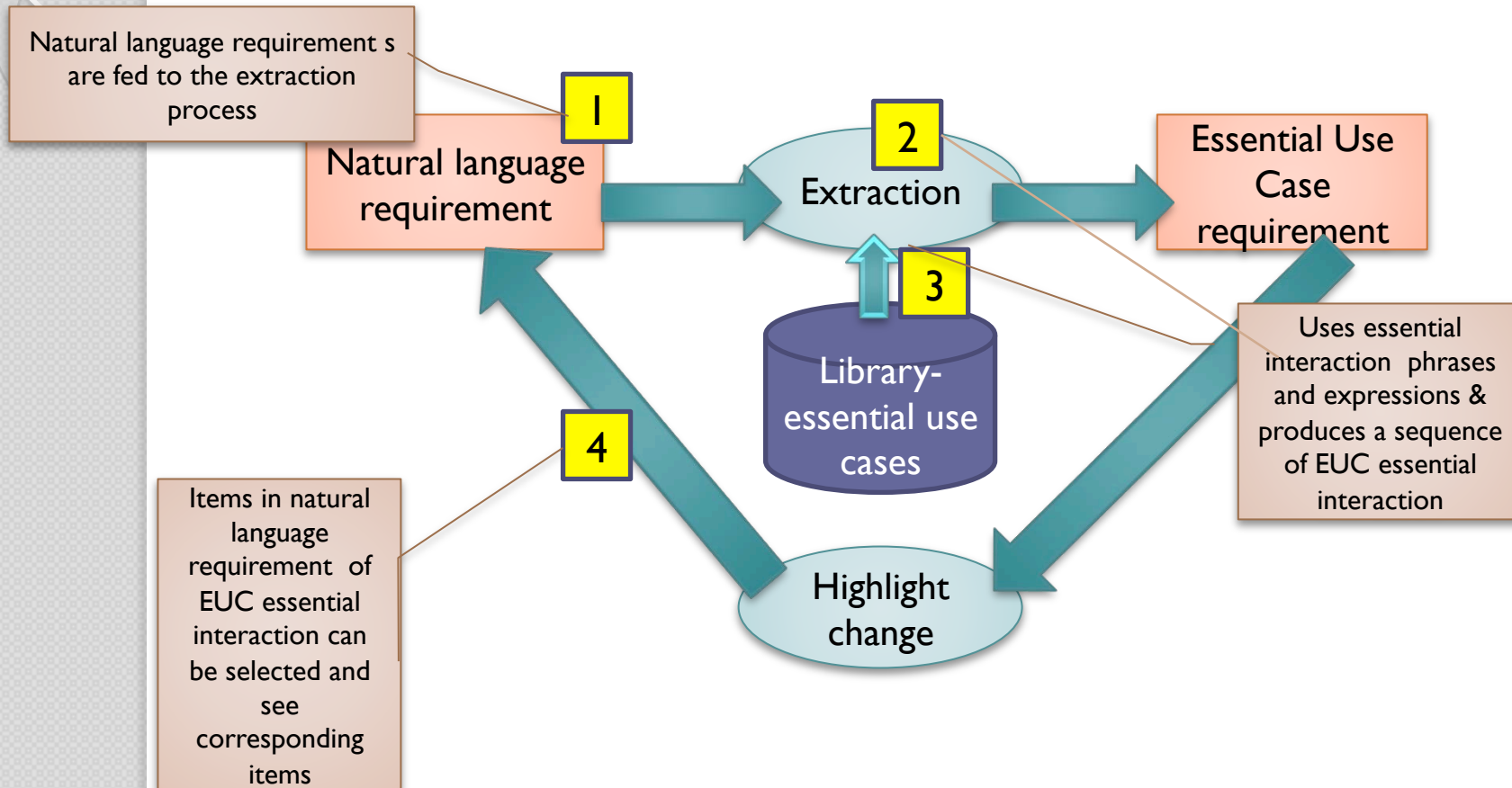
- Lightweight tool support vs heavy weight NL processing
- Domain specific – use knowledge of EUC domain
- Develop a library of “proven” essential interactions → textual phrases, phrase variants and limited regular expressions
  - Enables extraction of EUCs from NL textual requirements
- Library of *abstract interaction patterns*
  - collection of patterns developed by Constantine and Lockwood, Biddle et al. and us.
  - applicable across various domains
  - Enables deeper analysis of extracted requirements

# HOW DOES THE ABSTRACTION OF EUCs WORK?

- Each essential interaction pattern is:
  - associated with a collection of alternative sequences of textual requirement phrases that could match to the pattern
  - Each sequences relates to a more concrete version of the abstract interaction pattern
  - Textual natural language requirements were analyzed → match against the concrete versions and look for the best match
- Abstraction → instantiating an instance of the more abstract interaction pattern associated with the concrete one.
  - Similar to the process of keyword searching



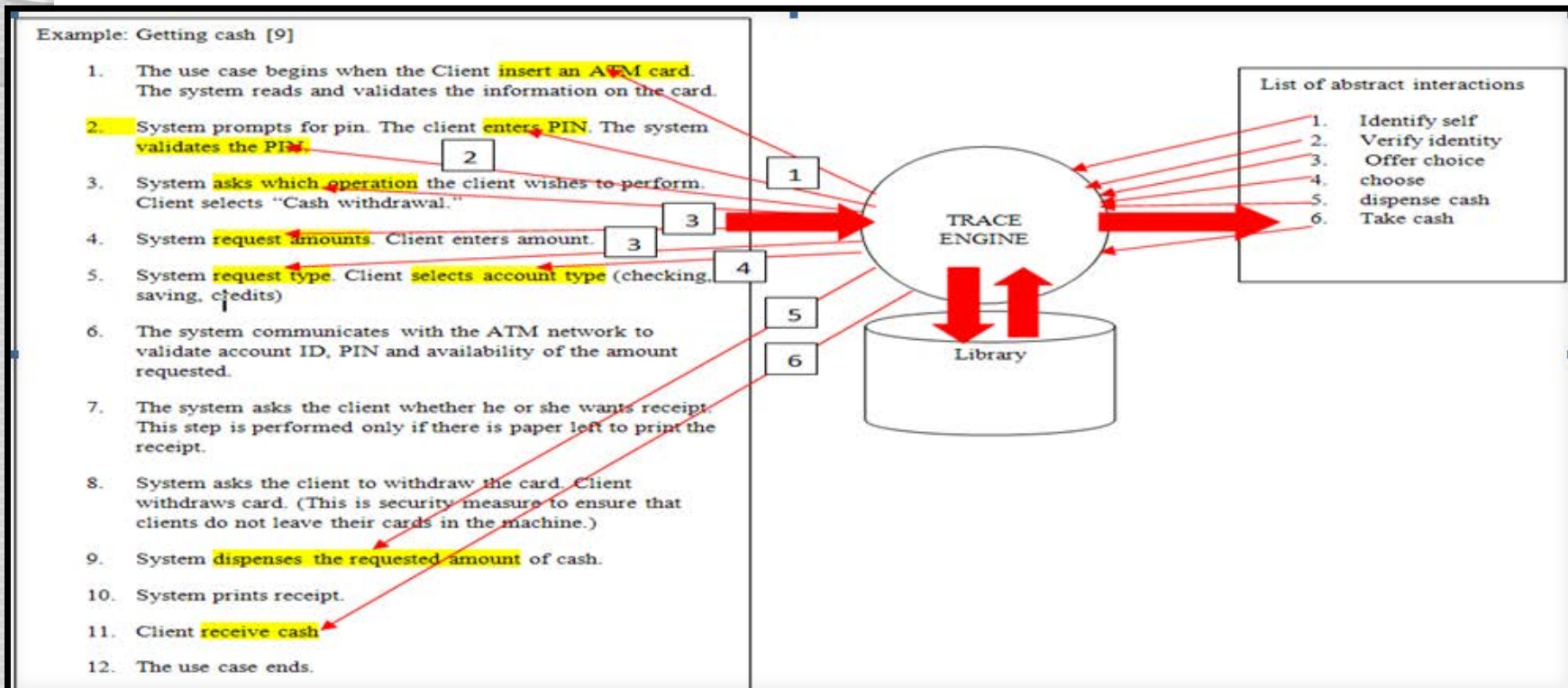
# ESSENTIAL INTERACTION EXTRACTION



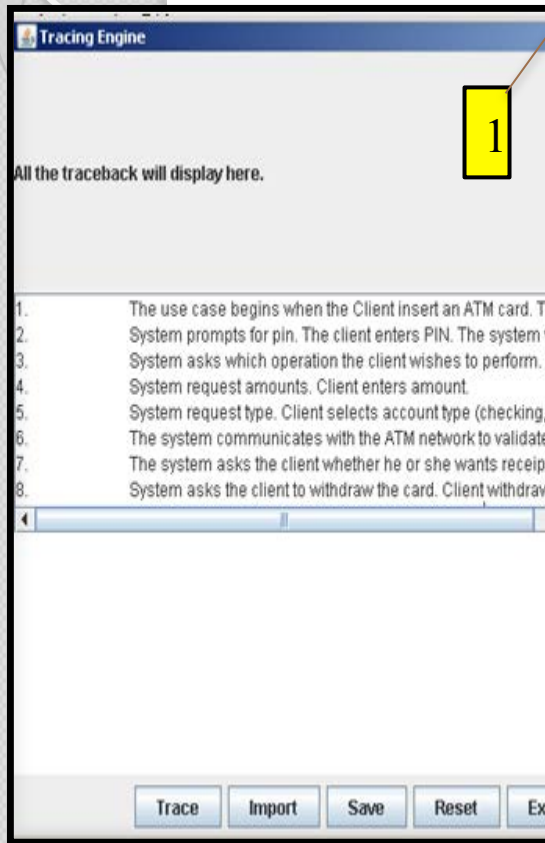
# TOOL SUPPORT

- Developed a prototype EUC essential interaction extraction tool
  - Requirements engineers can do initial essential interaction extraction from textual natural language requirements : this gives us an initial EUC model
- Tool provides traceability support mechanisms between textual natural language requirements and derived EUC models
  - Trace-forward & Trace –back
- Guidelines of using the patterns are codified
  - Requirements engineers need to have an understanding of the EUC concept and methodology before using the tool

# TOOL SUPPORT

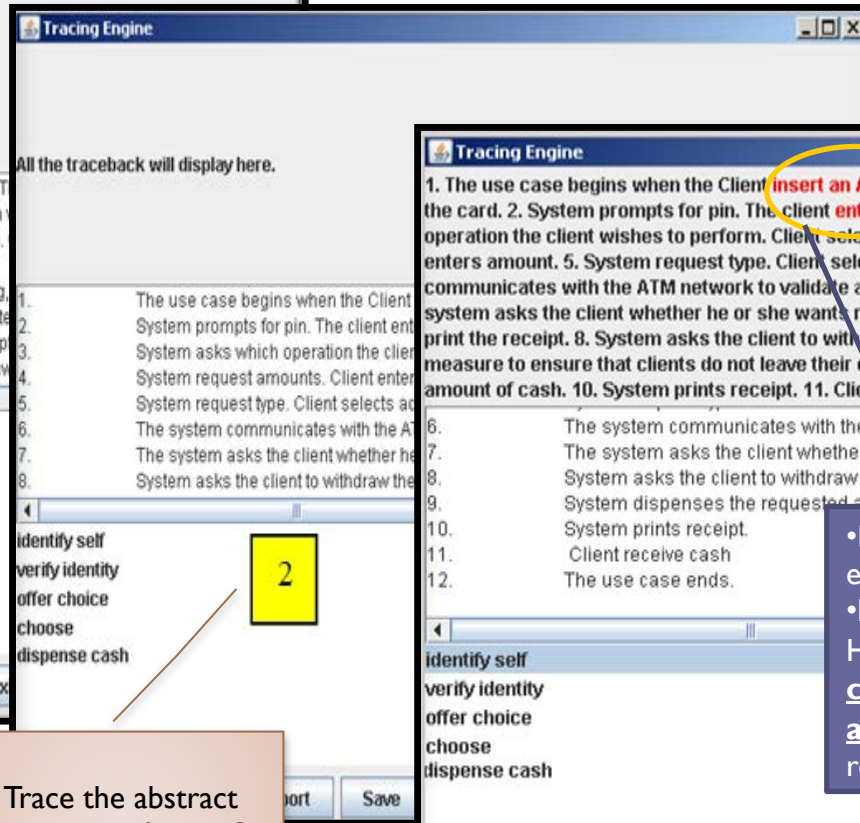


# OUR AUTOMATED TRACING TOOL



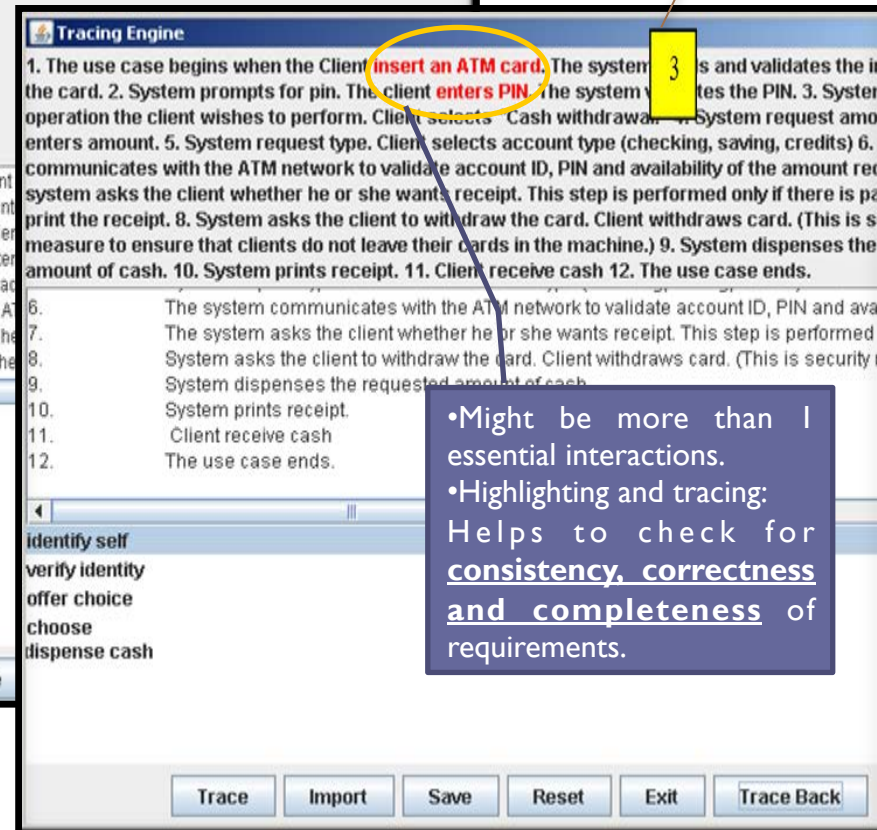
1

Input the textual natural language requirement



2

Trace the abstract interaction for EUC



3

Trace -back the identified abstract interaction to essential interaction

• Might be more than 1 essential interactions.  
• Highlighting and tracing: Helps to check for consistency, correctness and completeness of requirements.

# MARAMA ESSENTIAL

The screenshot displays the Marama Essential software interface. On the left, a toolbar includes options like 'Select', 'Marquee', 'Sketching tool', 'Shapes', 'Connectors', and 'VisualLink'. The main workspace is divided into two panels: 'ListOfAbstractInteraction' and 'EUCDiagram'. The 'EUCDiagram' panel shows a sequence of interactions: 'User Intention select option', 'System Responsi... request identific...', 'User Intention identify self', 'System Responsi... check status', 'System Responsi... provide identific...', and 'System Responsi... display error'. A red arrow points from the 'provide identification' interaction in the diagram to the corresponding trace result in the 'EUC Trace Result' panel on the right. The trace result shows a sequence of steps for the process 'Register for vote(Stephane S.Some 2005)', including 'Voter loads EVote system is online', 'Voter select voter registration option', 'EVote system ask for name, social security number, date of birth', 'Voter provide name and social security number, date of birth', 'EVote system checks Voter status', and 'Evote System \*\*\*generates Voter login id \*\*\*and password'. A blue box highlights the text '\*\*\*generates Voter login id \*\*\*and password' in the trace result.

•Integrate the automated tracing tool in Eclipse, MaramaAI.  
•The EUC in Marama called **Marama Essential**.  
•Develop using Marama meta-tools platform.

# ESSENTIAL INTERACTION EXTRACTION

- Collected and categorized phrases from a wide variety of textual natural language requirements documents.
- ≈ **300 phrases** from various requirements domains:
  - Online booking, online banking, mobile systems related to making and receiving calls, online election systems, online business, online registration and e-commerce.
- **88 patterns** of abstract interaction → on **average** **3-4** patterns /essential interactions per abstract interaction

Example of an Abstract interaction and associated Essential interactions

Abstract interaction	Essential interaction
Display error	Display time out
	Show error
	Display error message
	Show problem list

- Not categorized by 1 scenario.
- Associates with 5 concrete scenarios :
  - Online business, e-commerce, online booking, online banking and online voting system

# Key Textual Structures

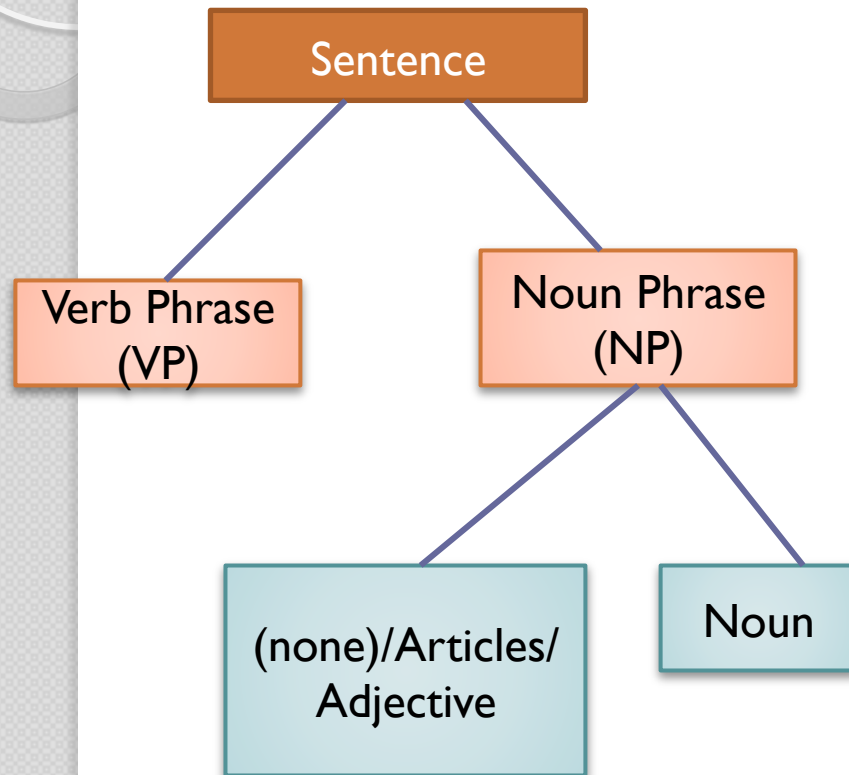


Figure 4. Tree structure for Key textual Phrases

Different sentence structures:

- Verb (V) + Noun (N) (only)
  - request (V) amount (N)
- Verb (V) + Articles (ART)+ Noun (N)
  - issue (V) a (ART) receipt (N)
- Verb (V) + Adjective (ADJ)+ Noun (N)
  - ask (V) which (ADJ) operation

Provides flexibility in the library:  
Accommodate **various types of sentences** containing essential requirements

# EVALUATION

- **Compare accuracy and performance** of our automated tracing tool with manual extraction
- Use same scenario & group of participants as we used earlier
- Survey their perception of the tool ease of use and utility for extraction and tracing

**Table 2. Comparison result of correctness between Manual Extraction and Automated Tracing Tool**

Answers	No. Correct answers		No. Wrong answers	
	Manual extraction	Automated Tracing	Manual extraction	Automated Tracing
Identify user	5	1	6	0
Verify Identity	4	1	7	0
Offer cash	4	1	7	0
Choose	6	1	5	0
Dispense cash	9	1	2	0
Take cash	3	0	8	1
Correctness ratio	47%	83%	53%	17%



# EVALUATION

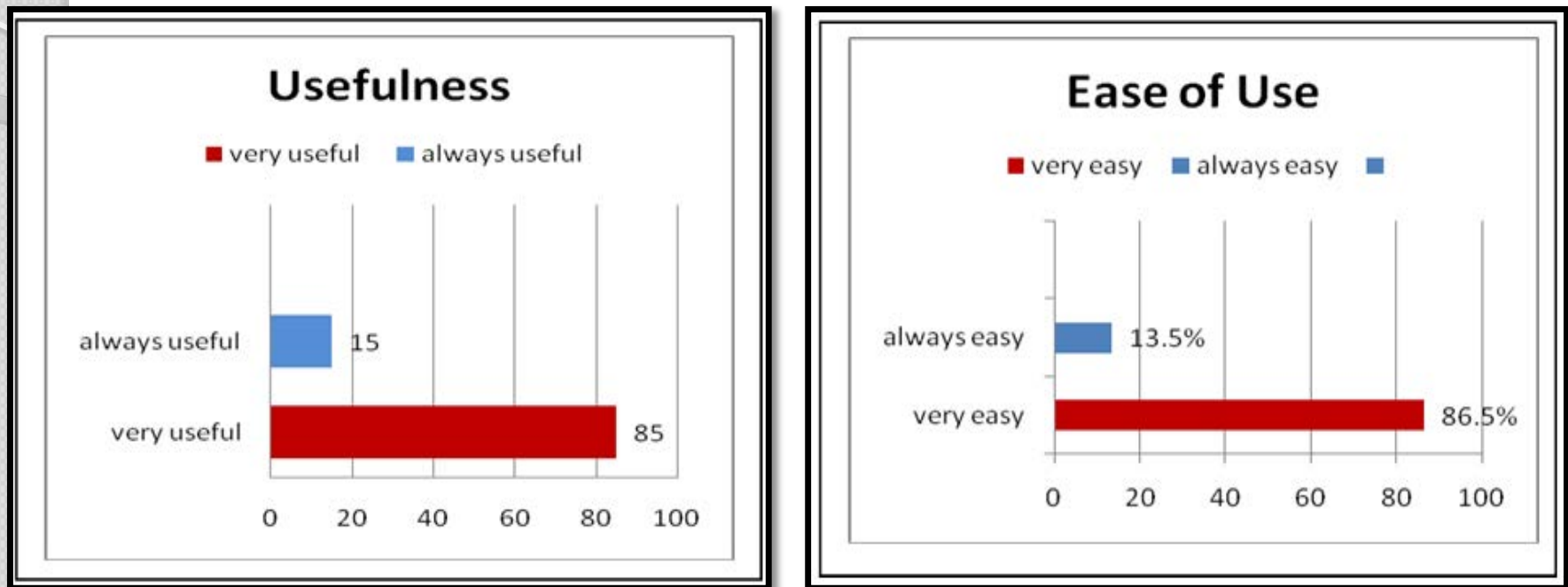


Figure 5: Result of the tool Usefulness and Ease of Use

## Recommendation:

- Better User interface with a more user friendly prototype
- Useful to be embedded within a tool that visually displays the EUCs to improve usability
- Time taken for trace & trace back:
  - fast and very fast,
  - noted some variation of speed for different scenarios

# EVALUATION

- **15 scenarios** from different domains derived from different researchers, developers and ourselves.
- Tool correctness evaluated by **comparing the answers with the actual interaction pattern** developed by Constantine and Lockwood, Biddle et al. and also pattern develop by us following Constantine and Lockwood methodology

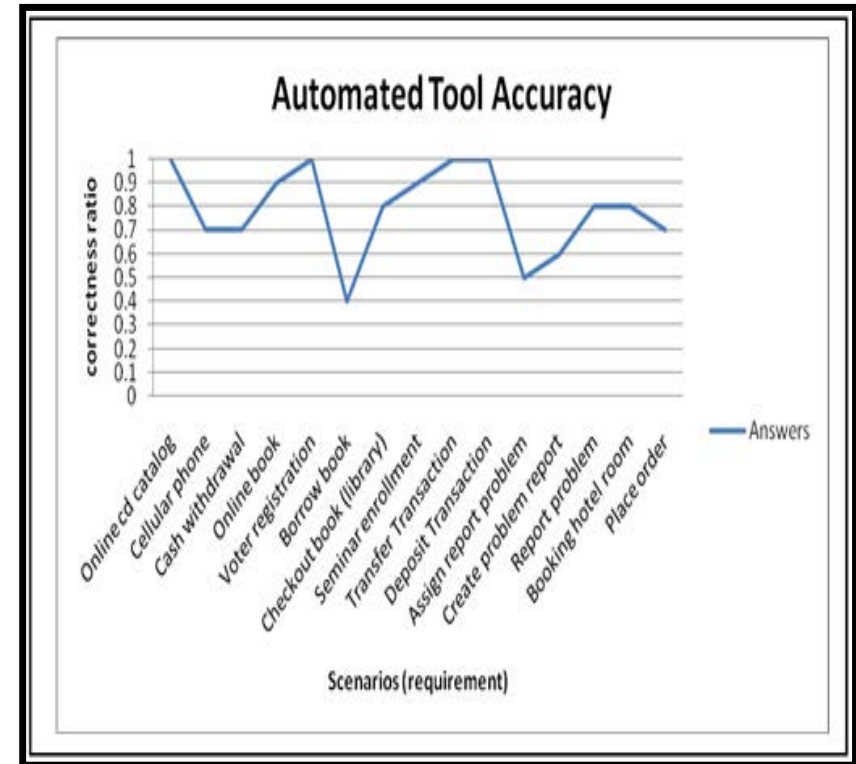


Figure 6. Accuracy across different scenarios

shows some **variability** across the range of scenarios, average correctness across all scenarios and interactions  $\approx 80\%$ , so the “getting cash” scenario used in the earlier evaluation was not unusual.

**Not 100%** : incorrectness and incompleteness issue of textual requirements → linguistic issues, parentheses existence and grammar.

# SUMMARY

- Identified problem faced by requirements engineers and end user while using EUC approach → our preliminary study
- Developed a prototype EUC essential interaction extraction and tracing tool
  - Key aim: to support EUC by extracting the essential requirement (abstract interactions) automatically and facilitate tracing between EUC and textual natural language requirements.
- Collection and categorization of terminology for the library of abstract interactions
  - assists in structuring EUC expressed requirements using common terminology and also helps prevent the textual requirements from being vague and error-prone
- Automated extraction and tracing tool
  - to increase the ratio of correctness in extracting EUC requirements from textual natural language requirements and eases the effort of users or requirements engineers in handling the EUC, significantly reducing the time taken.


# FUTURE WORK

- Embed our extraction approach into an integrated EUC Diagram tool (Marama Essential) developed using the Marama meta tool
  - will enable users to generate and maintain the consistency of visual EUC models automatically from lists of abstract interaction.
- Embed a glossary and template authoring support to the tool
  - to assist improved natural language-based requirements authoring and update.
- Add additional support for inconsistency, incompleteness and redundancy detection using our extraction approach and round-trip engineering of natural language and EUC model requirements
  - explore a complementary approach using a composite EUC pattern template library
- Plan to explore relating EUCs to further artefact views including generating UI and OO design models in our Eclipse prototype, with round-trip engineering support to consistency with textual natural language requirements.



# ACKNOWLEDGEMENT

This research is funded by the Ministry of Higher Education Malaysia (MOHE), Universiti Teknikal Malaysia Melaka (UTeM) the PReSS Account of the University of Auckland and the FRST Software Process and Product Improvement project.

- 
- TQ.
  - Q&A?

# References

- Kamalrudin, M., Hosking, J.G, and Grundy, J.C. Improving requirements quality using essential use case interaction patterns, In **Proceedings of the 2011 International Conference on Software engineering (ICSE 2011)**, Hawaii, USA, May 21-28 2011.
- Kamalrudin, M. and Grundy, J.C. Generating Essential User Interface Prototypes to Validate Requirements, In proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering, Nov 6-10 2011, IEEE Press. Kalmalrudin, M., Grundy, J.C. and Hosking, J.G., Managing consistency between textual requirements, abstract interactions and Essential Use Cases, In Proceedings of the 2010 IEEE International Conference on Computer Software and Applications (COMPSAC 2010), Seoul, South Korea, July 2010, IEEE CS Press.
- Kalmalrudin, M., Grundy, J.C. and Hosking, J.G. Tool Support for Essential Use Cases to Better Capture Software Requirements, In Proceedings of the 25<sup>th</sup> IEEE/ACM International Conference on Automated Software Engineering, Antwerp, Belgium, 20-24 Sept 2010, ACM Press.
- Kalmalrudin, M., Grundy, J.C. and Hosking, J.G. MaramaAI: Automated and Visual Approach for Inconsistency Checking of Requirements, Demo/Poster track at IEEE International Conference on Requirements Engineering, Sydney, Australia, September 27 - October 1, 2010.  
-- Final version available from [DOI](#) Author pre-published version [PDF](#)
- Grundy, J.C., Hosking, J.G., Li, N. and Huh, J. Marama: an Eclipse meta-toolset for generating multi-view environments, Formal demonstration at the 30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 2008, ACM Press.
- Olsen, T. and Grundy, J.C. Supporting traceability and inconsistency management between software artefacts, In Proceedings of the 2002 International Conference on Software Engineering and Applications, Boston, MA, 2-5 Nov 2002.
- Grundy, J.C., Hosking, J.G., Mugridge, W.B. Inconsistency Management for Multi-view Software Development Environments, IEEE Transactions on Software Engineering: Special Issue on Managing Inconsistency in Software Development, Vol. 24, No. 11, 1998, IEEE CS Press.