

SWIN
BUR
NE

SWINBURNE
UNIVERSITY OF
TECHNOLOGY

IMPROVING REQUIREMENTS QUALITY VIA ROUND-TRIP ENGINEERING WITH ESSENTIAL USE CASES

MASSILA KAMALRUDIN¹, JOHN GRUNDY², JOHN HOSKING³

¹Department of Electrical & Computer Engineering &

³Department of Computer Science,

University of Auckland, New Zealand

²Centre for Computing & Engineering Software Systems,
Swinburne University of Technology, Melbourne, Australia



Introduction



- Requirements engineering is hard
 - Software engineers often focus on doing the thing right, but...
 - Need to do the right thing!!
- Need to ensure requirements analysed for the “three Cs” - Consistency, Completeness and Correctness
- We are interested in all three of these
 - Previous work – supporting consistency of particular interest
 - Current work – analysing for completeness, correctness
 - Future work – better negotiation with stakeholders
- When and how do we do this analysis?
 - As early as possible!
 - Need good tool support

Motivation



- We conducted a study of experienced requirements engineers (REs) extracting semi-formal requirements from structure natural language documents
 - 11 people, most experienced industry REs
 - Basic natural language requirements -> (Essential) use cases
- They did really, really badly!!
- An initial exploratory tool with basic automation support did a lot better!
- Supporting tracability between the natural language requirements & semi-formal models helped REs a lot to improve both models

Example tool



The image displays three screenshots of a 'Tracing Engine' application window, illustrating the tool's output for an ATM transaction. Each window has a title bar with 'Tracing Engine' and standard window controls. The main content area contains a list of steps and a list of actions.

Top Left Screenshot: Shows the first step of the transaction. A yellow box with the number '1' highlights the first step in the list.

- 1. The use case begins when the Client insert an ATM card. The system reads and validates the information on the card.
- 2. System prompts for pin. The client enters PIN. The system validates the PIN.
- 3. System asks which operation the client wishes to perform. Client selects "Cash withdrawal."
- 4. System request amounts. Client enters amount.
- 5. System request type. Client selects account type (checking, saving, credits)
- 6. The system communicates with the ATM network to validate account ID, PIN and availability of the amount requested.
- 7. The system asks the client whether he or she wants receipt. This step is performed only if there is paper left to print the receipt.
- 8. System asks the client to withdraw the card. Client withdraws card. (This is security measure to ensure that clients do not leave their cards in the machine.)

Top Right Screenshot: Shows the same list of steps. A yellow box with the number '2' highlights the actions performed by the system.

- identify self
- verify identity
- offer choice
- choose
- dispense cash

Bottom Screenshot: Shows the complete transaction trace. A yellow box with the number '3' highlights the first step. The text is bolded and color-coded (red for client actions, black for system actions).

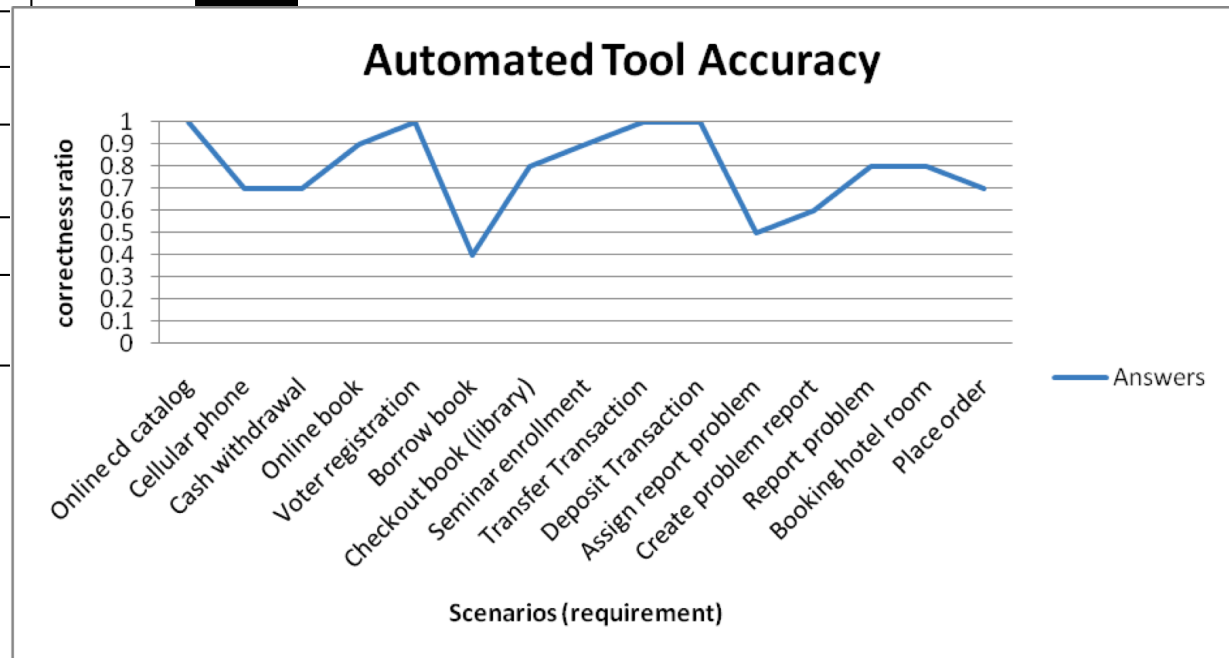
1. The use case begins when the Client **insert an ATM card**. The system reads and validates the information on the card. 2. System prompts for pin. The client **enters PIN**. The system validates the PIN. 3. System asks which operation the client wishes to perform. Client selects "Cash withdrawal." System request amounts. Client enters amount. 5. System request type. Client selects account type (checking, saving, credits) 6. The system communicates with the ATM network to validate account ID, PIN and availability of the amount requested. 7. The system asks the client whether he or she wants receipt. This step is performed only if there is paper left to print the receipt. 8. System asks the client to withdraw the card. Client withdraws card. (This is security measure to ensure that clients do not leave their cards in the machine.) 9. System dispenses the requested amount of cash. 10. System prints receipt. 11. Client receive cash 12. The use case ends.

- 6. The system communicates with the ATM network to validate account ID, PIN and availability of the amount requested.
- 7. The system asks the client whether he or she wants receipt. This step is performed only if there is paper left to print the receipt.
- 8. System asks the client to withdraw the card. Client withdraws card. (This is security measure to ensure that clients do not leave their cards in the machine.)
- 9. System dispenses the requested amount of cash.
- 10. System prints receipt.
- 11. Client receive cash
- 12. The use case ends.

Evaluation – text <-> EUCs for ATM



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	
Choose	6	1	5	
Dispense cash	9	1	2	
Take cash	3	0	8	
Correctness ratio	47%	83%	53%	



Our Aims/Research Goals



- Provide requirements engineers with an environment to support:
 - extraction of requirements from text into semi-formal models
 - consistency checking
 - traceability
 - completeness, correctness checkingbetween requirements expressed in natural language and semi-formal models of requirements expressed as essential use cases
- To provide REs with a lightweight approach c.f. natural language processing, formal methods
- We are using Constantine & Lockwood's Essential Use Cases as the semi-formal representation...

Essential Use Cases (EUCs)



“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 +Lockwood 1999].

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

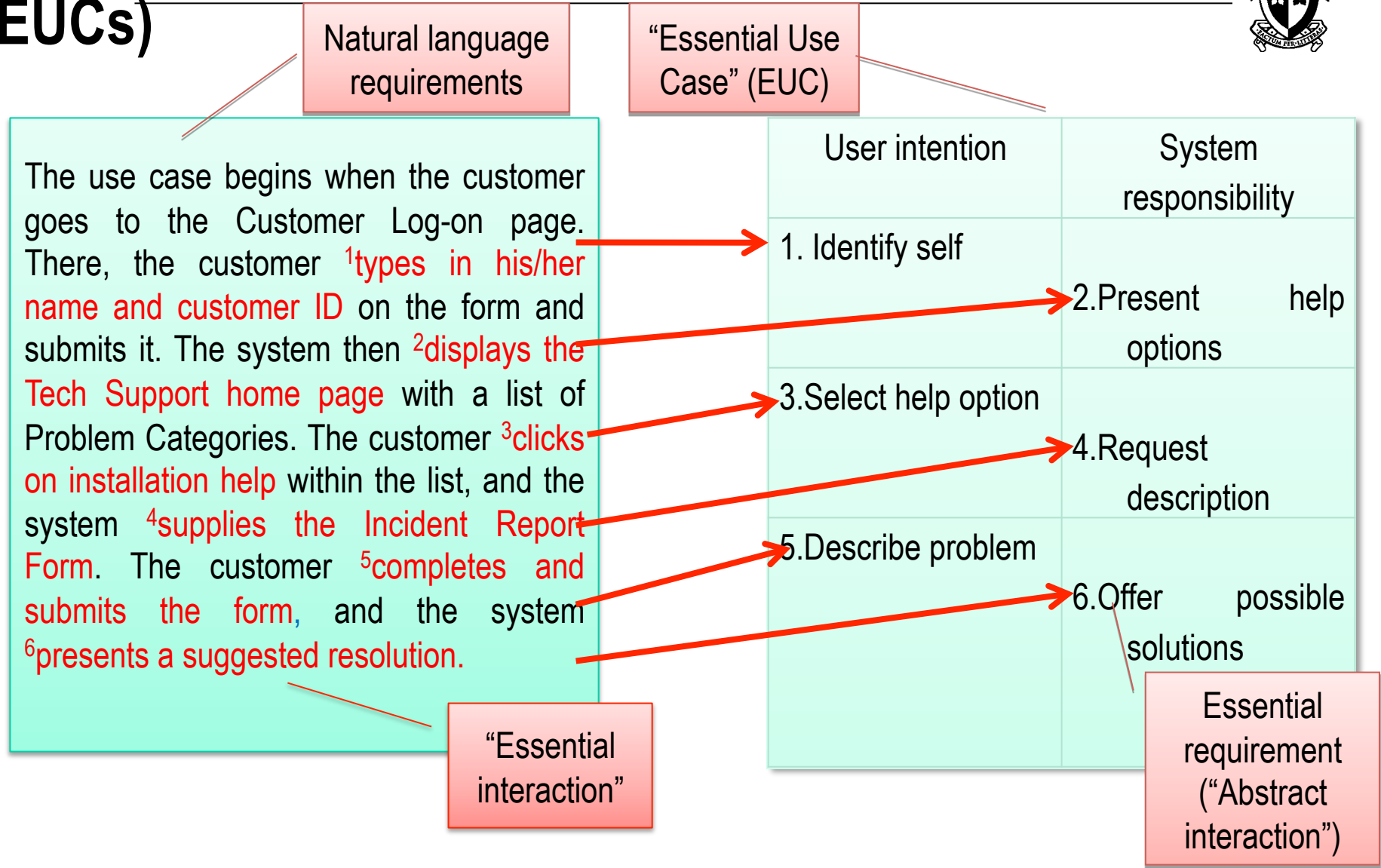
Shorter and simpler than conventional use cases, and is in the form of a dialogue between the user and system.

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)

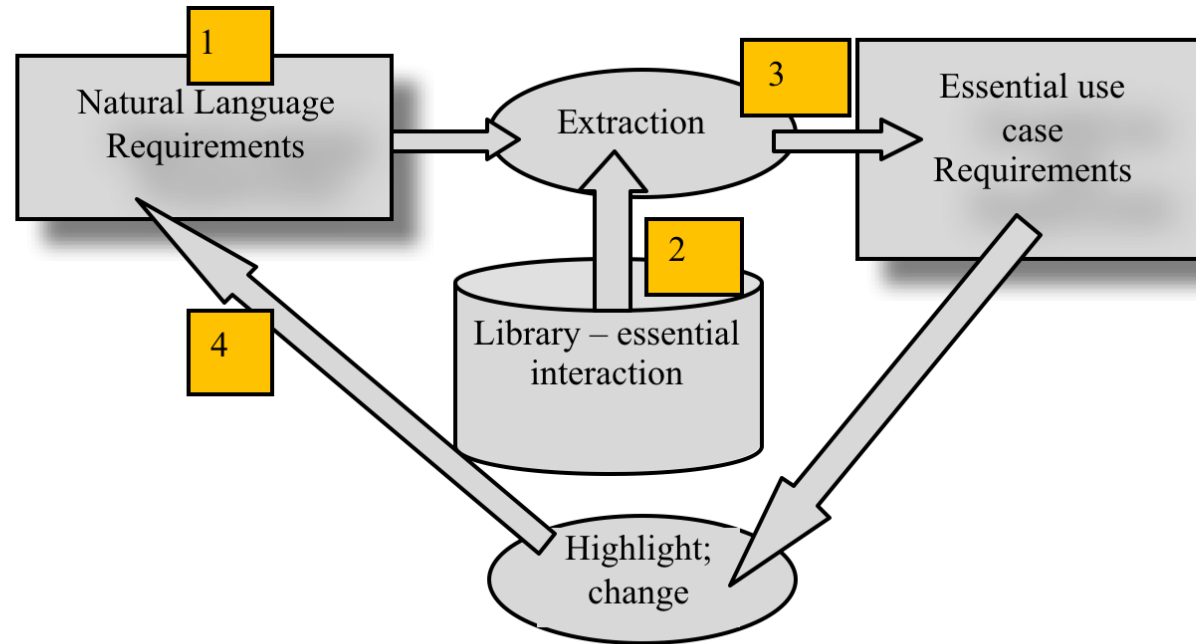


Example EUC abstract/essential interactions



Abstract interaction	Essential interaction	Example of Domains
Verify user	verify customer credential	Online banking, online booking, online business, e-commerce,online reservation
	verify customer id	Online banking, online booking, online business, e-commerce, online reservation
	verify username	Online banking, online booking, online business, e-commerce, online voting system, online reservation
	check the username	Online banking, online booking, online business, e-commerce, online voting system, online reservation
	check the password	Online banking, online booking, online business, e-commerce, online voting system, online reservation
Ask help	help desk	Online banking, online booking, online business, e-commerce, online reservation
	request for help	Online banking, online booking, online business, e-commerce, online voting system, mobile system, online reservation
	ask for help	Online banking, online booking, online business, e-commerce, online voting system, online reservation
	clicks help	Online banking, online booking, online business, e-commerce, online voting system,online reservation
	complete help form	Online banking, online booking, online business, e-commerce, online voting system, online reservation
Offer choice	prompt for amount	Online booking, online banking, online business, e-commerce
	display account menu	Online banking

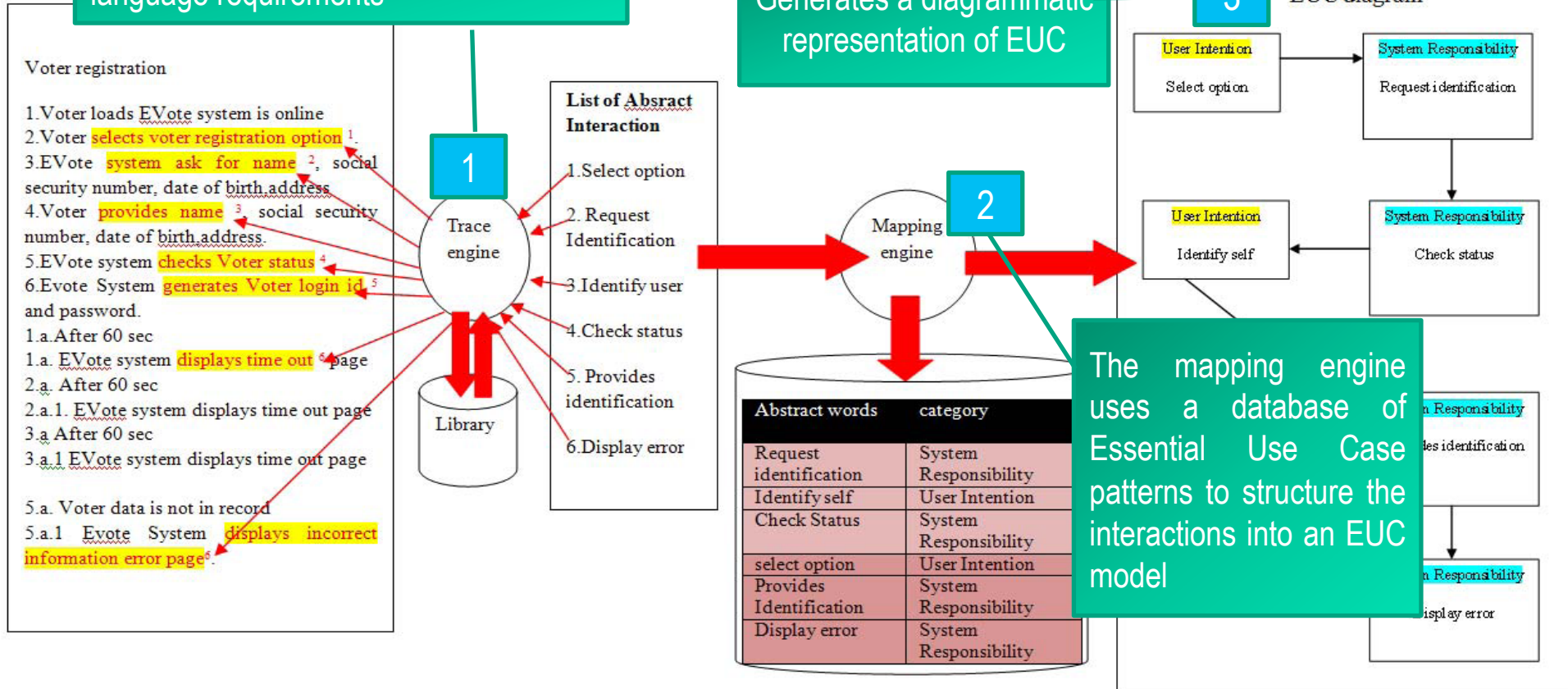
Our Approach(1)



Our Approach (2)

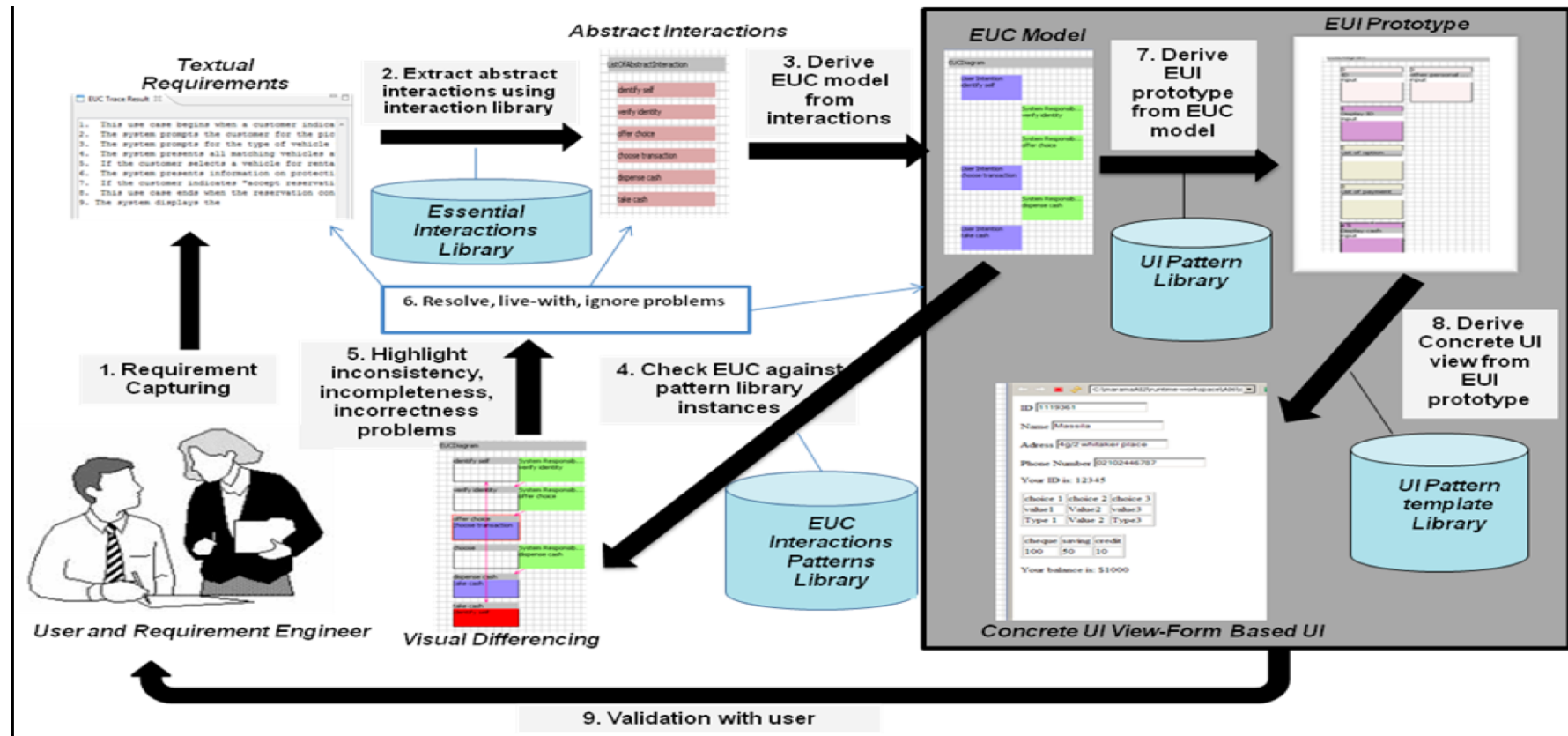


Extraction of a set of abstract interactions from the textual, natural language requirements



Our framework for extracting requirements: (1) mapping text to interactions; (2) mapping interactions to EUCs; and (3) creating the EUC

Our Approach (3)



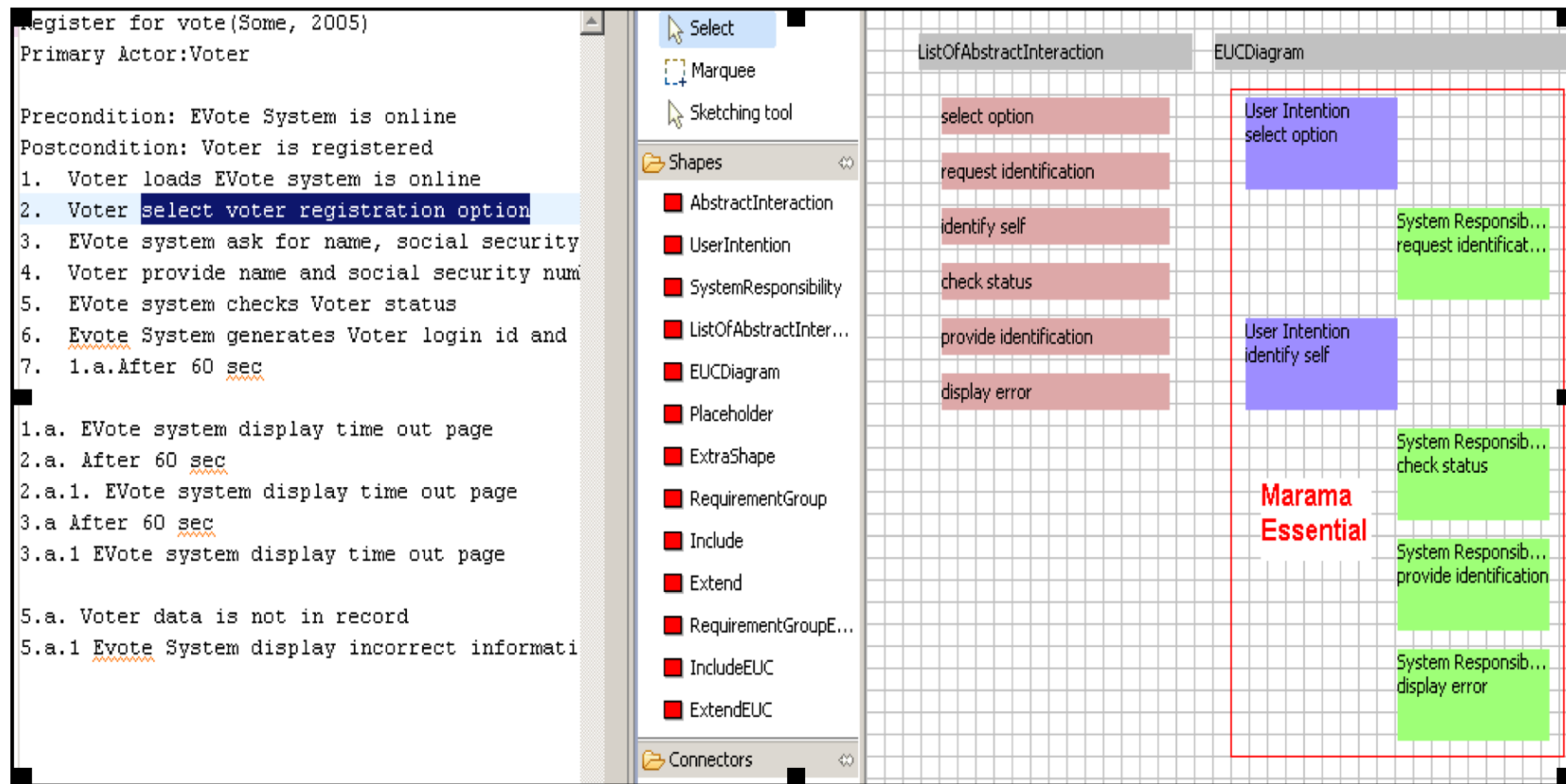
Validating EUCs with EUC pattern library (left); and supporting dialogue with the stakeholders via EUIs and form-based rapid prototypes (right)

Tool Support



- Developed an automated tracing tool, Marama AI, and EUC diagram editor, Marama Essential:
 - Provides support to extract EUCs automatically from text
 - Increases correctness of the abstract interactions produced
 - Lessens the need for manual checking of software requirements - provides consistency checking and notification support
 - Requirements that are detected as incomplete and/or inconsistent are highlighted - provides glossary and guidelines
- Comparison of extracted EUC to “best practice” EUC patterns:
 - Developed library of common EUC patterns (templates)
 - Compare extracted EUC to “best fit” pattern
 - Helps detect incompleteness, incorrectness in extracted EUC
 - Use a novel visual differ to highlight pattern/extracted EUC differences
- Generation of rapid user interface prototypes
 - Aid dialogue between requirements engineer and stakeholders

Extract Essential Interactions from text



Tracing Abstract Interactions

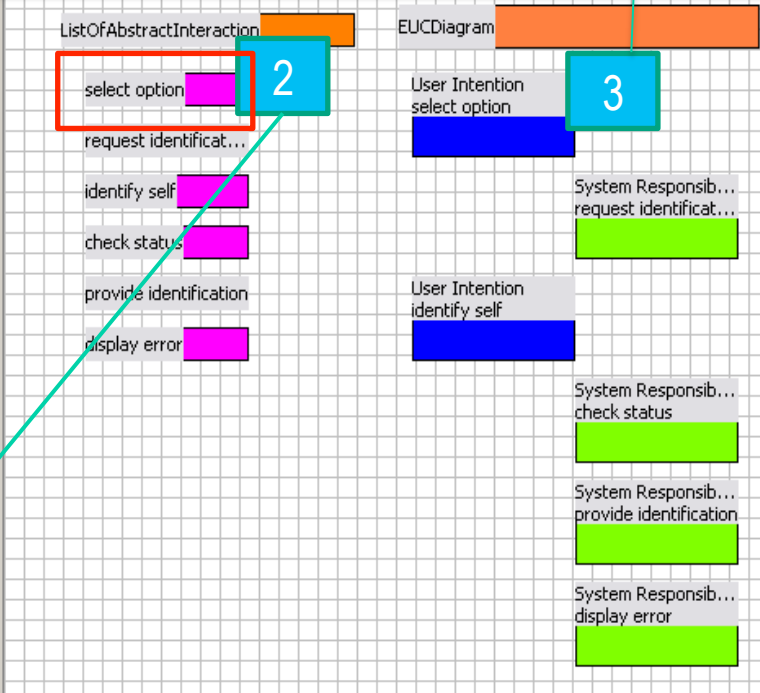


input requirement.txt

```
. Register for vote(Stephane S.Some 2005)
primary Actor:Voter
goal: An unregistered voter want to register in orde
recondition: EVote System is online
postcondition: Voter is registered
. Voter loads EVote system is online
. Voter select voter registration option
. EVote system ask for name, social security numbe
. Voter provide name, social security number, date
. EVote system checks Voter status
. EVote System generates Voter login id and passwo
. 1.a.After 60 sec
.a. EVote system display time out page
.a. After 60 sec
.a.1. EVote
.a After 60
.a.1 EVote s
.a. Voter de
.a.1 EVote S
```

Map to the EUC diagram and falls under the “user intention” category and select option interaction

“select voter registration option (1)” is traced to a particular abstract interaction – “select option (2)”



Tracing an abstract interaction from textual requirement and mapping to the Marama Essential representation

Inconsistency Checking



The screenshot shows a software development environment with three main components:

- Input Requirements File (input requirement.txt):** Contains text-based requirements for a voting system, such as "Register for vote", "Voter select voter registration option", and "EVote system display time out page".
- Diagram Editor (*diagram1.maramaDiagram):** Displays a sequence of interaction components including "request identificat...", "identify self", "check status", "provide identification", "display error", and "select option". A red arrow points from the "select option" component in the diagram to the warning dialog.
- Warning Dialog:** A blue box with a yellow warning icon and the text: "Warning: Abstract Interaction sequence is inconsistent with the input sequence of EUC components."

abstract interaction - "select option" is moved.
→ produces an inconsistency in the requirements and the tool detects this and provides a warning about the inconsistency.

Inconsistency Checking (2)



The screenshot displays a software interface with three main windows and a warning dialog:

- input requirement.txt**: Contains a requirement for "Register for vote" with a primary actor of "Voter", a goal, preconditions, and postconditions. It lists steps from 1 to 3.a.1, including a 60-second timeout.
- *diagram1.maramaDiagram**: Shows a "ListOfAbstractInteraction" window with a list of actions: "select option", "request identificat...", "identify self", "check status", "provide identification", and "display error".
- EUCDiagram**: Shows a sequence of components: "User Intention select option", "System Responsib... request identificat...", "System Responsib... check status", "System Responsib... provide identification", and "System Responsib... display error".
- Warning Dialog**: A yellow warning icon and text: "EUC component is inconsistent with the property in textual requirement and abstract interaction. keyword: select date".

A red box highlights a component in the EUCDiagram labeled "User Intention select date", which is connected by a red line to the warning dialog.

New component of EUC is added and tool detects an inconsistency with the textual requirements and abstract interaction. An inconsistency warning appears and informs the requirements engineer where the inconsistency occurs. This warning shows dependencies that occur between the textual requirement, abstract interaction and EUC diagram.

Text/Interaction patterns



The screenshot displays a software development environment with three main components:

- Text Editor (Left):** Contains a use case description with steps 2 through 8. Step 8 is highlighted in red.
- Diagram Editor (Right):** Shows a sequence diagram with two lifelines: 'User Intention' and 'System Responsib...'. The diagram includes messages: 'choose', 'offer choice', 'view detail', 'request identification', 'identify self', and 'confirm booking'. A red box highlights a 'print' message on the 'User Intention' lifeline.
- Inconsistency Warning Dialog (Center):** A dialog box with a yellow warning icon. The text reads: "The following abstract interaction is inconsistent with the textual requirements. Do you want to UPDATE the textual requirement or DELETE the new Abstract Interaction or Continue with Inconsistency?". Below the text is the word 'print'. At the bottom are three buttons: 'Update', 'Delete', and 'Continue'.

Detect textual essential interactions are inconsistent with abstract interactions

- option to change text
- option to change abstract interaction
- option to ignore and fix later (or not) 😊

Interaction -> text



The screenshot displays a software development environment. On the left, a text editor shows a use case description. In the center, a 'Text input' dialog box is open, prompting the user to input a new textual requirement. The dialog includes a list of 'Correct and Complete words' and a text input field containing 'system print bill.'. On the right, an 'EUCDiagram' (Essential Use Case Diagram) is shown, featuring a 'List of Abstract Interaction' panel with items like 'choose', 'offer choice', 'view detail', 'request identification', 'identify self', 'confirm booking', and 'print'. The diagram also shows 'User Intention' and 'System Responsibility' blocks. A red box highlights the 'print' interaction in the list, and a green arrow points from this box to the 'Text input' dialog.

Update text.
-manual update
-semi-automatic update from abstract interaction -> essential interaction

Ignoring inconsistency...



The screenshot shows the Eclipse IDE interface. On the left, a text editor contains use case requirements. The main workspace displays a UML diagram with two swimlanes: 'ListOfAbstractInteraction' and 'EUCDiagram'. The diagram includes several use case elements: 'choose', 'offer choice', 'view detail', 'request identification', 'identify self', and 'confirm booking' in the 'ListOfAbstractInteraction' swimlane; and 'User Intention choose', 'System Responsib... offer choice', 'System Responsib... view detail', 'System Responsib... request identific...', 'User Intention identify self', and 'System Responsib... confirm booking' in the 'EUCDiagram' swimlane. A new 'print' use case element is being added to the 'ListOfAbstractInteraction' swimlane, highlighted with a red box. The 'Problems' view at the bottom shows a warning: 'The new abstract interaction is inconsistent with textual requirement'.

Description	Resource	Path	Location	Type
Warnings (1 item)				
The new abstract interaction is inconsistent with textual requirement	diagram1.m...	AI6	print	Problem

Ignore (for now)
-Eclipse problem marker tracks

Renaming items (eg choose->check)



This use case begins when a customer indicates he wishes to make a reservation for a rental car.

- The system prompts the customer for the pickup and return dates of the reservation, as well as the pickup and return dates.
- The system prompts for the type of vehicle the customer indicates the vehicle type.
- The system presents all matching vehicles available at the customer indicates the vehicle type.
- If the customer selects a vehicle for rental, the system presents information on protection products.
- If the customer indicates "accept reservation," the system confirms the reservation.
- This use case ends when the reservation confirmation is received.

8. This use case ends when the reservation confirmation is received.

Inconsistency Warning

The following abstract interaction is inconsistent with the textual requirements.
Do you want to UPDATE the textual requirement or DELETE the new Abstract Interaction or Continue with Inconsistency?

check item

Update Delete Continue

Text input

Please input new textual requirement. You can use the list of words below or glossary for help;

Correct and Complete words:

scans label
check book
check an item
system scan label,

OK Delete

Names used to link parts

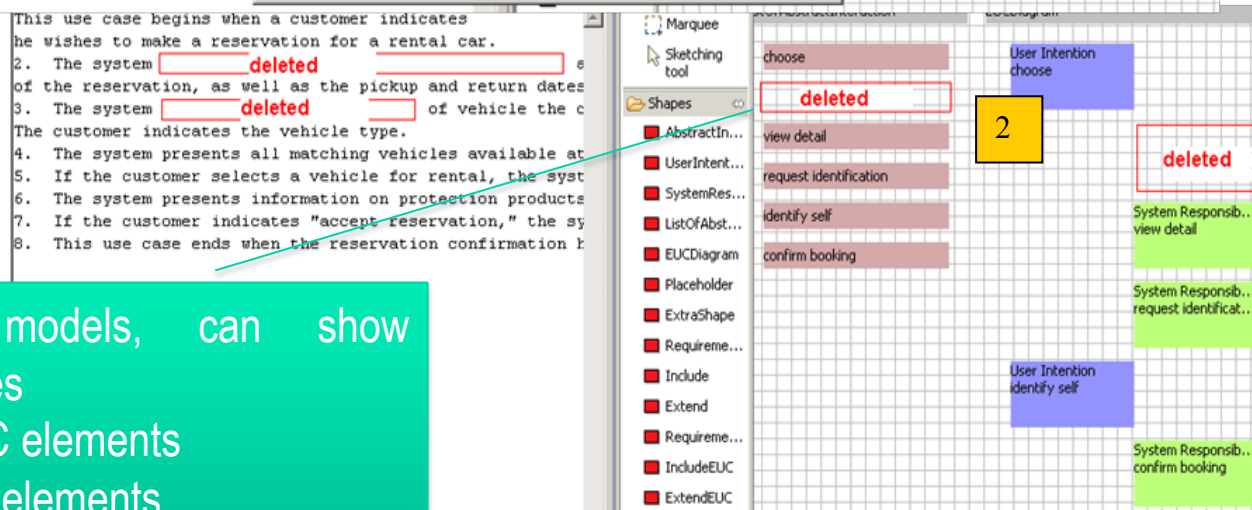
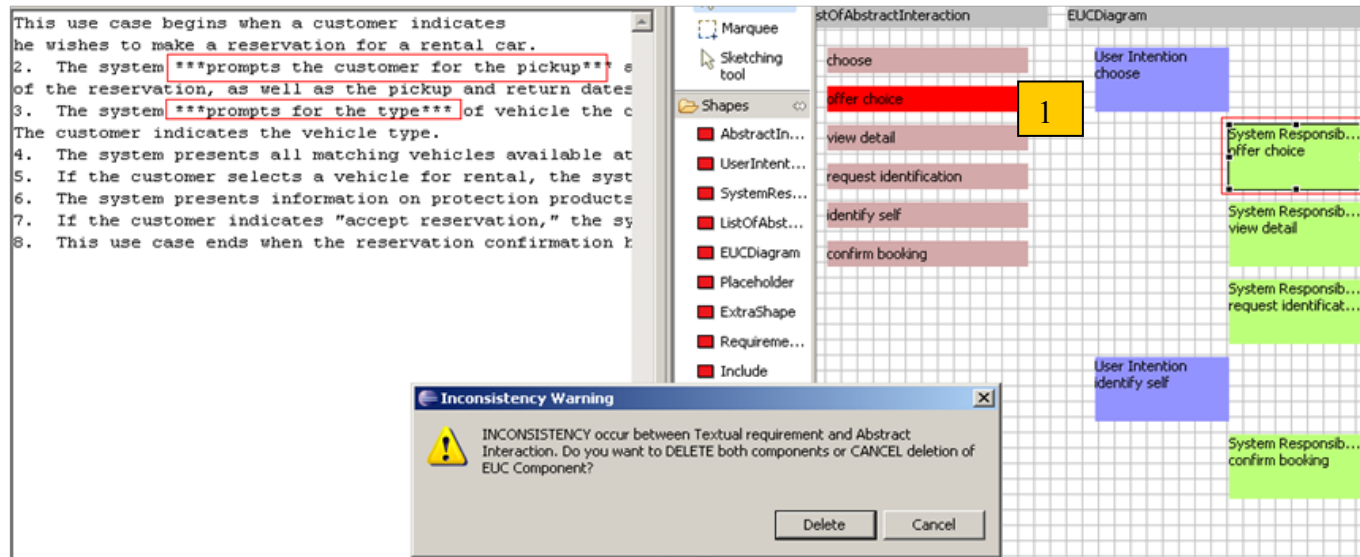
- need to remap
- need to update
- semi-automated support using abstract & essential interaction keywords

EUC Patterns – a few examples...



Scenarios/ Use Case stories	User intention Abstract Interaction	System responsibility Abstract Interaction
Reserve item	choose	
		offer choice
		view detail
		request identification
	identify self	
Purchase item		confirm booking
	choose	
		check status
	identify self	
	provides detail	
		verify identity
Make a transaction		request confirmation
		view detail
	select option	
	choose	
	select amount	
Book item		verify identity
		print
	identify self	
	select option	
	select item	
Make a registration	insert information	
		Print
	select option	
		request identification
	identify self	
		check status
	provide identification	
	display error	

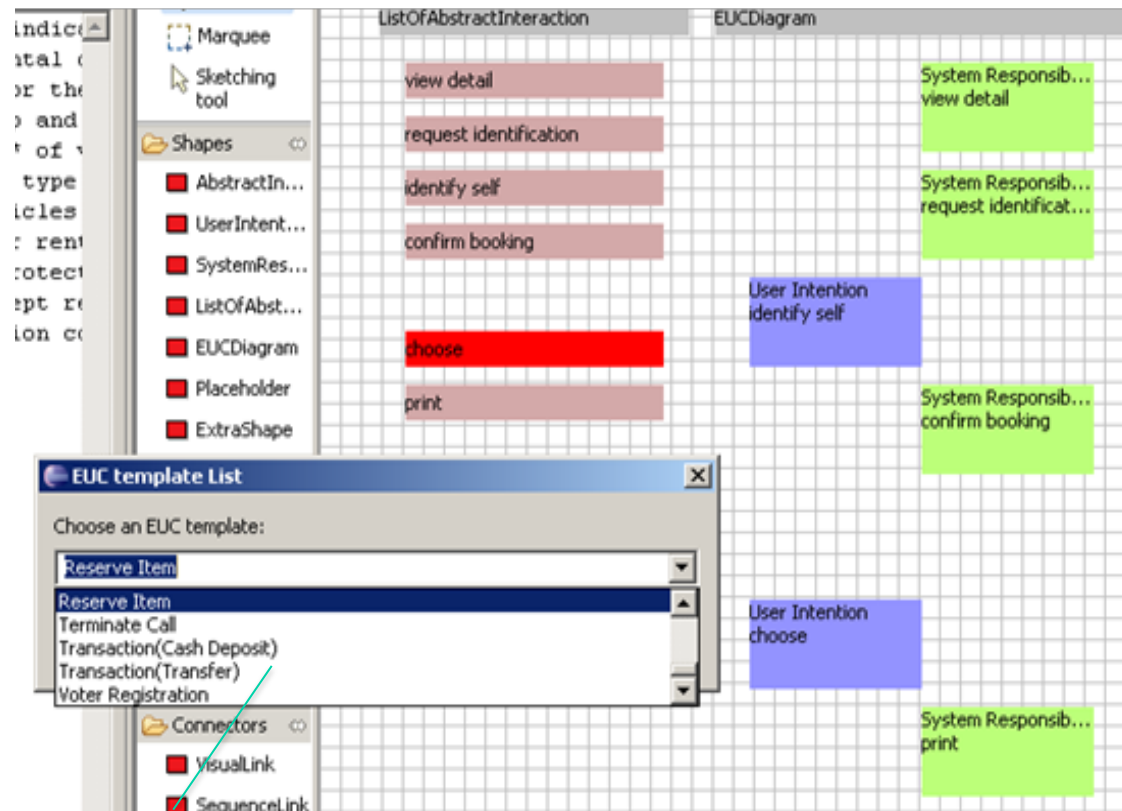
Visual diffing - consistency



Comparing models, can show inconsistencies

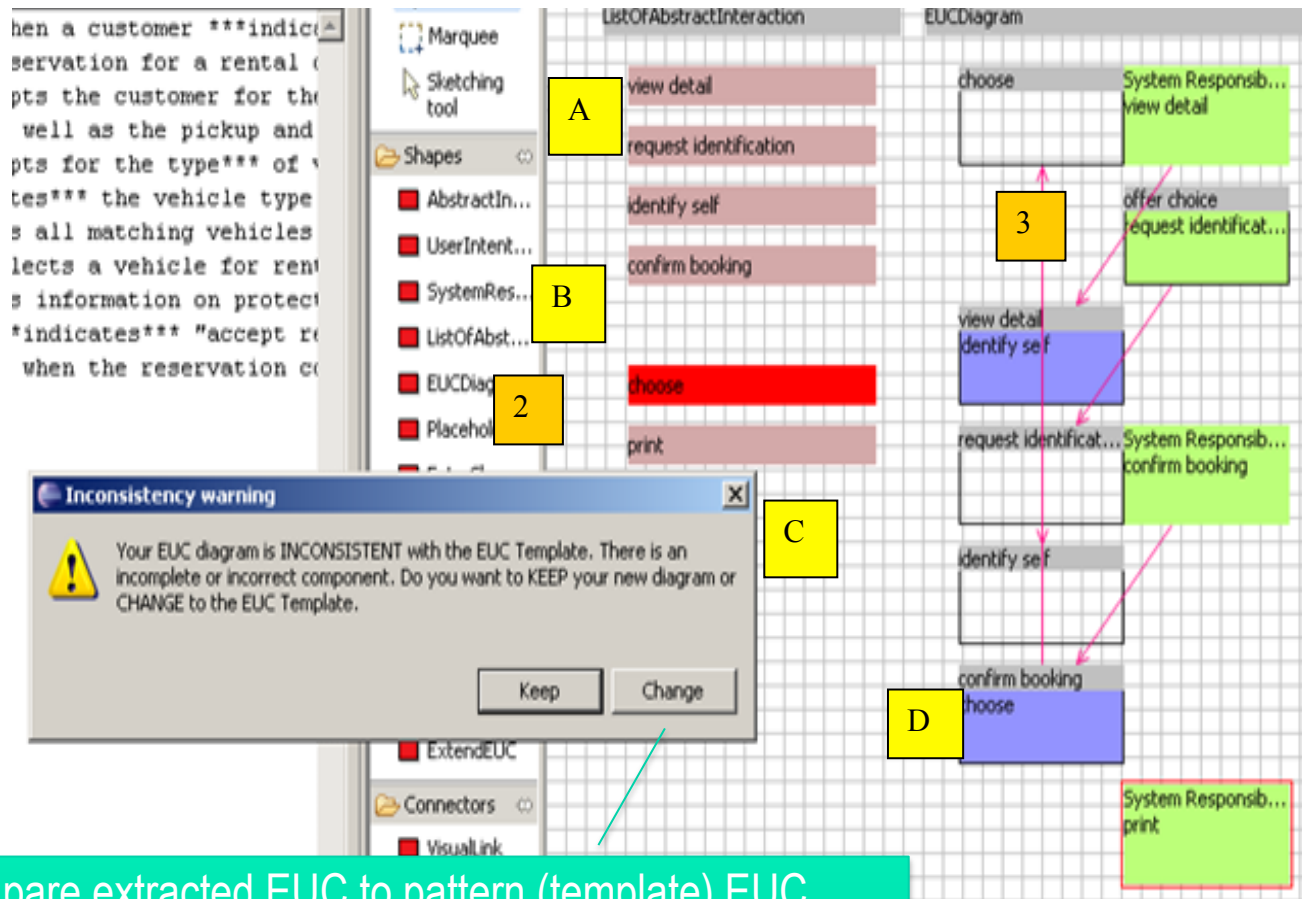
- highlight EUC elements
- highlight text elements
- show changes

Visual diffing – correctness/completeness



Choose EUC pattern to compare to extracted pattern
Detect “best fit” EUC pattern to compare

Compare to template EUC pattern



Compare extracted EUC to pattern (template) EUC
-highlight items added – incorrect
-highlight items missing - incomplete
-highlight items in diff order – incorrect/inconsistent
Allow semi-automated update of extracted to pattern

Update based on template



This use case begins when a customer ***indic...
he wishes to make a reservation for a rental...
2. The system ***prompts the customer for the...
of the reservation, as well as the pickup and...
3. The system ***prompts for the type*** of...
The customer ***indicates*** the vehicle type...
4. The system presents all matching vehicles...
5. If the customer selects a vehicle for rent...
6. The system presents information on protect...
7. If the customer ***indicates*** "accept r...
8. This use case ends when the reservation c...

Modify EUC to template
Update abstract interactions
Update text

ListOfAbstractInteraction

- view detail
- request identification
- identify self
- confirm booking
- choose
- print

EUCDiagram

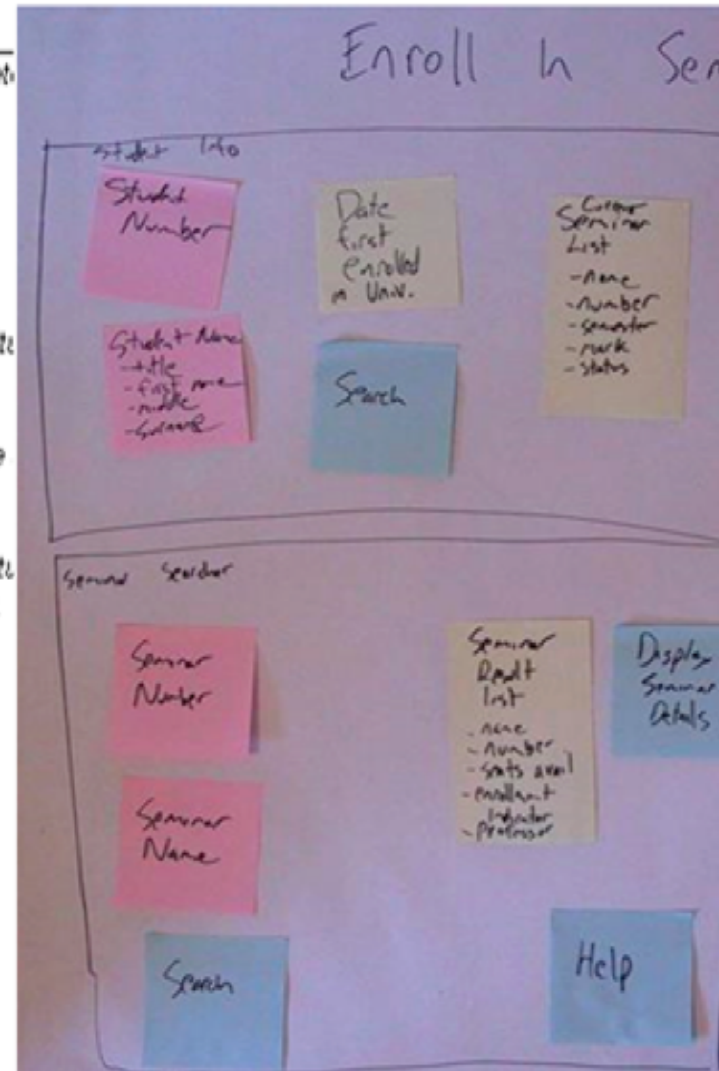
- User Intention choose
- System Responsib... offer choice
- System Responsib... view detail
- System Responsib... request identifi...
- User Intention identify self
- System Responsib... confirm booking

Description	Resource	Path	Location	Type
Warnings (2 items)				
The EUC diagram is changed to EUC Template	diagram1.m...	A16	EUC di...	Problem
The Highlighted textual requirements is inconsistent with the sequence of A	diagram1.m...	A16	choose	Problem

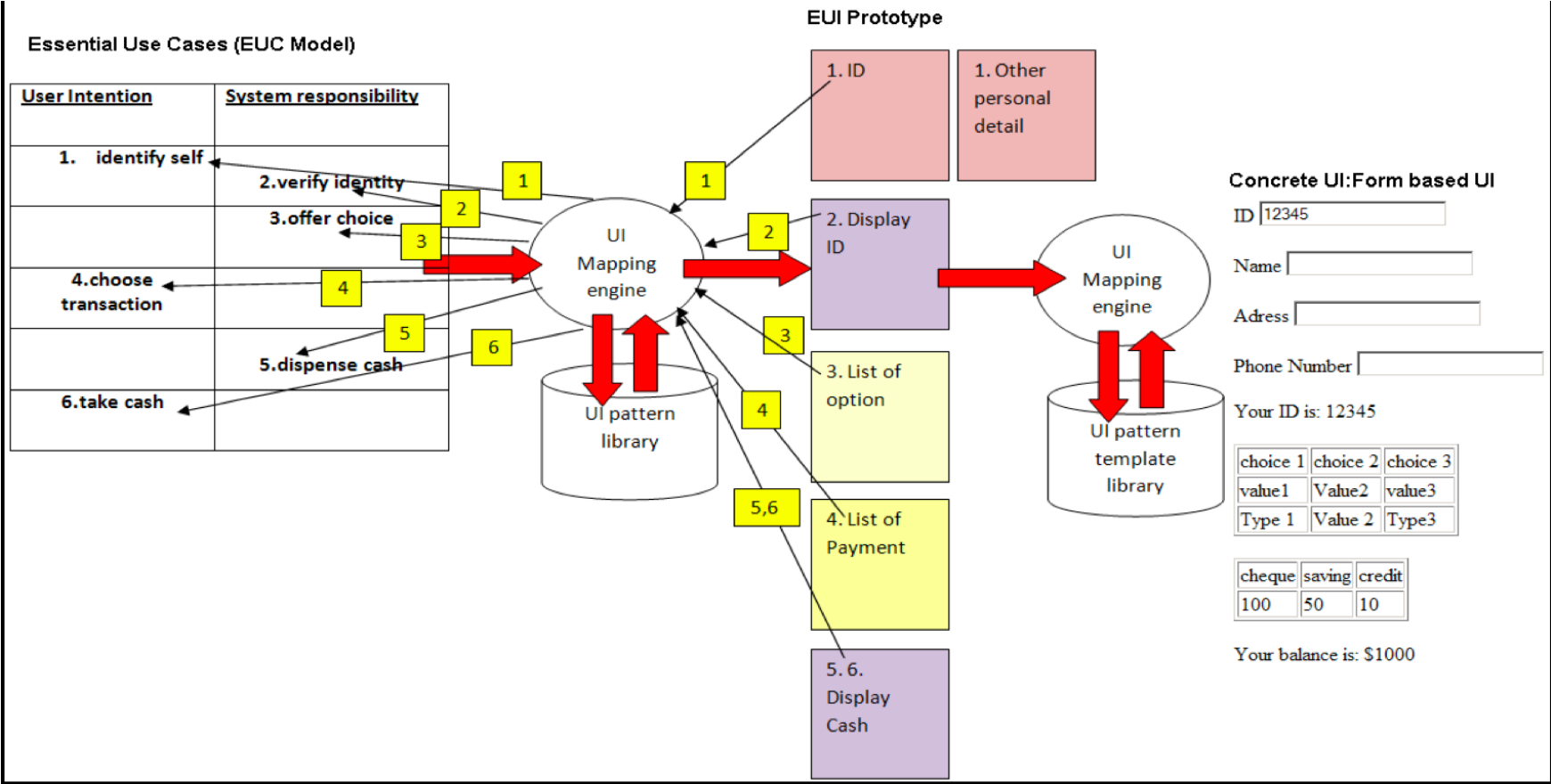
EUI rapid prototypes



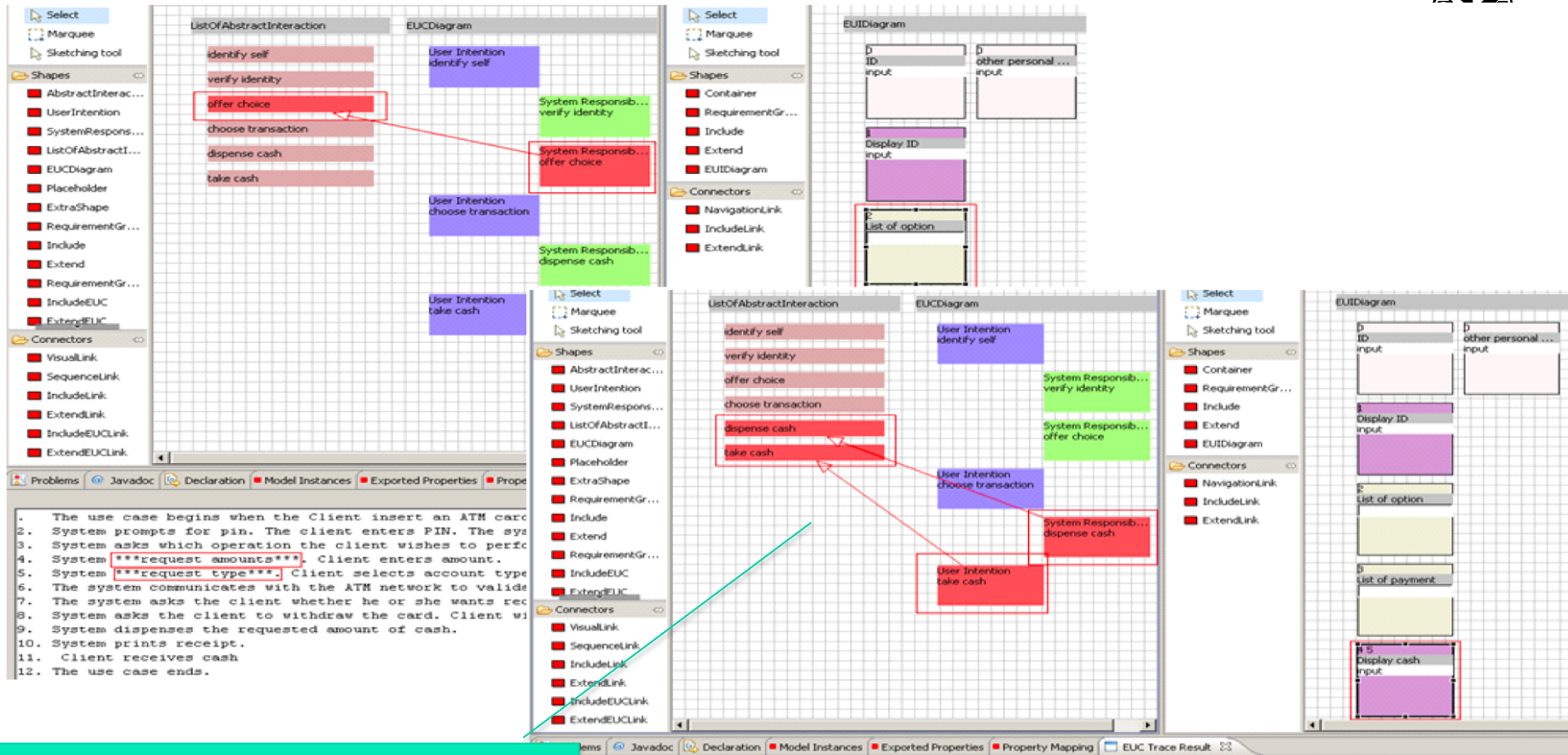
User Intention	System Responsibility
Student identifies himself	Verifies eligibility to enroll via <i>BR129 Determine Eligibility to Enroll</i> .
Choose seminar	Indicate available seminars
	Validate choice via <i>BR130 Determine Student Eligibility to Enroll in a Seminar</i> .
	Validate schedule fit via <i>BR143 Validate Seminar Schedule</i>
Confirm enrollment	Calculates fees via <i>BR 180 Calculate Student Fees</i> and <i>BR45 Calculate Taxes for Seminar</i> .
	Summarize fees
	Request confirmation
	Enroll student in seminar
	Add fees to student bill
	Provide confirmation of enrollment



EUI Generation



Generate an EUI from an EUC...



Take EUC and generate EUI model

- set of EUI element patterns
- map EUC items to EUI items
- generate EUI layout
- allow editing of both (plus text)

The use case begins when the Client insert an ATM card. The system reads and validates the information on the card. The system prompts for pin. The client enters PIN. The system validates the PIN. The system asks which operation the client wishes to perform. Client selects "Cash withdrawal." The system request amounts. Client enters amount. The system request type. Client selects account type (checking, saving, credits) The system communicates with the ATM network to validate account ID, PIN and availability of the amount requested. The system asks the client whether he or she wants receipt. This step is performed only if there is paper left to print. The system asks the client to withdraw the card. Client withdraws card. (This is security measure to ensure that clients system ***dispenses the requested amount*** of cash. The system prints receipt. Client ***receives cash*** The use case ends.

Generating an HTML Form from an EUI



The screenshot shows a software interface for generating HTML forms from EUI diagrams. On the left, a toolbar contains tools like 'Select', 'Marquee', and 'Sketching tool'. Below it are 'Shapes' (Container, RequirementGr..., Include, Extend, EUIDiagram) and 'Connectors' (NavigationLink, IncludeLink, ExtendLink). The central 'EUIDiagram' window shows a grid with several elements: two input fields labeled 'ID' and 'other personal ...', a 'Display ID' input field, a 'List of option' box, a 'List of payment' box, and a 'Display cash' input field. On the right, a browser window displays the generated HTML form with the following content:

ID

Name

Adress

Phone Number

Your ID is: 12345

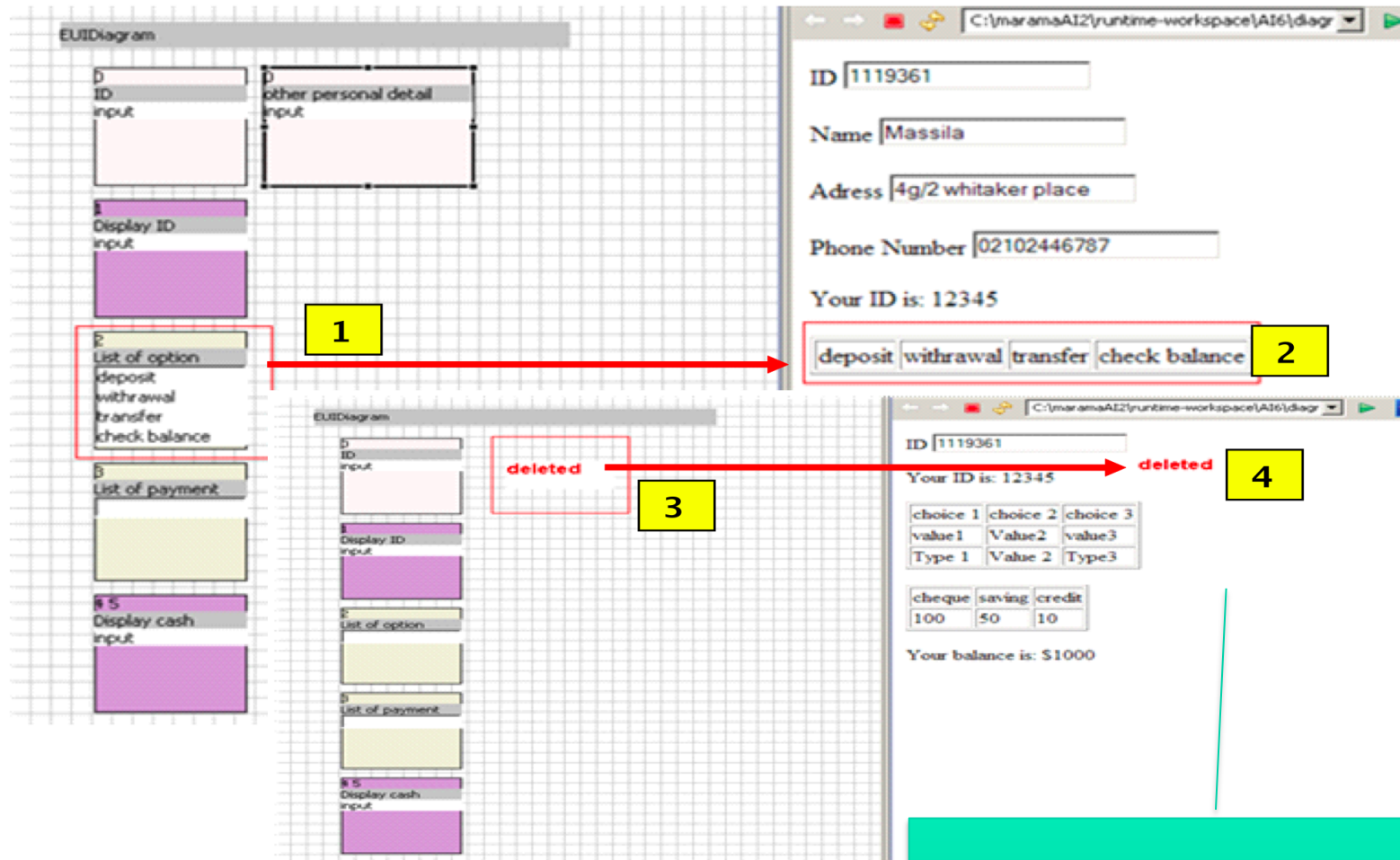
choice 1	choice 2	choice 3
value1	Value2	value3
Type 1	Value 2	Type3

cheque	saving	credit
100	50	10

Your balance is: \$1000

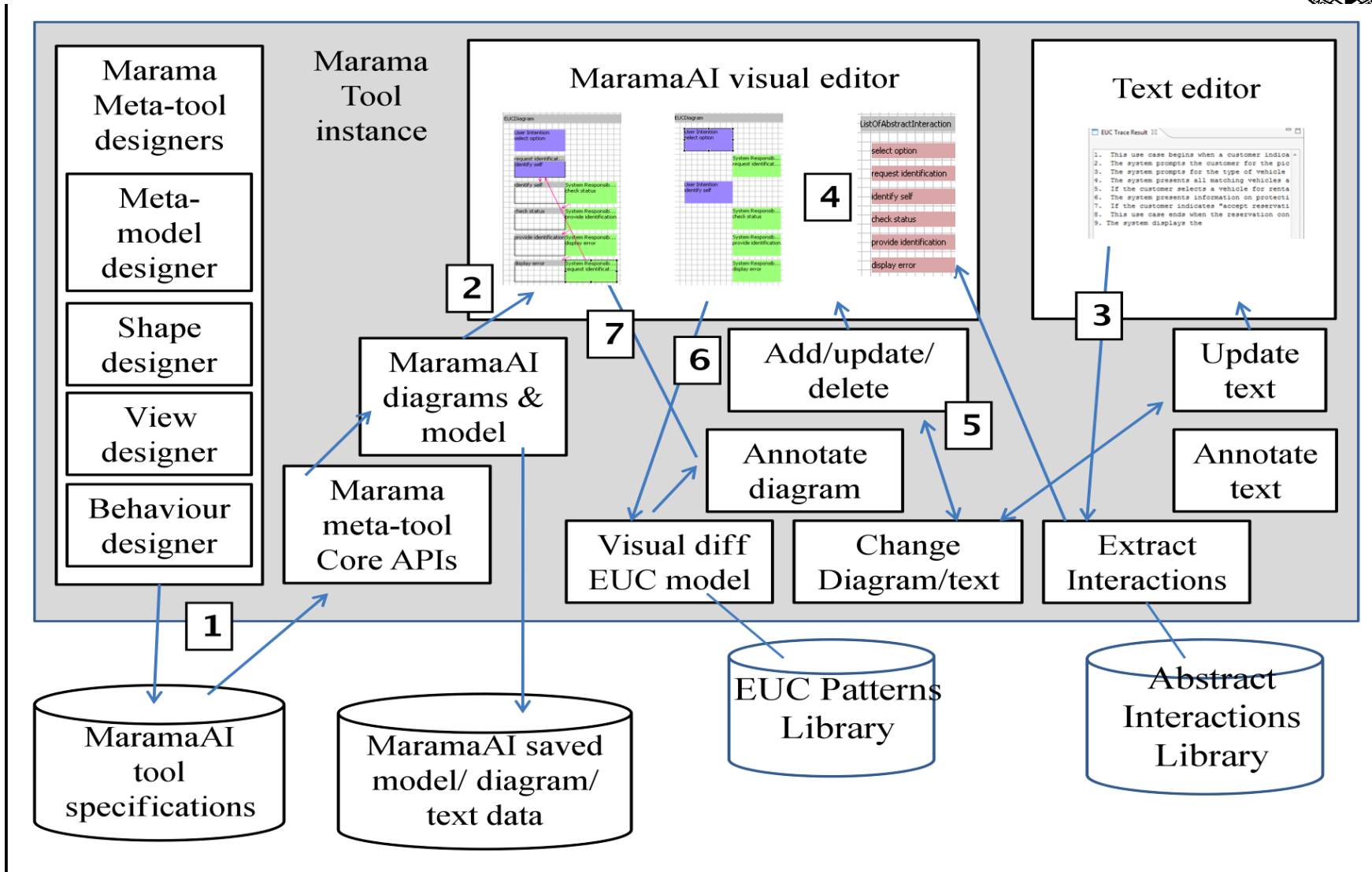
Generate HTML form from EUI
-library of HTML items
-library of EUI->HTML items
-layout & sizing heuristics
-can interact with form to "try" UI
[-can edit & keep consistent]

Consistency management, checking



Modifying EUI item -> modify HTML form
Add/delete EUI item -> modify form

Architecture



Architecture & Implementation cont..



Types of MaramaEUC Event handlers:

- ❑ ExtractInteractions: extracts abstract interactions from text
- ❑ Trace: Trace the textual requirement to the abstract interaction: extract key phrases which are analyzed and matched by the interaction pattern library.
- ❑ Trace back: Traces back from abstract interaction or EUC component to its source.
- ❑ MaptoEUC: Maps an abstract interaction to an EUC component - helps to auto-generate the EUCs.
- ❑ Index Checker: Checker for the consistency of the sequence of abstract interaction and EUC Component.
- ❑ Pattern comparison: Check match of EUC against a pattern (or patterns)
- ❑ Visual difference: visually compare extracted EUC against a chosen or best-fit pattern
- ❑ Map EUC to EUI: generate a EUI rapid prototype by mapping EUC essential interaction groups to EUI items
- ❑ EUI to HTML form: generate prototype HTML form from EUI

Evaluations



- Conducted preliminary evaluations with 8 Software Engineering postgraduate students
- Several work(ed) as developers/requirements engineers in industry
- Participants were given a tutorial on how to use the tool and examples of how an EUC model is derived from textual natural language requirement and how to manage requirement consistency using Marama AI
- Participants rated the usefulness and the usability of the tool together with its inconsistency detection
- The evaluation is conducted using a standard method - Likert scale with a five part answers (1 – not useful to 5 – always useful)

Evaluation #1 – Trace/consistency



Category	Abstract Interaction (%)	Marama Essential (%)	Consistency Management (%)
Very Useful	68.8	59.4	56.3
Always useful	25.0	34.4	37.5
Sometime Useful	6.2	6.2	6.2
Little useful	0	0	0
Not Useful	0	0	0
Save Time	100	100	100

Feedback :

- Abstract interaction:** The tool might be/is constrained by the domains available in the interaction pattern.
- MaramaEssential (EUCs):** Users more familiar with UML diagrams.
- Consistency Management:** Users would like to have more complex consistency checking by the tool.

Evaluation #1 cont..



Category	Automated Tracing Tool (%)	Inconsistency Management (%)
Very Easy	59.4	62.5
Always Easy	37.5	37.5
Sometimes Easy	3.1	0
Little Easy	0	0
Not Easy	0	0
User Friendly	100	100

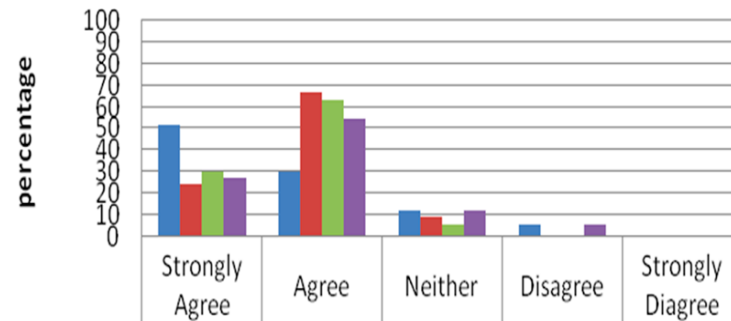
Feedback :

- Automated Tracing Tool:** Users had difficulty understanding layout used by Marama AI.
- Inconsistency Management:** Tool currently provides good warnings but limited ways of resolving the inconsistency (sometimes wrong)
- Multiple models:** useful for dialogue with stakeholders but want other formats

Evaluation #2 - EUC Patterns



User Study Result- usefulness, ease of use, ease of learning and satisfaction



	Strongly Agree	Agree	Neither	Disagree	Strongly Disagree
Usefulness	51.5	30.3	12.1	6.1	0
Ease of Use	24.2	66.7	9.1	0	0
Ease of Learning	30.3	63.6	6.1	0	0
satisfaction	27.3	54.5	12.1	6.1	0

Cognitive dimensions evaluation:
does MaramaEUC support ...?

Cognitive dimension	Strongly Disagree	Disagree	Neither	Agree	Strongly Agree
Visibility	0.0	0.0	0.0	90.9	9.1
Viscosity	0.0	9.1	9.1	72.7	18.2
Diffuseness	0.0	0.0	9.1	63.6	27.3
Hard-mental effort	9.1	27.3	45.5	18.2	0.0
Error-Proneness	0.0	54.5	45.5	0	0.0
Closeness of Mapping	0.0	9.1	9.1	72.7	9.1
Consistency	0.0	0.0	18.2	72.7	9.1
Hidden Dependencies	0.0	0.0	18.2	54.5	27.3
Progressive Evaluation	0.0	0.0	18.2	54.5	27.3
Premature Commitment	0.0	0.0	18.2	45.4	36.4

Summary & Future Research



- Extracting semi-formal models of requirements from natural language text is hard
- Keeping semi-formal models consistent with NL text is challenging
- Checking completeness, correctness of extracted semi-formal models very hard
- Negotiating with stakeholders using natural language text or semi-formal models limited effectiveness c.f. rapid prototypes of interfaces
- Developed MaramaEUC:
 - Supports extraction of semi-formal EUC requirements models from natural language text
 - Supports consistency management between different notations (text, essential interactions, EUCs, EUIs, rapid prototypes)
 - Supports analysis of extracted EUCs against “best practice” EUC patterns
 - Supports visual diffing of EUC vs best practice pattern
 - Supports generation of EUI and HTML form rapid prototypes from EUCs to aid negotiation with stakeholders
 - Evaluation of tool prototypes undertaken with experienced REs

- Want to further extend libraries of interactions, patterns, UIs - support wider domains
- Larger evaluation of the tool including in industrial domain to be undertaken
- Want to assess not only impact of our tool - both in terms of improving the adoption and use of the Essential Use Case method - but also its impact on improving the efficacy of the method itself. This may include integration with other requirements and design modeling views.

Acknowledgement



Support for Massila Kamalrudin PhD is from Ministry of Higher Education (MOHE), University of Auckland and the FRST Software Process and Product Improvement project.

Questions?