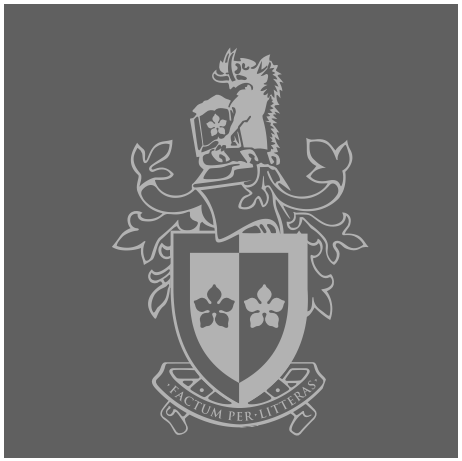
IMPROVING REQUIREMENTS QUALITY VIA ROUND-TRIP ENGINEERING WITH ESSENTIAL USE CASES

MASSILA KAMALRUDIN[1], JOHN GRUNDY[2], JOHN HOSKING[3]

[1]Department of Electrical & Computer Engineering &
[3]Department of Computer Science,

University of Auckland, New Zealand

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



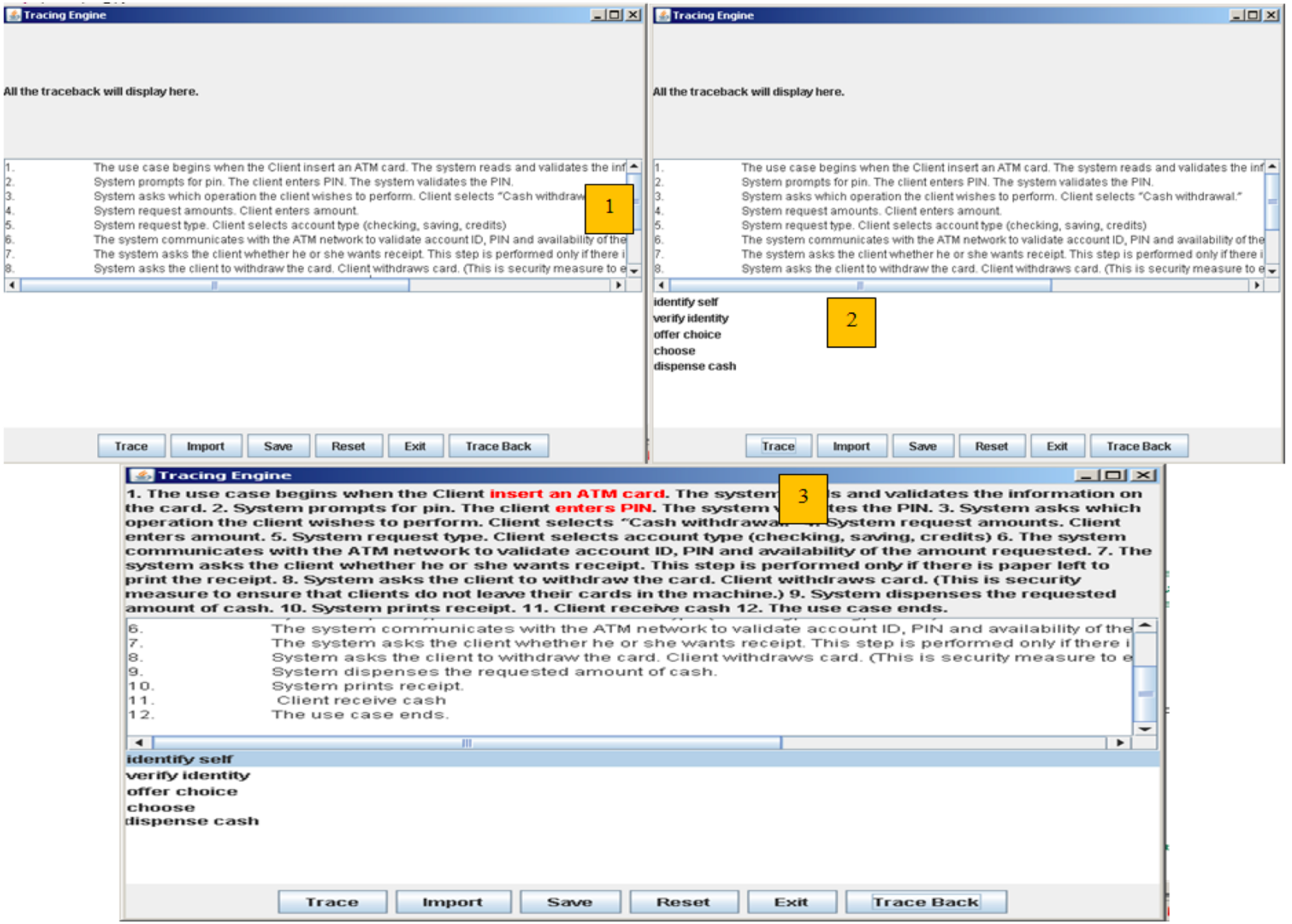
Introduction

- 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

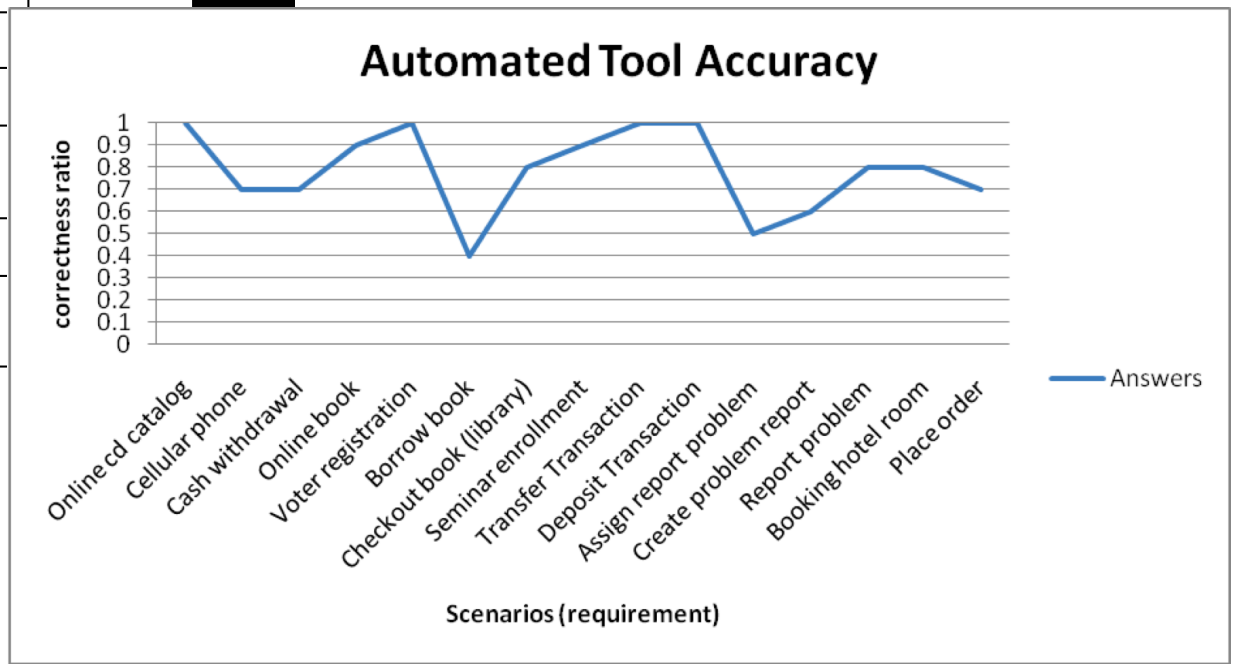
- We conducted a study of experienced requirements engineers (REs) extracting semi-formal requirements from structure natural language documents
  - 11 people, most experienced industry REs
  - Basic natural language requirements -> (Essential) use cases
- They did really, really badly!!
- An initial 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



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 checking

  between 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)

Natural language requirements

"Essential Use Case" (EUC)

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

"Essential interaction"

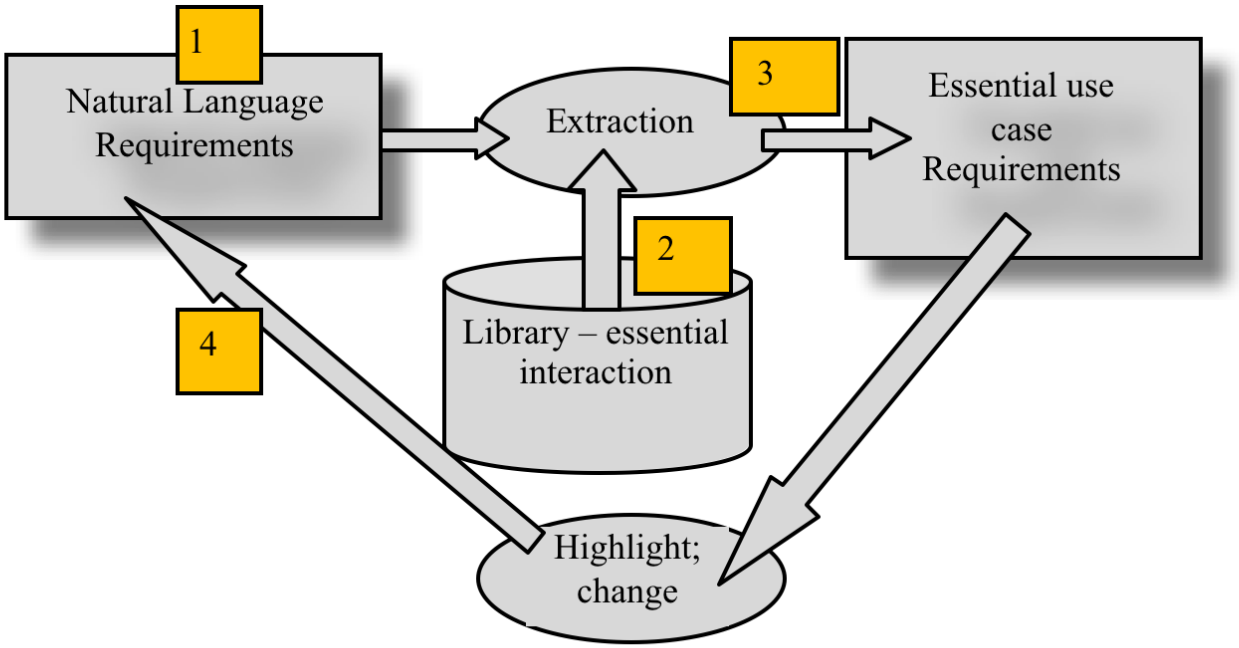
User intention	System responsibility
1. Identify self	
	2.Present help options
3.Select help option	
	4.Request description
5.Describe problem	
	6.Offer possible solutions

Essential requirement ("Abstract interaction")

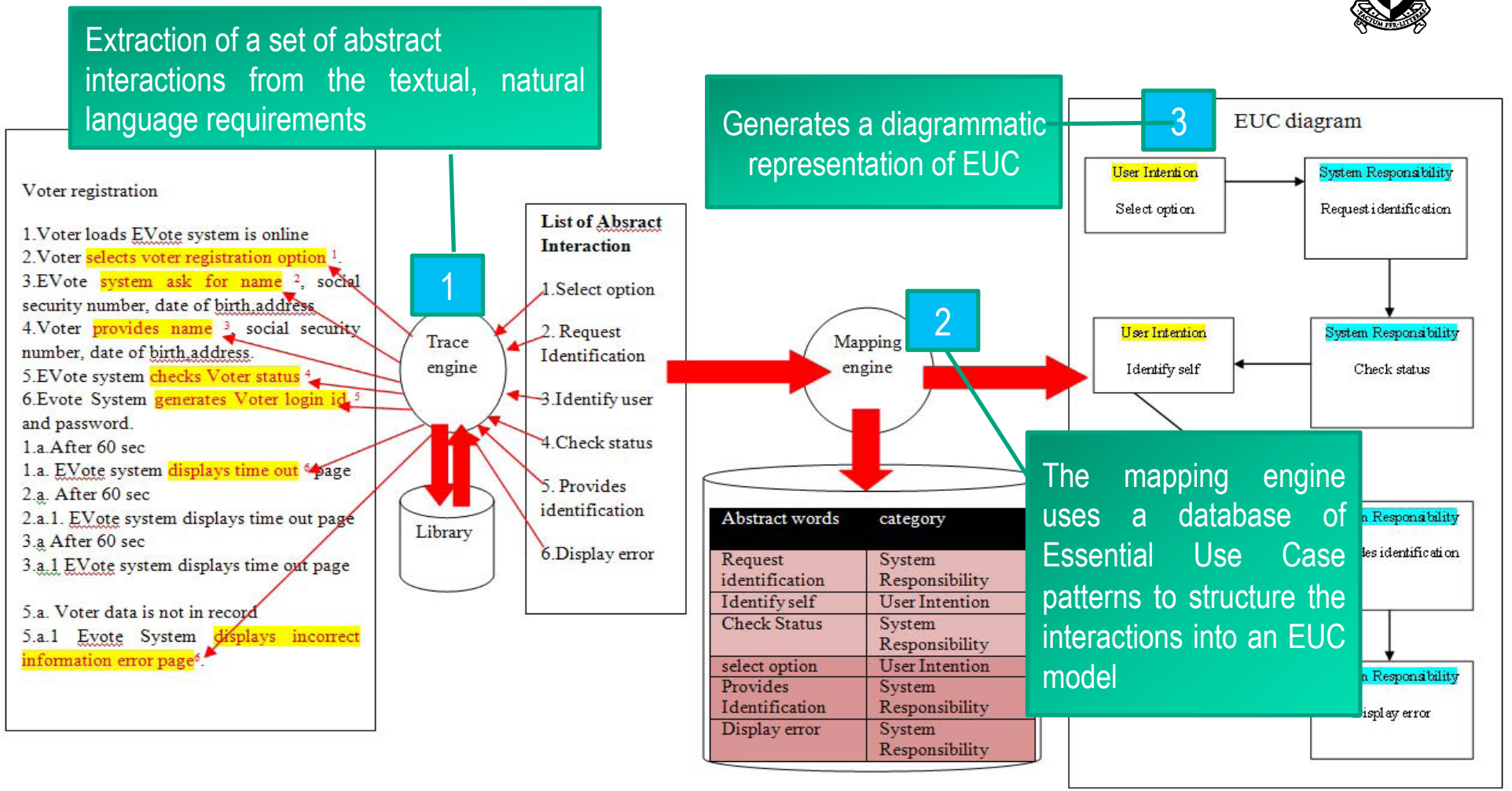
Example EUC abstract/essential interactions

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)

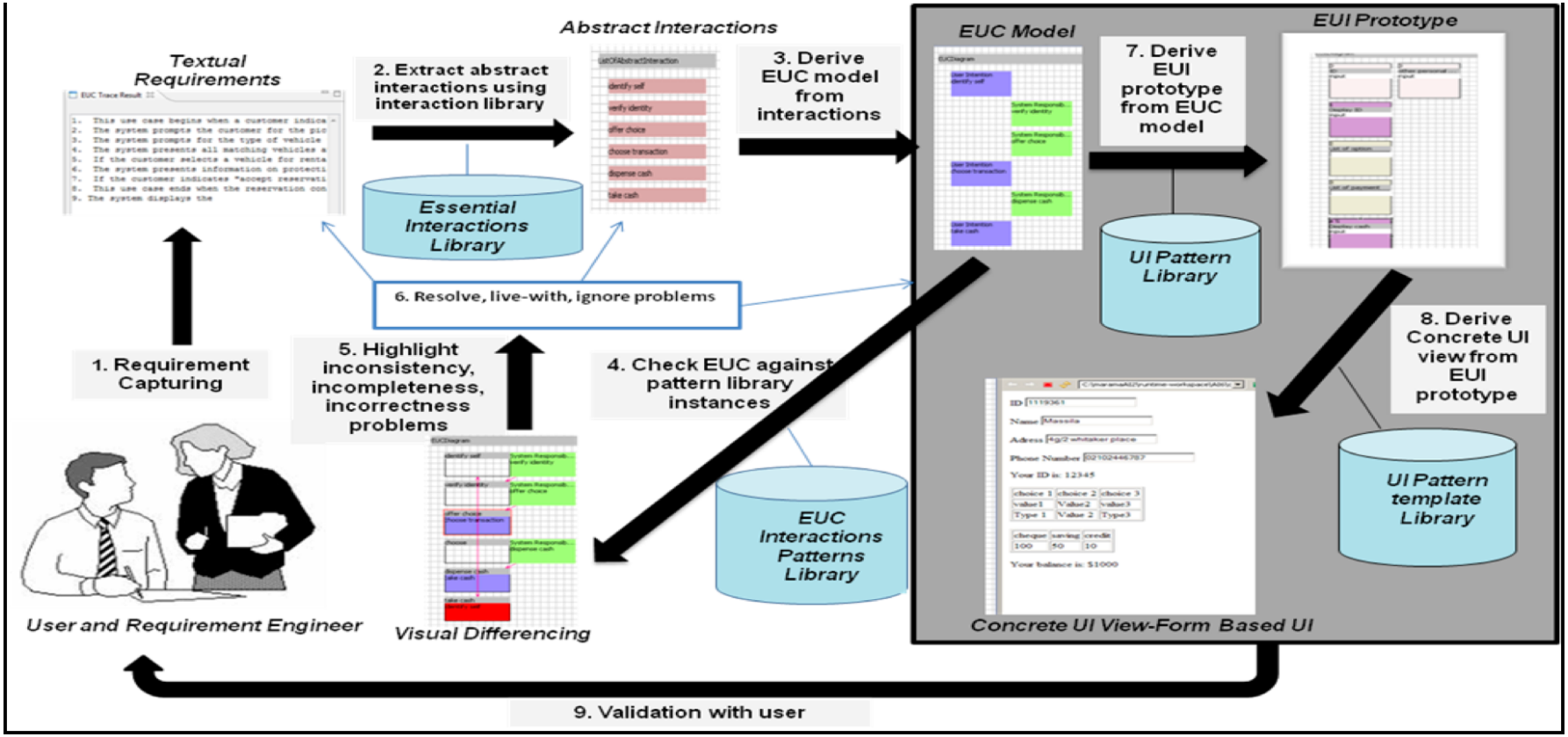


Our Approach (2)



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

Our Approach (3)

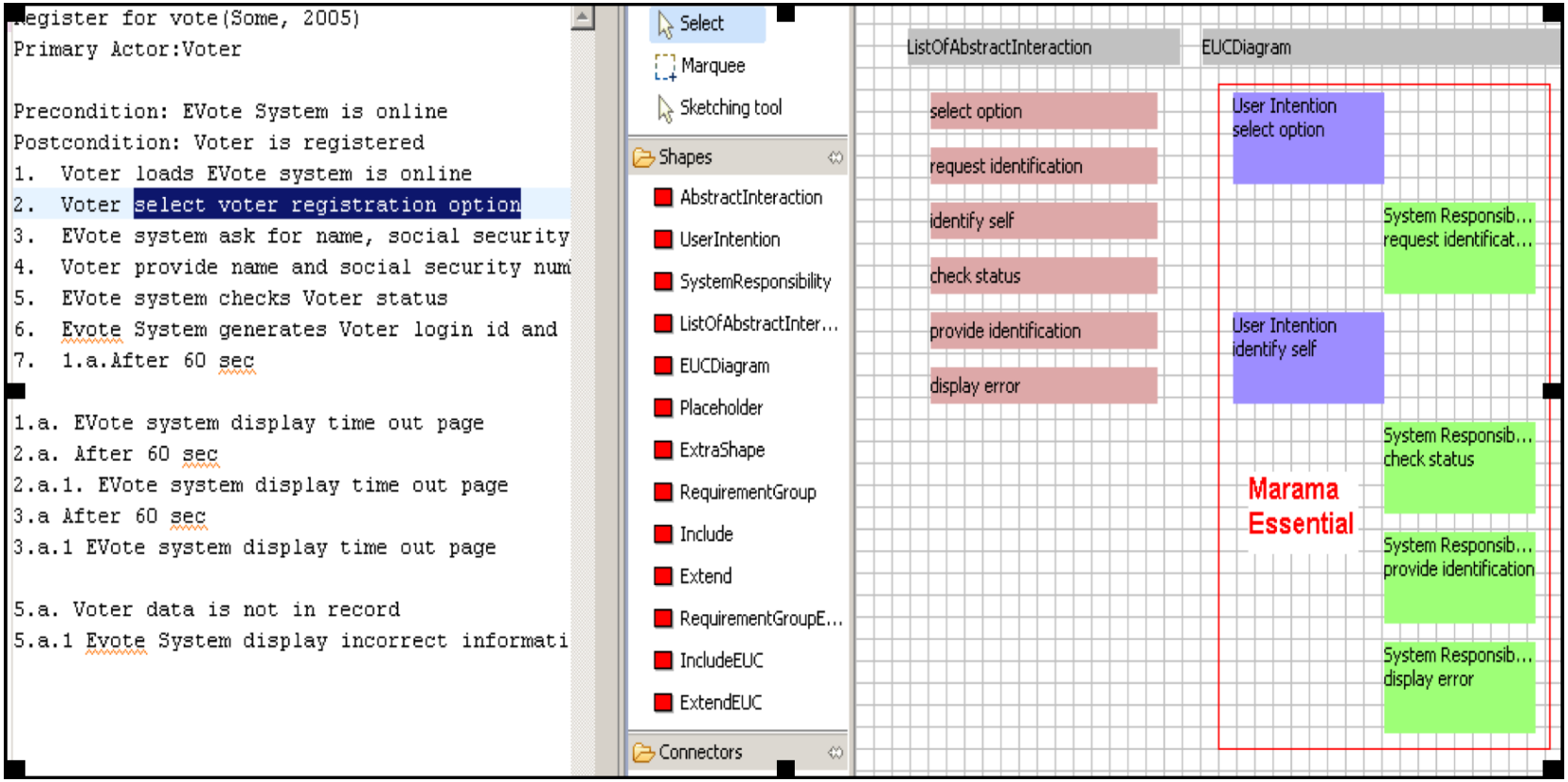


Validating EUCs with EUC pattern library (left); and supporting dialogue with the stakeholders via EUIs and form-based rapid prototyes (right)

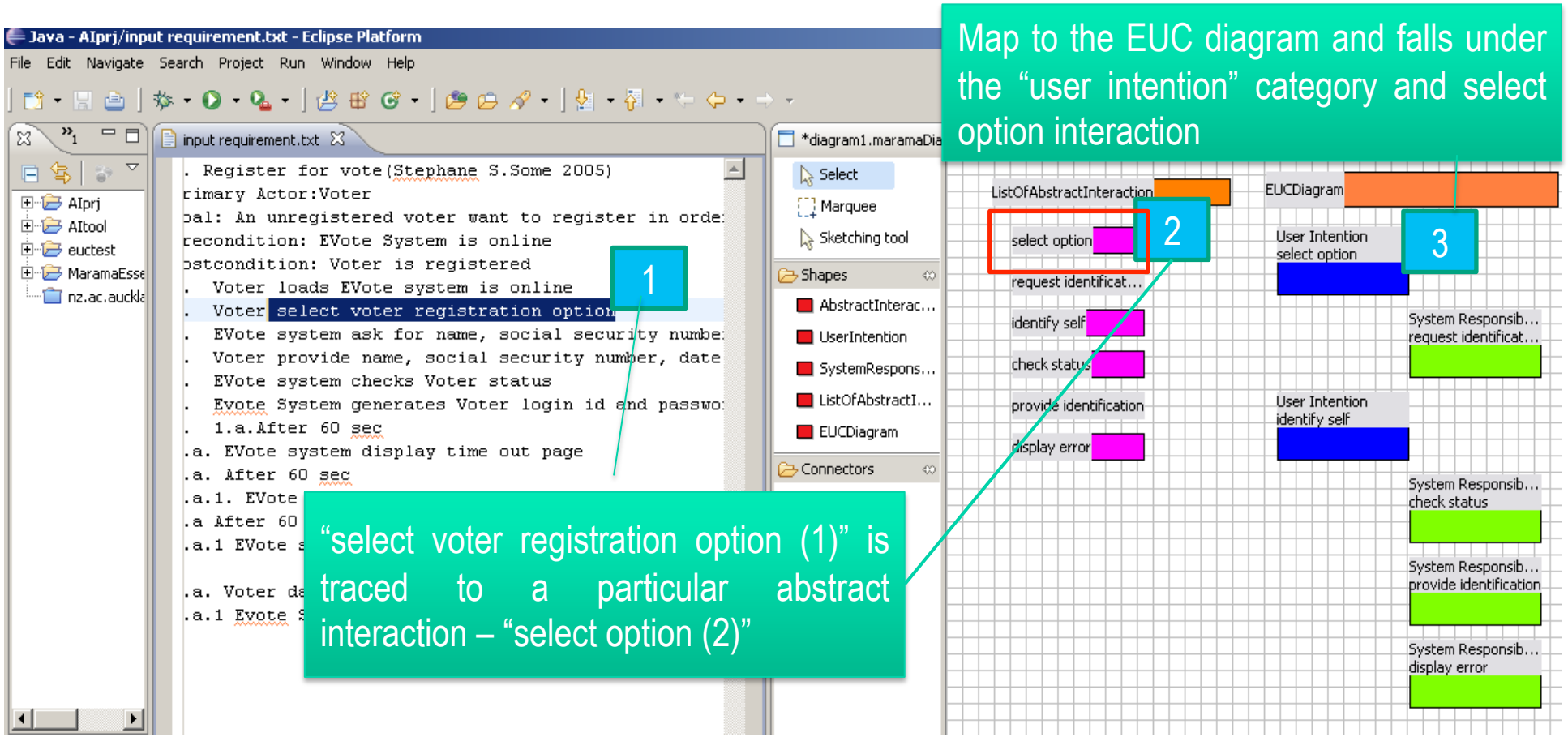
Tool Support

- 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

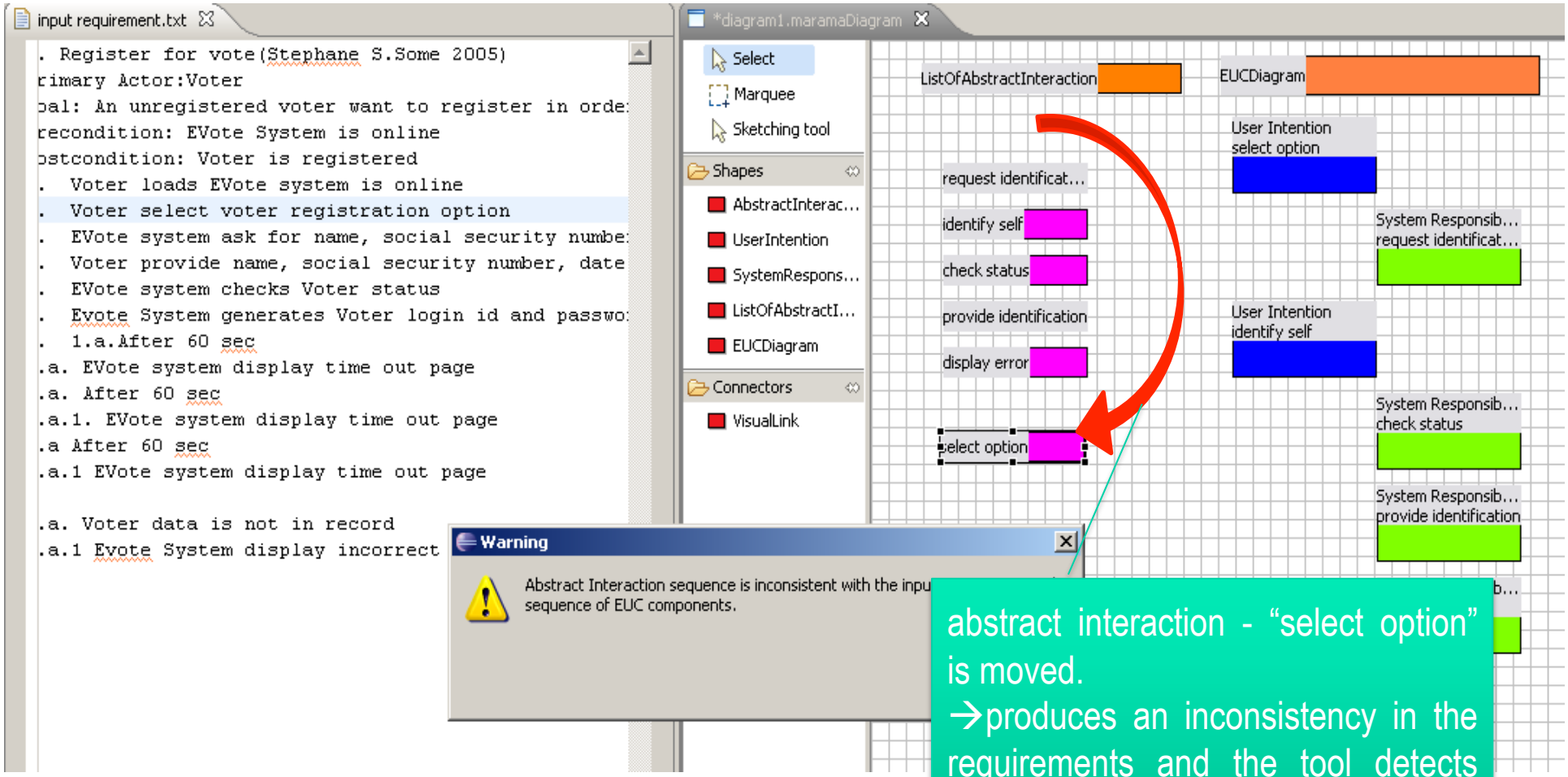


Tracing Abstract Interactions

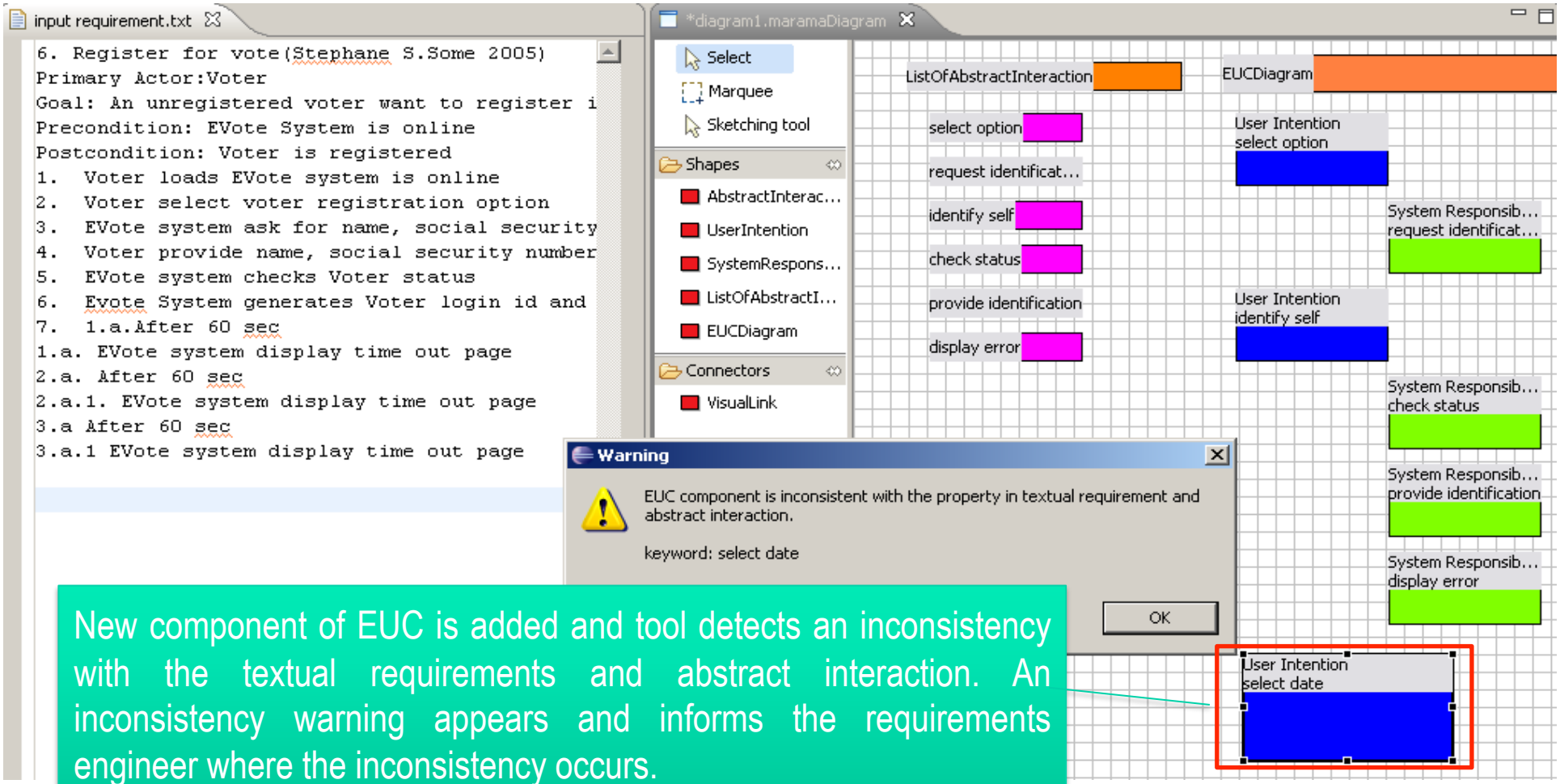


Tracing an abstract interaction from textual requirement and mapping to the Marama Essential representation

Inconsistency Checking



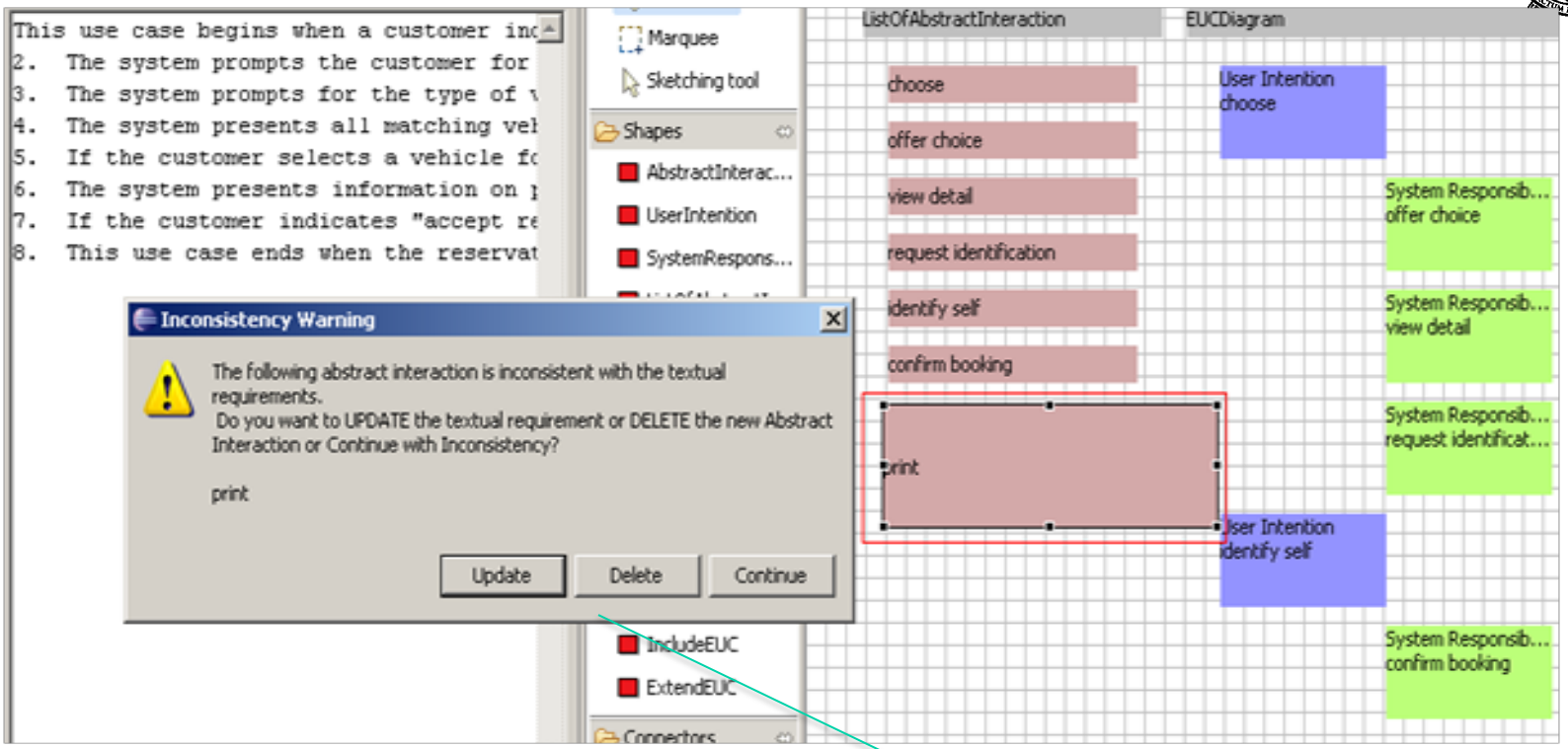
Inconsistency Checking (2)



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.

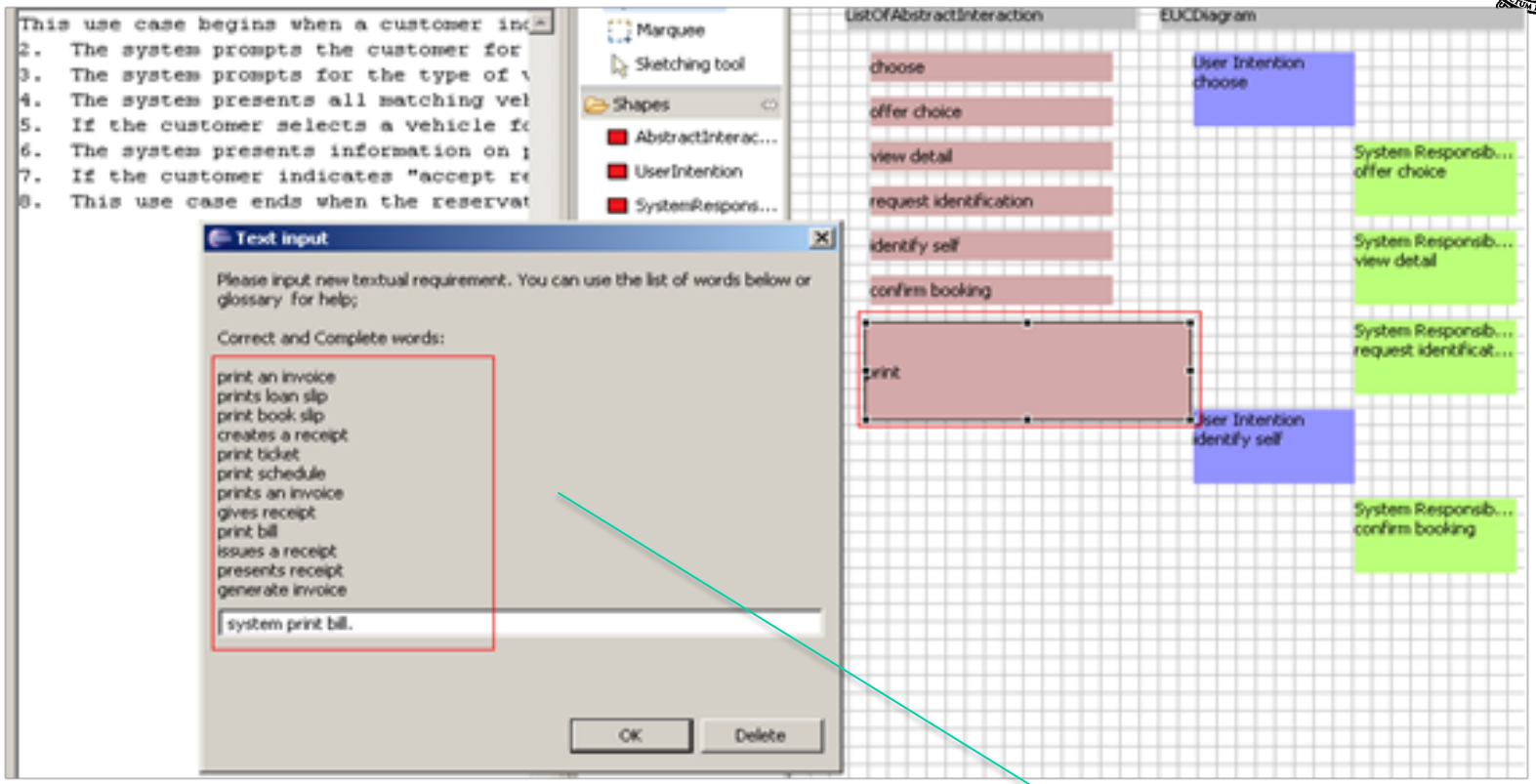
This warning shows dependencies that occur between the textual requirement, abstract interaction and EUC diagram.

Text/Interaction patterns



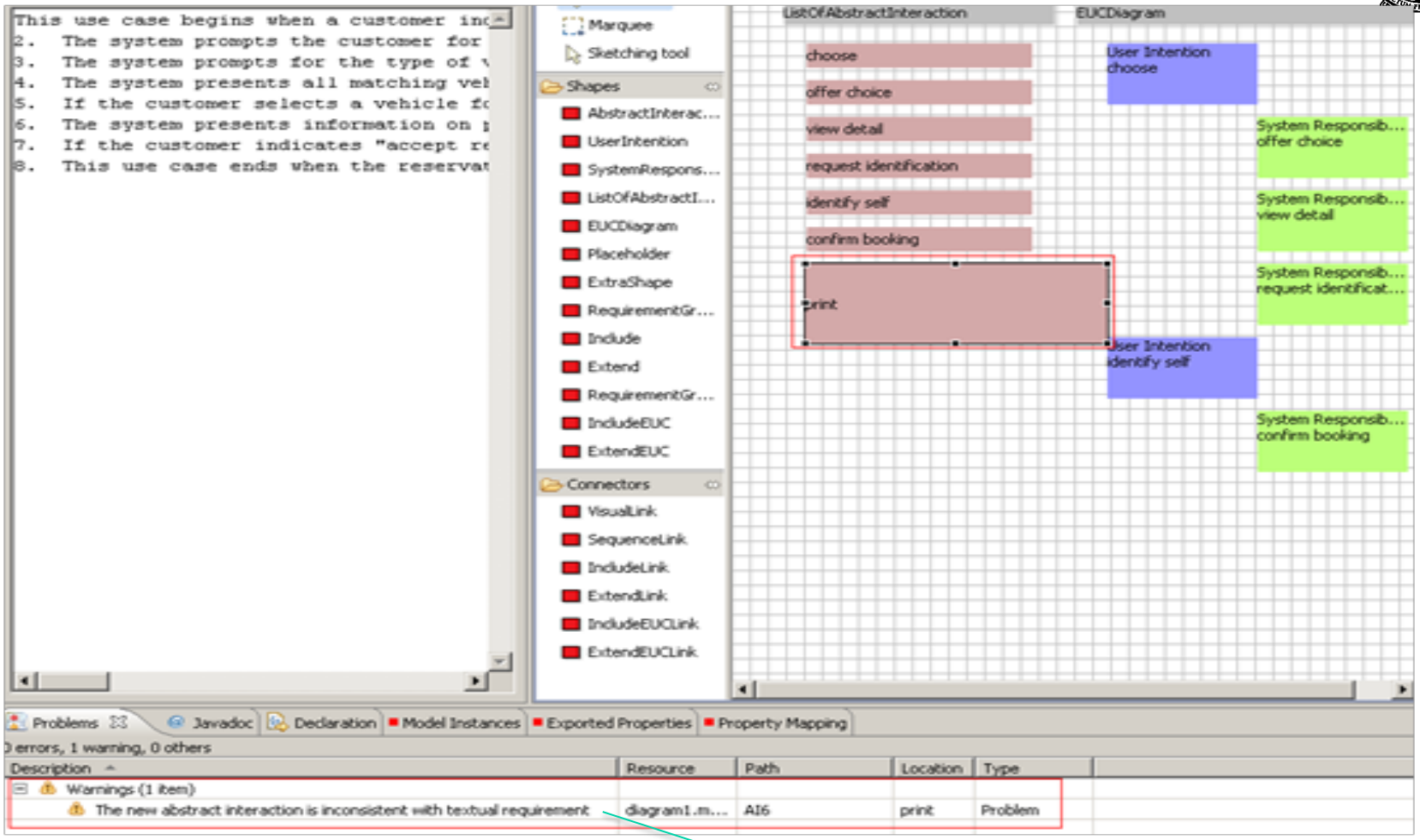
Detect textual essential interactions are inconsistent with abstract interactions
-option to change text
-option to change abstract interaction
-option to ignore and fix later (or not) ☺

Interaction -> text



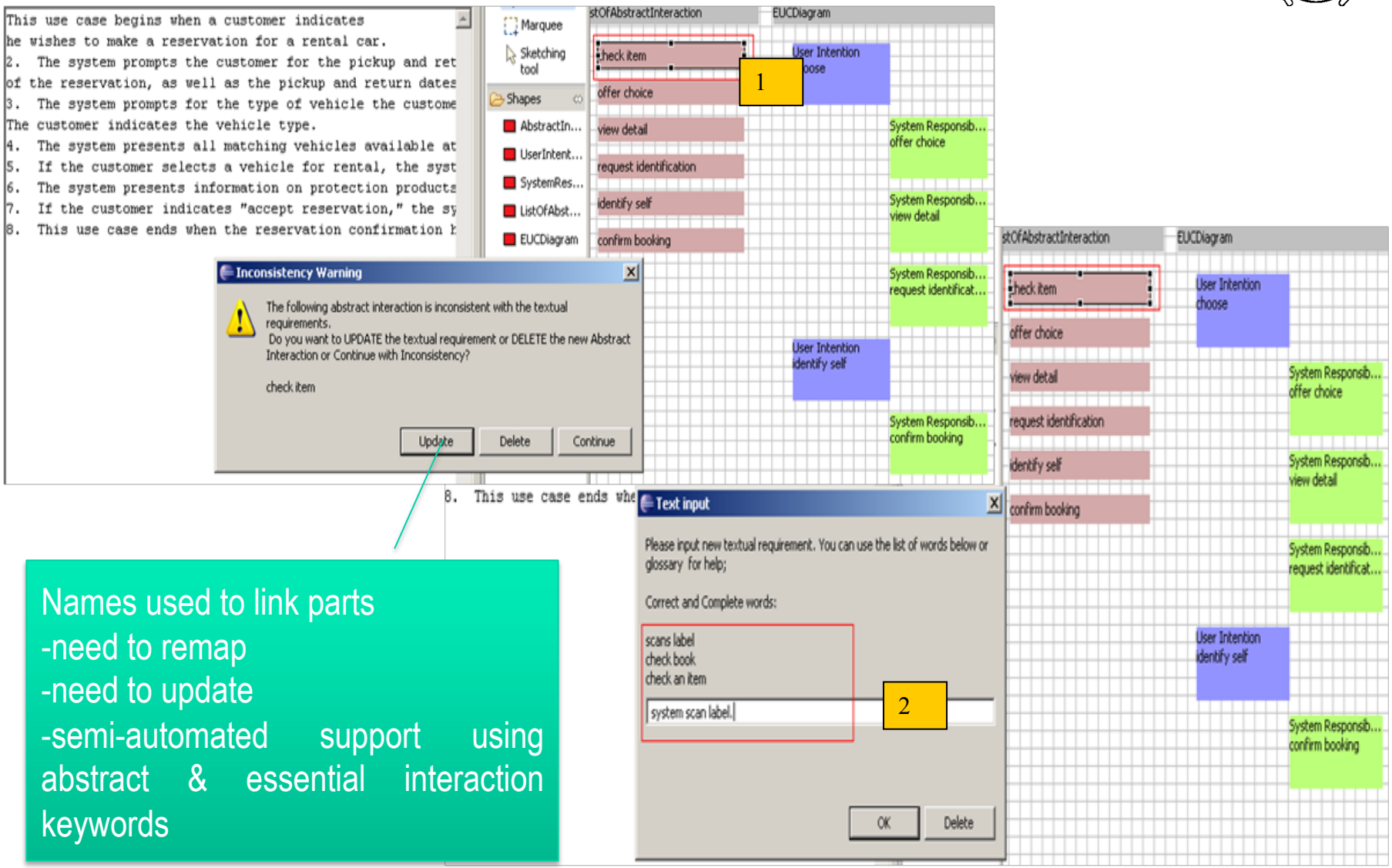
Update text.
-manual update
-semi-automatic update from abstract interaction -> essential interaction

Ignoring inconsistency…



Ignore (for now)
-Eclipse problem marker tracks

Renaming items (eg choose->check)

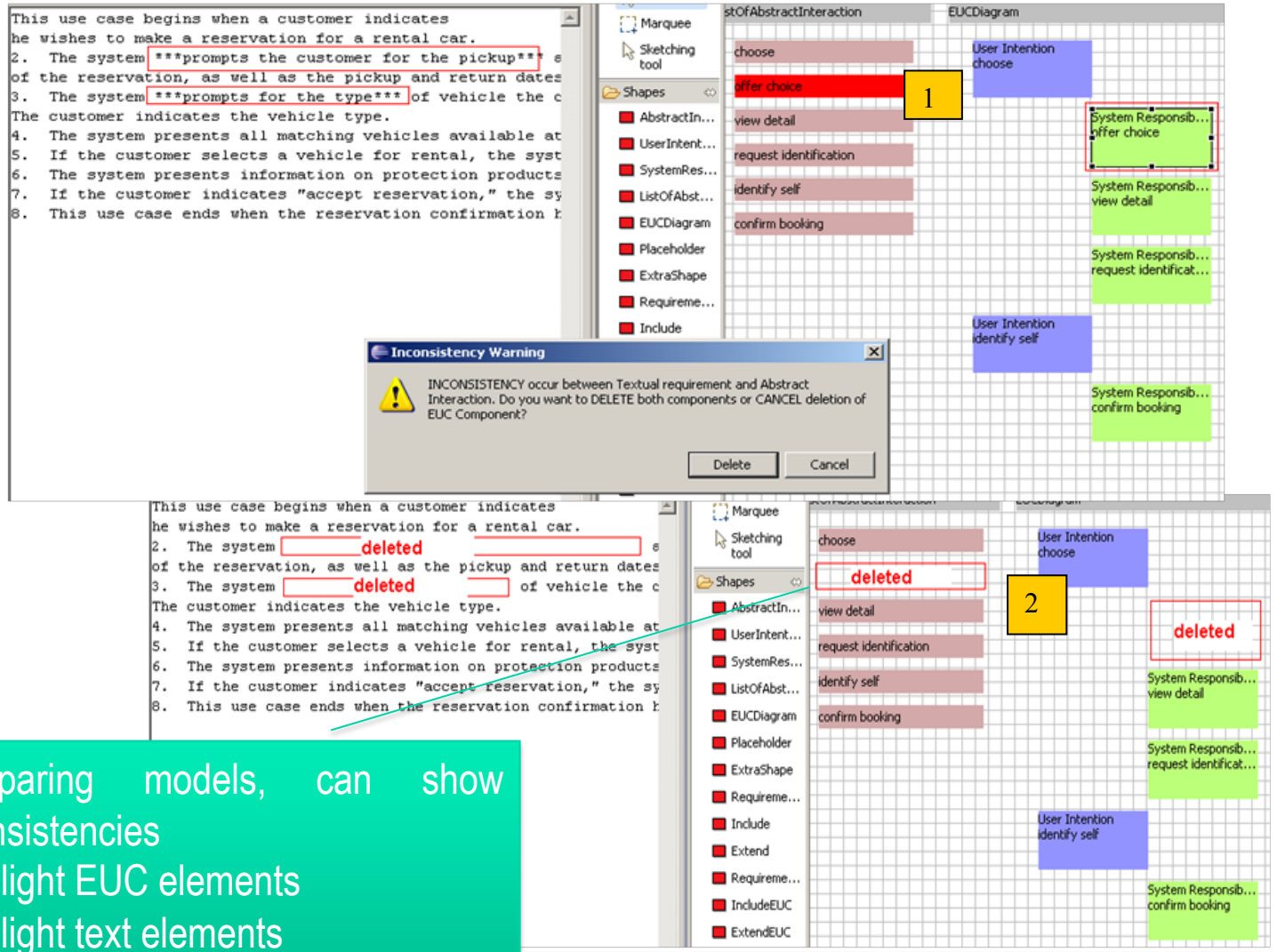


Names used to link parts
-need to remap
-need to update
-semi-automated support using abstract & essential interaction keywords

EUC Patterns – a few examples…

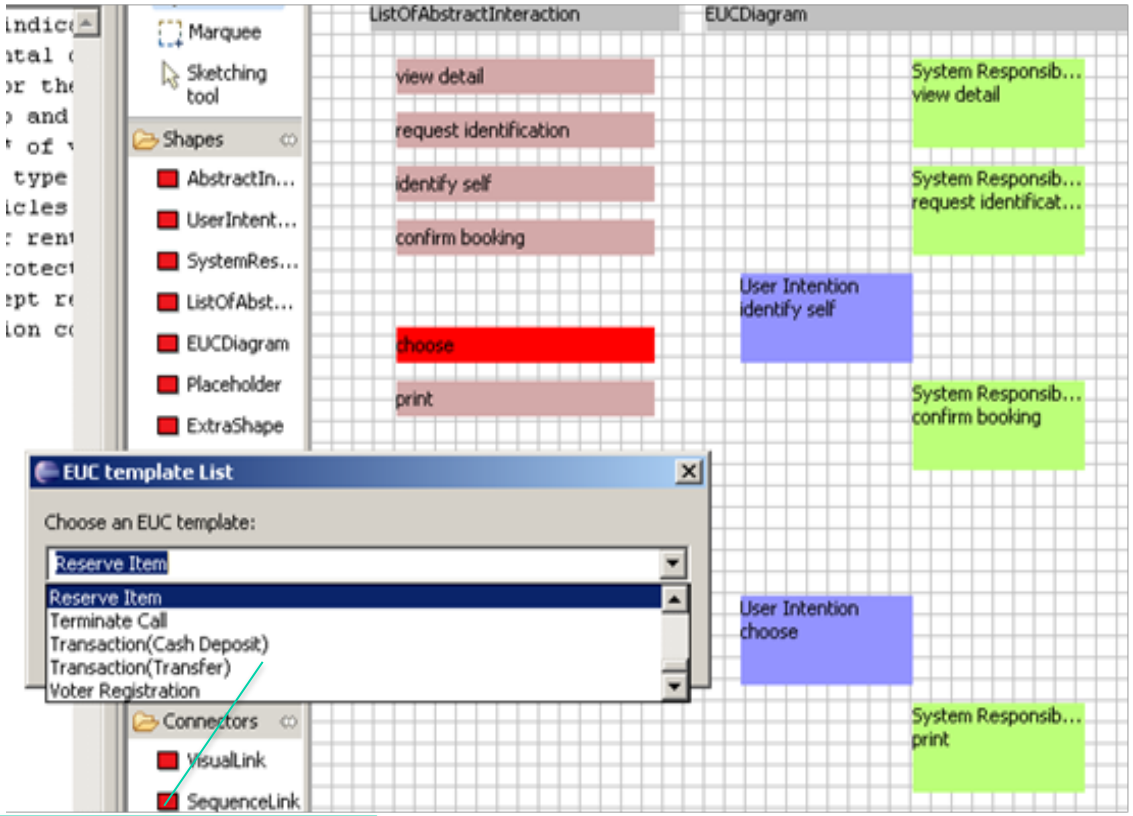
Scenarios/ Use Case stories	User intention Abstract Interaction	System responsibility Abstract Interaction
Reserve item	choose	
		offer choice
		view detail
		request identification
	identify self	
		confirm booking
Purchase item	choose	
		check status
	identify self	
	provides detail	
		verify identity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

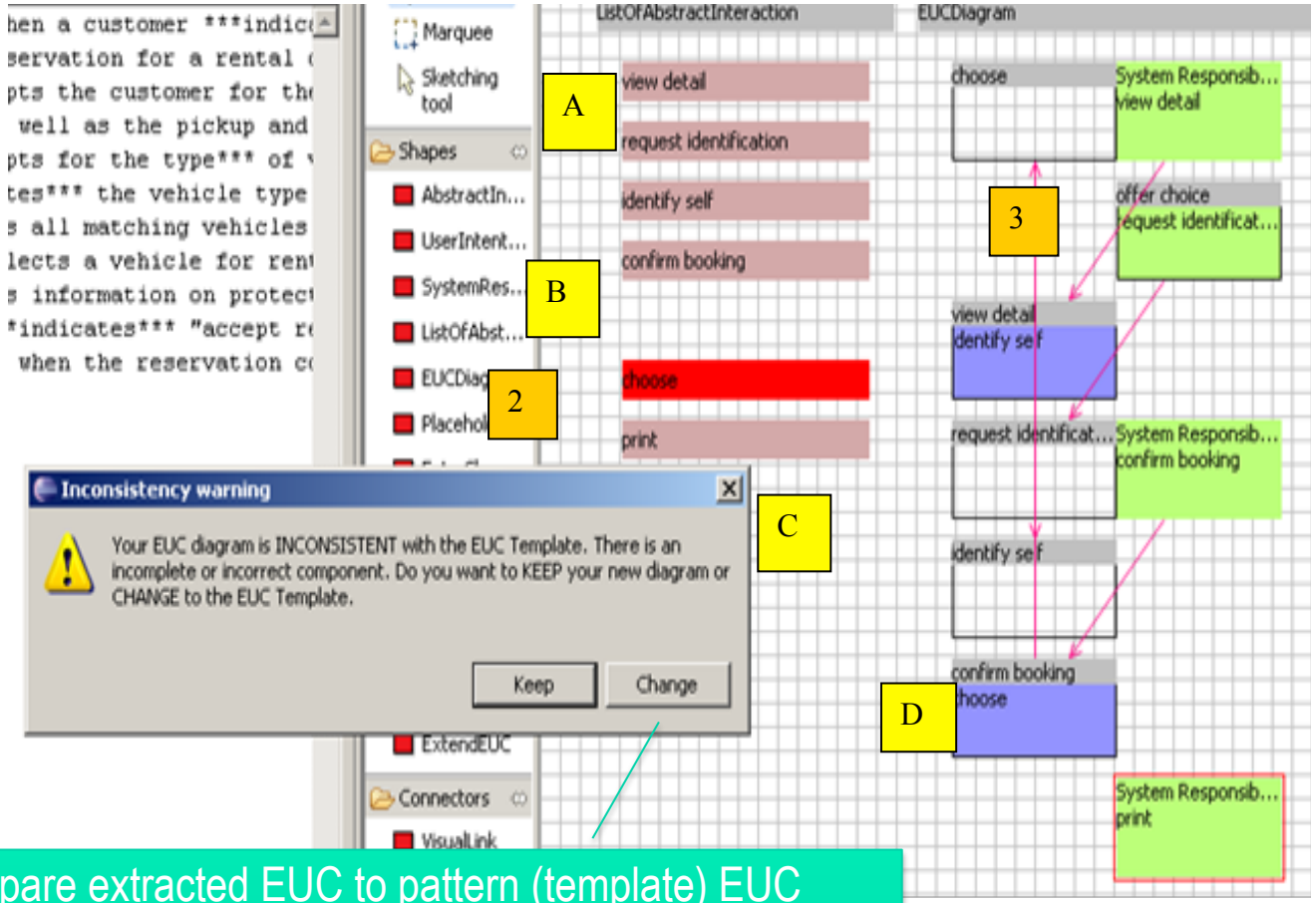
Visual diffing – correctness/completeness



Choose EUC pattern to compare to extracted pattern

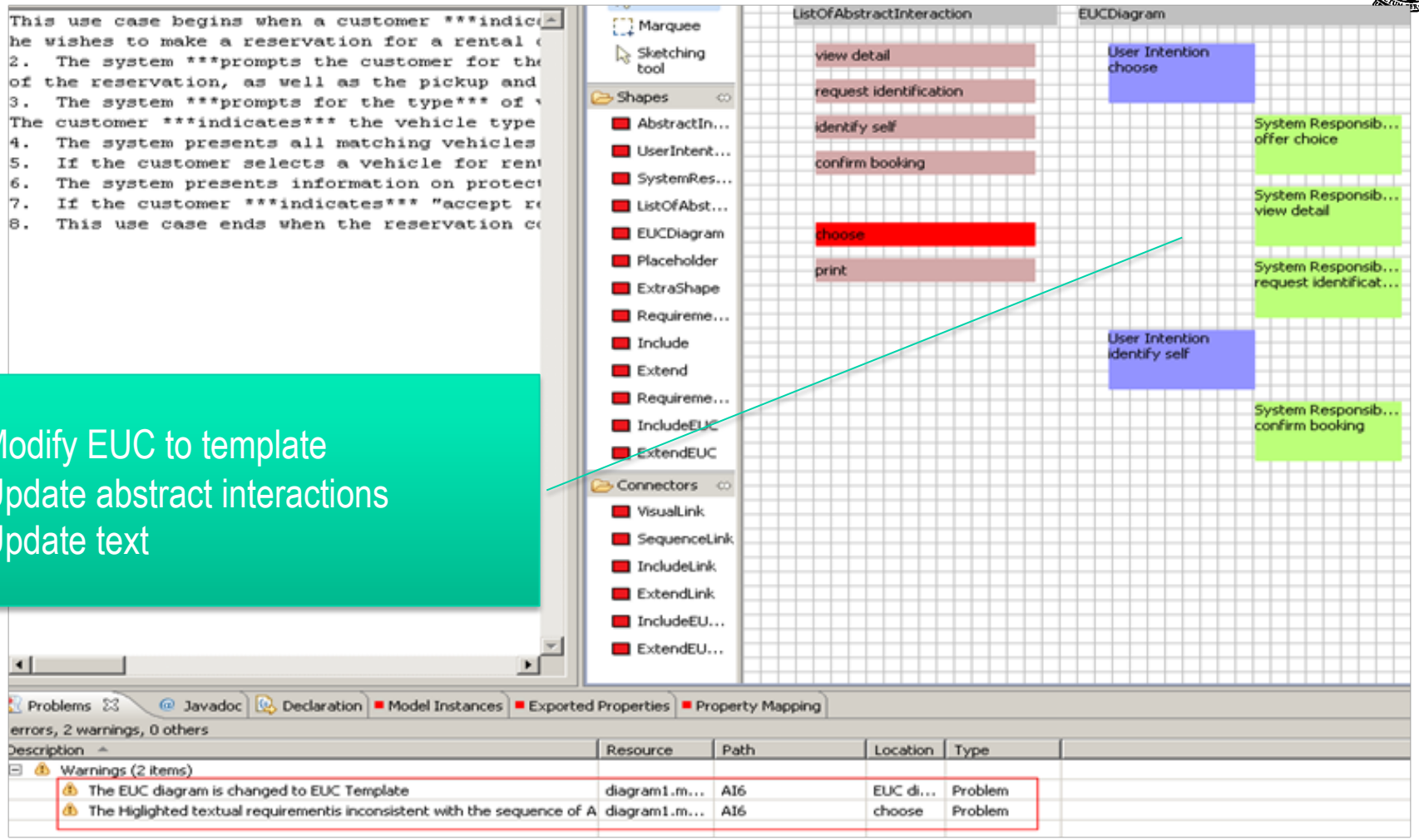
Detect “best fit” EUC pattern to compare

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

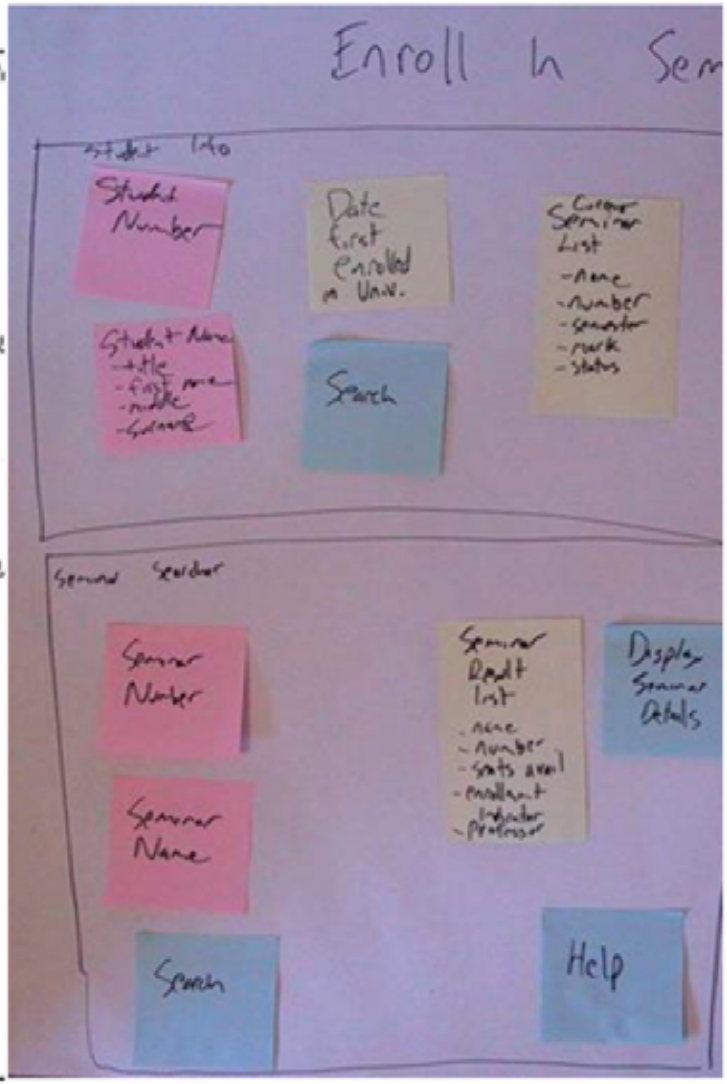
Update based on template



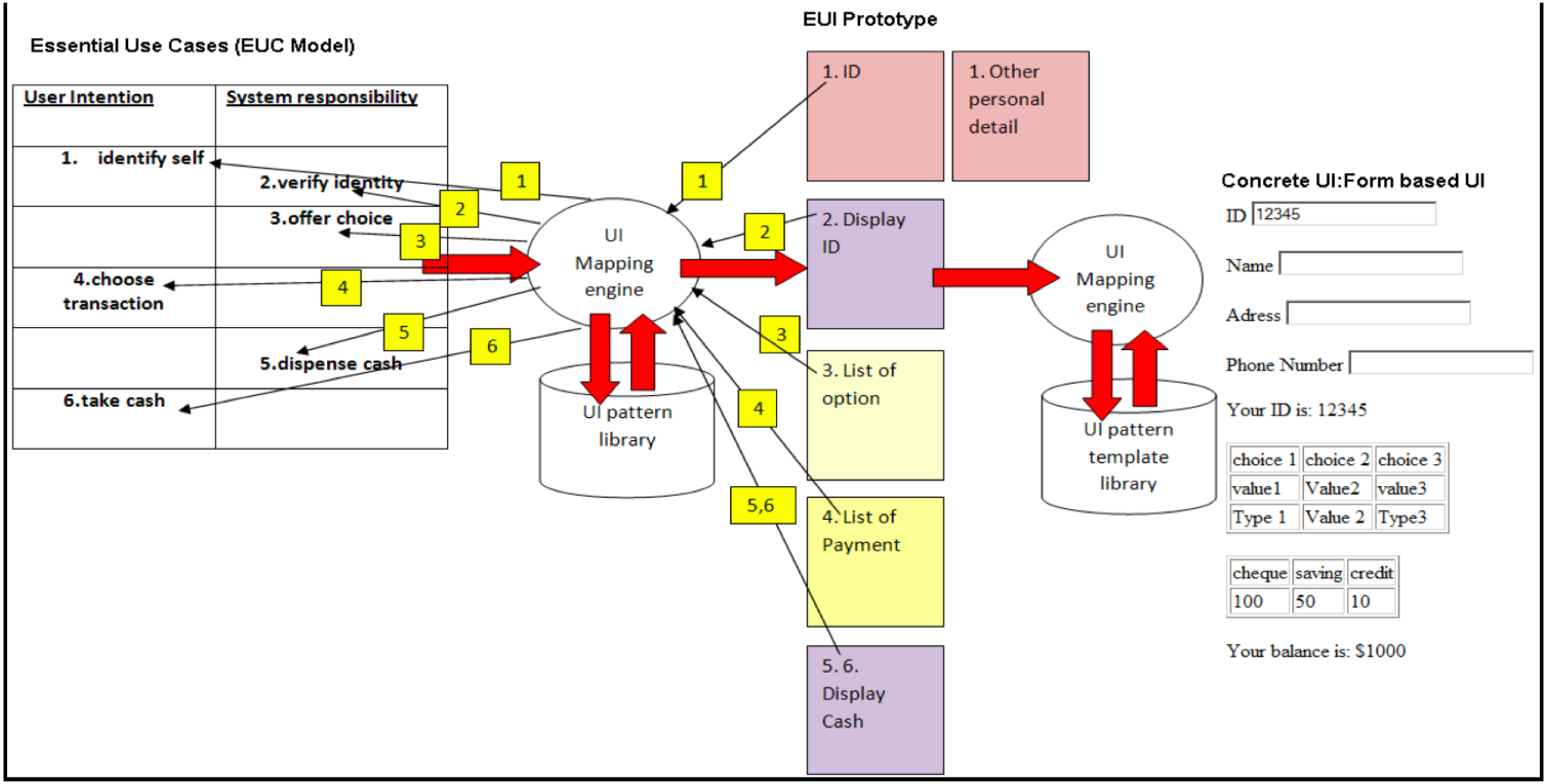
Modify EUC to template
Update abstract interactions
Update text

EUI rapid prototypes

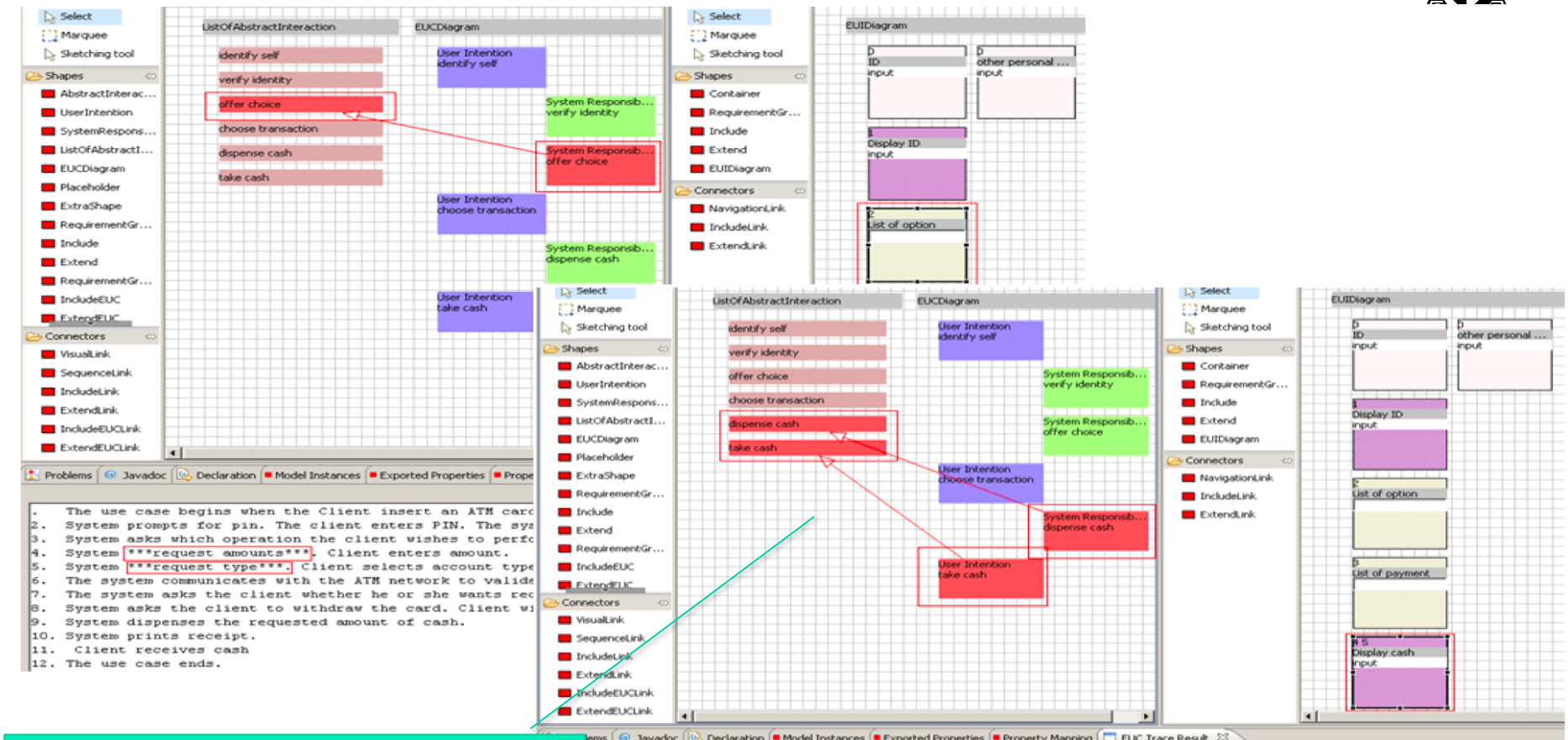
User Intention	System Responsibility
Student identifies himself	Verifies eligibility to enroll via *BR129 Det… Eligibility to Enroll.*
	Indicate available seminars
Choose seminar	Validate choice via *BR130 Determine St… Eligibility to Enroll in a Seminar.*
	Validate schedule fit via *BR143 Validate Seminar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



EUI Generation

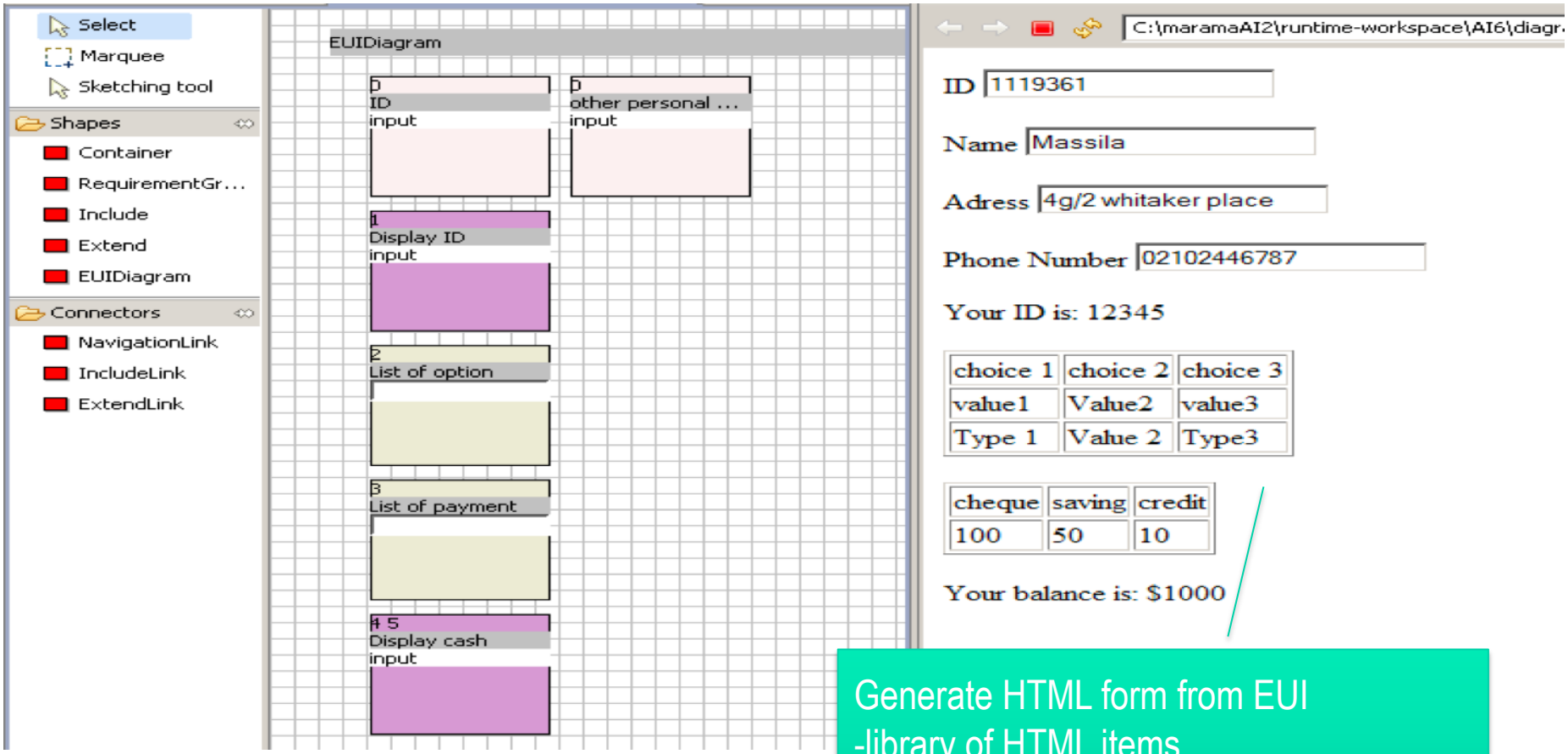


Generate an EUI from an EUC...



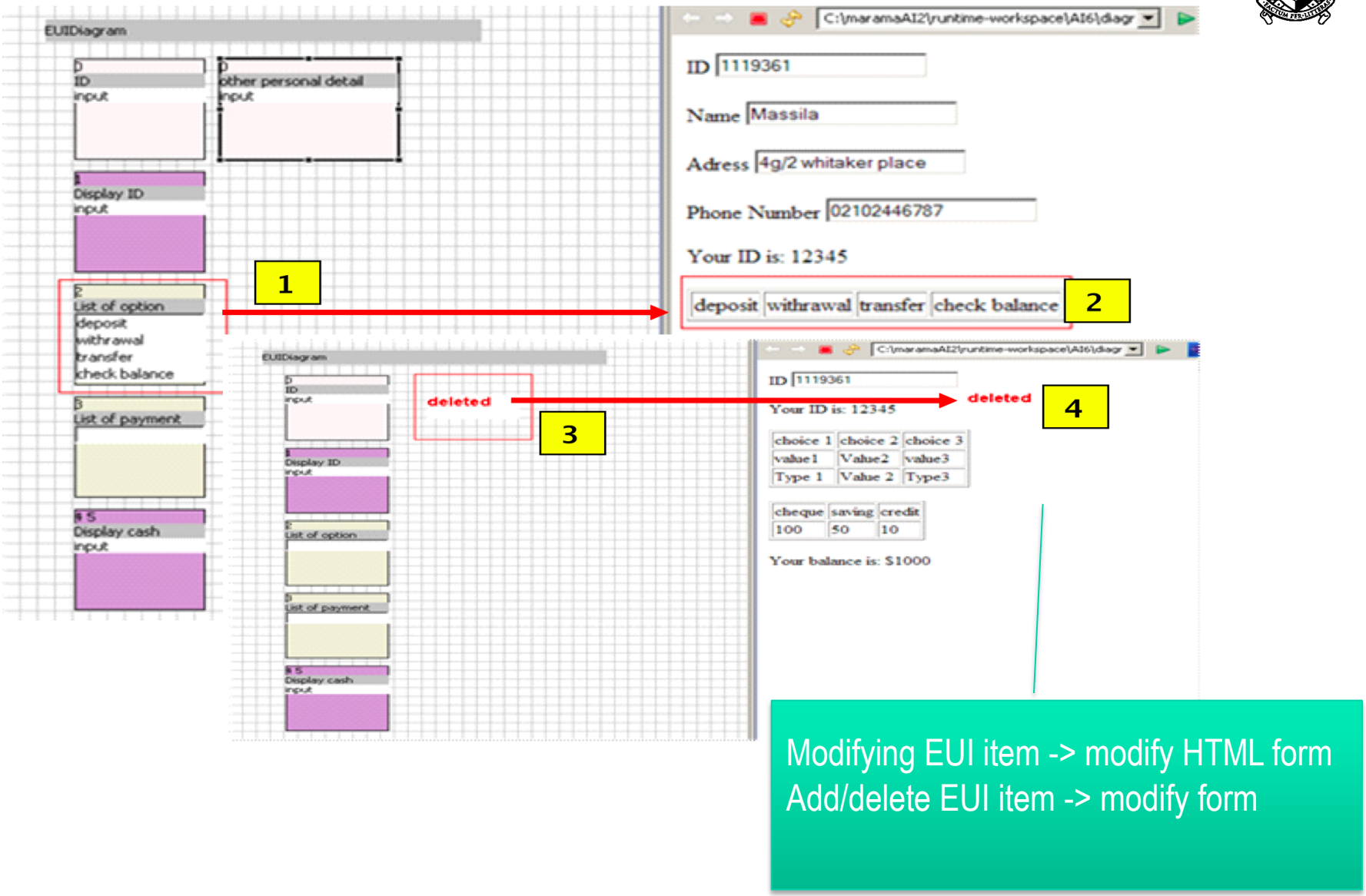
Take EUC and generate EUI model
-set of EUI element patterns
-map EUC items to EUI items
-generate EUI layout
-allow editing of both (plus text)

Generating an HTML Form from an EUI

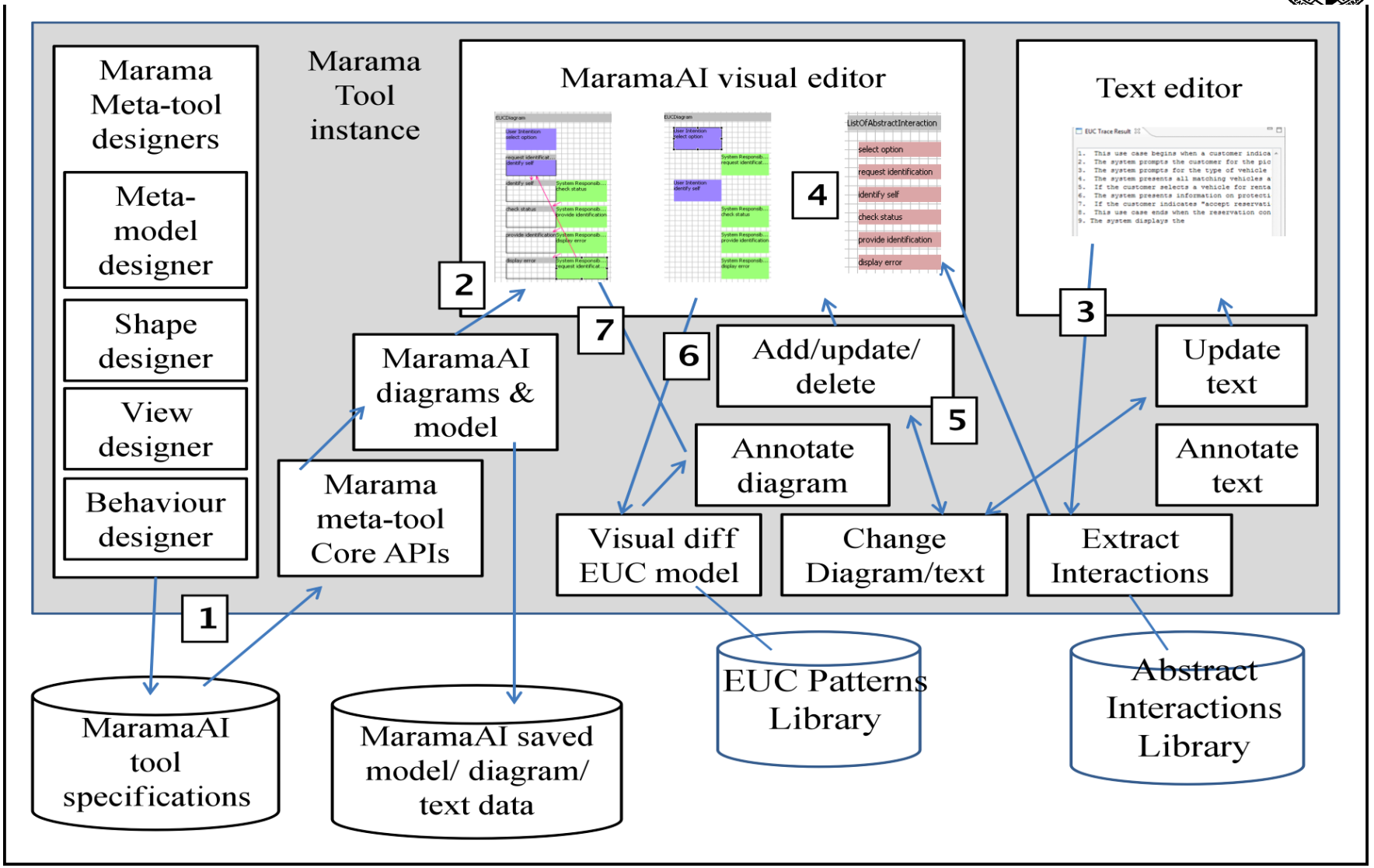


Generate HTML form from EUI
-library of HTML items
-library of EUI->HTML items
-layout & sizing heuristics
-can interact with form to “try” UI
[-can edit & keep consistent]

Consistency management, checking



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

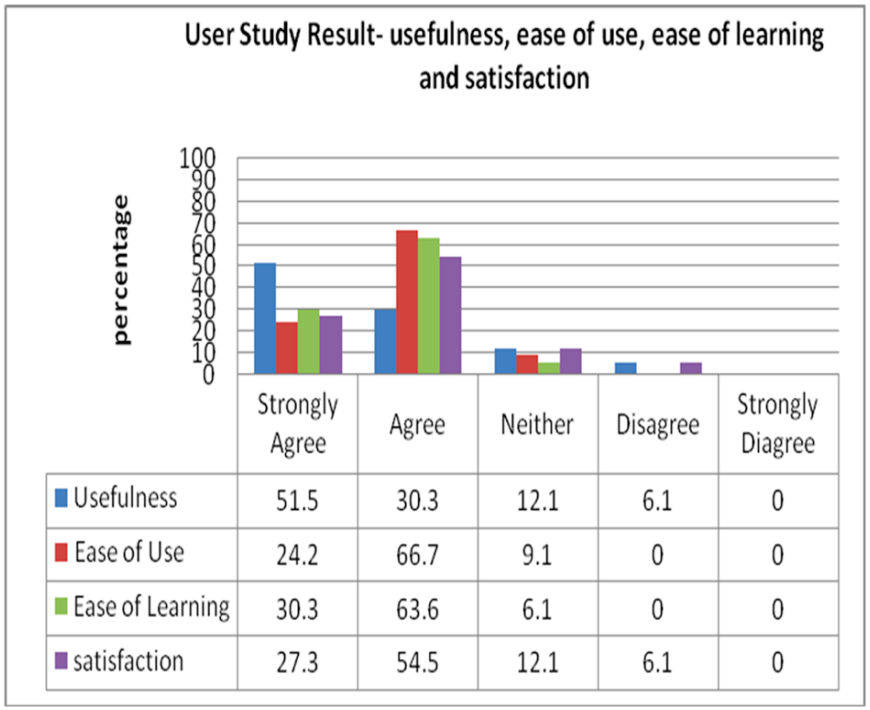
Feedback :

❑**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



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 interaitons, 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?