Object-Oriented Software Engineering
Conquering Complex and Changing Systems

REQuest Tutorial

November 9, 2001

Odds & Ends

Object-Oriented Software Engineering for the lecture is now available at Computer Books at the Obelisk

- First 35 buyers: DM 89,90
- Subsequent buyers: DM 99,95

- List price (e.g., amazon.de): DM 107,03 + delivery

Tutorial outline

- Requirements engineering
  - **Summary from last 2 lectures**
  - **Levels of descriptions**
  - **Guidelines**
- Requirements engineering process for TRAMP & ARENA
  - **Elicitation & review with REQuest**
  - **Analysis with TogetherJ**
- REQuest: SuperMarket example
- Summary & next steps

Tutorial goal

Goal of this tutorial

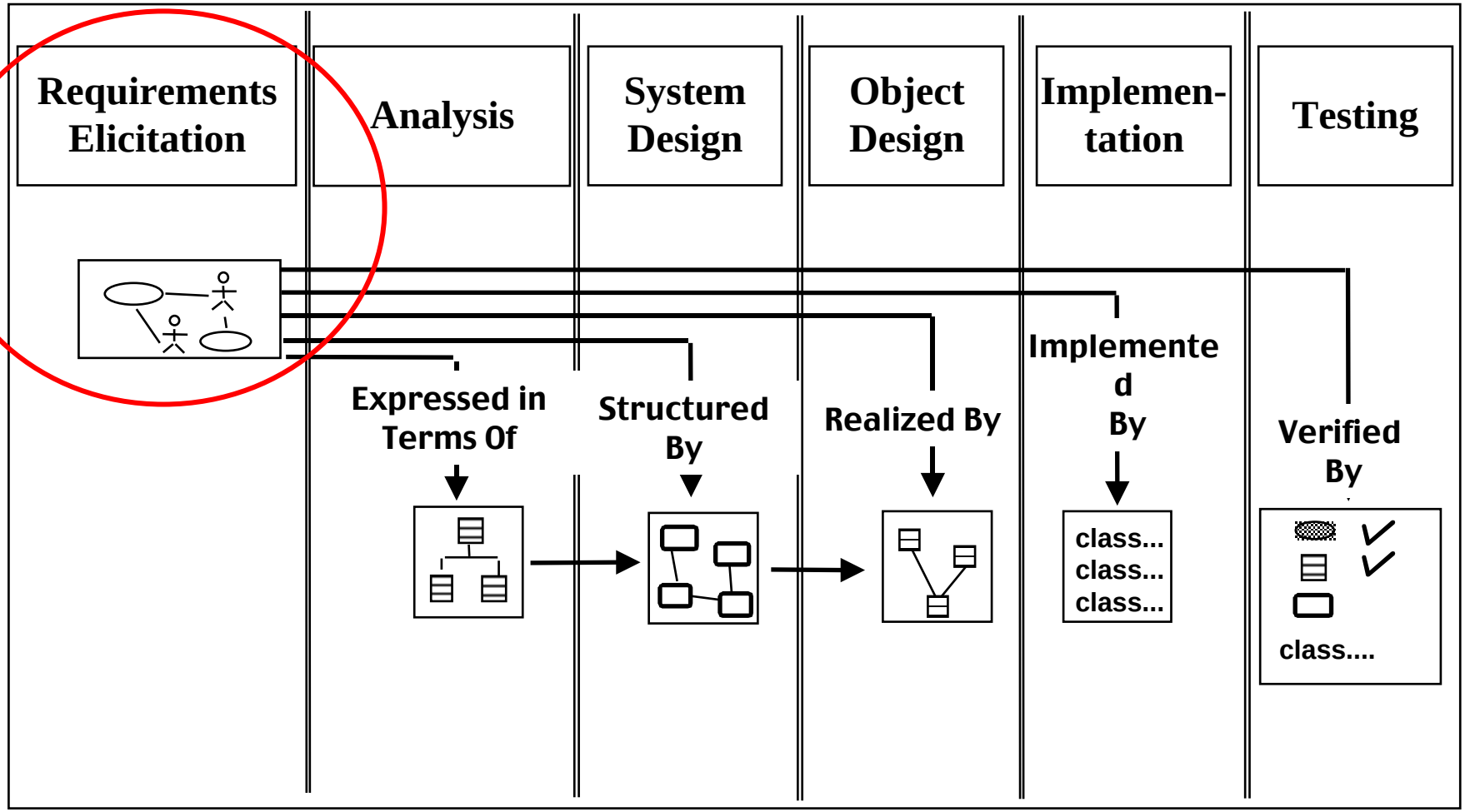
- To operationalize ***requirements elicitation*** concepts you have learned in the lecture for an actual tool and set of guidelines.

Desired outcome

- To provide you with sufficient skills so that you can do a first requirements elicitation exercise.
- To point out the range and difficult of the issues you may face during requirements elicitation.

Note: ***analysis*** will be the topic of next week's lecture and tutorial.

Requirements Engineering Summary



RAD

A RAD includes 3 descriptions:

Requirements Elicitation:

Use case model →

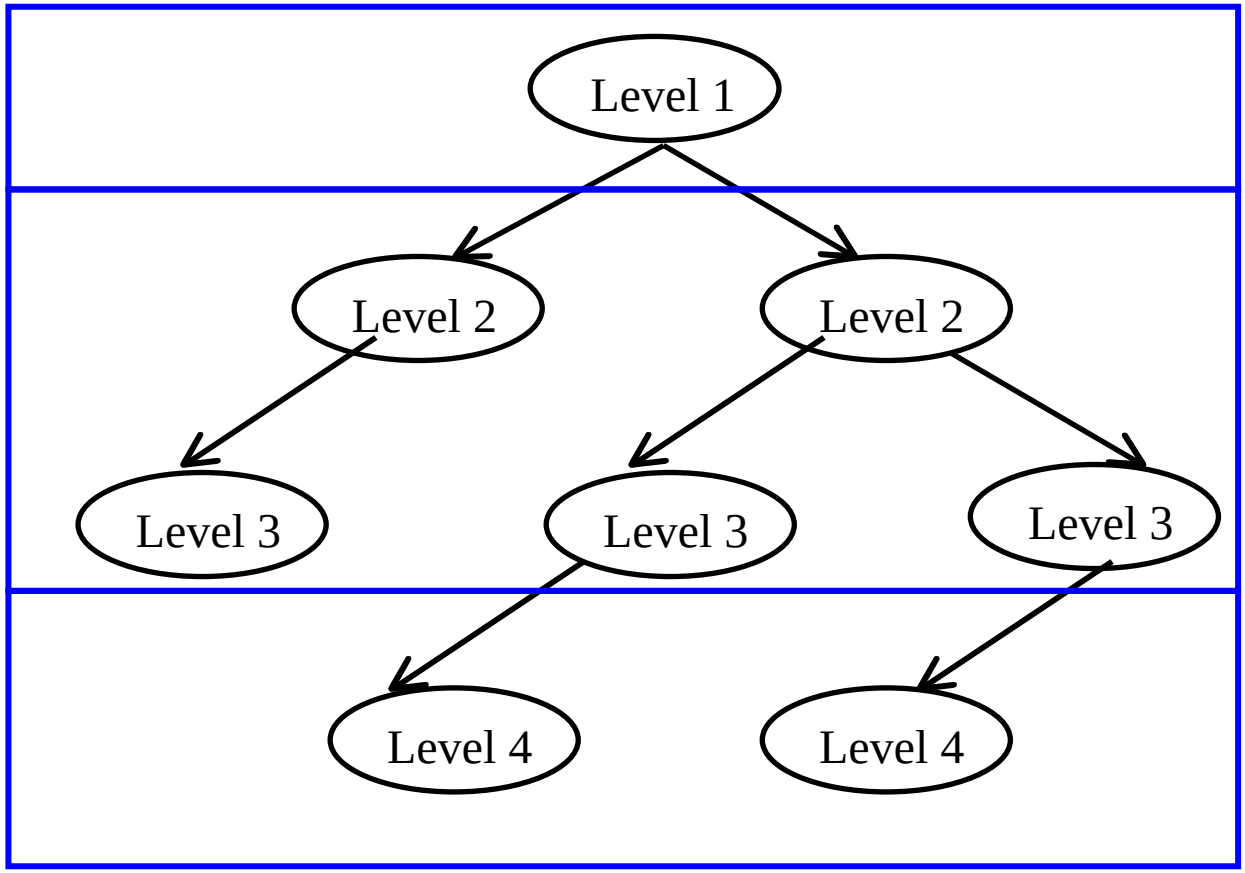
- *Requirements:* What do users do?
- *Interactions:* How do users use the system to accomplish their work?

Analysis:

Requirements analysis model (object model) →

- *Specification:* What does the system do?

Levels of descriptions



User tasks
describe domain

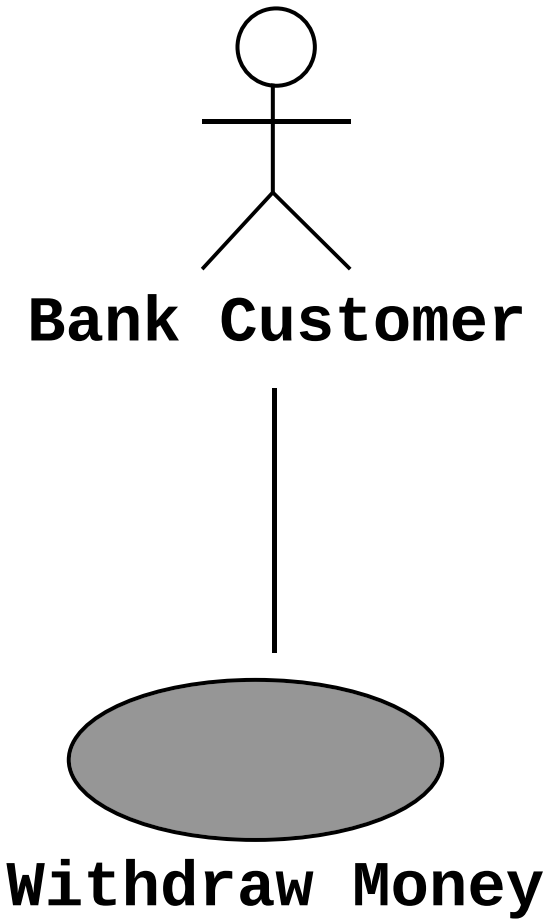
Use Cases
describe interactions

Services
describe system

Requirements elicitation activities

- Define the boundary of the system:
  - **Identify and describe actors**
- Define the needs of the user
  - **Describe one or more user tasks per actor**
- Describe the interactions between the actors and the system
  - **Describe one or more use cases per user task**
  - **Exceptions & nonfunctional constraints**
- Describe the functionality of the system
  - **Identify all services needed to realize the use cases**
  - **Each use case uses one or more services**
  - **Each service can be used by one or more use cases**
- Review the system specification with the client

Requirements: What do users do?



- Brief high-level descriptions elicited from client
- ***Actors*** represent roles, that is, a type of user of the system
  - **Bank Customer**
  - **Bank Teller**
- ***User task*** represents work accomplished by the user, independently of the system.
  - **Open Account**
  - **Withdraw Money**

Requirements (2): Examples

Actor **Bank Customer**

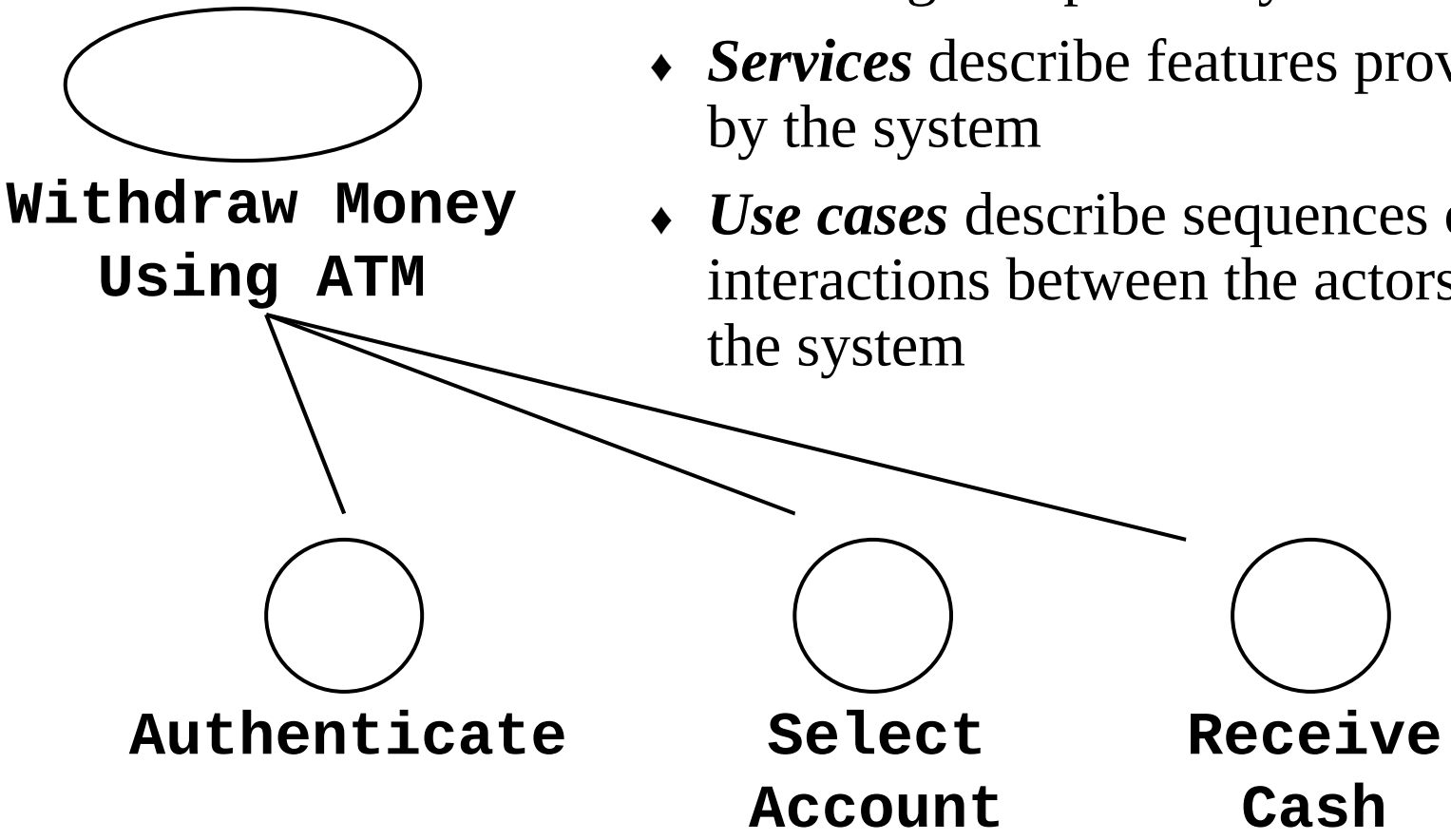
- Person who owns one or more Accounts in the Bank.

User Task **Withdraw Money**

- The Bank Customer specifies a Account and provides credentials to the Bank proving that s/he is authorized to access the Bank Account.
- The Bank Customer specifies the amount of money s/he wishes to withdraw.
- The Bank checks if the amount is consistent with the rules of the Bank and the state of the Bank Customer's account. If that is the case, the Bank Customer receives the money in cash.

Specification: What does the system do?

- Complete description of the system including exceptions by developers
- ***Services*** describe features provided by the system
- ***Use cases*** describe sequences of interactions between the actors and the system



Specification (2): Example of use case attributes

Use Case **Withdraw Money Using ATM**

Initiatiating actor:

- Bank Customer

Preconditions:

- Bank Customer has opened a Bank Account with the Bank ***and***
- Bank Customer has received an ATM Card and PIN.

Postconditions:

- Bank Customer has the requested cash ***or***
- Bank Customer receives an explanation from the ATM about why the cash could not be dispensed.

Specification (3): Example of use case flow of events

Actor steps	System steps
1. The Bank Customer inputs her card into the ATM.	
	2. The ATM requests the input of a four-digit PIN.
3. The Bank Customer types in PIN.	
	4. If several accounts are recorded on the card, the ATM offers a choice of the account numbers for selection by the Bank Customer
5. The Bank Customer selects an account .	
	6. If only one account is recorded on the card or after the selection, the ATM requests the amount to be withdrawn.
7. The Bank Customer inputs an amount.	
	8. The ATM outputs the money and a receipt and stops the interaction.

Specification (4): Example services

Service **Authenticate**

- Inputs: card, PIN
- Output: account menu (if multiple accounts) ***or*** message requesting cash amount (if one account)

Service **Select Account**

- Input: one account (from menu)
- Output: message requesting cash amount

Service **Receive Cash**

- Input: Cash amount
- Outputs: Cash ***or*** error message

Specification (5): Exceptions

Actor steps

1. The Bank Customer inputs her card into the ATM**.[Invalid card]**

3. The Bank Customer types in PIN. **[Invalid PIN]**

5. The Bank Customer selects an account .

7. The Bank Customer inputs an amount. **[Amount over limit]**

[Invalid card]
The ATM outputs the card and stops the interaction.

[Invalid PIN]
The ATM announces the failure and offers a second try as well as canceling the whole use case. After three failures, it announces the possible retention of the card. After the fourth failure it keeps the card and stops the interaction.

[Amount over limit]
The ATM announces the failure and the available limit and offers a second try as well as canceling the whole use case.

Nonfunctional requirements

Domain constraints

- Domain facts
- Applicable to user tasks

Global functional constraints

- Functionality that is easier to describe in terms of constraints
- Applicable to use cases

Quality constraints

- Constraint on the attribute of a user task, use case, or service.

Guidelines for use cases (1)

Name

- Use a verb phrase to name the use case.
- The name should indicate what the user is trying to accomplish.
- Examples:
  - **“Request Meeting”, “Schedule Meeting”, “Propose Alternate Date”**

Length

- A use case should not exceed 2 A4 pages. If longer, use *include* relationships.
- A use case should describe a complete set of interactions.

Counter examples:

  - **“Add Participants”, “Add Date To Exclusion Set”, ...**

Guidelines for use cases (2)

Flow of events

- The active voice should be used. Steps should start either with “The Actor …” or “The System …”.
- The causal relationship between the steps should be clear.
- All flow of events should be described (not only the main flow of event).
- The boundaries of the system should be clear. Components external to the system are described as such.
- Define important terms in the glossary.

Negative example:

- **The driver arrives at the parking gate, the driver receives a ticket from the distributor, the gate is opened, the driver drives through.**

Guidelines for use cases (3)

Exceptions

- Exceptions should be attached to the step where they are detected.
- If an exception can occur in any step, describe it only in the exception section.
- Exception handling is described as flow of events.
- At the end of the exception handling, it should be clear what happens next (if the use case is terminated or if it is resumed in a particular step).

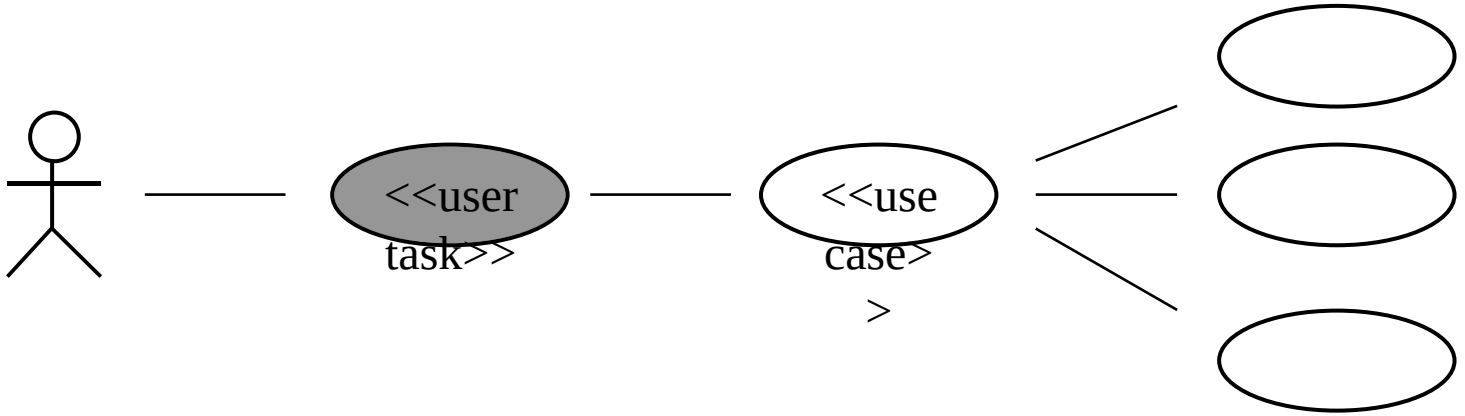
Preconditions

- If a case is excluded with a precondition, then it should not be handled as an exception.

Guidelines for use cases (4)

User task/use case decomposition

- Write one high-level use case per user task
- Include lower-level use cases into the high-level use case
- If a use case includes only one or two steps, it should probably be a service, not a use case.
- If a sequence of steps is identical in several use cases, it should be factored out into a separate use case and included in the original use cases (eliminate redundancy).



Rationale

Question: Alternative Authentication Mechanisms?

References: Service: Authenticate

Decision: Smart Card + PIN

	Criteria 1: ATM Unit Cost	**Criteria 2:** Privacy
Option 1: Account number	+	–
Option 2: Finger print reader	–	+
Option 3: Smart Card + PIN	+	+

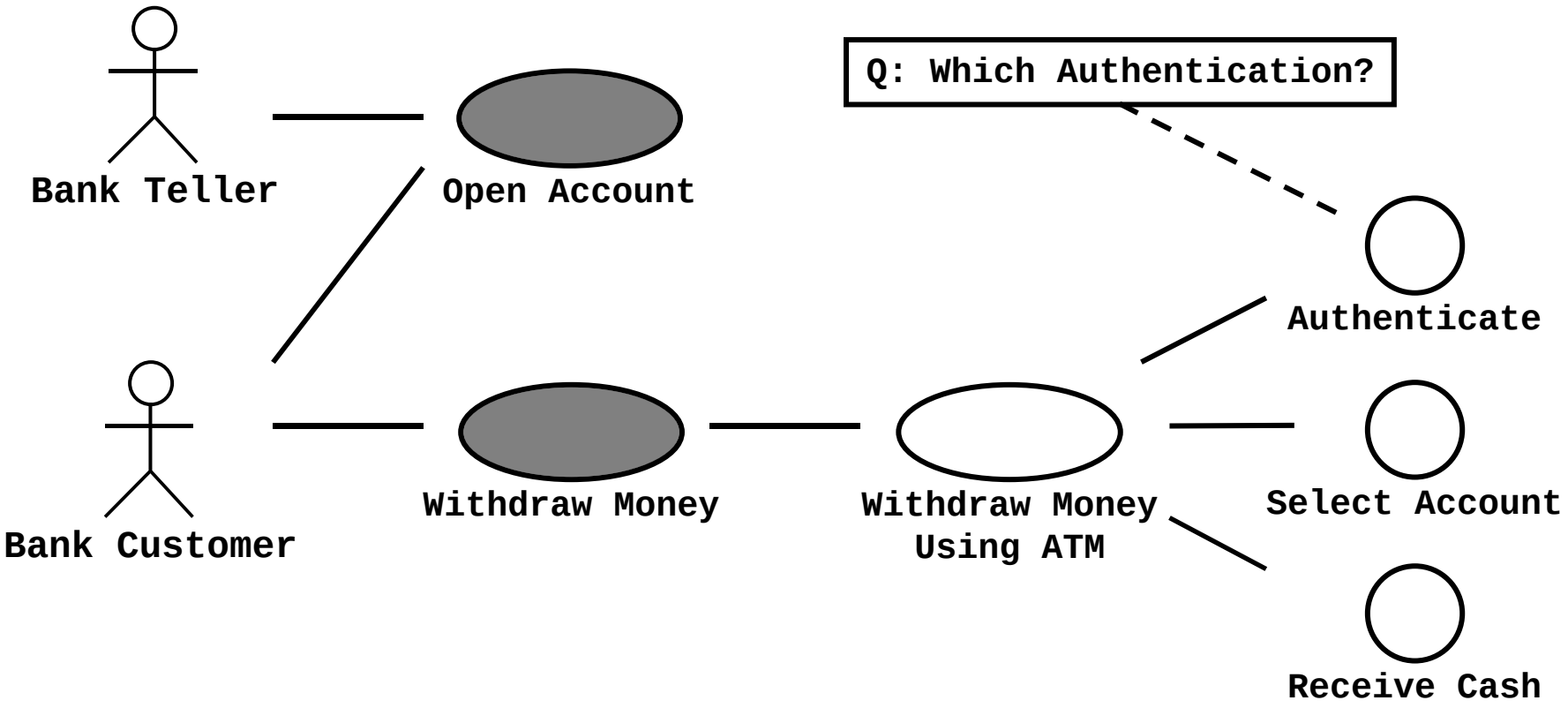
Rationale (2)

Questions can be used to:

- Request a clarification — *How is a bank account identified?*
- Indicate a defect — *Can't more than one person own a single account?*
- Justify a use case or a service — *Which solution is the best for …?*

Questions are asked during review and consolidated into justifications during revisions.

Putting everything together



Process for TRAMP (1)

- REQuest for the requirements specification
  - **Web-based tool**
  - **Actors, User Tasks, Use Cases, & Services**
  - **Constraints & Glossary**

- REQuest for review and negotiation
  - **Questions, Options, Criteria, Assessments**
  - **Discussion**
  - **What's new, what's revised, conflict detection**

- TogetherJ for Analysis (class diagram)

Process for TRAMP (2)

Instructors/coaches

Nov 12: RAD v.0 from coaches

- **Actors & user tasks**

Dec 3: Questions from coaches

Dec 10: More questions ...

Teams

Before Nov 30:

- **Questions to coaches**

Nov 30: RAD v.1 due

- **Add'l user tasks**
- **Use cases, Services, Constraints**
- **Glossary**
- ***Analysis Object Models and sequence diagrams (Together)***

Dec 3: Requirements review

Dec 7: RAD v.2 due

- **Options, assessments**
- **Decisions**
- **Revised requirements elements**
- ***Revised analysis (Together)***

Process for ARENA

ARENA

- **a web-based tournament management system**
- **Simpler than TRAMP**

Intended audience

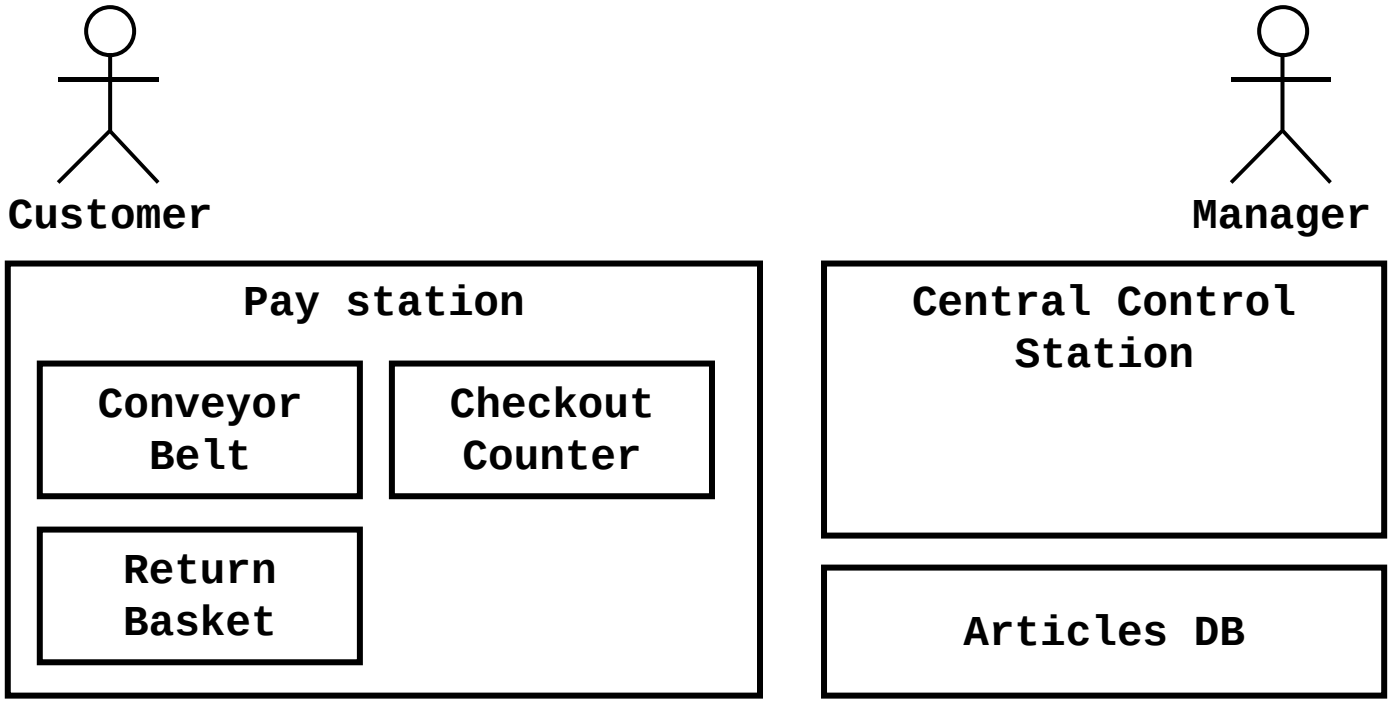
- **Lecture students who are not part of the Praktikum**
- **Anybody interested in using REQuest**

Process

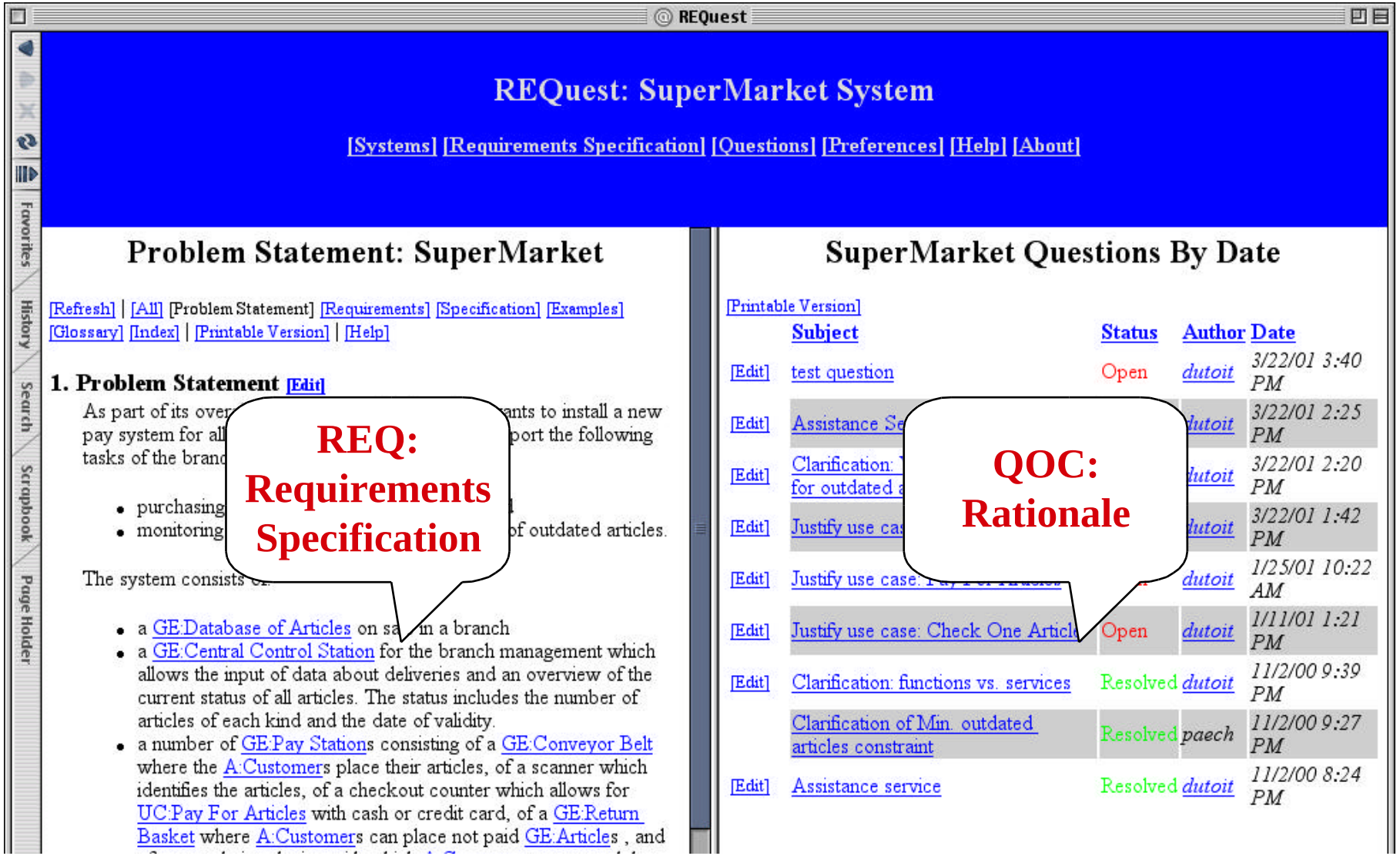
- **Problem statement will be posted on the lecture bboard**
- **Ask for a REQuest account if interested (mailto:dutoit@in.tum.de)**
- **Partial solution explained in two weeks**

SuperMarket: Problem statement

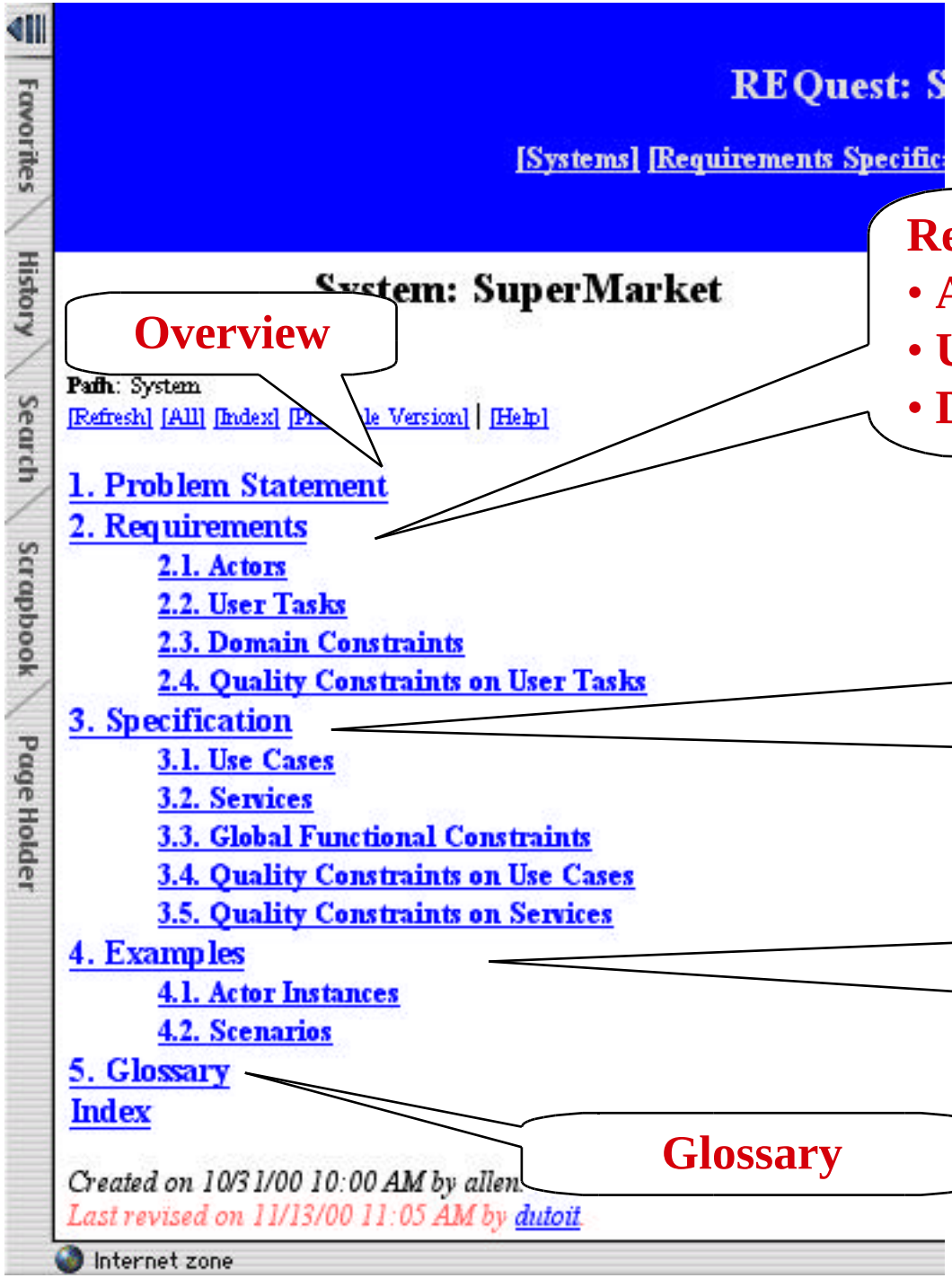
- Food vendor needs new pay system:
  - **Articles can be purchased by customers**
  - **Manager can monitor inventory**
  - **System periodically checks for outdated articles**



REQuest: Overview



REQuest: Overview (2)



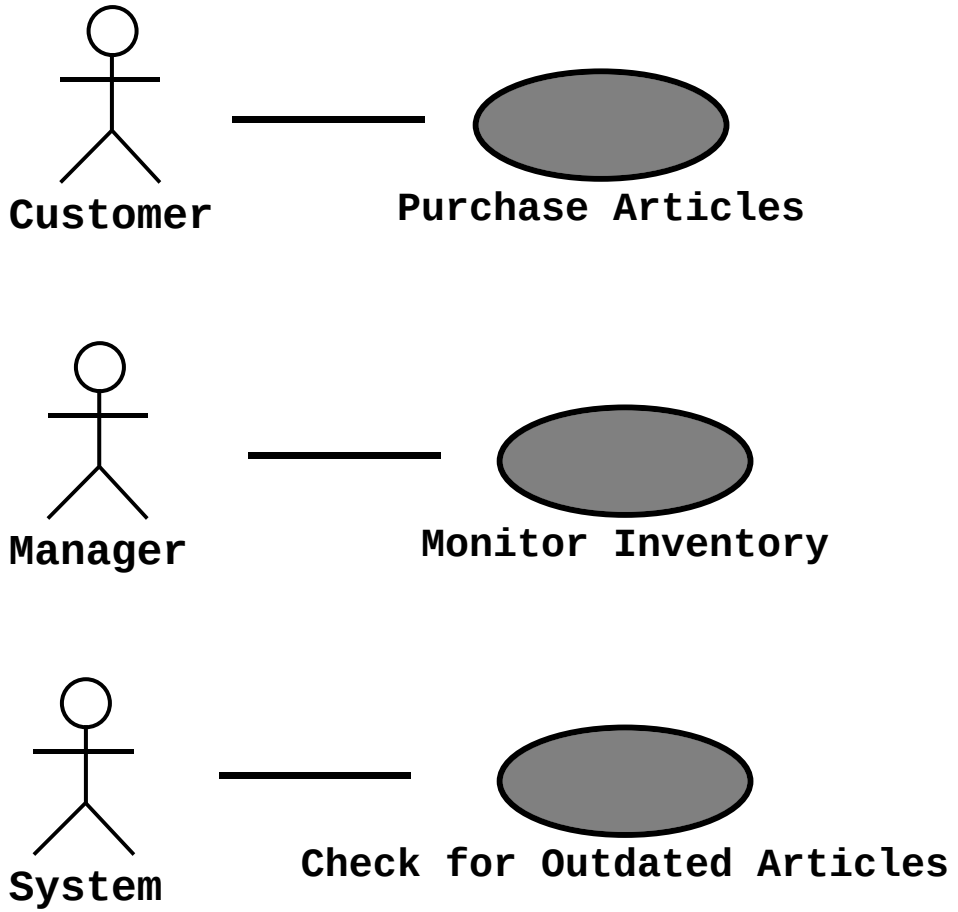
Requirements:
- Actors
- User Tasks
- Domain Constraints

Specification:
- Use Cases
- Services
- Quality Constraints

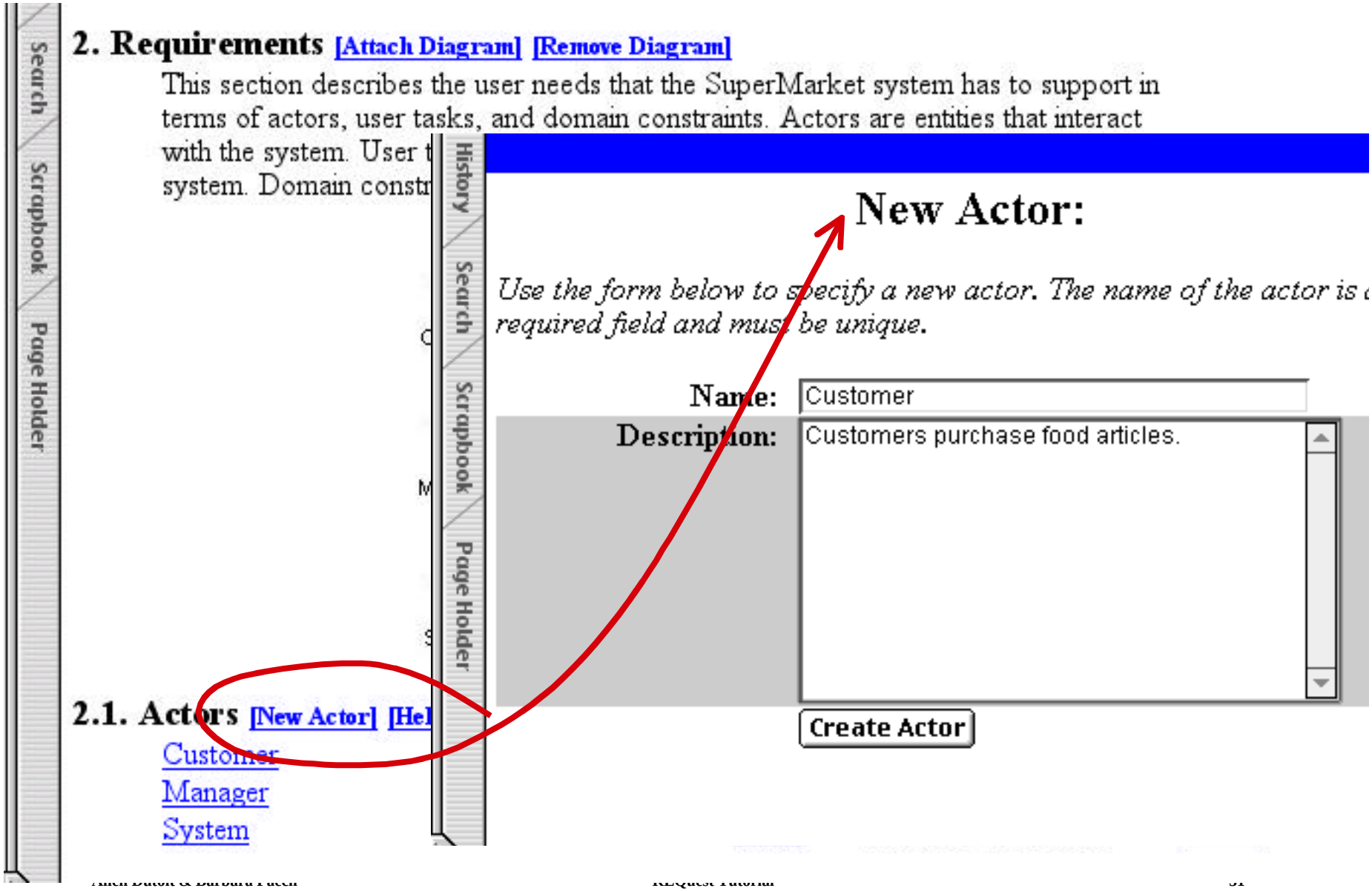
Examples:
- Actor Instances
- Scenarios

Glossary

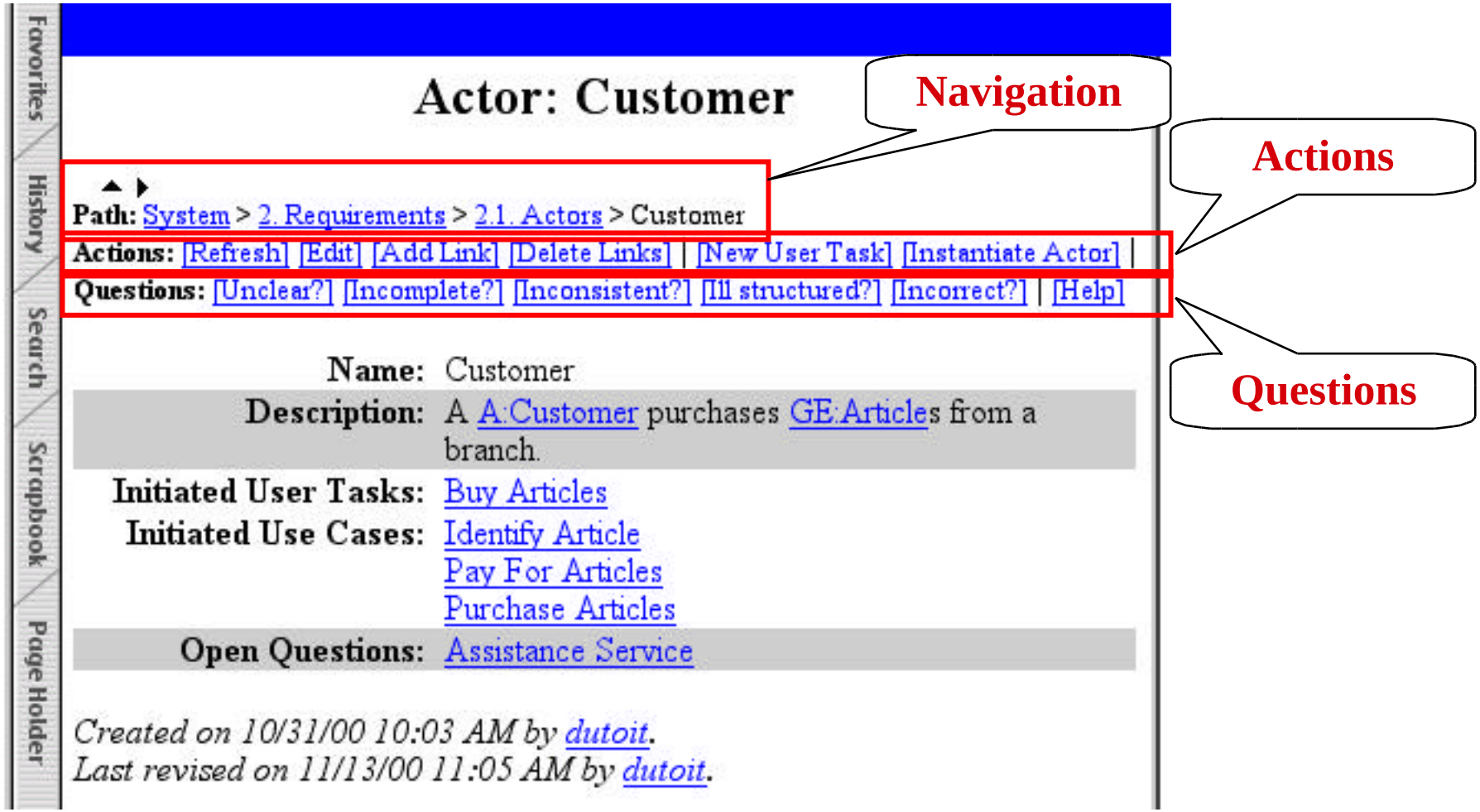
SuperMarket: Identifying Actors & User Tasks



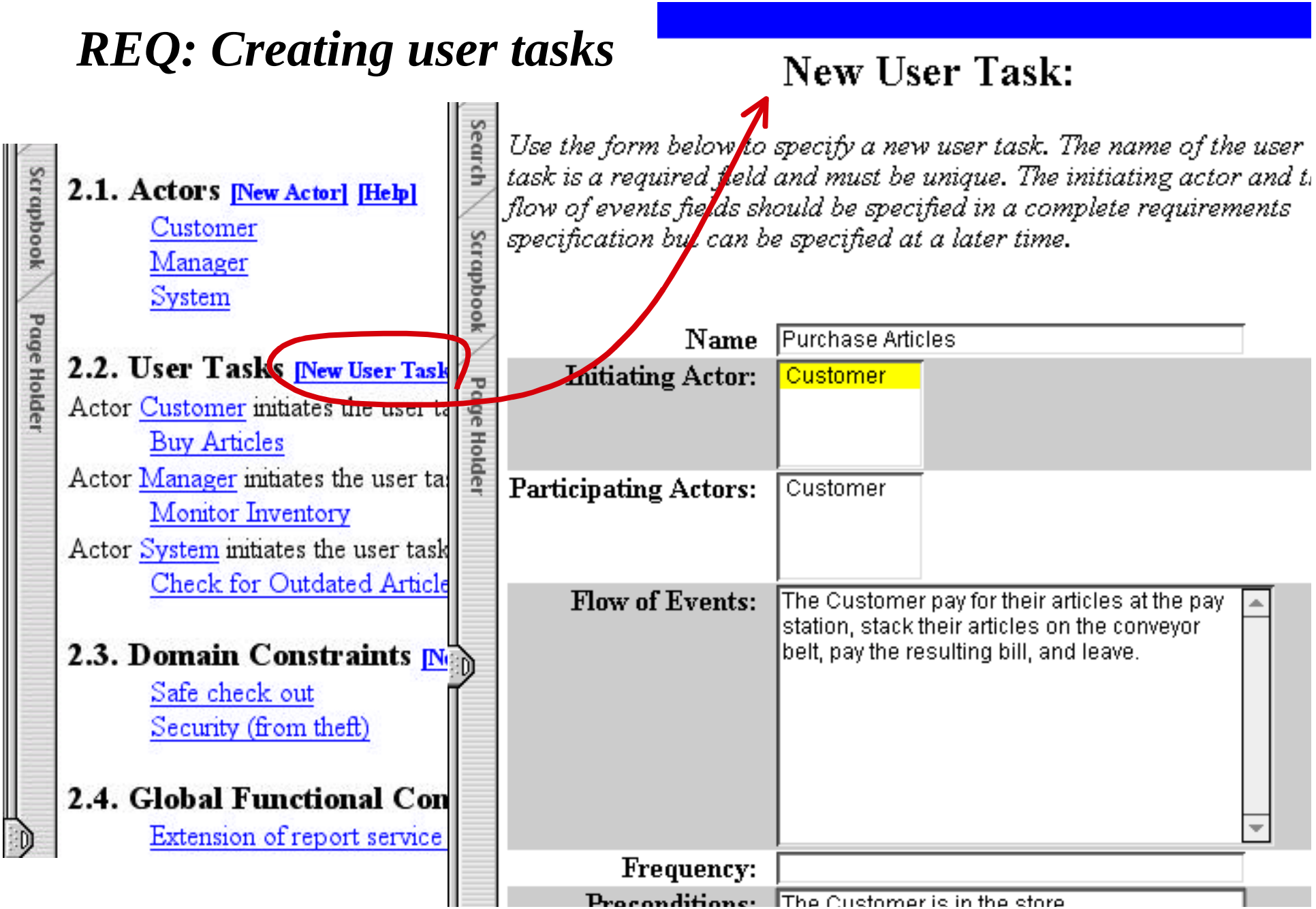
REQ: Creating actors



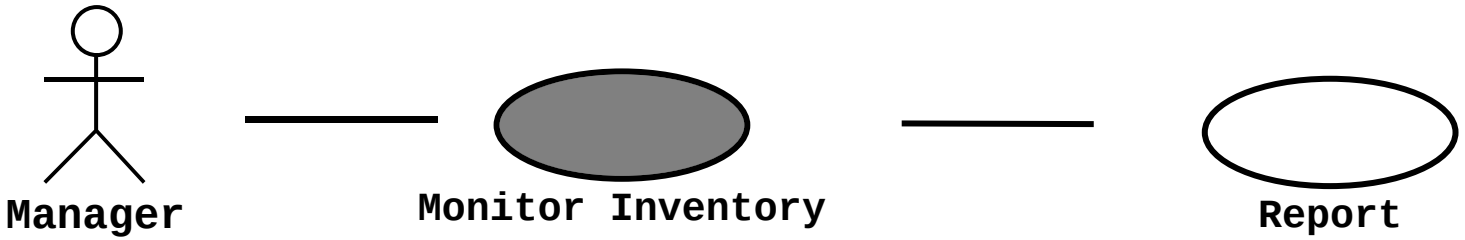
REQuest Creating actors (2)



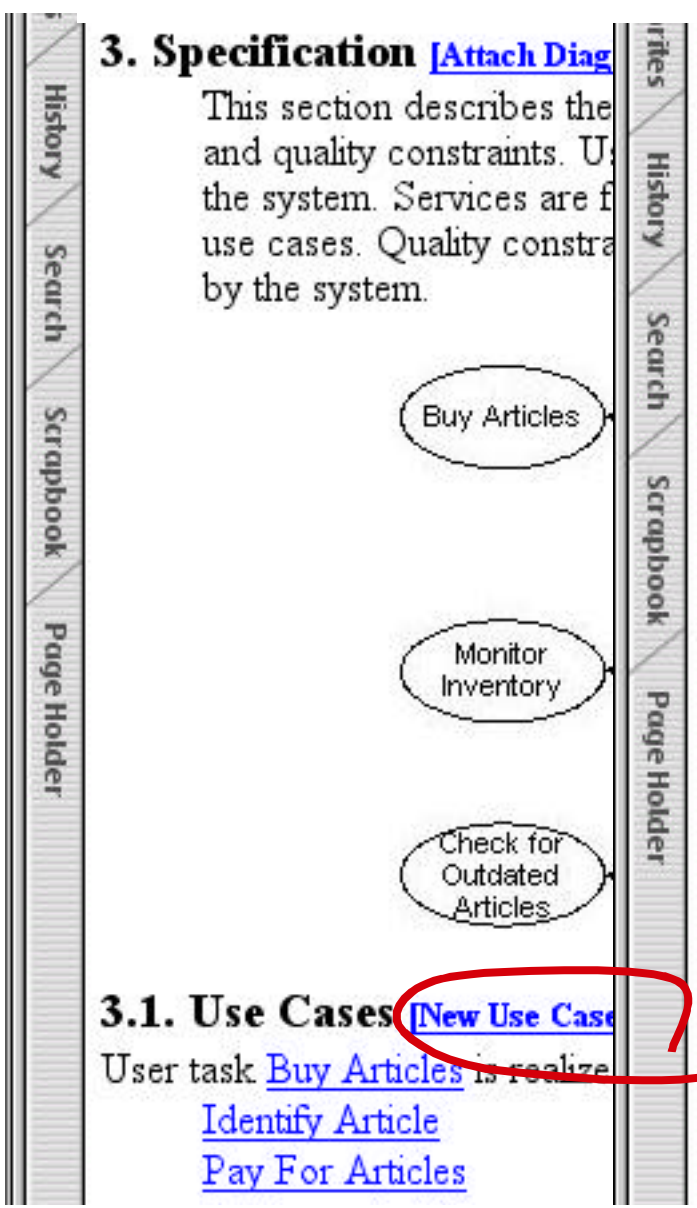
REQ: Creating user tasks

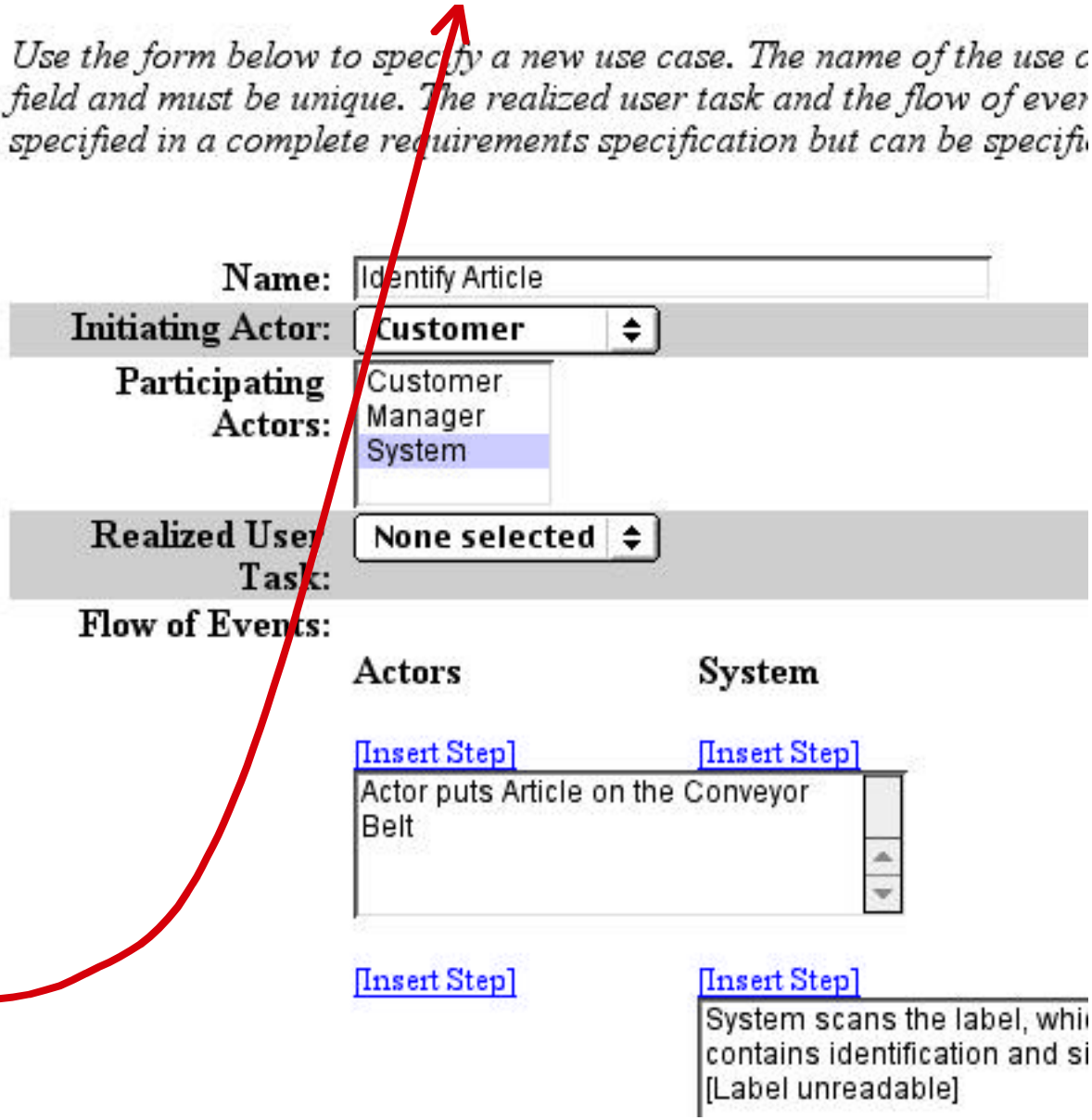


SuperMarket: Identifying Use Cases

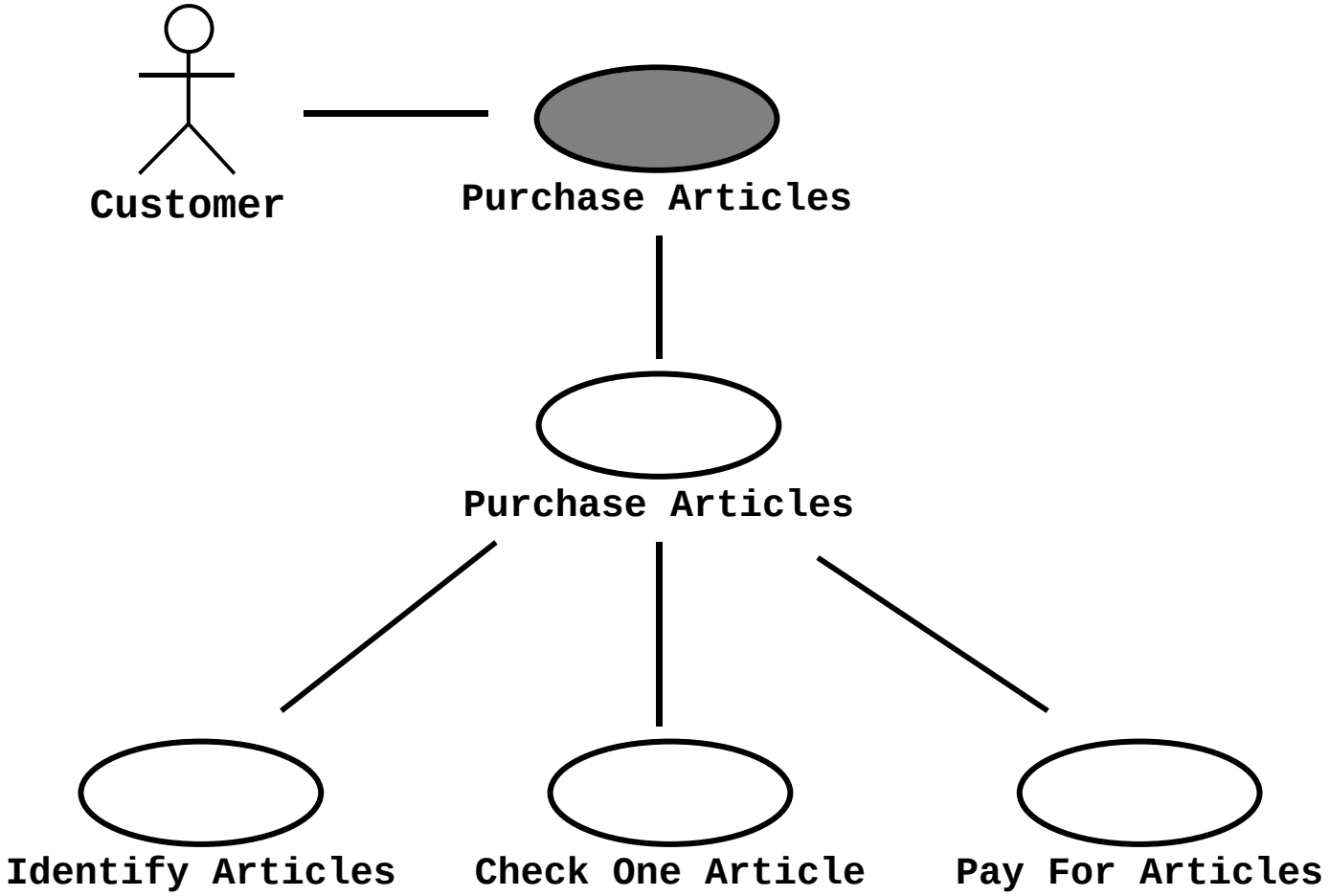


REQ: Creating Use Cases

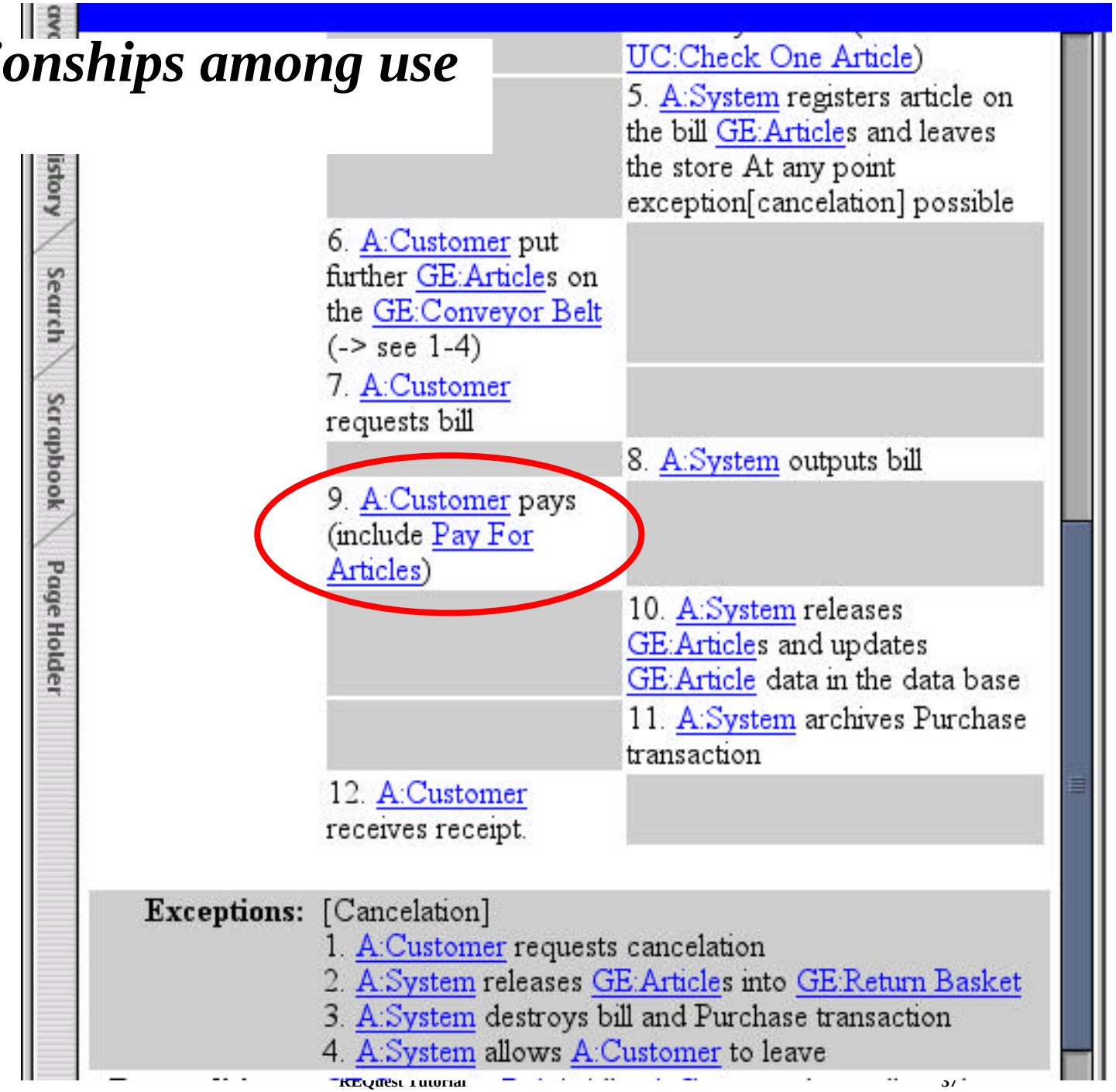




SuperMarket: Refining Use Cases



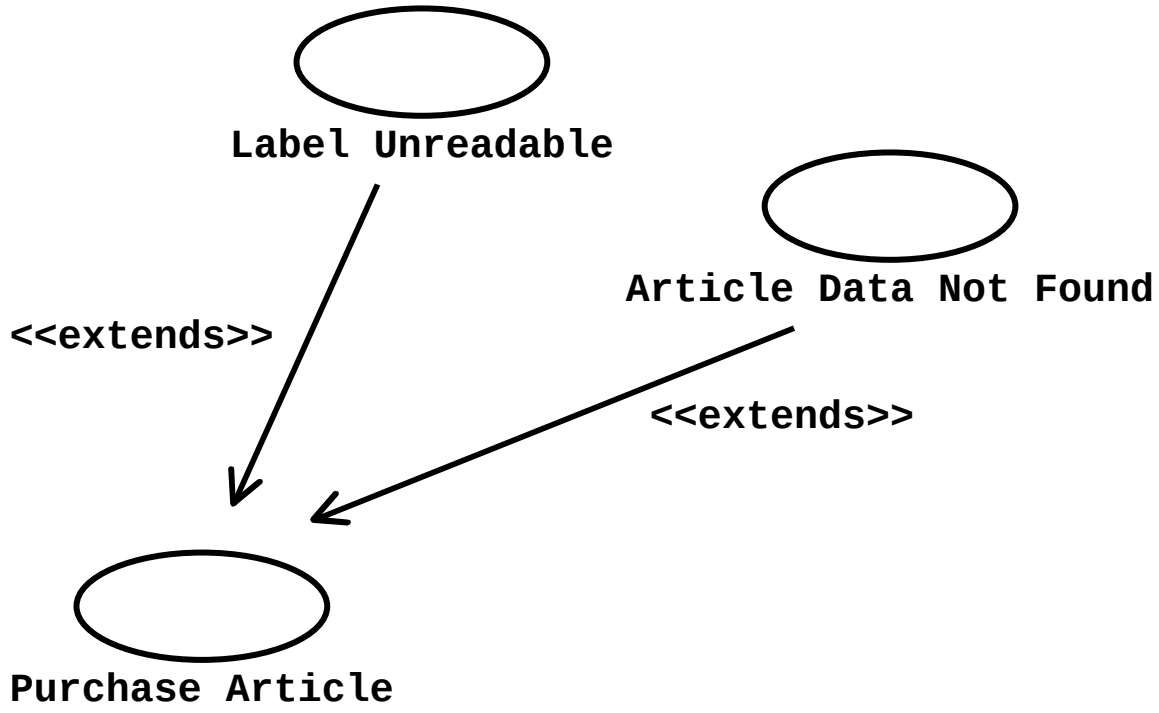
REQ: Relationships among use cases



	UC:Check One Article)
	5. A:System registers article on the bill GE:Articles and leaves the store At any point exception[cancelation] possible
6. A:Customer put further GE:Articles on the GE:Conveyor Belt (-> see 1-4)	
7. A:Customer requests bill	
	8. A:System outputs bill
9. A:Customer pays (include Pay For Articles)	
	10. A:System releases GE:Articles and updates GE:Article data in the data base
	11. A:System archives Purchase transaction
12. A:Customer receives receipt.	

Exceptions: [Cancelation]
1. A:Customer requests cancelation
2. A:System releases GE:Articles into GE:Return Basket
3. A:System destroys bill and Purchase transaction
4. A:System allows A:Customer to leave

SuperMarket: Identifying Exceptions



REQ: Describing exceptions

Realized User Task: Buy Articles

Flow of events:

Actors [Include Use Cases]	**System** [Use Services]
1. Actor puts GE:Article on the GE:Conveyor Belt	
	2. A:System scans the label, which contains identification and size [Label unreadable] (use Scan Article Label)
	3. A:System searches for GE:Article data [GE:Article data not found] (use Search Article in Database)
	4. A:System displays name and price (use Display Message to Customer)

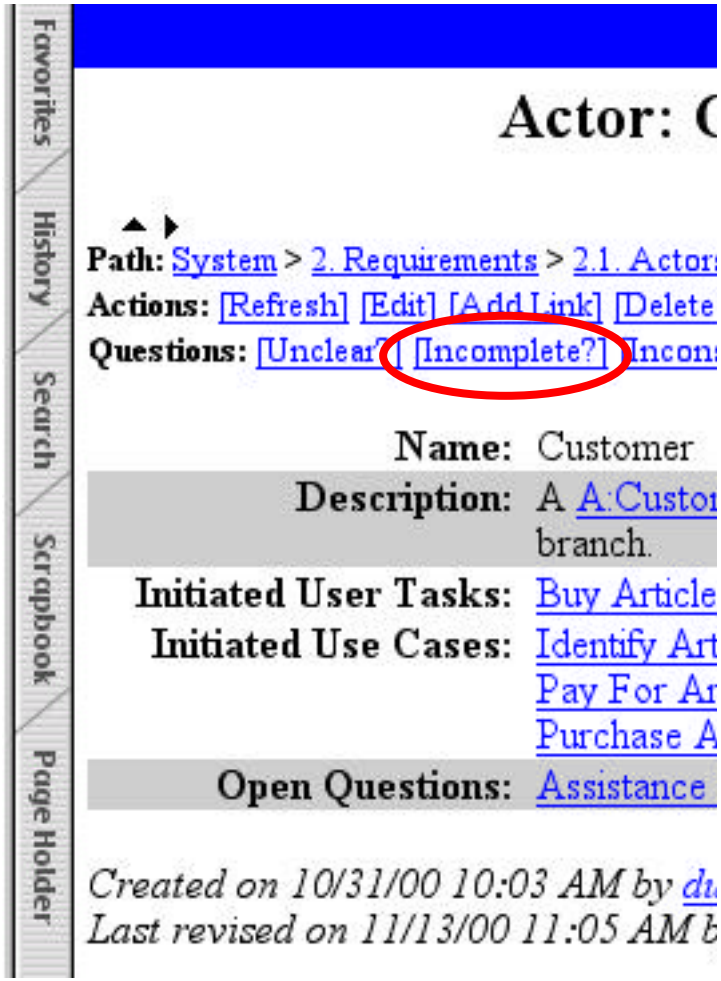
Exceptions: [Label unreadable]

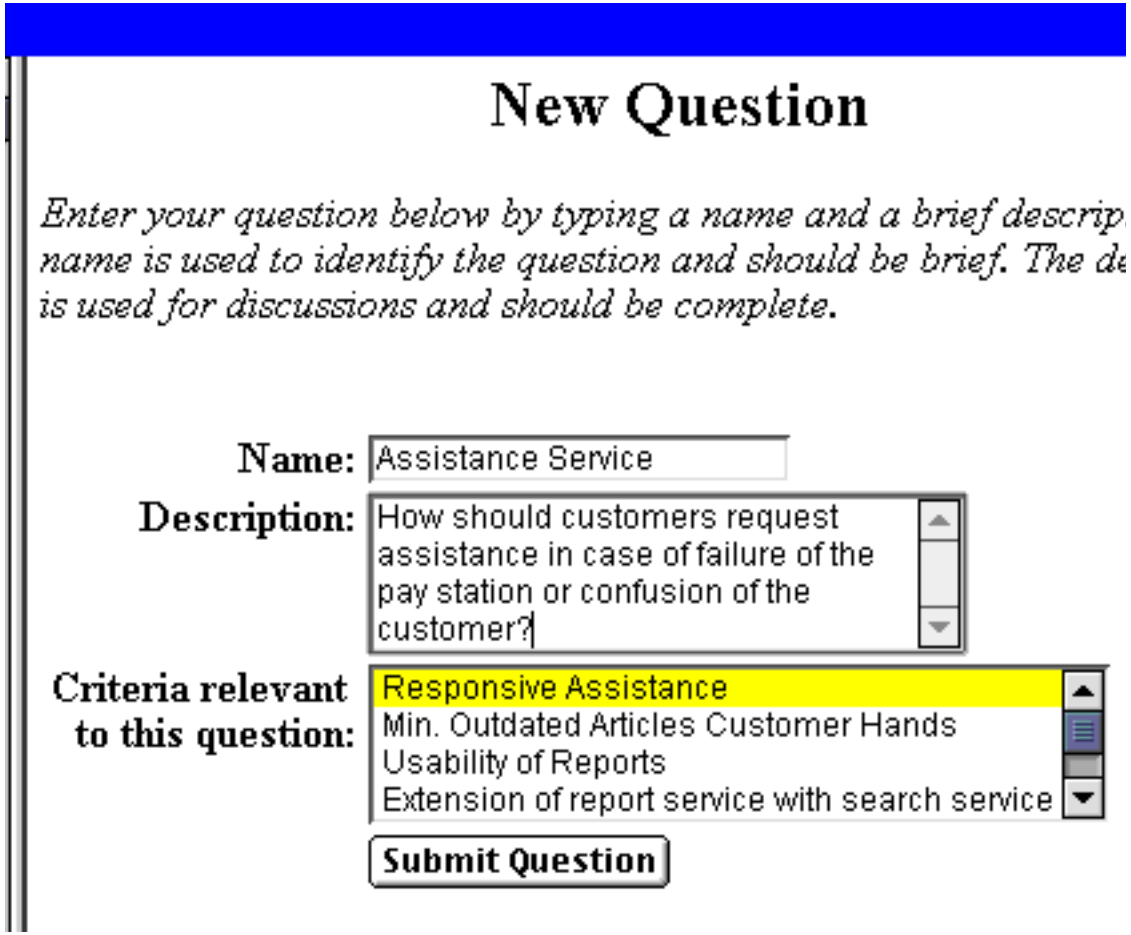
1. A:System displays error message
2. Actor replaces GE:Article on the GE:Conveyor Belt (->see description above) or places GE:Article into the GE:Return Basket

[GE:Article data not found]

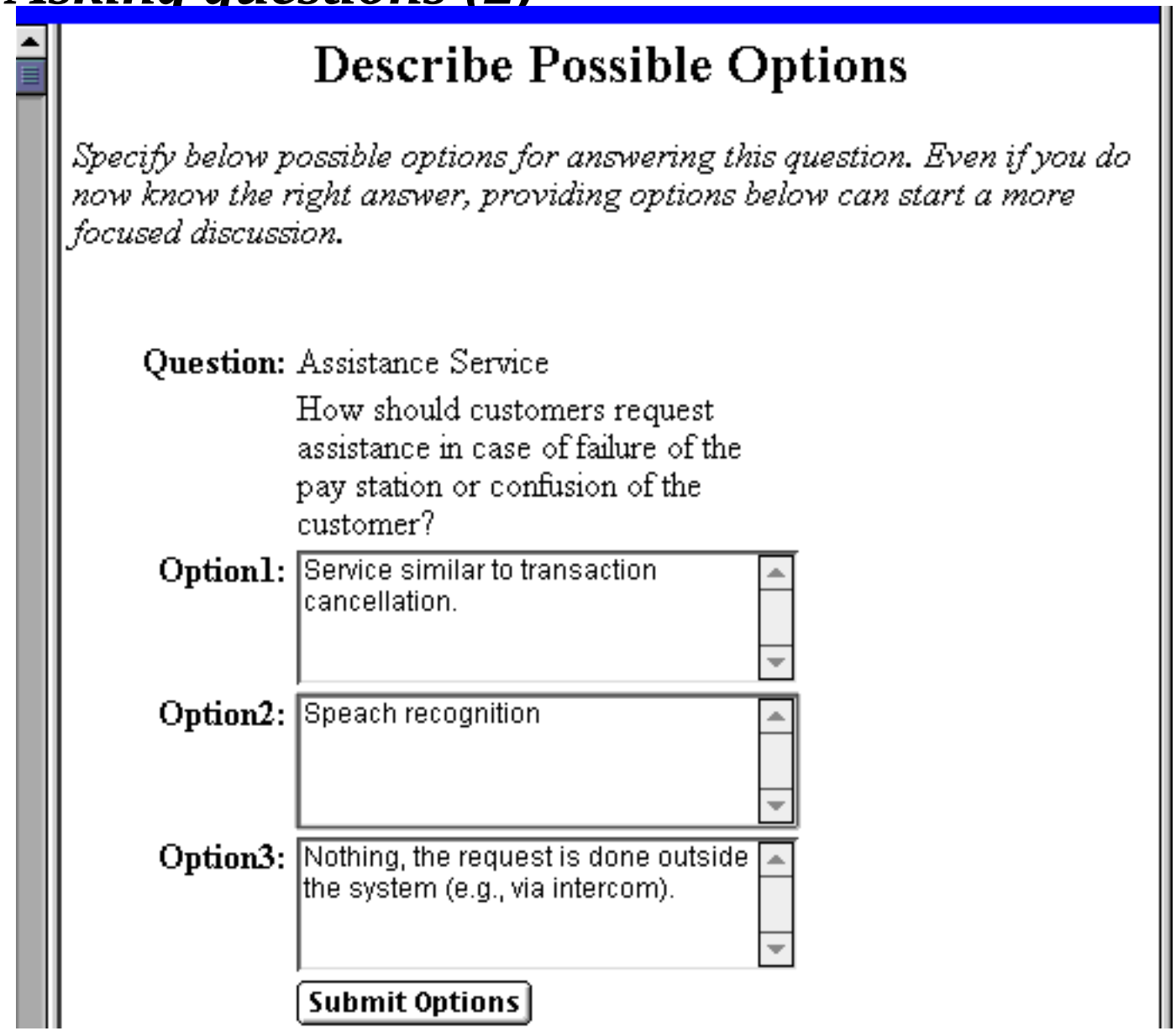
1. A:System displays error message
2. Actor places GE:Article into GE:Return Basket

QOC Asking questions (1)

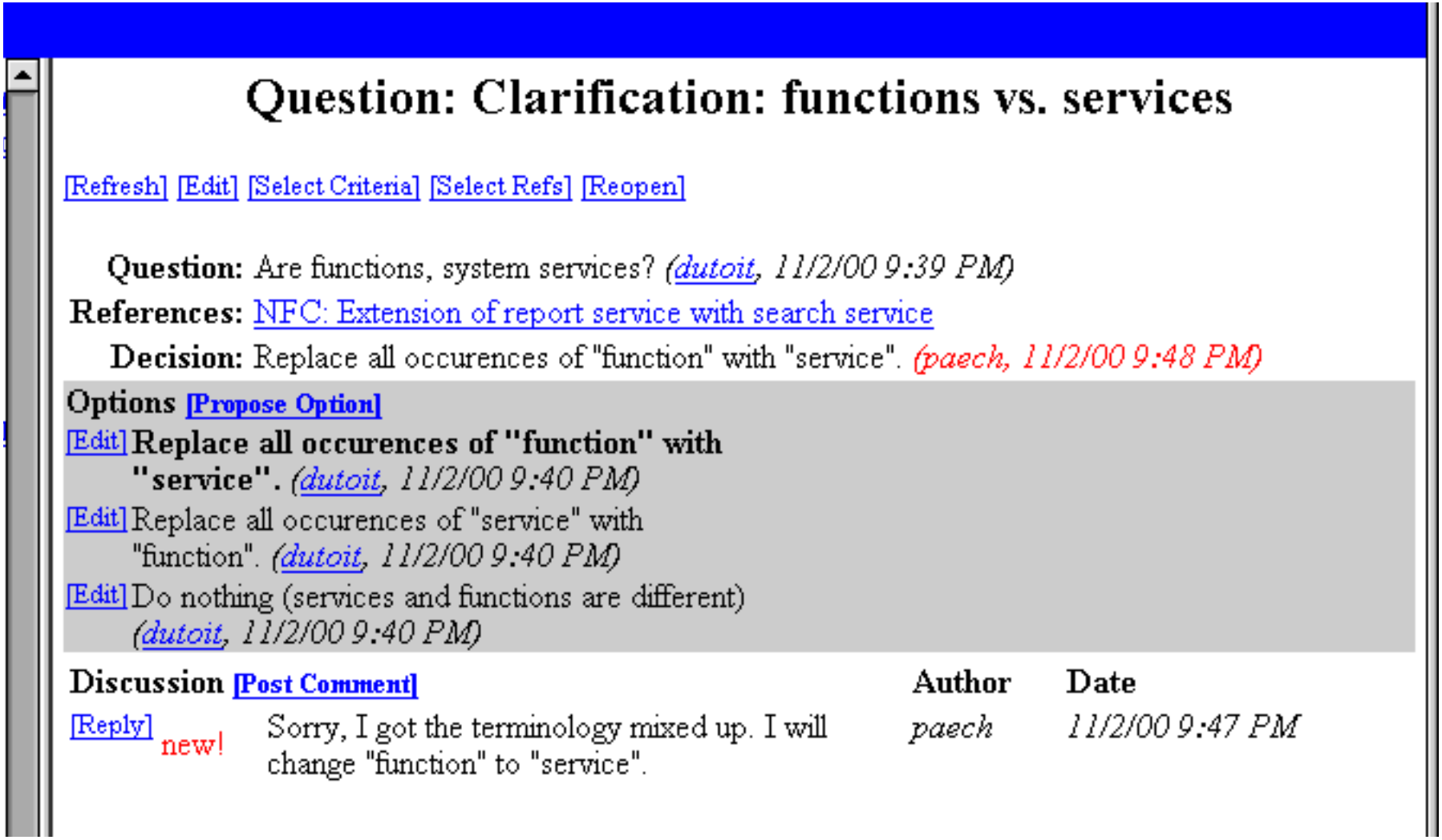




QOC: Asking questions (2)



QOC: Resolving questions



Question: Clarification: functions vs. services

[Refresh] [Edit] [Select Criteria] [Select Refs] [Reopen]

Question: Are functions, system services? *(dutoit, 11/2/00 9:39 PM)*

References: NFC: Extension of report service with search service

Decision: Replace all occurences of "function" with "service". *(paech, 11/2/00 9:48 PM)*

Options **[Propose Option]**

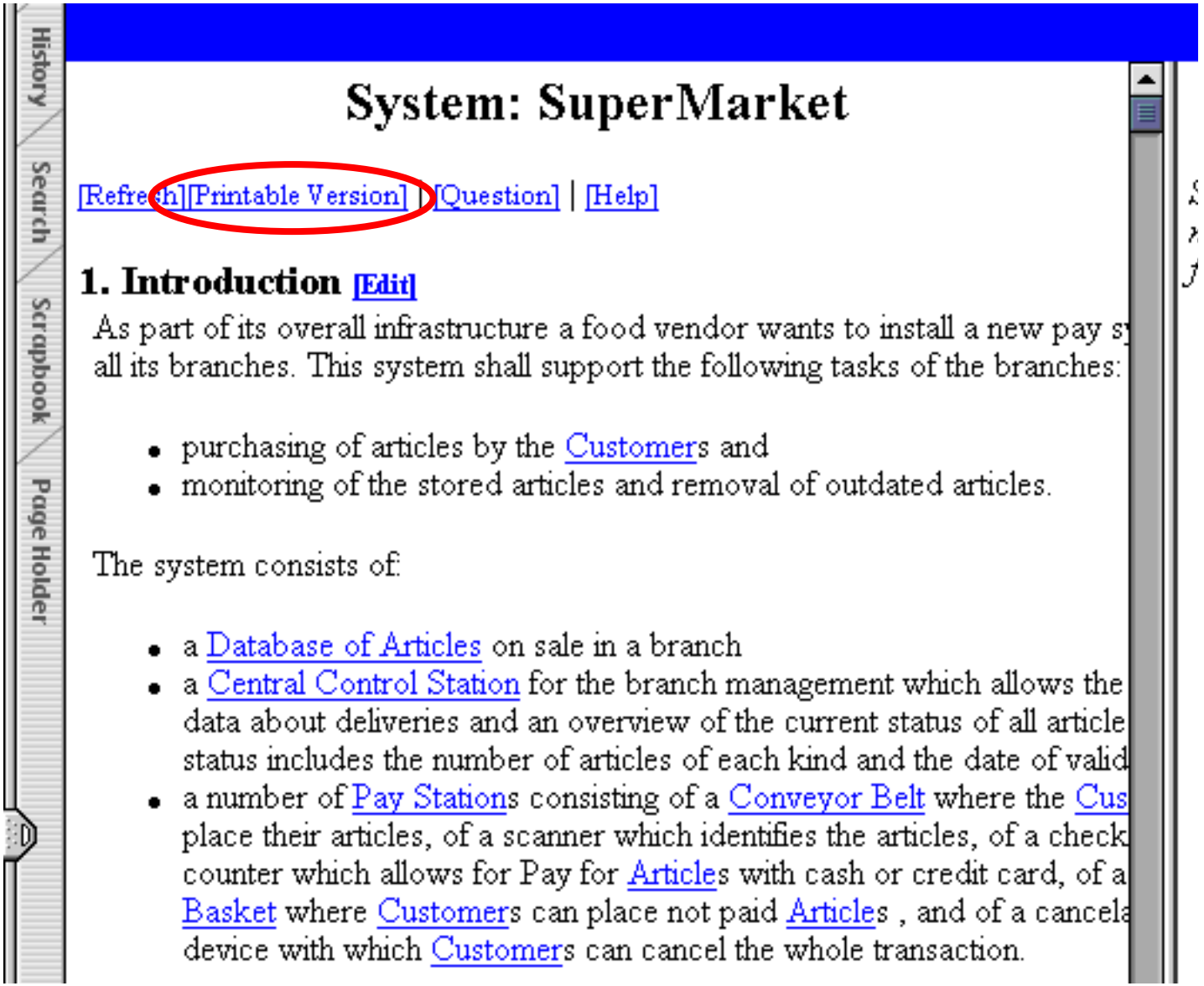
[Edit] **Replace all occurences of "function" with "service".** *(dutoit, 11/2/00 9:40 PM)*

[Edit] Replace all occurences of "service" with "function". *(dutoit, 11/2/00 9:40 PM)*

[Edit] Do nothing (services and functions are different) *(dutoit, 11/2/00 9:40 PM)*

Discussion **[Post Comment]**		**Author**	**Date**
[Reply] new!	Sorry, I got the terminology mixed up. I will change "function" to "service".	*paech*	*11/2/00 9:47 PM*

REQ: Exporting the requirements specification



Summary

- REQuest supports
  - **Definition of requirements specification**
  - **Questions about the requirements elements**
  - **Discussion, negotiation, and resolution of questions**
  - **Finding out what others have done**
- Together supports
  - **Definition of use case diagrams**
  - **Definition of class diagrams**
  - **Definition of sequence diagrams**
- More on Analysis next week
- More about QOC in 2 weeks

- TRAMP deadlines
  - **RAD v.1 November 30**
  - **RAD v.2 December 7**