

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.

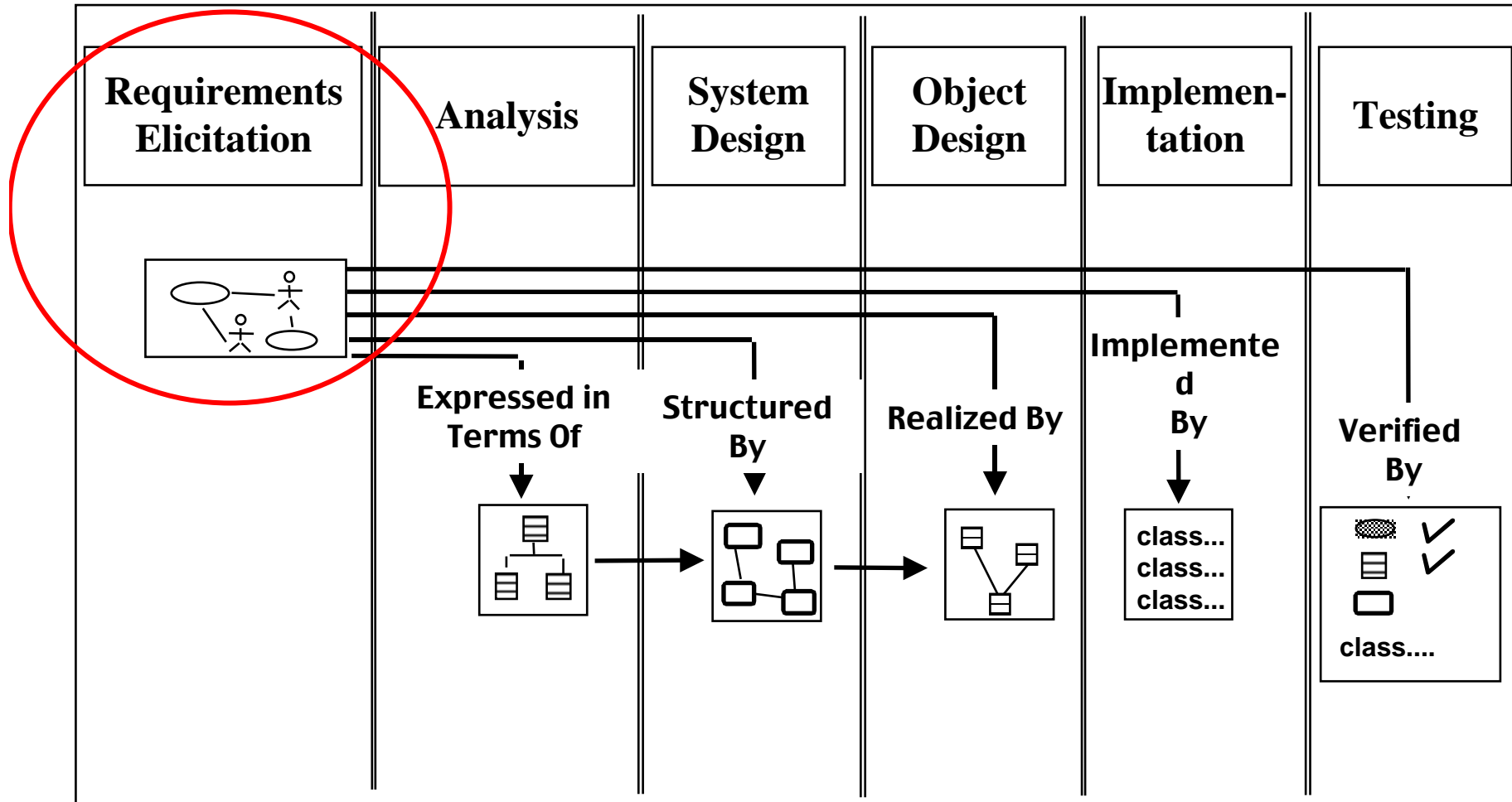
Desired outcome

To provide you with sufficient skills so that you can do a first requirements elicitation exercise.

To point out the range and difficulty 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

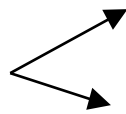


RAD

A RAD includes 3 descriptions:

Requirements Elicitation:

Use case model



Requirements: What do users do?

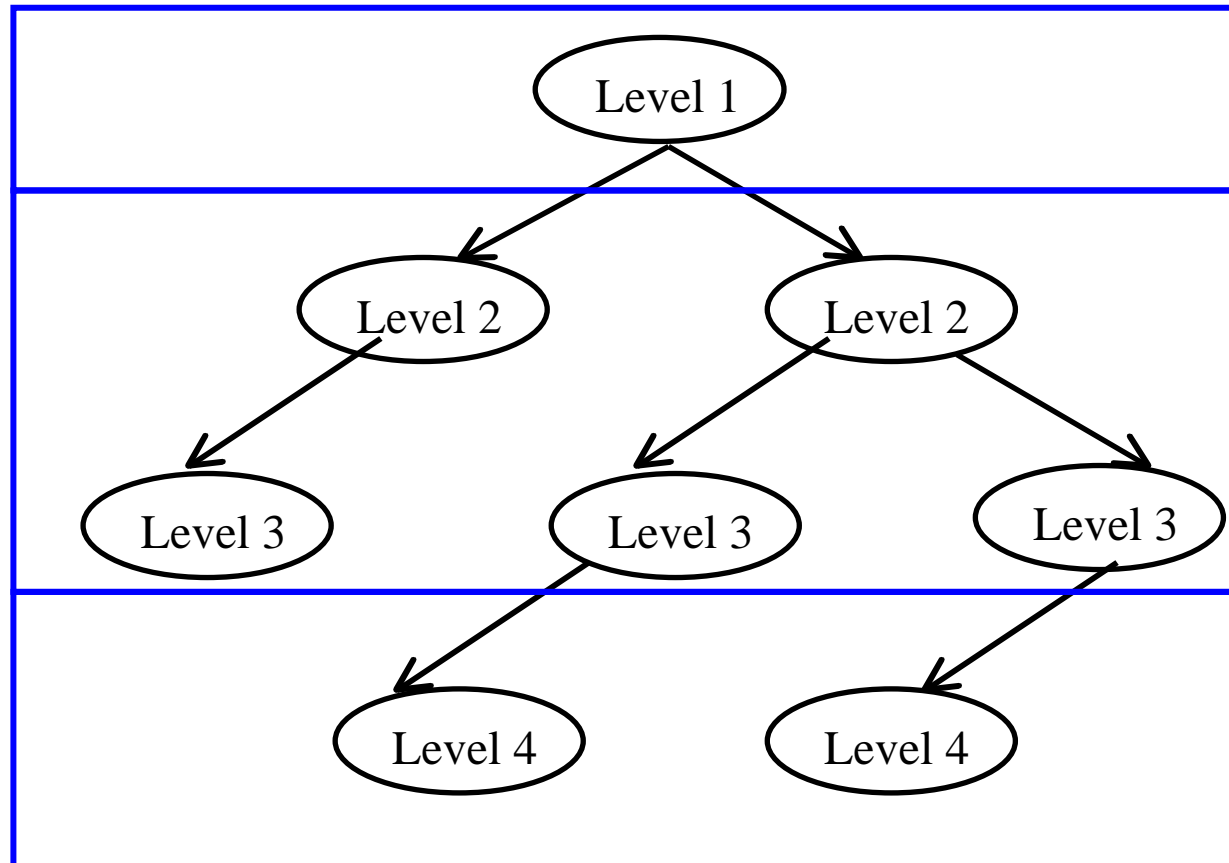
Interactions: How do users use the system to accomplish their work?

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

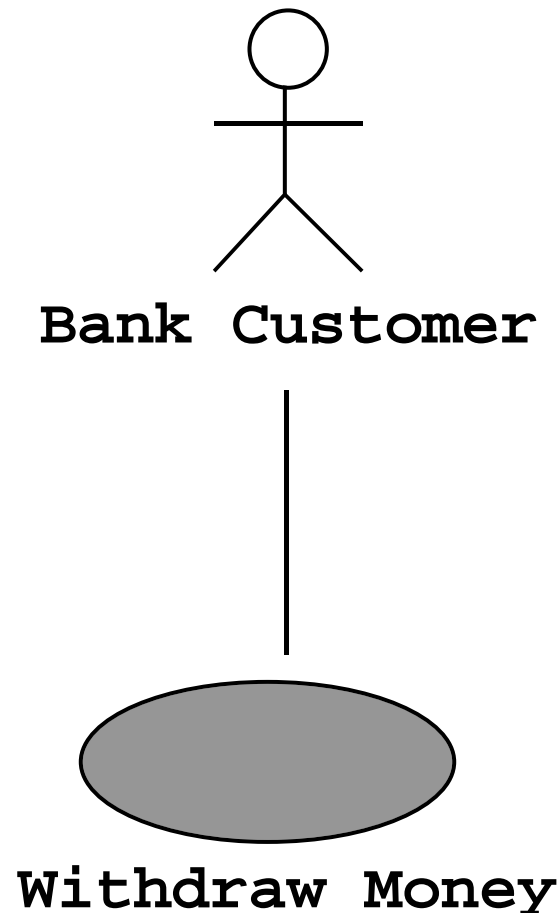
- ♦ **Describe one or more use cases per user task**
- ♦ **Exceptions & nonfunctional constraints**

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.

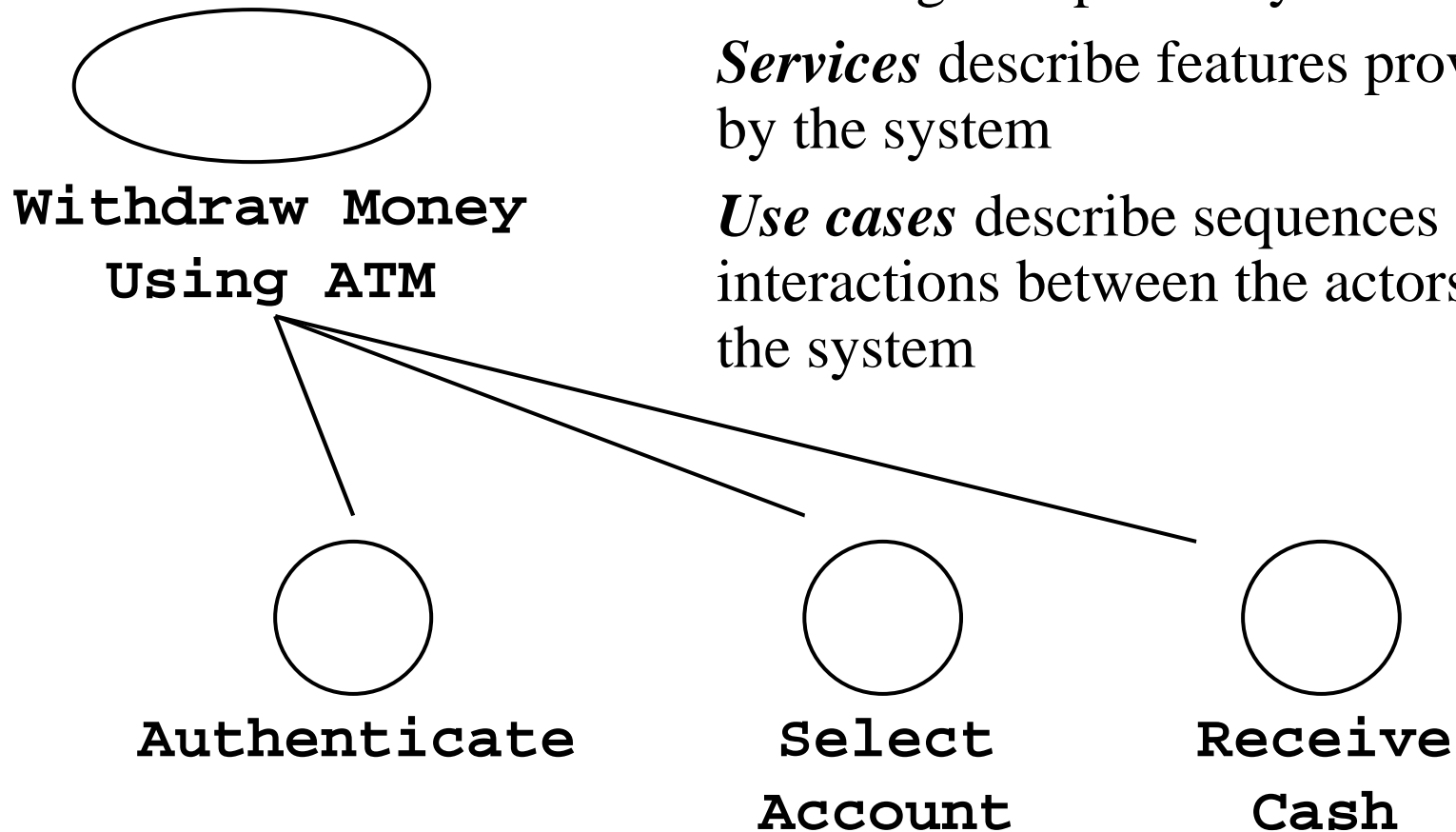
The Bank checks if the amount is consistent with the rules of the Bank and the state of the Bank Customer's account. If that is the case, the Bank Customer receives the money in cash.

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**

Initiating 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

1. The Bank Customer inputs her card into the ATM.
3. The Bank Customer types in PIN.
5. The Bank Customer selects an account .
7. The Bank Customer inputs an amount.

System steps

2. The ATM requests the input of a four-digit 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
6. If only one account is recorded on the card or after the selection, the ATM requests the amount to be withdrawn.
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)

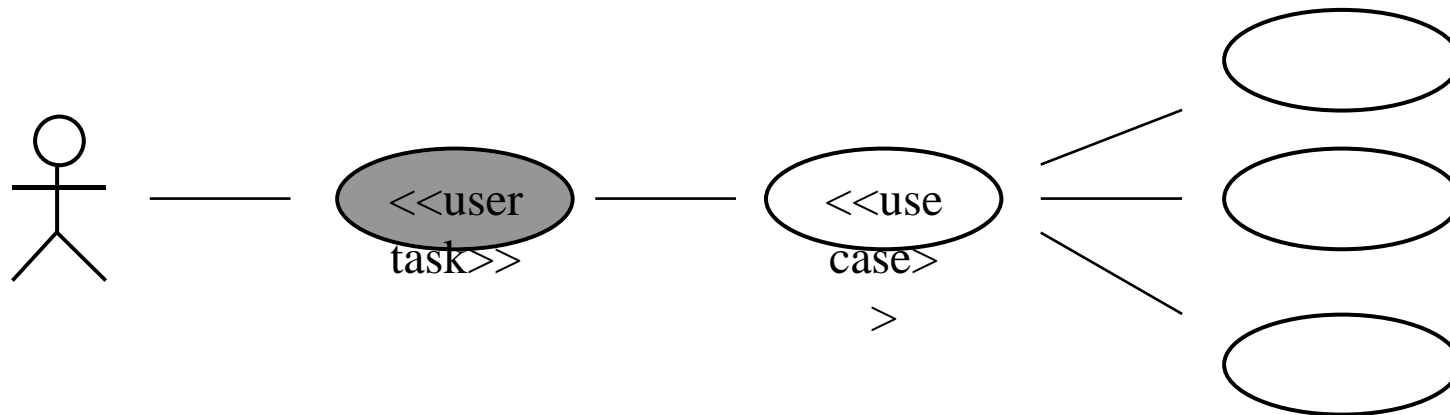
User task/use case decomposition

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

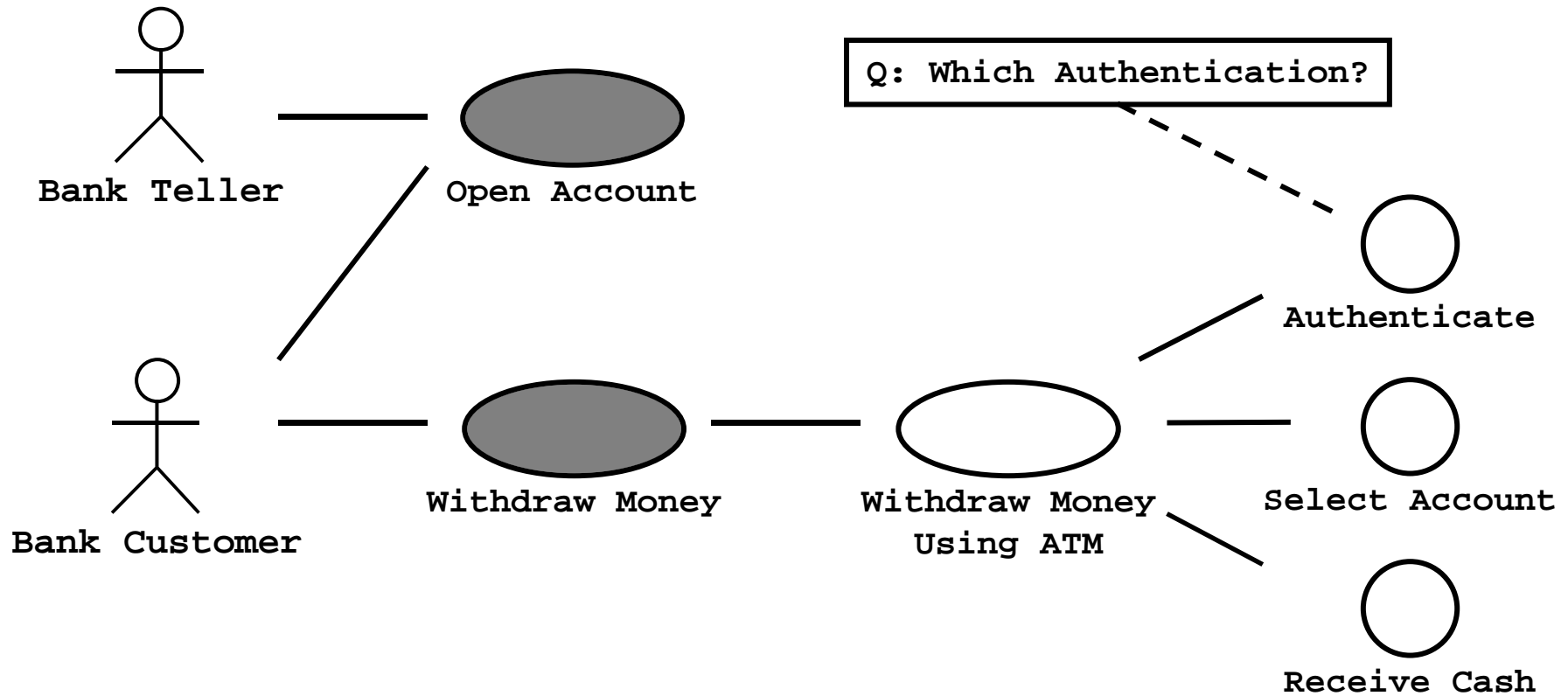
Can't more than one person own a single account?

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**

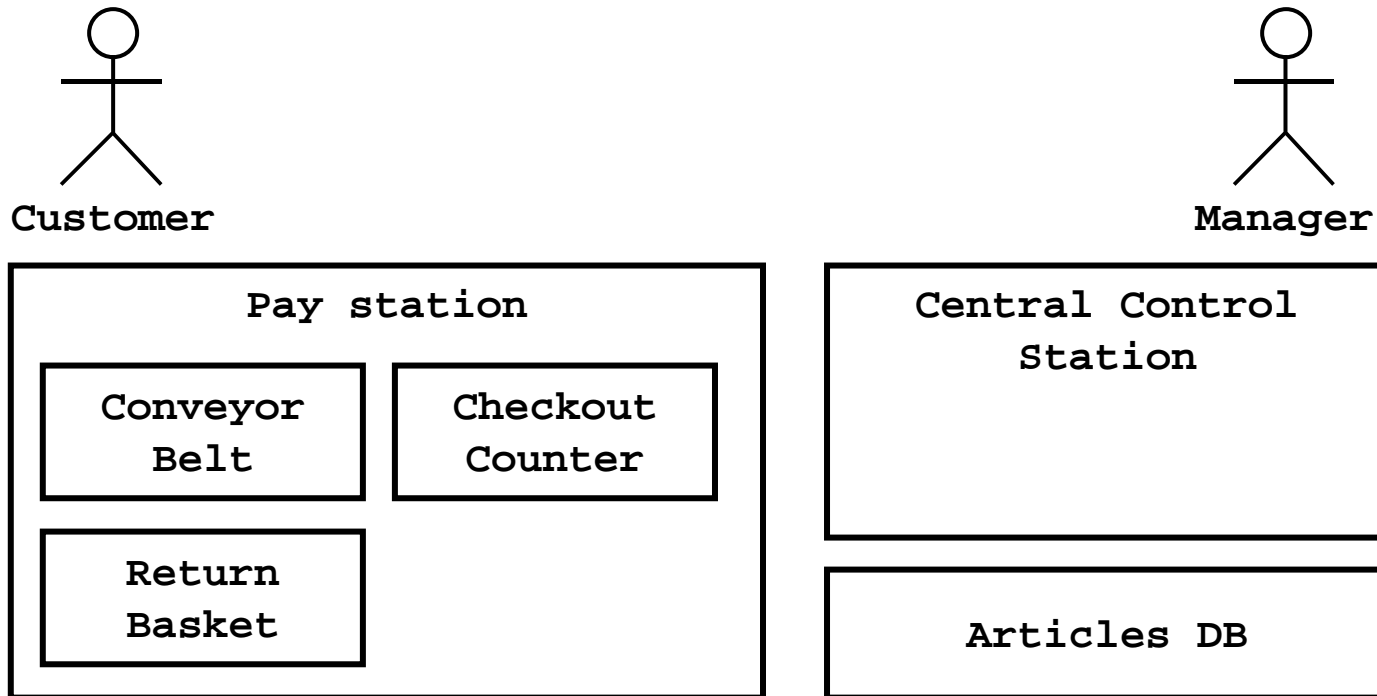
Process

- ♦ **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

The screenshot shows the REQuest web application interface. The title bar reads "REQuest". The main header is "REQuest: SuperMarket System" with navigation links: [Systems] [Requirements Specification] [Questions] [Preferences] [Help] [About].

The left pane displays the "Problem Statement: SuperMarket" section. It includes a navigation menu with links like [Refresh], [All], [Problem Statement], [Requirements], [Specification], [Examples], [Glossary], [Index], [Printable Version], and [Help]. The main content starts with "1. Problem Statement [Edit]" and describes the system's purpose: "As part of its overall strategy, the company wants to install a new pay system for all branches to support the following tasks of the branches: purchasing and monitoring of outdated articles." It then lists system components: a GE:Database of Articles, a GE:Central Control Station, and a number of GE:Pay Stations (including GE:Conveyor Belt, A:Customers, UC:Pay For Articles, GE:Return Basket, and GE:Articles).

The right pane displays "SuperMarket Questions By Date" with a [Printable Version] link. It contains a table of questions:

Subject	Status	Author	Date
[Edit] test question	Open	dutoit	3/22/01 3:40 PM
[Edit] Assistance Se		dutoit	3/22/01 2:25 PM
[Edit] Clarification: for outdated a		dutoit	3/22/01 2:20 PM
[Edit] Justify use cas		dutoit	3/22/01 1:42 PM
[Edit] Justify use case: 1 of 2 of 2000		dutoit	1/25/01 10:22 AM
[Edit] Justify use case: Check One Article	Open	dutoit	1/11/01 1:21 PM
[Edit] Clarification: functions vs. services	Resolved	dutoit	11/2/00 9:39 PM
[Edit] Clarification of Min. outdated articles constraint	Resolved	paech	11/2/00 9:27 PM
[Edit] Assistance service	Resolved	dutoit	11/2/00 8:24 PM

Two callout boxes are overlaid on the screenshot: one on the left pane labeled "REQ: Requirements Specification" and one on the right pane labeled "QOC: Rationale".

REQuest: Overview (2)

The screenshot shows the REQuest application interface. At the top, a blue header contains the text 'REQuest: S' and navigation links '[Systems] [Requirements Specific]'. Below this, the system name 'System: SuperMarket' is displayed. A left sidebar contains navigation options: Favorites, History, Search, Scrapbook, and Page Holder. The main content area lists a table of contents with the following items: '1. Problem Statement', '2. Requirements' (with sub-items: 2.1. Actors, 2.2. User Tasks, 2.3. Domain Constraints, 2.4. Quality Constraints on User Tasks), '3. Specification' (with sub-items: 3.1. Use Cases, 3.2. Services, 3.3. Global Functional Constraints, 3.4. Quality Constraints on Use Cases, 3.5. Quality Constraints on Services), '4. Examples' (with sub-items: 4.1. Actor Instances, 4.2. Scenarios), '5. Glossary', and 'Index'. At the bottom, it states 'Created on 10/31/00 10:00 AM by allen.' and 'Last revised on 11/13/00 11:05 AM by dutoit.' The status bar at the very bottom shows 'Internet zone'.

Overview

Requirements:

- Actors
- User Tasks
- Domain Constraints

Specification:

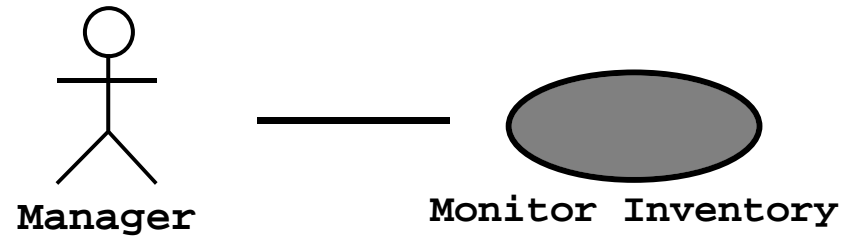
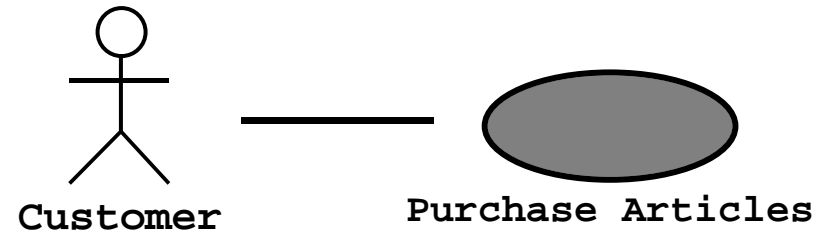
- Use Cases
- Services
- Quality Constraints

Examples:

- Actor Instances
- Scenarios

Glossary

SuperMarket: Identifying Actors & User Tasks



REQ: Creating actors

2. Requirements [\[Attach Diagram\]](#) [\[Remove Diagram\]](#)

This section describes the user needs that the SuperMarket system has to support in terms of actors, user tasks, and domain constraints. Actors are entities that interact with the system. User tasks are actions that the user performs on the system. Domain constraints are rules that govern the system's behavior.

New Actor:

Use the form below to specify a new actor. The name of the actor is a required field and must be unique.

Name:	<input type="text" value="Customer"/>
Description:	<input type="text" value="Customers purchase food articles."/>

Create Actor

2.1. Actors [\[New Actor\]](#) [\[Help\]](#)

- [Customer](#)
- [Manager](#)
- [System](#)

Search
Scrapbook
Page Holder

History
Search
Scrapbook
Page Holder

REQuest Creating actors (2)

Actor: Customer

Navigation

Path: [System](#) > [2. Requirements](#) > [2.1. Actors](#) > [Customer](#)

Actions: [\[Refresh\]](#) [\[Edit\]](#) [\[Add Link\]](#) [\[Delete Links\]](#) | [\[New User Task\]](#) [\[Instantiate Actor\]](#) |

Questions: [\[Unclear?\]](#) [\[Incomplete?\]](#) [\[Inconsistent?\]](#) [\[Ill structured?\]](#) [\[Incorrect?\]](#) | [\[Help\]](#)

Questions

Name: Customer

Description: A [A.Customer](#) purchases [GE:Articles](#) from a branch.

Initiated User Tasks: [Buy Articles](#)

Initiated Use Cases: [Identify Article](#)
[Pay For Articles](#)
[Purchase Articles](#)

Open Questions: [Assistance Service](#)

Created on 10/31/00 10:03 AM by [dutoit](#).
Last revised on 11/13/00 11:05 AM by [dutoit](#).

REQ: Creating user tasks

New User Task:

Use the form below to specify a new user task. The name of the user task is a required field and must be unique. The initiating actor and the flow of events fields should be specified in a complete requirements specification but can be specified at a later time.

Scrapbook Page Holder

2.1. Actors [\[New Actor\]](#) [\[Help\]](#)
[Customer](#)
[Manager](#)
[System](#)

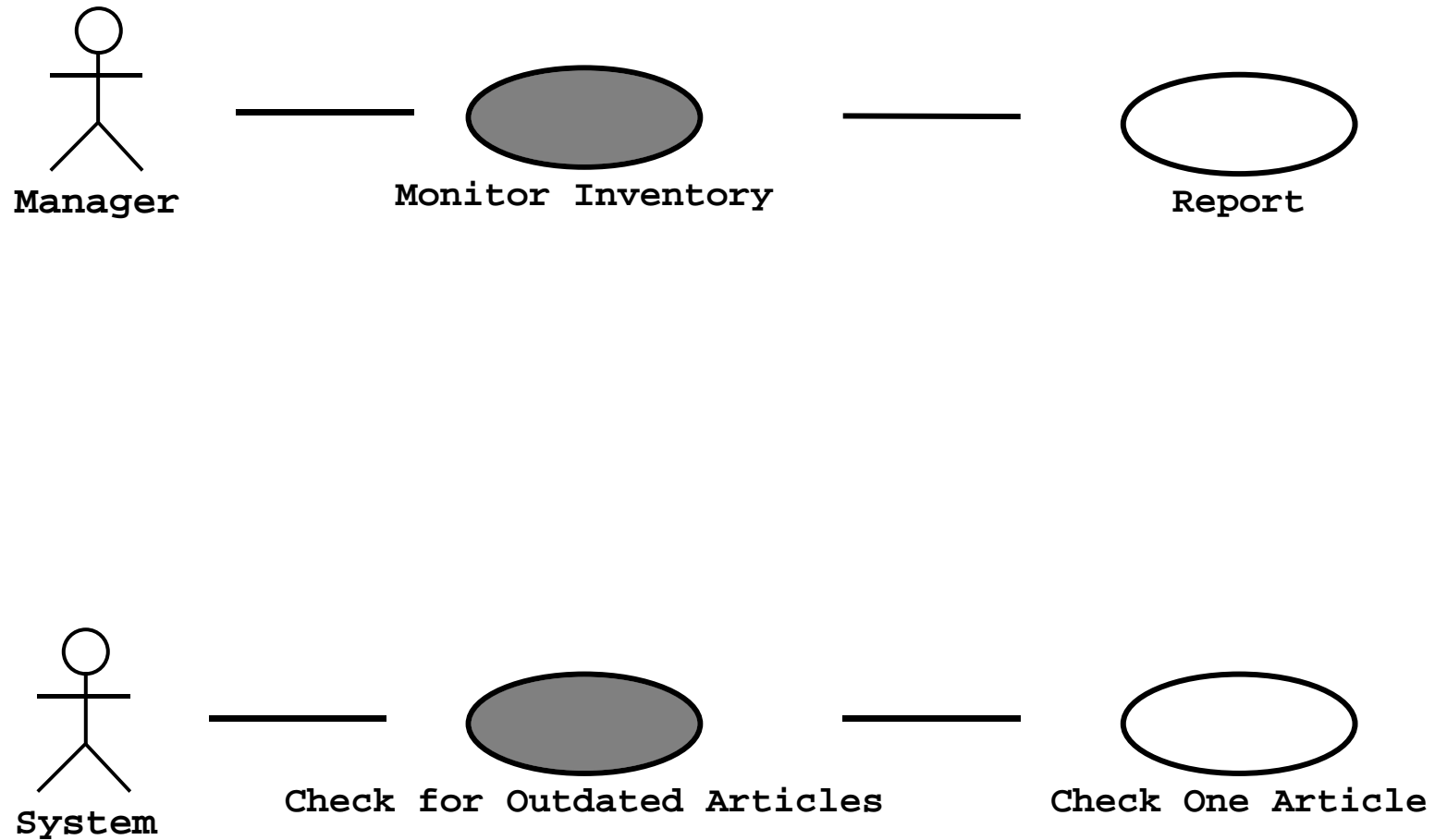
2.2. User Tasks [\[New User Task\]](#)
Actor [Customer](#) initiates the user task
[Buy Articles](#)
Actor [Manager](#) initiates the user task
[Monitor Inventory](#)
Actor [System](#) initiates the user task
[Check for Outdated Article](#)

2.3. Domain Constraints [\[New\]](#)
[Safe check out](#)
[Security \(from theft\)](#)

2.4. Global Functional Constraints
[Extension of report service](#)

Name	<input type="text" value="Purchase Articles"/>
Initiating Actor:	<input type="text" value="Customer"/>
Participating Actors:	<input type="text" value="Customer"/>
Flow of Events:	<input type="text" value="The Customer pay for their articles at the pay station, stack their articles on the conveyor belt, pay the resulting bill, and leave."/>
Frequency:	<input type="text"/>
Preconditions:	<input type="text" value="The Customer is in the store"/>

SuperMarket: Identifying Use Cases



REQ: Creating Use Cases

New Use Case

3. Specification [\[Attach Diagram\]](#)

This section describes the and quality constraints. Use the system. Services are for use cases. Quality constraints by the system.

Buy Articles

Monitor Inventory

Check for Outdated Articles

3.1. Use Cases [\[New Use Case\]](#)

User task [Buy Articles](#) is realized by

- [Identify Article](#)
- [Pay For Articles](#)

Use the form below to specify a new use case. The name of the use case field and must be unique. The realized user task and the flow of events specified in a complete requirements specification but can be specified

Name:

Initiating Actor:

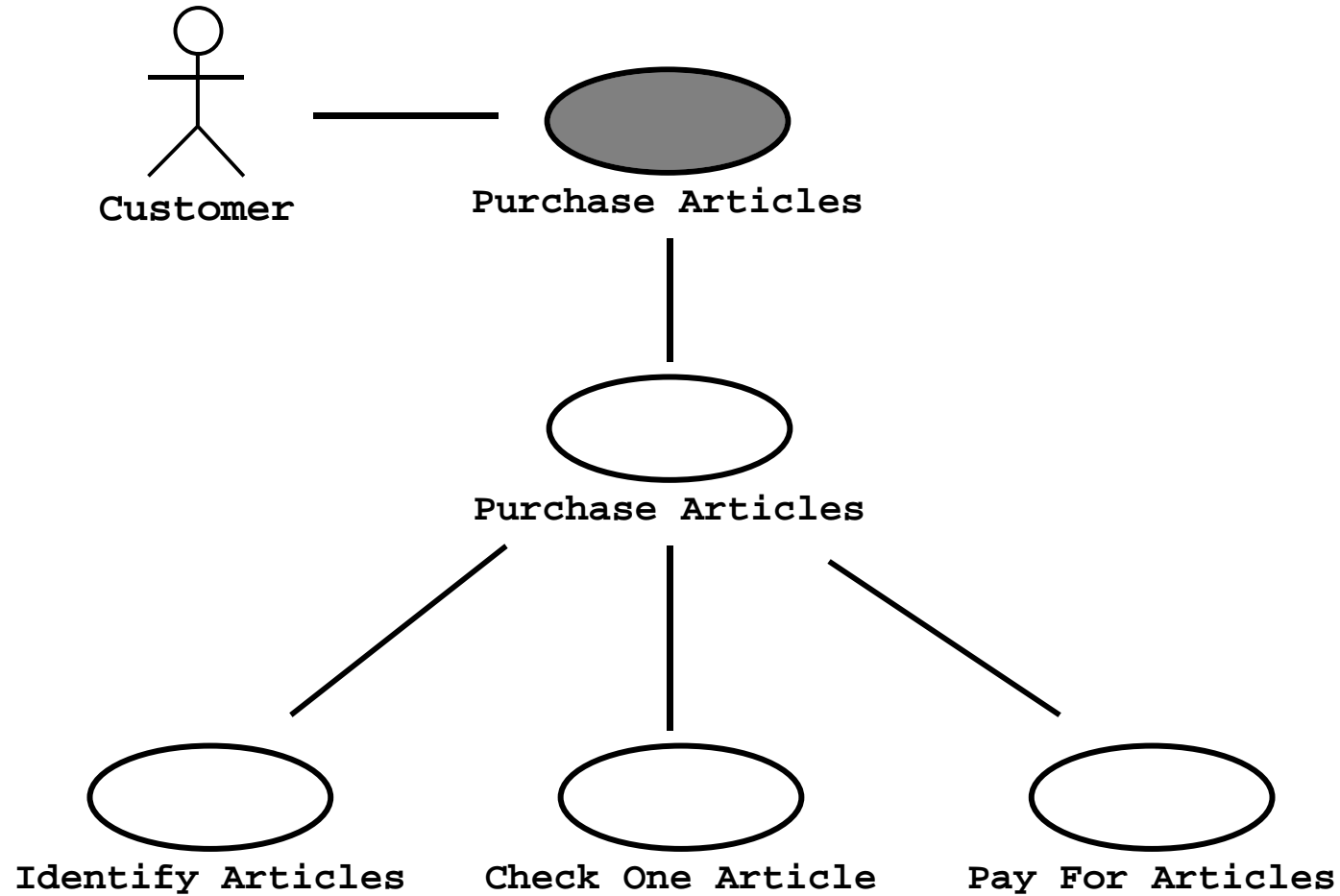
Participating Actors:

Realized User Task:

Flow of Events:

Actors	System
[Insert Step]	[Insert Step]
Actor puts Article on the Conveyor Belt	
[Insert Step]	[Insert Step]
	System scans the label, which contains identification and si [Label unreadable]

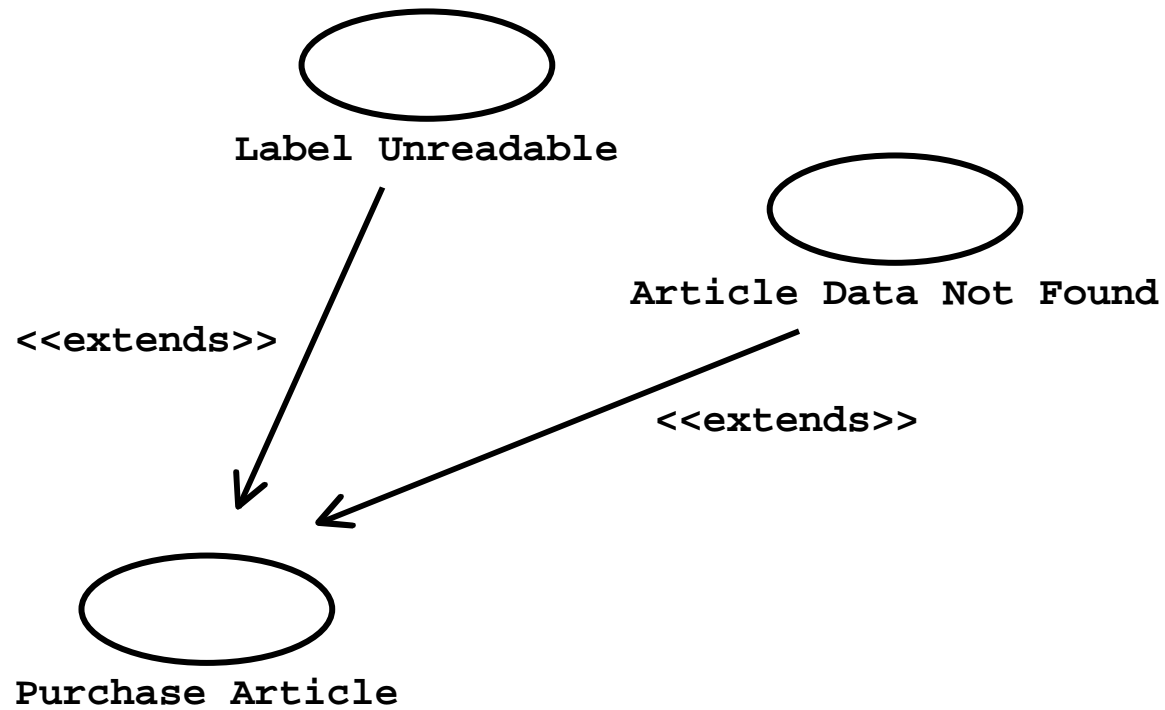
SuperMarket: Refining Use Cases



REQ: Relationships among use cases



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)

Actor: C

▲ ▶
Path: [System](#) > [2. Requirements](#) > [2.1. Actor](#)
Actions: [\[Refresh\]](#) [\[Edit\]](#) [\[Add Link\]](#) [\[Delete\]](#)
Questions: [\[Unclear?\]](#) [\[Incomplete?\]](#) [\[Incon\]](#)

Name: Customer
Description: A [A.Custor](#)
branch.

Initiated User Tasks: [Buy Article](#)
Initiated Use Cases: [Identify Art](#)
[Pay For Ar](#)
[Purchase A](#)

Open Questions: [Assistance](#)

Created on 10/31/00 10:03 AM by [du](#)
Last revised on 11/13/00 11:05 AM by

New Question

Enter your question below by typing a name and a brief description. The name is used to identify the question and should be brief. The description is used for discussions and should be complete.

Name:

Description:

Criteria relevant to this question:

- Responsive Assistance
- Min. Outdated Articles Customer Hands
- Usability of Reports
- Extension of report service with search service

QOC: Asking questions (2)

Describe Possible Options

Specify below possible options for answering this question. Even if you do now know the right answer, providing options below can start a more focused discussion.

Question: Assistance Service
How should customers request assistance in case of failure of the pay station or confusion of the customer?

Option1:

Option2:

Option3:

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 occurrences of "function" with "service". (*paech, 11/2/00 9:48 PM*)

Options [\[Propose Option\]](#)

- [\[Edit\]](#) Replace all occurrences of "function" with "service". (*dutoit, 11/2/00 9:40 PM*)
- [\[Edit\]](#) Replace all occurrences 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".	<i>paech</i>	<i>11/2/00 9:47 PM</i>

REQ: Exporting the requirements specification

System: SuperMarket

[Refresh](#) | [Printable Version](#) | [Question](#) | [Help](#)

1. Introduction [Edit](#)

As part of its overall infrastructure a food vendor wants to install a new pay system in all its branches. This system shall support the following tasks of the branches:

- purchasing of articles by the [Customers](#) and
- monitoring of the stored articles and removal of outdated articles.

The system consists of:

- a [Database of Articles](#) on sale in a branch
- a [Central Control Station](#) for the branch management which allows the data about deliveries and an overview of the current status of all article status includes the number of articles of each kind and the date of valid
- a number of [Pay Stations](#) consisting of a [Conveyor Belt](#) where the [Cus](#) place their articles, of a scanner which identifies the articles, of a check counter which allows for Pay for [Articles](#) with cash or credit card, of a [Basket](#) where [Customers](#) can place not paid [Articles](#) , and of a cancel device with which [Customers](#) can cancel the whole transaction.

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**