Software Engineering

Computer Science

7. Requirements Engineering

Prof. Dr. Klaus Baer

SS 20

Objectives

The objectives of the learning unit are:

- Understanding the importance of requirements engineering
- Learning a methodology for object-oriented requirements analysis
- Learn the appropriate UML elements and how to use them correctly

Content

Terms
Requirement and Requirements Management

- **Requirement**

A requirement is a statement about a property to be fulfilled or the performance of a product, a process or the persons involved in the process.

- **Requirements Management**

Comprehensive, systematic approach to identifying, documenting, organizing and tracking requirements.

Requirements
Critical and difficult topic

- **Garbage in, garbage out**

  Peter Drucker:
  “Doing the wrong thing right is not nearly as good as doing the right thing wrong.”

  really worth watching:
  https://www.youtube.com/watch?v=OqEeIG8aPPk

- **Why is it so difficult?**

  Peter Wegner, 1997:
  “It is impossible to fully specify or test an interactive system designed to respond to external inputs”

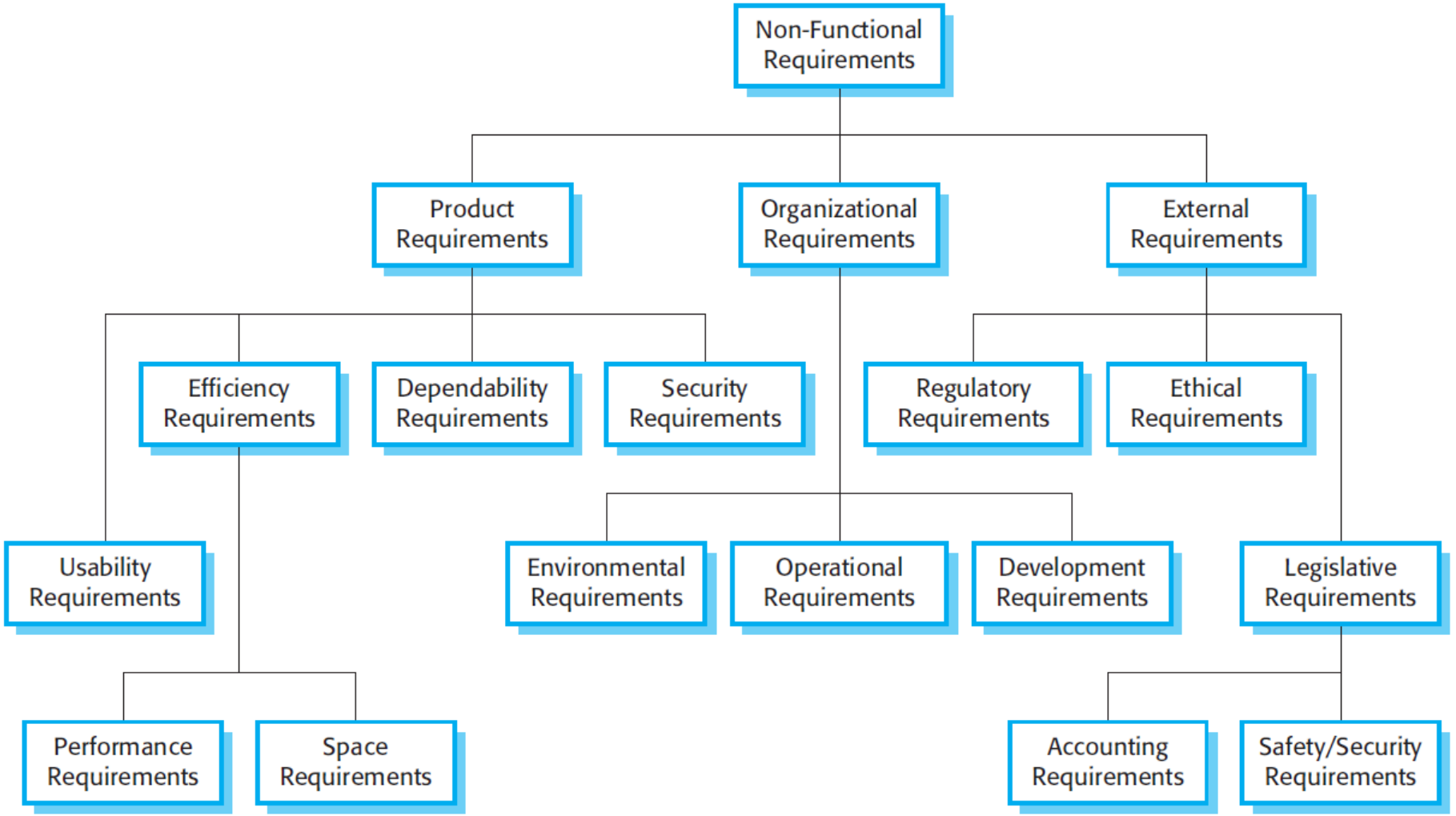
  Watts Humphrey, 1995:
  “For a new software system, the requirements will not be completely known until after the users have used it.”

Classification of Requirements
Functional and Non-Functional

- **Functional requirements**
  - describe the domain-oriented functionalities or services that the system is supposed to provide from the user's point of view.
  - depend on the type of system to be developed and its users.

- **Non-functional requirements**
  - describe all other required characteristics of the system
  - do not directly affect the special functions of the system
  - often refer to important characteristics that affect the system as a whole, e.g.
    - user-friendliness
    - trustworthiness
    - efficiency
    - maintainability

Classification of Requirements
Non-Functional Requirements



Sommerville, I: Software Engineering. 6. Auflage, Addison Wesley, 2001.

Classification of Requirements

- **Operational requirements**
  - describe interfaces, data, functionalities and reactions of the system to events (also extraordinary events → exceptions).
  - operational requirements are also referred to as a business concept.
- **Quality requirements**
  - concern software quality criteria such as reliability, maintainability, efficiency and usability.
  - should be specified quantitatively if possible (→ Verifiability).
- **Technical requirements**
  - include constraints such as devices to be connected, interfaces to external systems and the use of development tools.
- **Validity and maintenance requirements**
  - prepare tests (e.g. by specifying test cases)
  - define the acceptance test
  - describe the scope of warranty conditions, maintenance conditions, training,...
- **Implementation requirements** concern
  - the process model and the documentation,
  - the resources available (personnel, dates, costs),
  - Additional conditions such as legal regulations, (company-internal) guidelines and standards

Kühnel: Req. Eng. - Versuch einer Begriffsklärung. In: Informatik-Spektrum 10 (6), p. 433, 1987

Classification of Requirements

- **UP clasifies according to the “FURPS+ model”**
  - Functionality
  - Usability
  - Reliability
  - Performance
  - Supportability

- **The "+" in FURPS+ is intended to draw attention to other requirement categories like:**
  - design constraints
  - implementation requirements
  - interface requirements
  - physical requirements.

Grady, R.: Practical Software Metrics for Project Management and Process Improvement. Prentice-Hall,1992.

Requirements Analysis
Objectives

- A common understanding of the performance of the target system among all stakeholders involved in the project (customer, user, service provider, financier, interface partner,...).
- Giving software developers a deeper understanding of the system requirements.
- Determining the functionality of the system.

Based on the requirements, the planning of the project can be carried out including a well-founded effort and cost estimate.

The requirements describe the external behaviour of the system.

- The description of the internal structure of the system is subject of the design.

Requirements Analysis according to the Unified Process

- Requirements management is one of UP's six best practices:

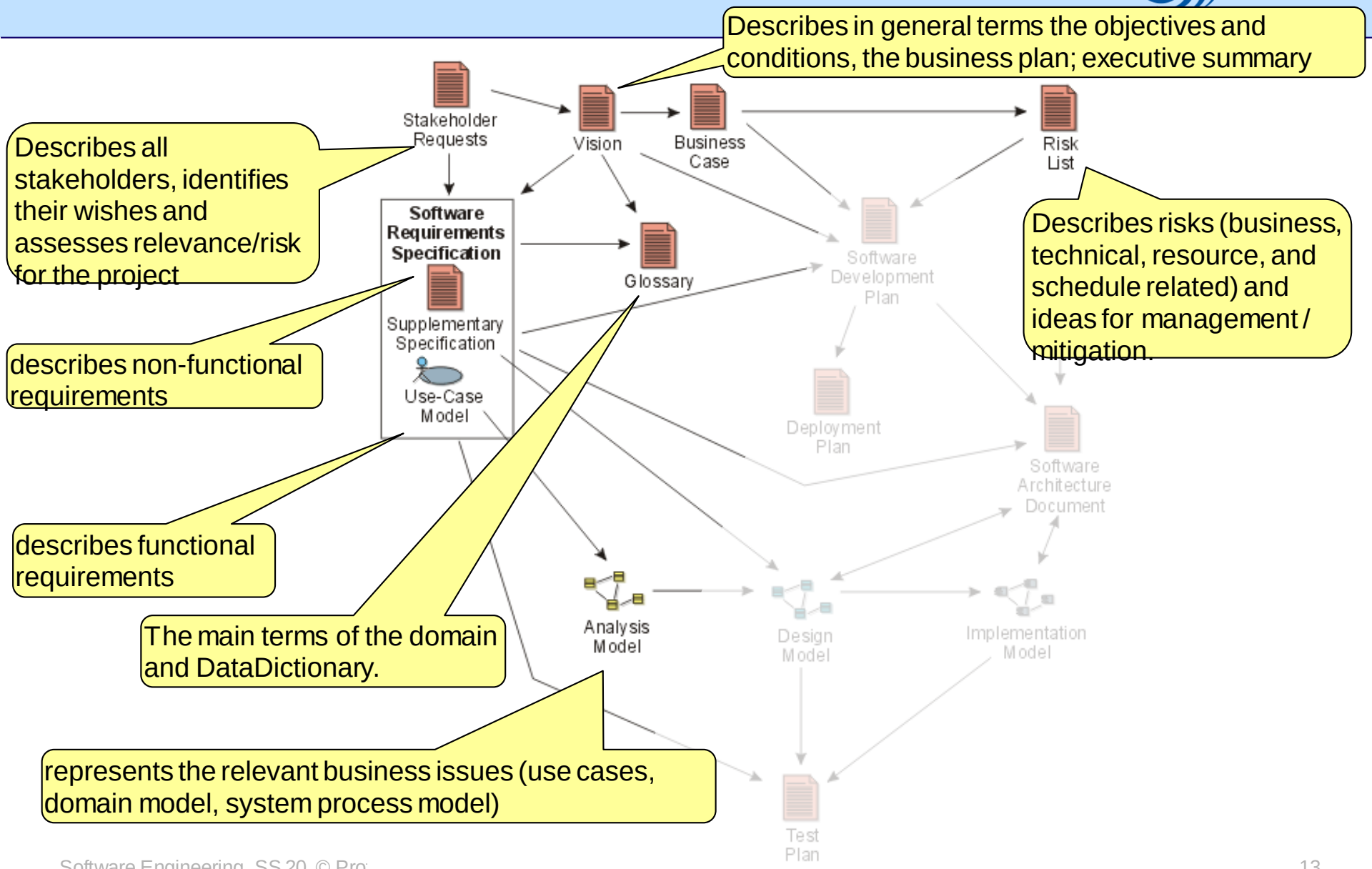
> A systematic approach to finding, documenting, organizing and controlling changes in the requirements of a system.

- Cornerstones:
  - Requirements are recorded iteratively and further developed.
  - Requirements analysis is not a phase but a discipline that lasts the whole project (with emphasis on inception and elaboration).
  - With iterative refinement, feedback from already completed software versions is included.
  - Requirements are developed in workshops in cooperation with customers, application experts, business people and technicians.
  - A list of requirement documents is proposed. Functional requirements are specified with the help of use cases.

Requirements Analysis Challenges

- Requirements are neither obvious nor unchangeable over time.
- An incorrect level of detail in the formulation of requirements eliminates the flexibility for design decisions.
- 100% actuality and consistency of all recorded requirements can hardly be guaranteed with increasing complexity of the project.
- The identification of requirements requires competence in many areas, especially in the project domain and information technology.

Requirements Analysis
The most important artifacts of the UP for requirements analysis



Content

Requirements Analysis Methods

- **Actual state analysis:**
  What is the situation?
  Describes the current status
  (System environment, if necessary old system, …)

- **Structured Analysis (SA)** → **Procedural Programming**
  Data flows, data dictionary, pseudocode, decision trees

- **Object-Oriented Analysis (OOA)**
  Use cases, class diagrams, interaction diagrams, state machines

Actual state analysis
Activities

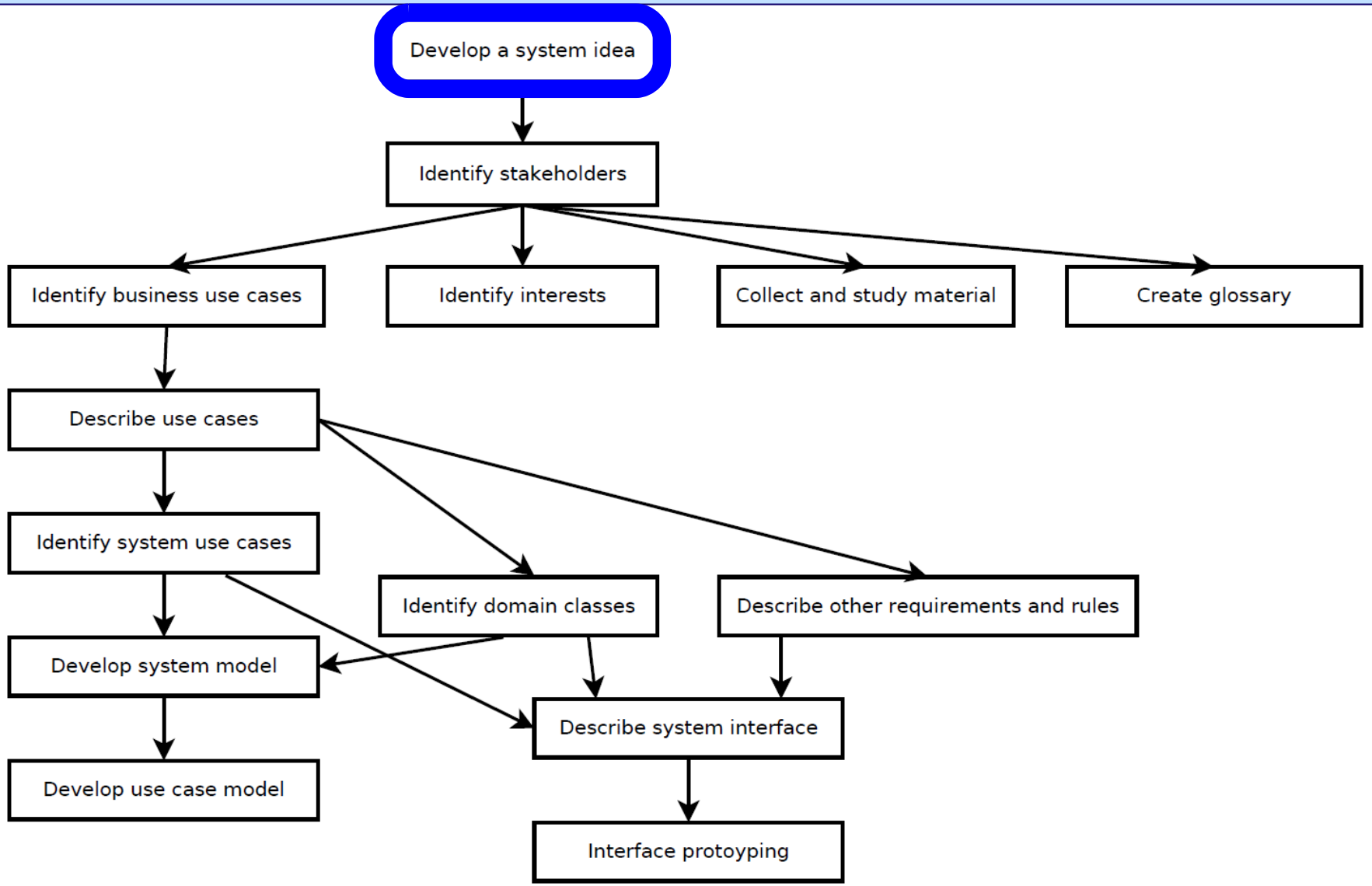
- Capturing of the user/user environment (structural and process organization, service regulations, decrees, laws, guidelines and similar more)
- Capturing existing data, programs
- Capturing existing IT equipment
- Capturing of time and quantity structure (current processing times, turnaround times, response times, waiting times, data quantities)
- Capturing of business and technical factors that cannot be influenced
- Presenting the threat and the equipment gap,
- Identification of weak points
- Identifying the causes of the identified vulnerabilities

Actual state analysis
Sources of information

Sources of information for the actual state analysis:

- conversations
  - Management, end users, IT managers, system administrators,...
- visits
  - Observe people at work
- action protocols
  - Records of the current task completion
- forms
  - Provide information on data requirements and workflows
- flyers, brochures
  - How a company wants to be seen, which competencies need to be strengthened

Object-Oriented Analysis
Proposal for a systematic approach



Oestereich, B: Analyse und Design mit UML 2. Oldenbourg, 7. Auflage, 2005.

Object-Oriented Analysis
1. Develop system idea and objectives

THU Technische Hochschule Ulm

Objective

- find the fundamental objective and system idea
- What should be achieved with the system to be developed?

Action

- Develop the system idea together with the client, product recipient, user and developer, actively clarifying conflicts of interest and contradictions.
- Formulate the system idea briefly and concisely but absolutely in writing with about 5 - 20 sentences. Consider the most important characteristics, features, framework conditions, prerequisites and explicit exclusions of performance.
- Make sure that clients, product recipients, users and developers know the system idea and support it without reservation.

Artifact

- Vision-Document

Well or not well?

In a car rental company, the reservation and rental of cars and the invoicing of rentals should be supported by an information system.

The new system to be developed will provide all functions directly related to customer care. This includes customer advice, administration of master data (addresses, bank details, etc.), reservation, rental of vehicles and billing with the customer.

Remote and indirect areas, such as internal accounting, tariff and product planning, vehicle transfer/disposition, etc., are not system components.

Object-Oriented Analysis
Example for the system idea / vision

Good:

Fast-Cars 1.0 is the software that supports you in all important car rental applications: You can reserve (incl. change and cancel), rent (incl. contract creation and withdrawal protocol) and invoice the rentals (individual, collective and monthly invoices).

Of course Fast-Cars 1.0 can also manage the vehicles to be hired, the existing rental stations and the customers with all important master data (addresses, bank details, etc.). Data can be transferred from the legacy system.

Other important features of Fast-Cars 1.0: multi-client capability, agency operation, participation in single sign-on, archiving of legacy data, various evaluations, reports and statistics (not on the Internet), 24*7 operation.

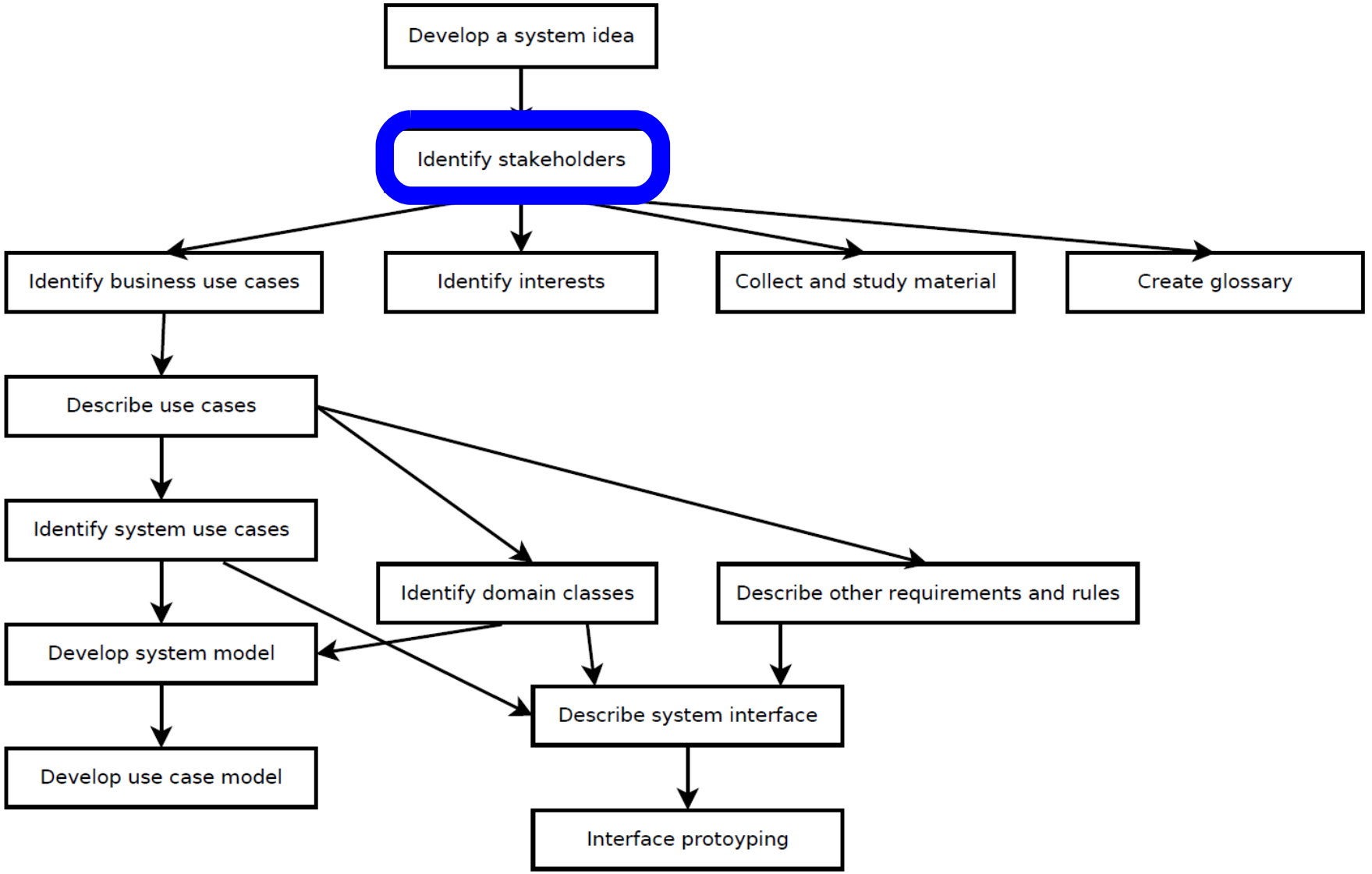
Fast-Cars 1.0 runs in your company's network and also offers Internet access for customers and agencies. System requirements are interfaces to the existing TSC tariff and product system and to SAP-FI financial accounting.

Additional modules for vehicle transfer/dispatch, claims processing and complaint management are planned as well as multi-currency and multi-language capability for Fast-Cars 2.0.

Checklist

- Is the size of the system idea about half a page?
- Does the system idea describe what is to be achieved with the system to be developed?
- Is the system idea formulated in such a way that the completion of the product was mentally anticipated (... and could be printed on a product carton)?
- Are the most important features / performance features listed?
- Are the most important prerequisites / general conditions listed?
- Is there a demarcation with regard to properties that the product does not (currently) offer?

Object-Oriented Analysis



Object-Oriented Analysis
2. Identify stakeholders

Objective

- Ensure that all relevant stakeholders are taken into account.
- Find out which groups of people can provide requirements for the system

Action

- Identifying stakeholders.

- Assess the importance of stakeholders based on relevance and risk.
- Objective: classify stakeholders as "must - should - could" be considered.

- Identification of concrete project contact persons (name, function, contact data)

- Classify the contact persons into
- Experts, those responsible for requirements and those affected by the system.

Key stakeholder groups

- end users
- specialist department
- audit department
- Client, Financier, Management, Board of Directors, Management
- Legislators, Standards
- consumers
- System administrators, service personnel, training personnel, hotline, support
- System Developer, System Maintenance
- Purchaser of the system
- Marketing / Sales
- Project opponents and supporters

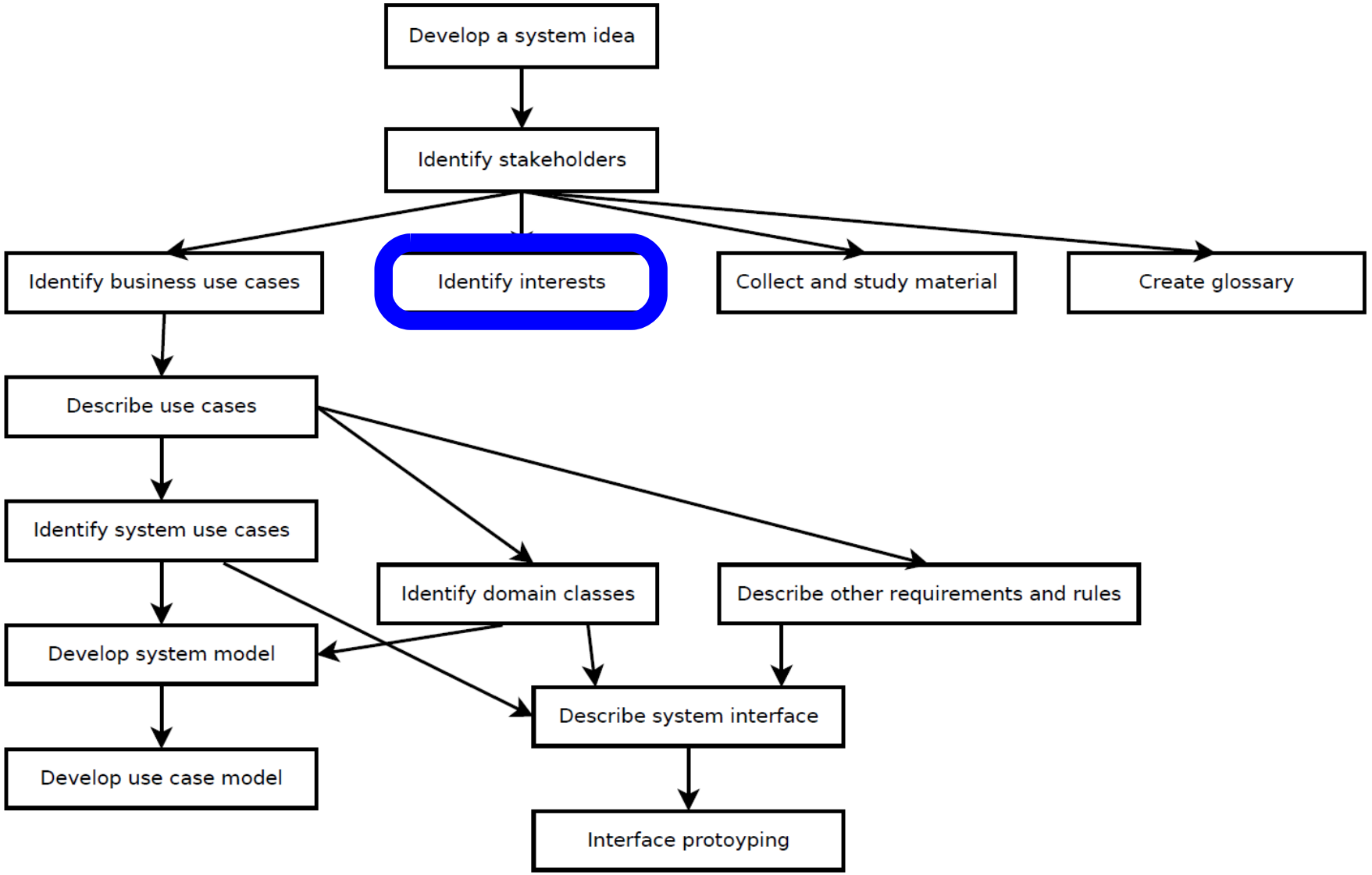
THU Technische Hochschule Ulm

Identification of project contact persons

- Contacts are selected from all stakeholders.
- There are three categories of contact persons:
  - domain experts
  - requirements manager
  - people affected by the system
- Important:
  - All categories must be represented!
  - All important stakeholders must be represented!

- The quality of the analysis results is decisively influenced by the contact persons!
- Therefore, communication with them is of crucial importance!

Object-Oriented Analysis



THU Technische Hochschule Ulm

Objective

- Ensure that the requirements of all stakeholders are taken into account.

Action

- Describe the objectives and interests of each stakeholder.
- Identify existing problems and vulnerabilities from the stakeholder perspective.
- Describe the important required system properties from the stakeholders' point of view.

Artifact

- Vision-Document → Summary of key interests
- Stakeholder Request → Detailed description of individual interests

Typical difficulties

- Interest holders know what they want, but they can't express it.
- Interest holders don't know what they want.
- Interest holders think they know what they want until you give them what they want.
- Analysts believe they understand user problems better than the users themselves.
- Everyone thinks everyone else is politically motivated.

Object-Oriented Analysis
3. Identify stakeholders interests - Example

- **The call center management**
  - sees it as a problem that the existing system is too slow during peak times
  - sees it as a problem that even minor deviations from the normal course of the reservation are not reasonably supported by the existing system, which causes a considerable expenditure of time.
  - expects the new system to reduce costs per reservation by at least 30 percent on average.

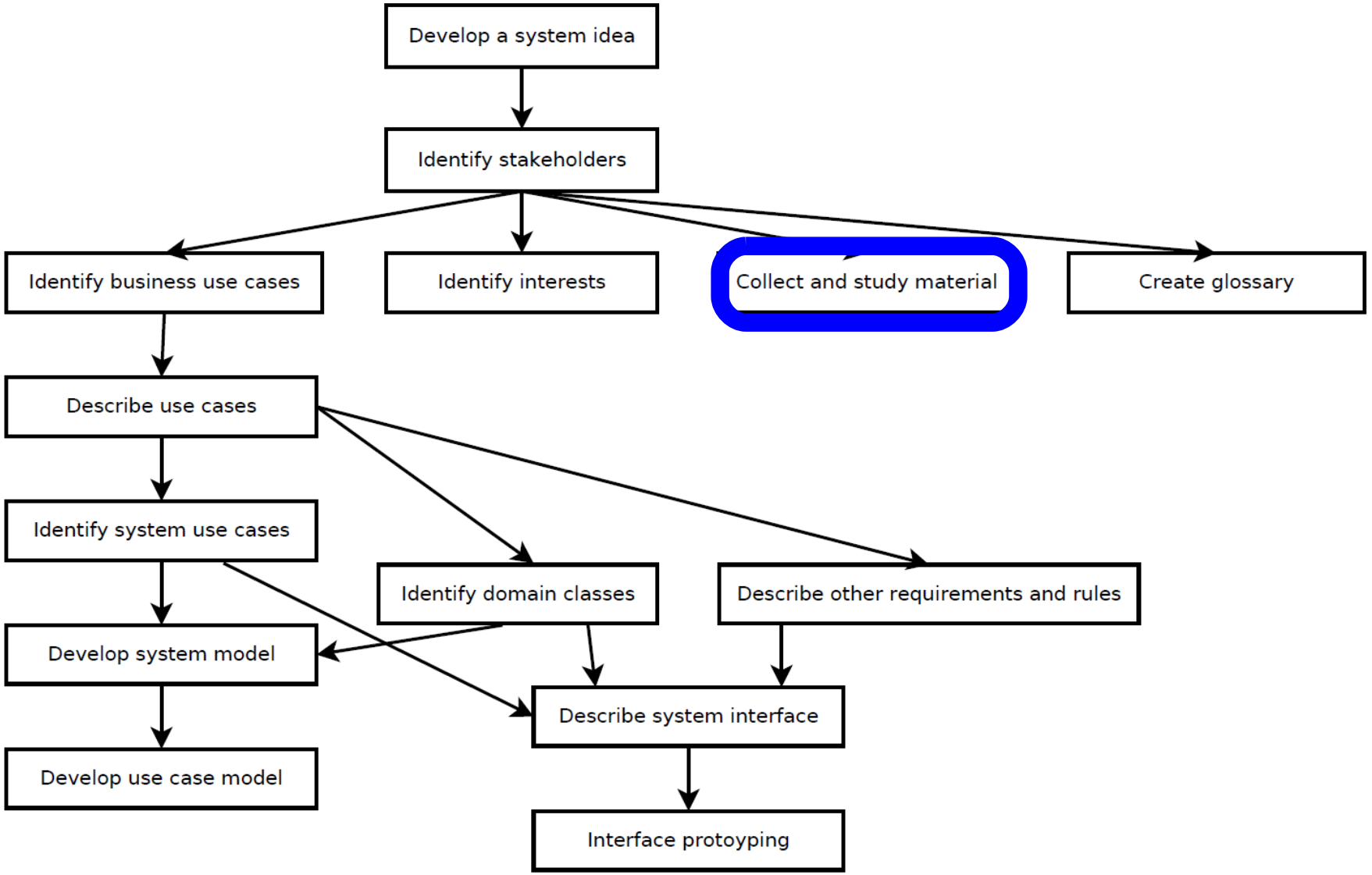
- **Call-Center-Agents**
  - According to the call center agents, the existing system does not adequately support their work, which is why they are often unable to provide the service desired by the customers.
  - There are fears that more direct Internet reservations could endanger jobs in the call centre.
  - The call center agents expect the new system to enable them to offer customers better and more attractive services than an Internet reservation.
- …

THU Technische Hochschule Ulm

Checklist

- How long has the stakeholder been considered "competent" in his area of expertise?
- What are the rights of the stakeholder?
- What interest does he have in the new system?
- What should the system do in any case?
- What fears / reservations does he have about the new system?
- ...

Object-Oriented Analysis



Objective

- Learning from previous practices and accessing additional sources of information.

Action

- Identification and analysis of objects, examples and patterns from the domain.
- Evaluation of the materials with regard to relevance and usability for the current project.

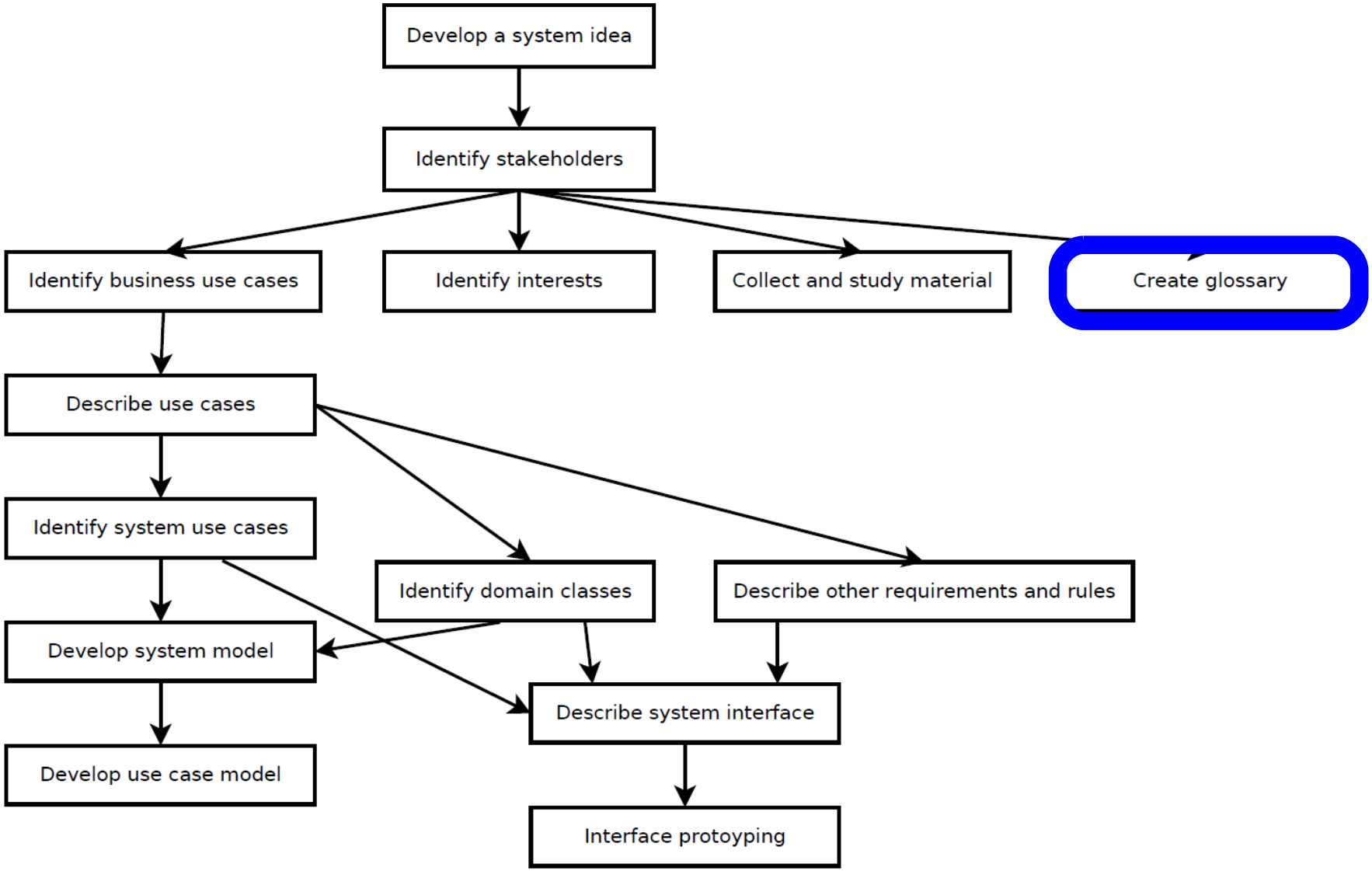
Artifact

- a list of all collected materials

Checklist

- Is the material accessible and easy to find for all project staff?
- Can the materials be referenced? (e.g. unique number)
- Is each material classified in terms of timeliness, commitment, correctness and importance?

Object-Oriented Analysis



THU Technische Hochschule Ulm

Objective

- Create a uniform, consistent understanding of terms and minimise misunderstandings between developers and users.

Action

- Creation of a technical glossary and definition of all important technical terms.
- Define all classes of the class model as a term in the glossary.
- Define all association roles as a term in the glossary.
- Define all other important technical subjects, concepts and states of these subjects in the glossary.
- Define all important general and technical process words in the glossary.

Artifact

- Glossary

Object-Oriented Analysis
5. Create glossary

Term	Invoice
Synonyms	-
Shortcut	-
Definition	Each invoice results from a contract. It invoices services rendered or deliveries and is addressed to a customer.
Delimitation	There are individual, monthly, partial and collective invoices. An invoice has an invoice recipient, a date, an invoice number, and invoice items that are used to list the individual services and deliveries to be billed for. Each item contains a description, a number, an individual amount and a total amount (item total). The invoice contains a final total (sum of all items). The sales tax is displayed separately for each item and all totals.
Constraints	
Contact person	
Status	Final
Changes	...

Object-Oriented Analysis
5. Create glossary - language consolidation

Problem:

- Language inaccurate, understanding of language different

Consequence:

- Different people may talk about the same subject for a long time using the same terms without realizing that they have different views of things.

Remedy:

- Adhering to rules for language consolidation can help to gain more security.

Copyright © 2001 United Feature Syndicate, Inc.

Interrogate process words

- Process words are verbs and substantiated verbs such as reserve, book, message (communicate).
- Ask the W-questions for each process word: who, where, what, when, how.
- For example "Who reserves?", "What is reserved?" etc....

Question all comparisons and comparison forms

- Comparisons are formulations containing words such as "better", "faster", "simpler".
- comparison forms contain superlatives of these words ("best", "fastest", "simplest").
- Ask what the comparisons/superlatives refer to ("faster than what?"), how the required property can be measured or compared and with what accuracy.

Question all universal quantizers

- Universal quantizers are words like "all", "never", "always", "everyone", "always", etc. that describe a set or number of something.
- Ask here for the possible exceptions and the associated assumptions:
  - "Is there really a monthly bill to write every month?"
  - → No, only if the invoice amount exceeds 20.00 €..

Review all conditions, exceptions and variants

- In formulations containing words such as "if", "then", "dependent on", etc., ask whether these really are all possibilities or whether there are more.
- Define every possible variant, create a complete list of all possibilities.

- **Use active instead of passive formulations**
  - Instead of "The rental contract is concluded",
    it is better to write: “the branch office concludes a contract with the tenant".
- **Do not use synonyms, homonyms or tautologies**
  - Synonym: meaning related word
  - Homonym: identical word of other meaning
  - Tautology: repeat what has been said with a related word
- **Use verbs instead of nouns that are not technical terms**
  - In "The customer receives a message about...", "message" is not a technical term, but a masked process.
- **Use terms only in justified cases in the plural**
  - It is better to write "the customer returns the car"
    than "the customers return the car".

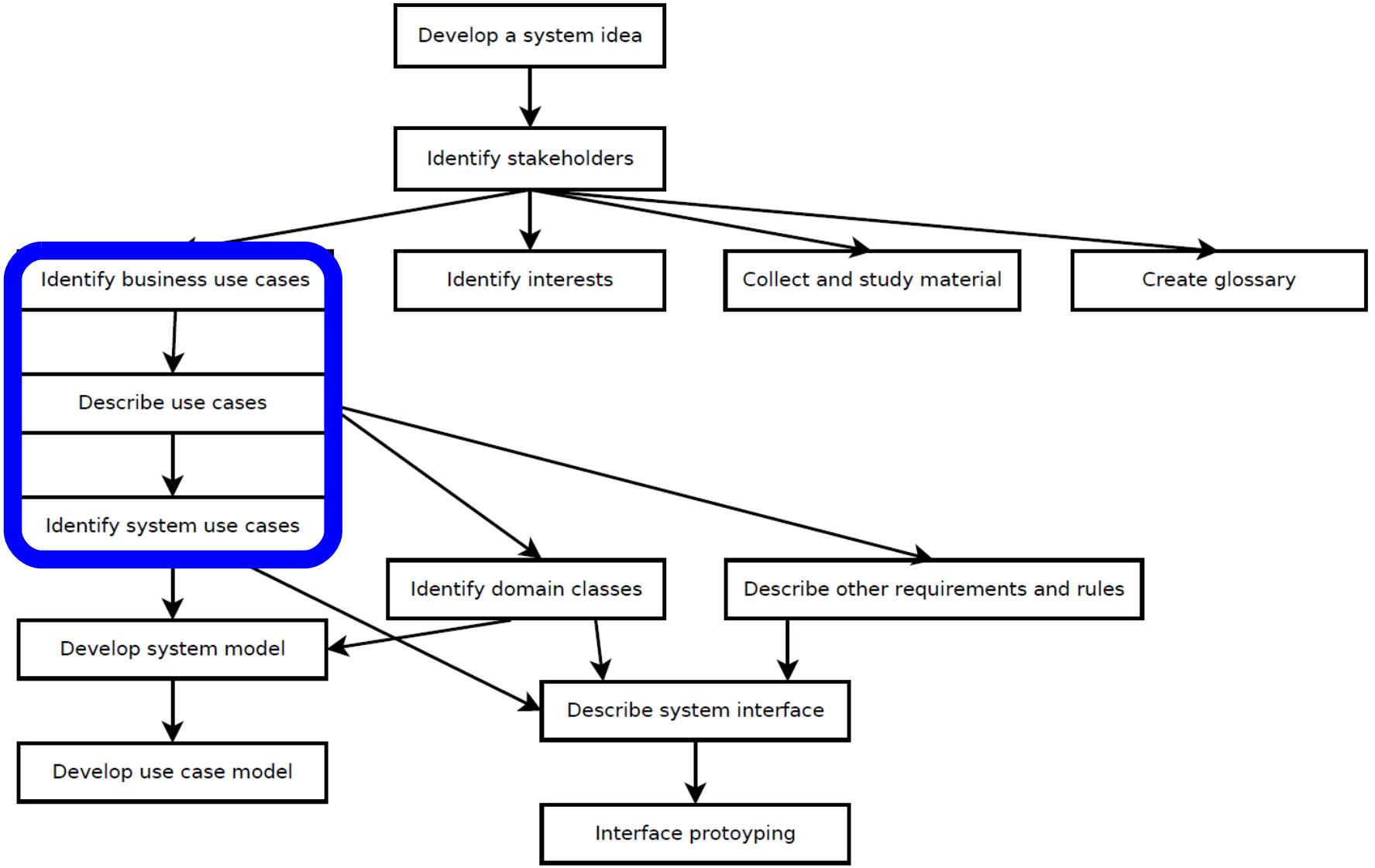
- **Use the most qualified terms possible**
  - "Quality Checklist" instead of "Checklist".
- **Do not confuse the information with the information carrier**
  - Example: "Customer File" and "Customer"
- **Pay attention to the possible miscommunication in case of wrong designations**
  - The "glove box" may continue to be called this way because there is no risk of miscommunication.
  - "Tray" would create more uncertainty.

THU Technische Hochschule Ulm

Checklist

- Has every glossary entry been checked using natural language methods?
- Are important verbs defined in the glossary?
- Is the glossary entry formulated as concisely as possible?
- Are important synonyms defined in the glossary?
- …

Object-Oriented Analysis



THU Technische Hochschule Ulm

What is a use case?

- A use case is a written story.
- It describes business processes or procedures when using a planned system at a high level of abstraction.

Use case:
A use case describes the interaction with a system by means of a coherent workflow. A use case is always initiated by an actor and usually leads to a visible result for the actors.

What is a scenario?

- A scenario is a specific sequence of actions and interactions between actors and the system.
- Each concrete path through a use case represents a scenario.
- A use case is a collection of related scenarios.

Distinguish between:

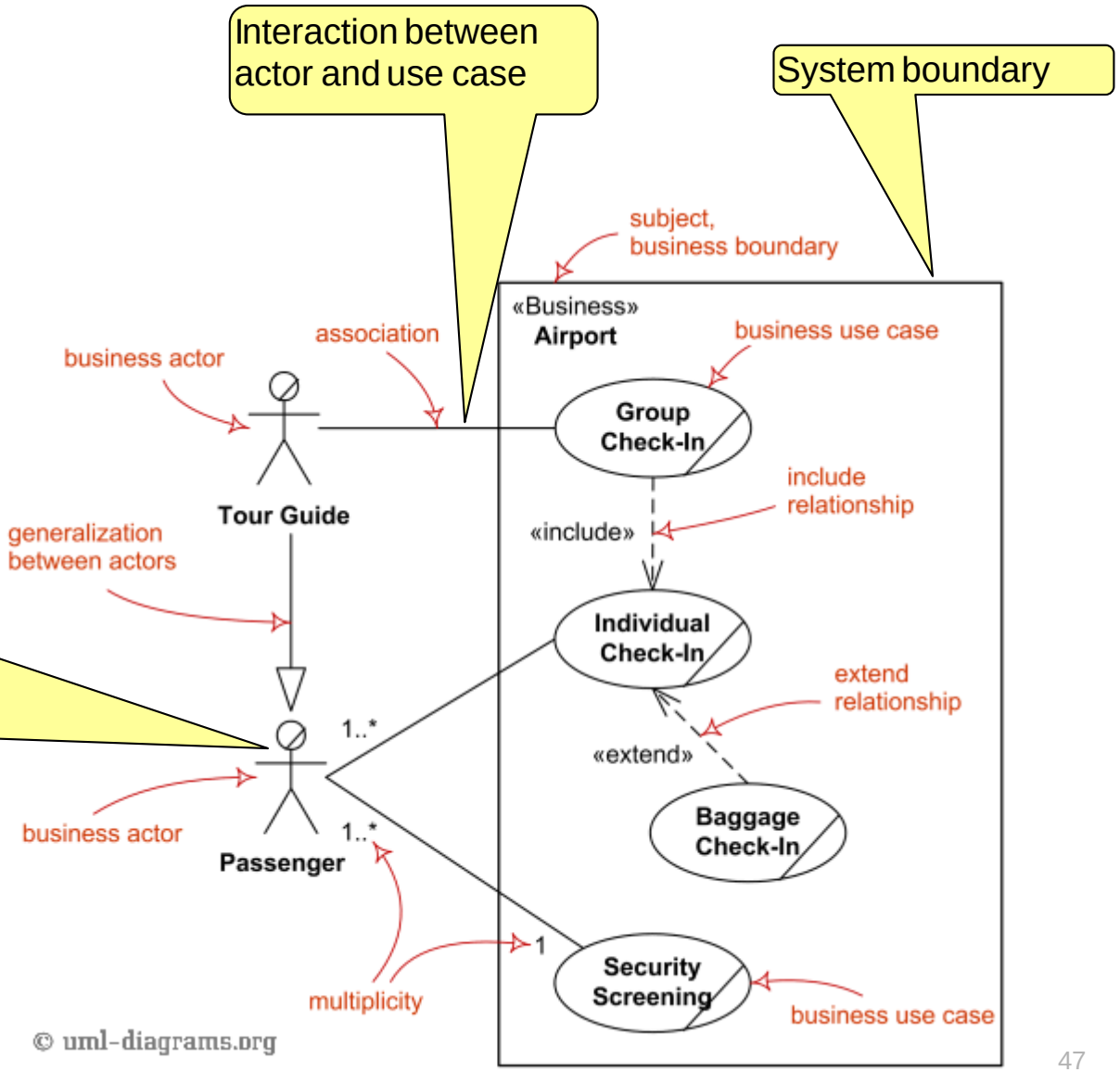
- **Business use case:**
  - describes a process flow on a business level independent of a technical system implementation. Triggered by a business event and typically leads to a result that represents a business value.

- **System use case:**
  - describes the behaviour of a system (hardware or software) that can be perceived by external actors (users, neighbouring systems).

THU Technische Hochschule Ulm

Use Case Diagram

- to get an overview



Actor:

- An actor is a role that a user of the software system plays.
- Actors can be people or automated systems.
- They are always outside the system.

- The graphic representation in the use case diagram provides an overview!
- The real work is the written development of the individual use cases!

Common notation forms:

- **Short notation in natural language** - often in tabular form, early requirement phase
  name
  short description
  Actor(s)
  triggers
  Result(s)

Description of business use case	
Name	Reserve car
Short description	A customer reserves a car at car rental XYZ
Actor(s)	Customer, call center agent
Triggers	A customer contacts car rental XYZ to rent a car.
Result(s)	Car is reserved for the customer

- **Informal - several paragraphs**
- **Fully elaborated -** all steps and variants are described in detail
  detailed description of the requirements

Description of a system use case	
name	Book a car by phone
specialization of	Reserve a car
short description	A car is reserved by telephone for a customer for a defined period of time.
actors	Caller, call center agent
trigger / Motivation	The caller wants to reserve a car.
preconditions	Customer is registered
incoming information	Customer no., customer name, caller name, reservation request
result	A car has been reserved for the customer.
postconditions	Vehicle reserved for customer.

... (Main section see next page)

Description of a system use case	
contact	Mr. John Doe
risk	modest
relevance	essential, high prio
effort	still to be estimated
stability	unstable

Description of a system use case	
Sequence	**1. identify customer** The caller states his name, his customer number and (if different from the caller) the customer name. The customer number is used to search for the customer; the stored customer name must match the customer name specified. **2. take up reservation request** The caller specifies his reservation request, i.e. vehicle type, reservation period, pick-up and return location and special features. **3. check reservation possibility** The system determines whether the given reservation request can be fulfilled. **4. reserve a car** A vehicle is reserved for the customer as desired, he receives a reservation number for confirmation. No specific vehicle is reserved, only a quantity of the desired vehicle type. **5 Confirm reservation** The caller receives verbal confirmation of the reservation and is given the reservation number. On request, the reservation will be confirmed in writing, by fax or e-mail.

use case section	comment
use case name	Name consisting of object and verb (vice versa!)
short description	Brief summary of the main steps
primary actors	Who are the users of the application case?
triggers	What triggers the use case?
preconditions	What must be guaranteed at the beginning and is it worth reporting to the reader?
postconditions	What is guaranteed after the successful completion and is worth reporting to the reader?
standard sequence	The scenario for a typical, unconditional successful run through the use case
alternative sequence(s)	Alternative scenarios for success or failure
Special requirements	Non-functional requirements belonging to the system

Problem: Search for a suitable level of abstraction

Challenge in the identification of use cases:

- What is a meaningful degree of abstraction and how do you find it?
- What granularity should the functionality be broken down into?

- Too fine:
  - Complexity problem, too many use cases.

- Too rough:
  - Many overlaps, redundancy, little benefit in development planning

Object-Oriented Analysis
Developing Use Cases - Level of Abstraction

What are useful use cases?

- Negotiate a supplier contract
- process returns
- Log In
- Move figure to playing field

Indication 1: The boss test

- The Boss test can be helpful in finding the right granularity:
  - Boss: What have you been doing all day?
  - I: I have logged in!
- Will the boss be happy?

THU Technische Hochschule Ulm

Indication 2: Elementary Business Process Test

- The EBP test can be helpful when searching for the correct granularity:
  - Test: Check if the application is an EBP!
  - An Elementary Business Process (EBP) is
    - a task performed by
    - one person at
    - one location at
    - one time
  - in response to a business event that creates measurable business value
    (such as approving credit or calculating order price).

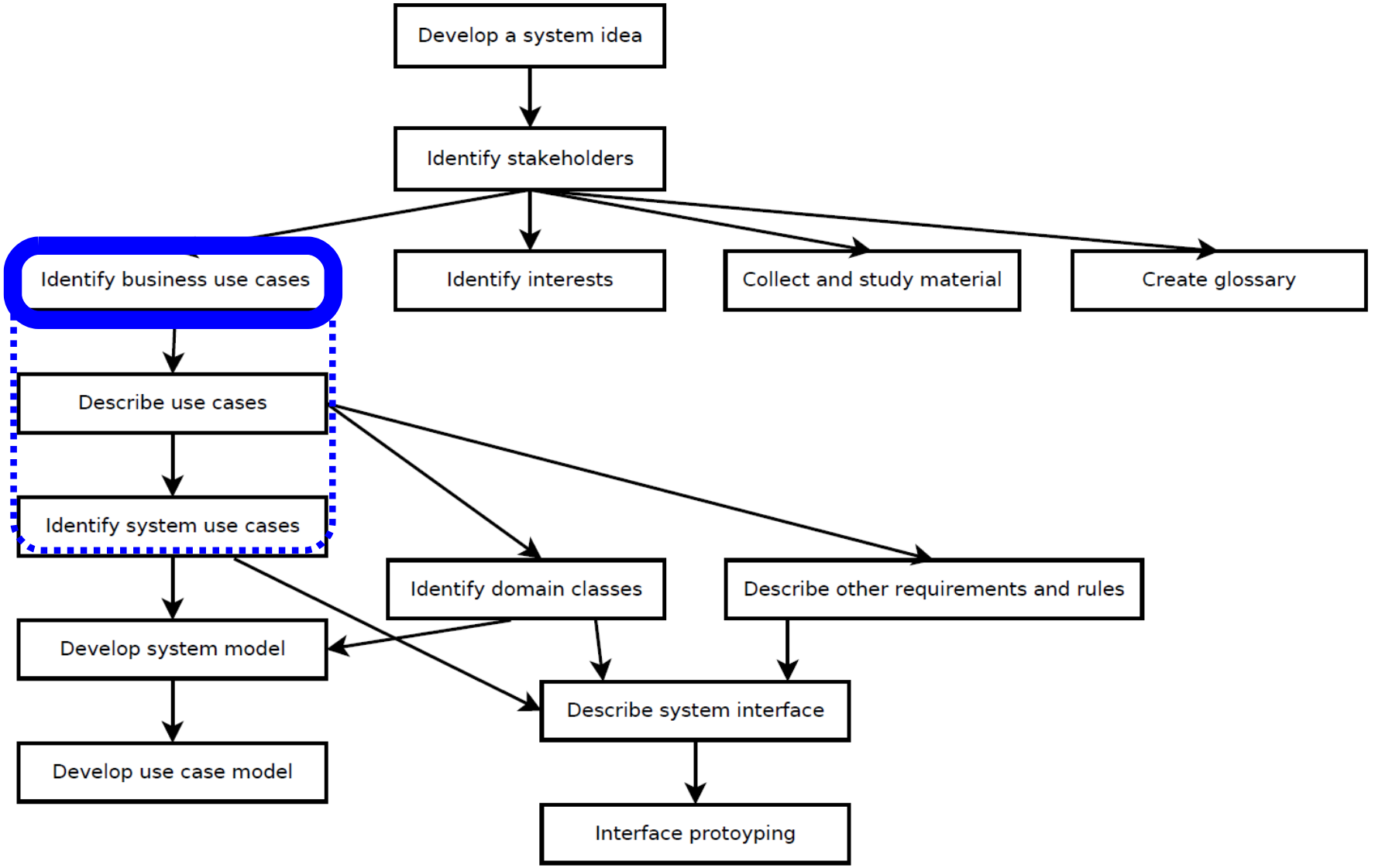
Indication 3: size test

- The size test can be helpful in the search for the correct granularity:

- Testing:
  - How long is the fully developed use case?

  - 6-10 pages of text: → OK
  - Only one step in the standard procedure: → oops??

Object-Oriented Analysis



Objective

- Initial recording and rough description of the requirements

Action

- Decide whether business use cases should be identified at all
- Identifying business use cases.
- Identification of triggers and results of use cases
- Identifying the use cases to be excluded

Artifact

- Use Case Model

Object-Oriented Analysis
6. Identify buiness use cases

- Business use cases are not suitable for functional decomposition.
  - in the corresponding UML use case diagram, there are therefore no chains of business use cases.
  - Business use cases combine as large and time-consuming processes as possible (e.g. "Add new customer").

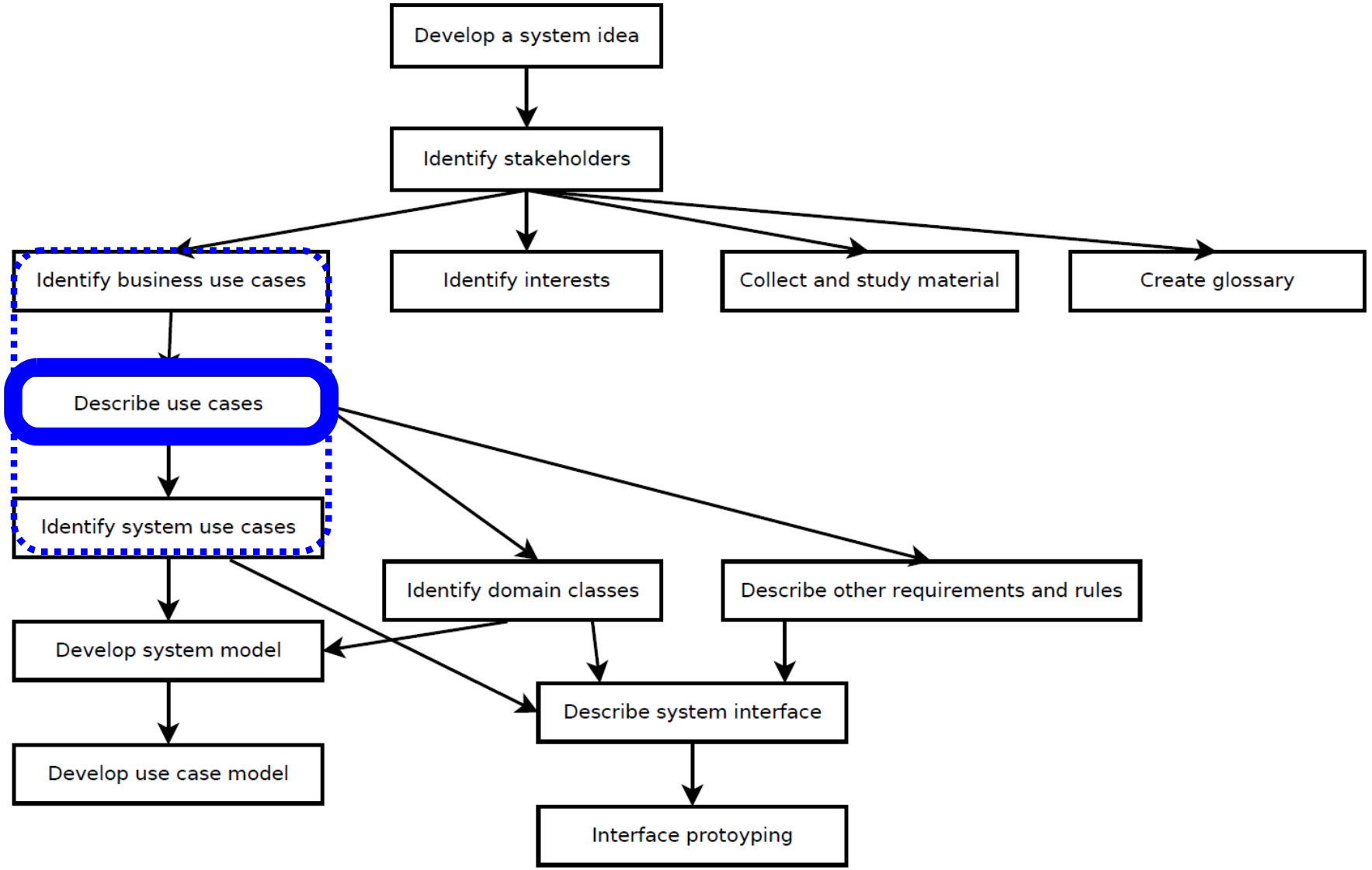
- A business use case describes a process flow,
  - which is not refined in the UML use case diagram,
  - but in use case process model with activity diagrams.

- In addition, users can write scenarios (XP: User Stories) for the business use cases.

Checklist

- Does the use case describe WHAT the system should do, but not yet HOW?
- Is the use case formulated from the system's point of view?
- Is the level of abstraction correct?
- ...

Object-Oriented Analysis



Objective

- elaboration of the actual business purpose of the business use cases

Action

- Identify and describe the business intentions underlying each business use case.
- Differentiating between stable requirements and likely changing requirements (business purpose is likely to remain stable, technical implementation may be unstable, e.g. "reserving cars" will always be part of car rental, "reserving by fax" could become unnecessary with further expansion of the Internet.
- Define the pre-conditions and post-conditions for each business use case:
- Trigger, preconditions, incoming information, result, subsequent conditions, outgoing information.

Artifact

- Use Case Model

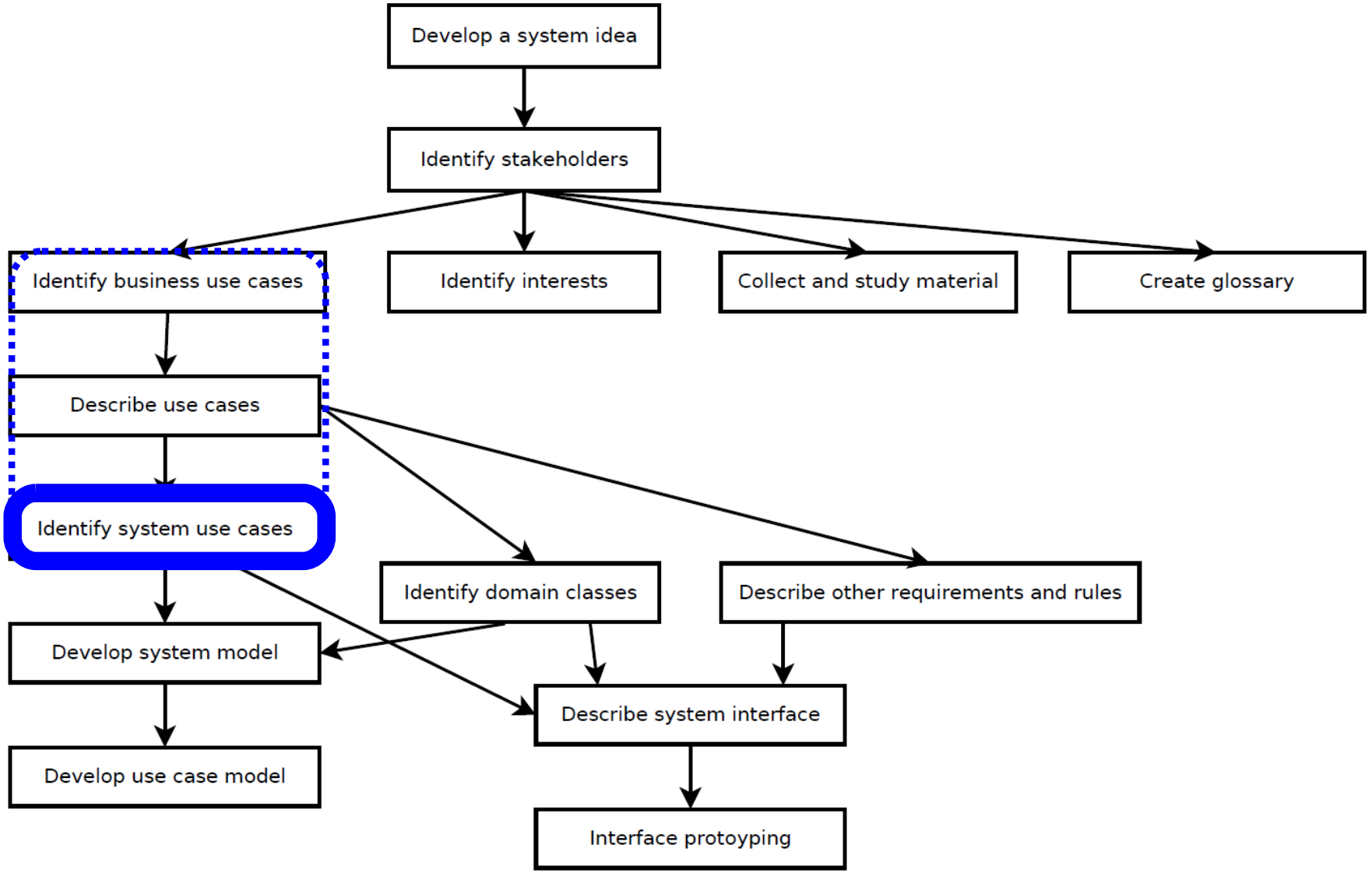
Object-Oriented Analysis
7. Describe use cases - rules

- At least one actor is involved in each use case.
- Use cases are generally formulated from the point of view of the business operator. The behaviour that is visible to the actor is described.
- A use case should always express a single, unambiguous fact.
  (→ decouple use cases, formulate several use cases).
- Each use case has a domain-oriented trigger and leads to a domain-oriented result.
- Consciously use singular and plural in the description of a use case.
  (Not "renting cars", but " rent a car", when usually only ONE car is rented)
- A use case should not contain any specific technological solutions.

Checklist

- Does each use case describe a single coherent business context?
- Does each step of the use case describe a coherent and unique issue with regard to trigger and result?

Object-Oriented Analysis



THU Technische Hochschule Ulm

Objective

- Concretising the existing application descriptions, taking into account concrete framework conditions, technical details and technical aspects.

Action

- Decide which business applications should be fully or partially supported by the system.
- Decomposition of the business use cases into time-consistent system use cases.
- Adding additional information (contact person, risk, importance, estimated effort, stability,…)

Artifact

- Use Case Model

- **Contributors to requirements**
  - Which contributors to the requirements require a specific use case?
- **Risk**
  - Which risks does this system use case induce (e.g. cost overrun, delay, technical problems,...)?
- **Relevance**
  - How important is the use case (indispensable, important, useful)?
  - What is the benefit of this application?
- **Effort**
  - How much effort is required to implement this use case (in PT)?
- **Stability**
  - What is the probability that the use case will change?
- **Time, urgency**
  - When at the latest must the use case be implemented (which iteration or which milestone)?
  - At what point in time should the use case be implemented (which iteration or which milestone)?

Checklist

- For each essential step of a use case: was it described how this can be defined in the concrete use case?
- Was each step described as concisely as possible but in such detail that it cannot be confused with others (in similar applications)?
- Were no synonymous descriptions used, but uniform terms and formulations?
- Were requirement contributors, existing risk, importance, estimated effort, expected stability and time/urgency named for each use case?

In Agile Software Development, so called User Stories will be used instead of Use Cases

User Stories

- are a short description of what your user will do when they use your software.
- are written in natural language

- Usual format:

  As an [actor] I want [action] so that [achievement].

"*As a traveller I want to be able to book a car by phone so that the car is reserved for me when I arrive at my destination.*"

"*As a customer, I want to update my customer profile so that future reservations are billed to a new credit card number*"

Acceptance criteria example
- Traveller must be registered in system
- Traveller must be able to identify himself
- Confirmation email is sent
- …

- **Acceptance criteria** required
  - which define the boundaries of a user story,
  - and are used to confirm when a story is completed and working as intended
  - written by the Product Owner (in Scrum)

In Agile Software Development, so called User Stories will be used instead of Use Cases

A good User Story should be:

- Independent (of all others),
- Negotiable (not a specific contract for futures),
- Valuable (It's all about the end-user. If you can't describe the value a customer is going to get out of your story, it's not a good story),
- Estimable (to a good approximation),
- Small (so as to fit within an iteration)
- Testable (in principle even if there isn't a test for it yet).

The three C's of the user story:

- Card
  stories are traditionally written on notecards, and these cards can be annotated with extra details
- Conversation
  details behind the story come out through conversations with the Product Owner
- Confirmation
  acceptance tests confirm the story is finished and working as intended.

In Agile Software Development, so called User Stories will be used instead of Use Cases

User Story and Use Case compared:

- User stories are about needs
- Use cases are about the behavior you'll build into the software to meet those needs.
- User stories are easy for users to read
- Use cases describe a complete interaction between the software and users (and possibly other systems)

Are User Stories the better concept?

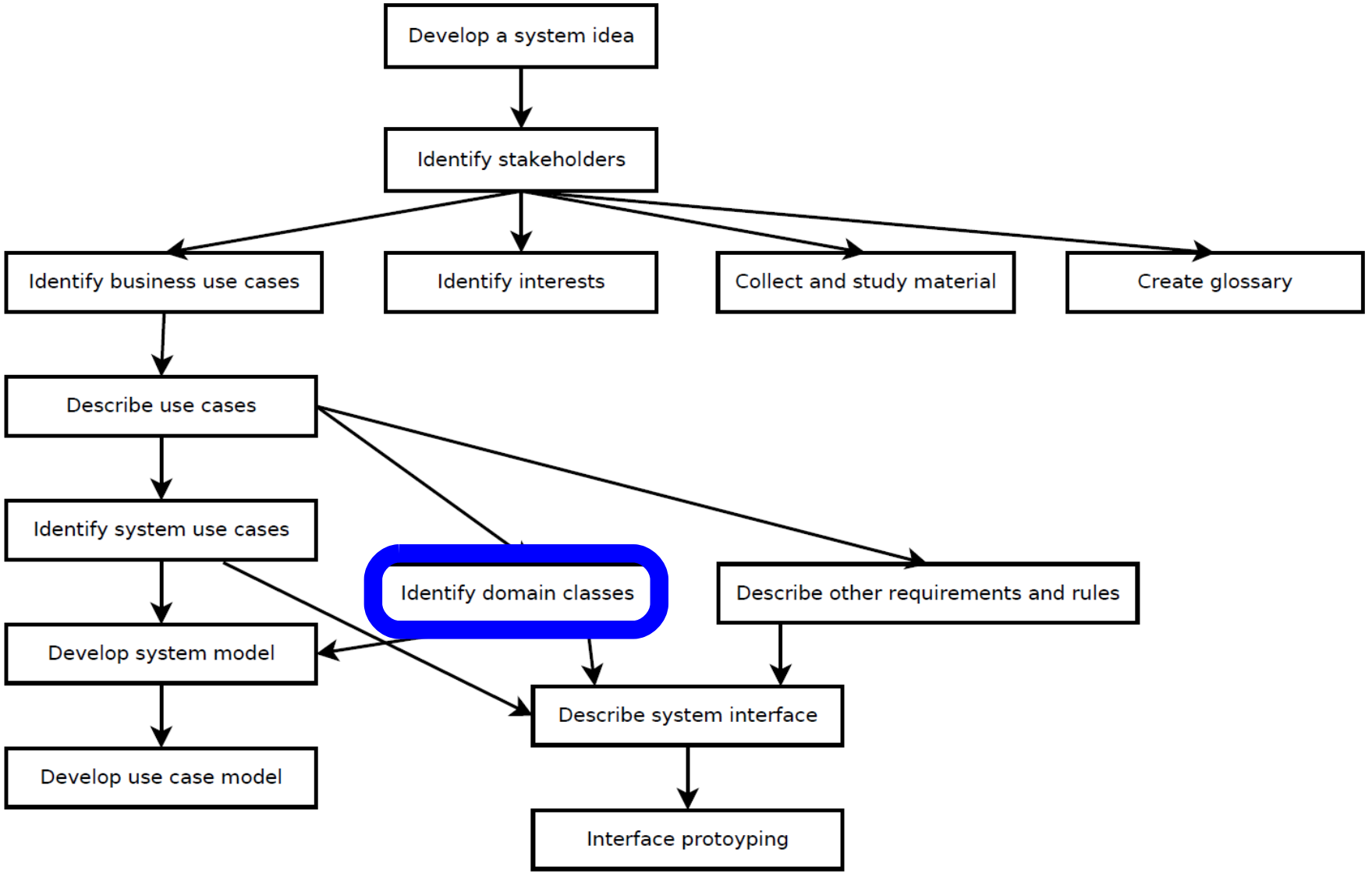
Obvious advantages of User Stories

- Simple concept
- Easy to understand
- Easy to communicate
- Easy to plan
- They seem to cause less effort to develop

Major drawbacks of using User Stories:

- Lack of context (what's the largest goal)
- Sense of completeness that you covered all bases relating to a goal.
- No mechanism for looking ahead at upcoming work
- What is the basis for regression tests?
  The description in user stories is insufficient to derive test cases.

Object-Oriented Analysis



THU Technische Hochschule Ulm

Objective

- Description of the fundamental, structural relationships.
- Conceptual representation of things from the domain
- (**NO** software objects!)

Action

- Identification of the most important technical subjects / concepts
- Model these objects as classes with their structural relationships in a UML class diagram.
- Assigning meaningful names to the classes, naming their associations and association roles and describing their cardinalities.

Artifact

- Domain model

A **domain class** describes
- an object,
- a thing,
- a concept,
- an idea,
- a place or
- a person

from the field of application (from the domain).

- Level of detail suitable for domain experts
- reduced to purely professionally motivated characteristics
- no software classes

Focus

- Relationships between classes
  - Association, Inheritance
  - roles
  - multiplicities

How to find the domain classes (conceptual classes)?

Three strategies:

- Reuse or modify existing models
  - from completed projects in similar domains
  - from books, e.g. Fowler: Analysis Patterns
- category list
- substantival expressions

Category list

- Processing a list of typical class categories can help to uncover overlooked concepts.

Category of conceptual classes	examples
commercial transaction	Sale, Payment, Reservation
transaction positions	SalesLineItem
Product or service associated with transaction	Item, Flight, Seat, Meal
Where is the transaction registered?	Register, Ledger, FlightManifest
Roles of people/organisations	Cashier, Customer, Store
Place of transaction, place of service	Store, Airport, Tarpaulin, Seat
Problem relevant events	sale, payment
Physical objects	Item, Register
Description of things	ProductDescriptionCatalogues

Identification of substantive expressions

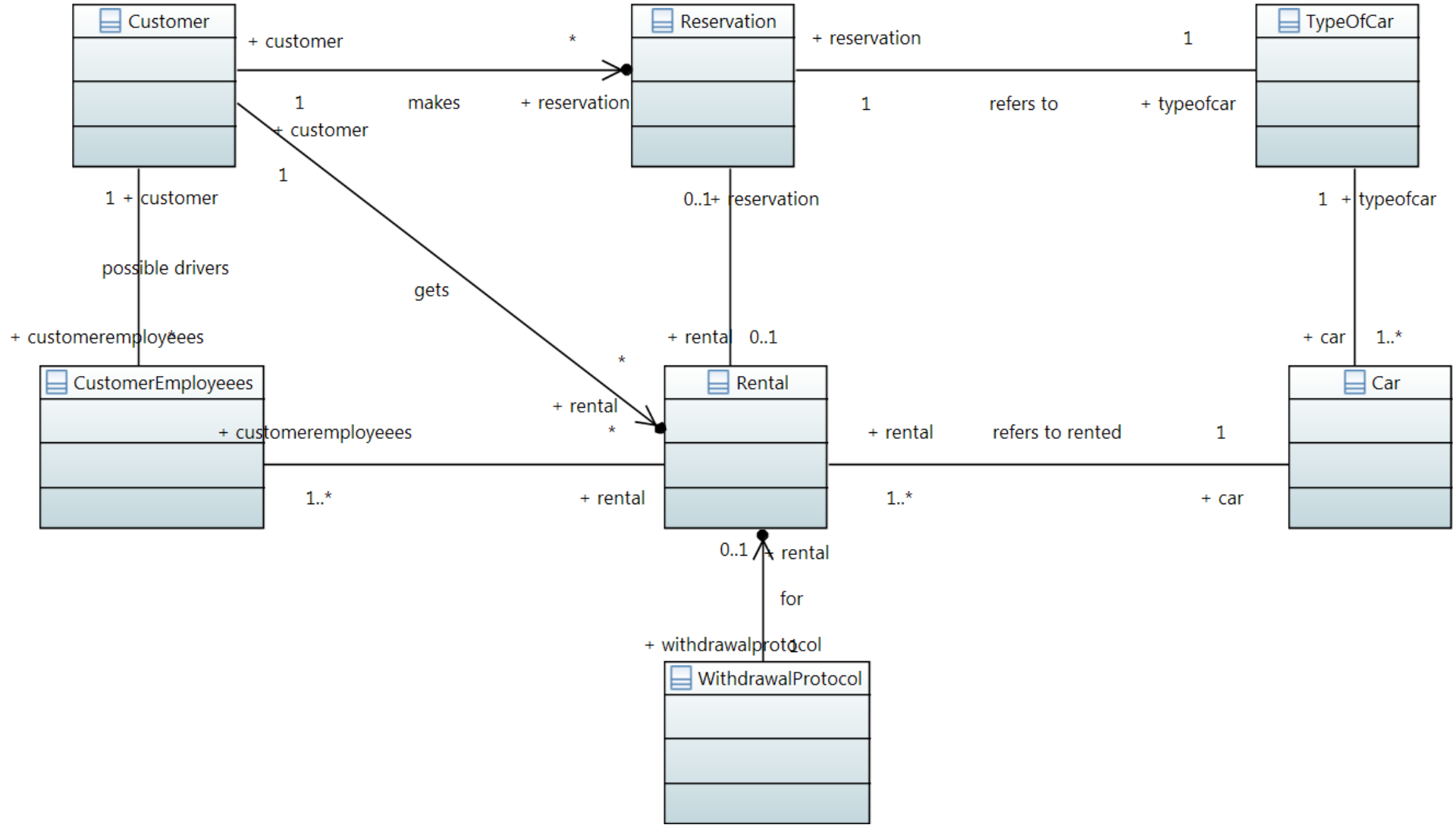
- from use case descriptions, vision document, etc.
- Example: Extract from the car rental vision document

> ...
> You can reserve cars (incl. change and cancel), rent (incl. contract creation and withdrawal protocol) and invoice the rentals (individual, collective and monthly invoices).
> Of course Fast-Cars 1.0 can also manage the vehicles to be hired, the existing rental stations and the customers with all important master data (addresses, bank details, etc.). Data can be transferred from the legacy system.
> ...

Watch the trap:

- Conceptual classes and simple attributes are often indistinguishable in the text.
- Synonyms can lead to class duplications/overlaps.

Object-Oriented Analysis
9. Identify domain classes – Example: Car Rental System

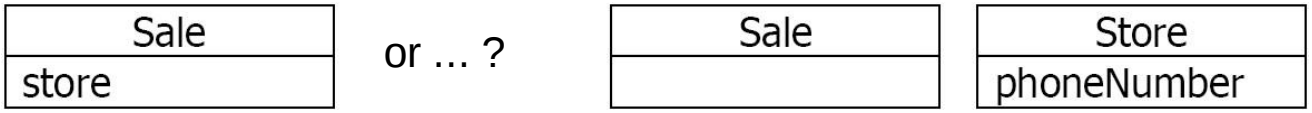


Object-Oriented Analysis
9. Identify domain classes - Remarks

- Domain classes are often further broken down during design.
- The cardinalities simply express the permissible technical proportions.
- Important: the model does not define everything!
- Example:
  - In a rental, a car can be referenced that belongs to a different type of car than it is specified in the reservation belonging to the rental!
  - If this is not desired, this must be defined by explicit requirements and/or the model must be extended by corresponding assertions.

Examples

- Excerpt from the modeling of a point-of-sale system
- Is shop an independent concept?



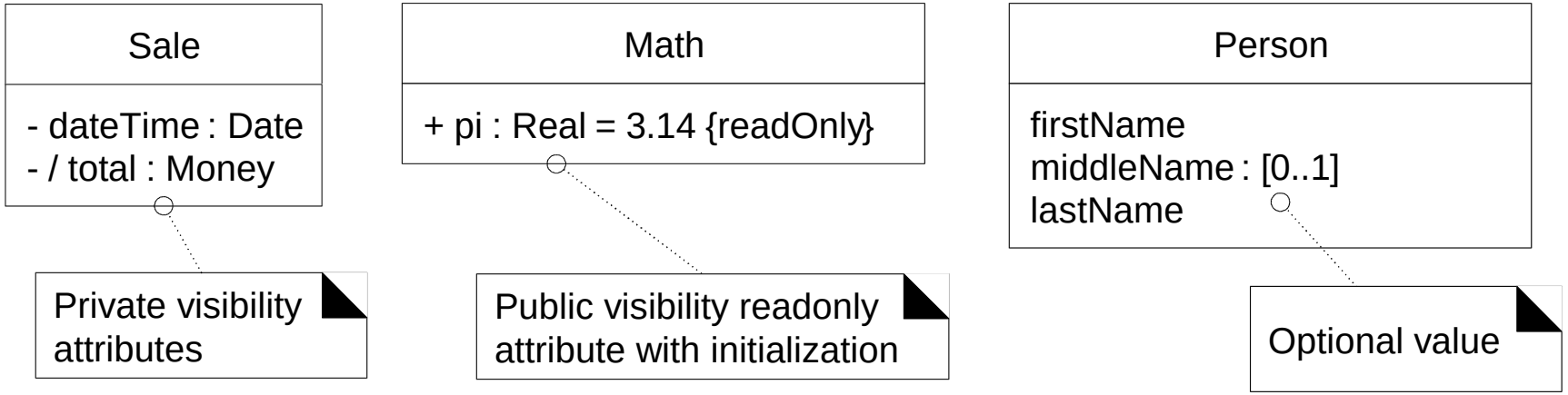
- Excerpt from the modeling of a flight booking system
- Is airport an independent concept?

If we do not see a conceptual class X in the domain as a number or text, X is probably a conceptual class and not an attribute.

General syntax of the attribute declaration:

Class name
visibility name : package::Type[multiplicity] = initial value {property = value}

Examples:



An association is a problem-relevant relationship between classes (more precisely: between instances of classes)

Guideline

- Associations (like attributes and classes) should only be included in the model if they are necessary to meet the information requirements of a relevant scenario.

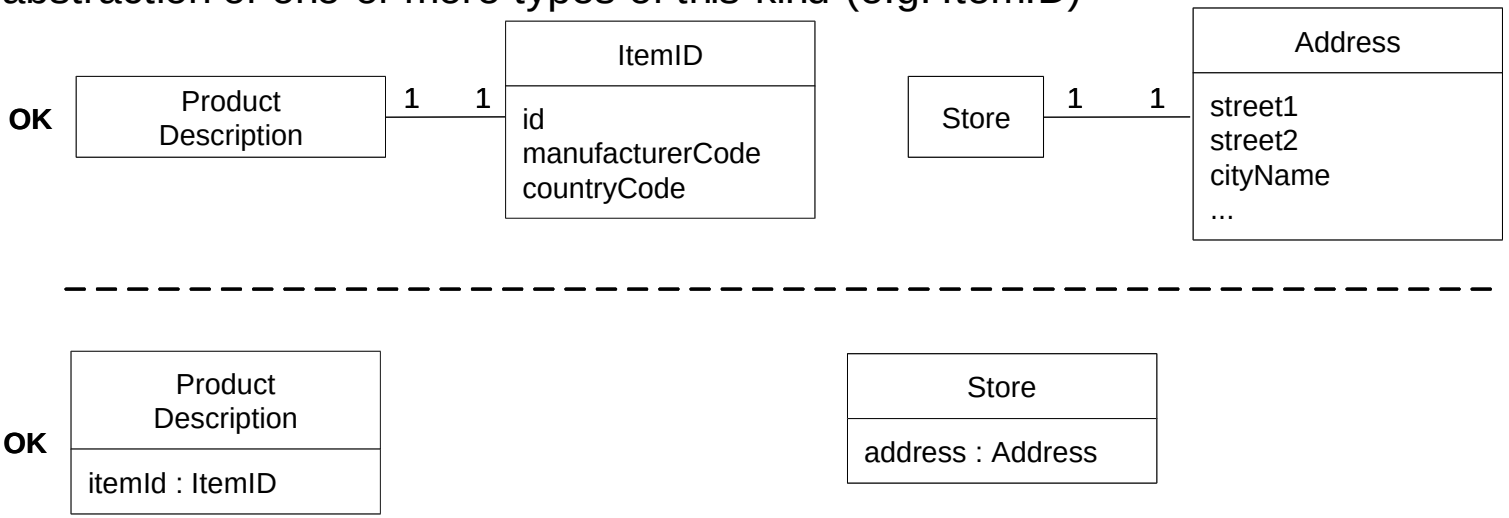
Example

- The relationships between customer - rental - car are necessary, otherwise no invoicing can take place.
- A relationship between vehicle type and manufacturing company would not contribute to the model.

When should new data types be introduced?

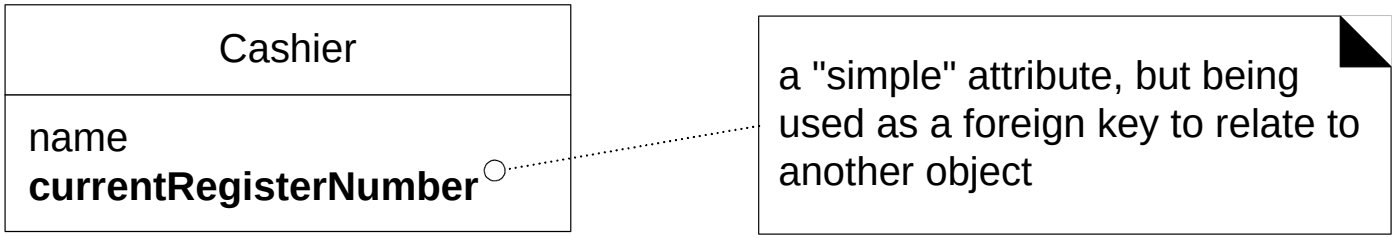
- A small set of standard types is sufficient for most attributes
  - such as Boolean, Integer, Double, String.

- New data types should be defined in the following situations:
  - If the attribute consists of several sections (for example, Address)
  - If special operations are required (e.g. Angle)
  - If it is a quantity with unit (e.g. Money, Weight)
  - If it is an abstraction of one or more types of this kind (e.g. ItemID)

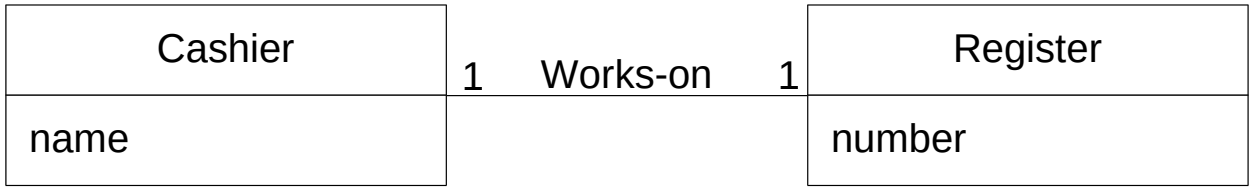


Do not model associations using attributes!

Worse



Better



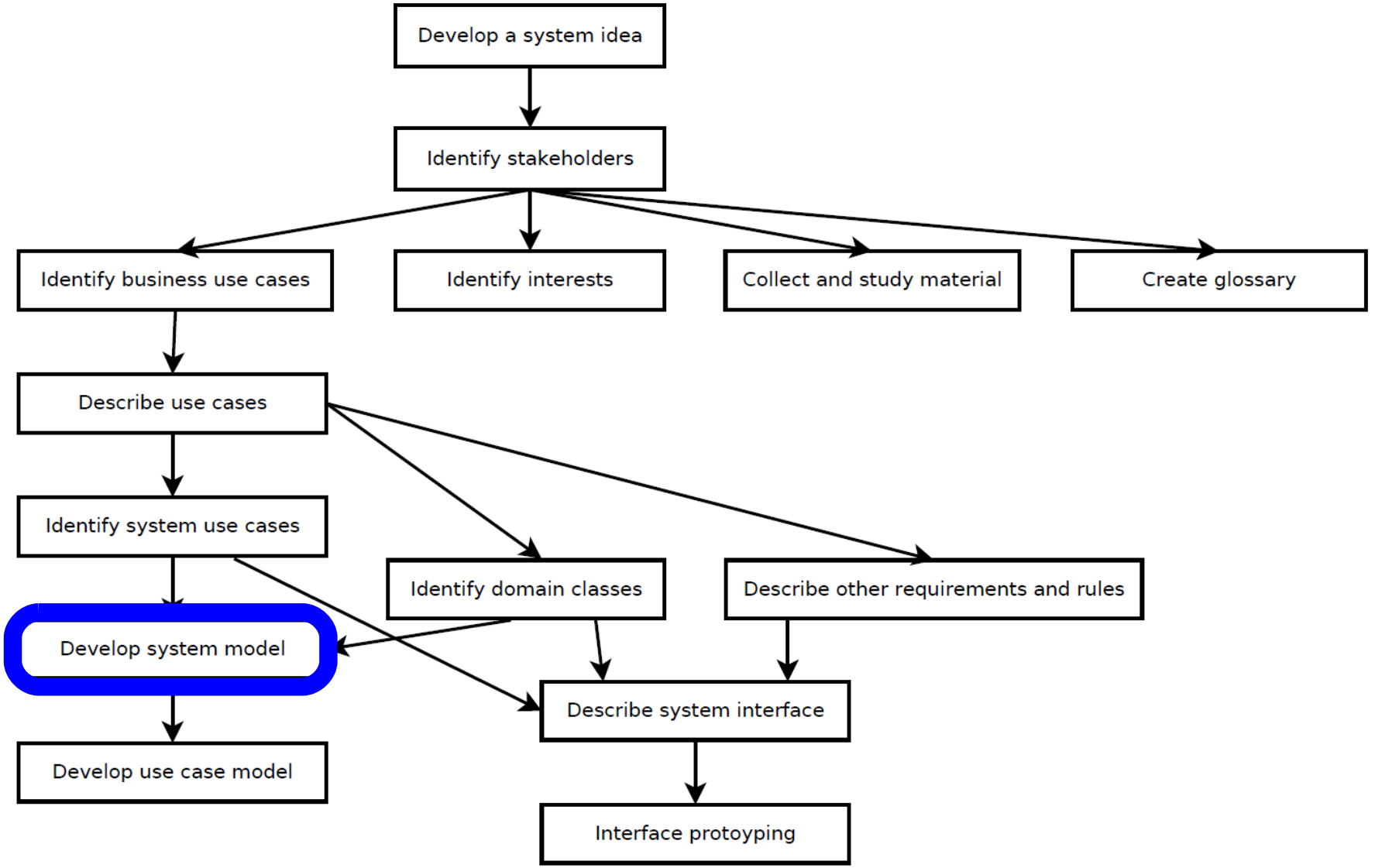
Object-Oriented Analysis
9. Identify domain classes – Tips

- Do not try to develop a comprehensive domain model in one step (→ feat of strength)!
- Let the domain model grow incrementally!
- Determine a precisely delimited scope in each increment (e.g. a use case)!
- Only include elements in the model that are relevant for the application to be developed!
- Talk to experts about domain models and take advantage of agile modeling!
- Prefer Whiteboard+digital camera to a CASE tool!

Checklist

- Does the business class have a meaningful name that is understandable to departments and decision-makers?
- Does each business class represent an object, a concept, a place or a person from the real domain?
- Does the class diagram only describe the structural relationships (inheritance, associations including role names and cardinalities), but no (only a few) further details?
- Do the cardinalities in the business class model represent the technically permissible proportions?
- Are the names of all business classes and all association roles defined in the glossary and are the glossary entries referenced?

Object-Oriented Analysis



Objective

- Detailed elaboration of the system use cases with special consideration of dynamic aspects.

Action

- Split each use case step into one or more elementary activities and model the flow of each use case with an activity diagram (standard flow).
- Model all planned exceptions and branching options for each activity (complete process flow).
- For each activity, model the necessary incoming objects and data as well as all results (objects, object states).

Artifact

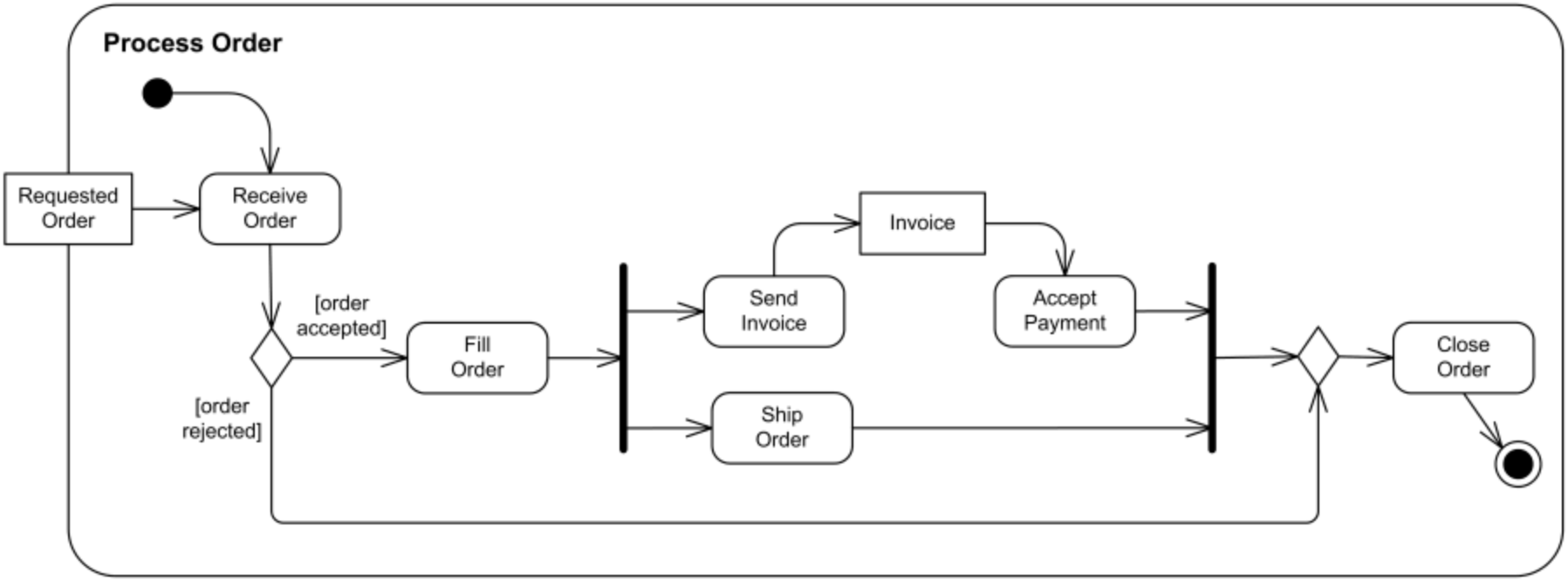
- Analysis model

UML acivity diagram:

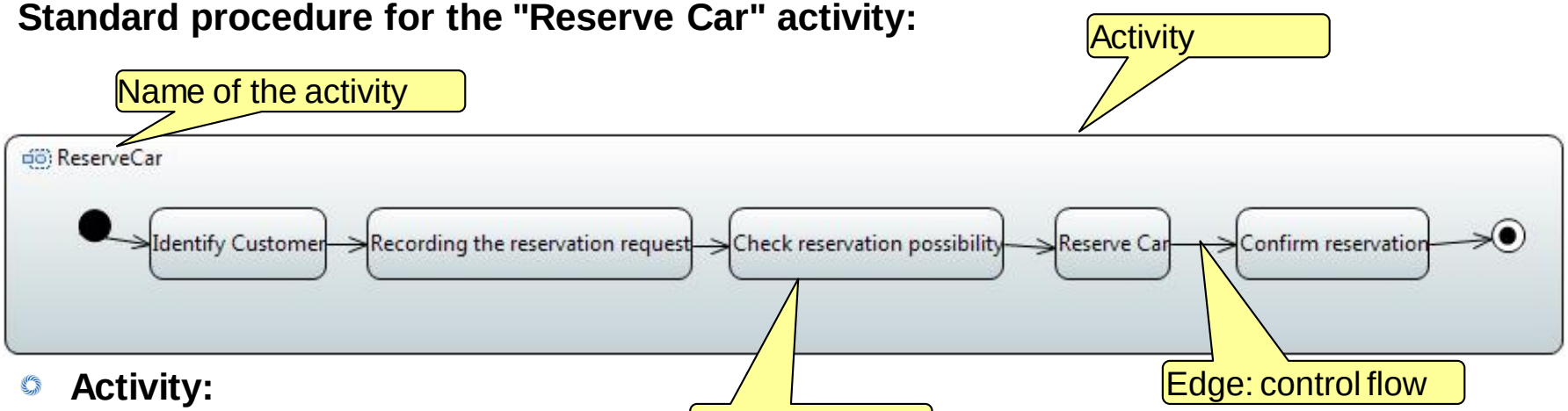
- Central question:
  - How does a certain process work?
  - How does the system implement a certain behavior?

- An activity diagram describes processes in the system including concurrent activities or alternative decision paths

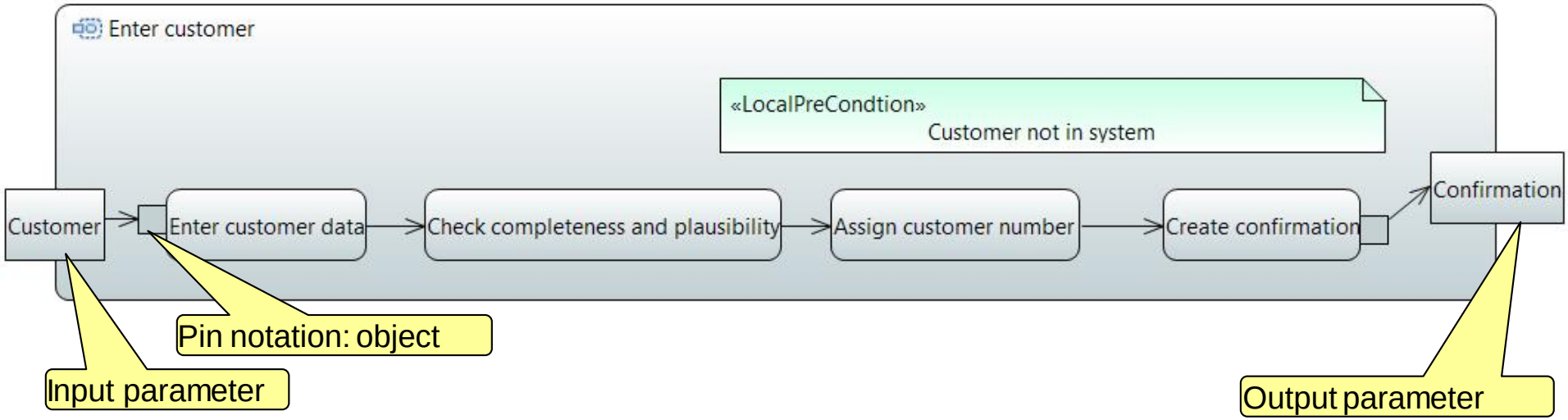
10. Develop system process model – Example: „Process order“

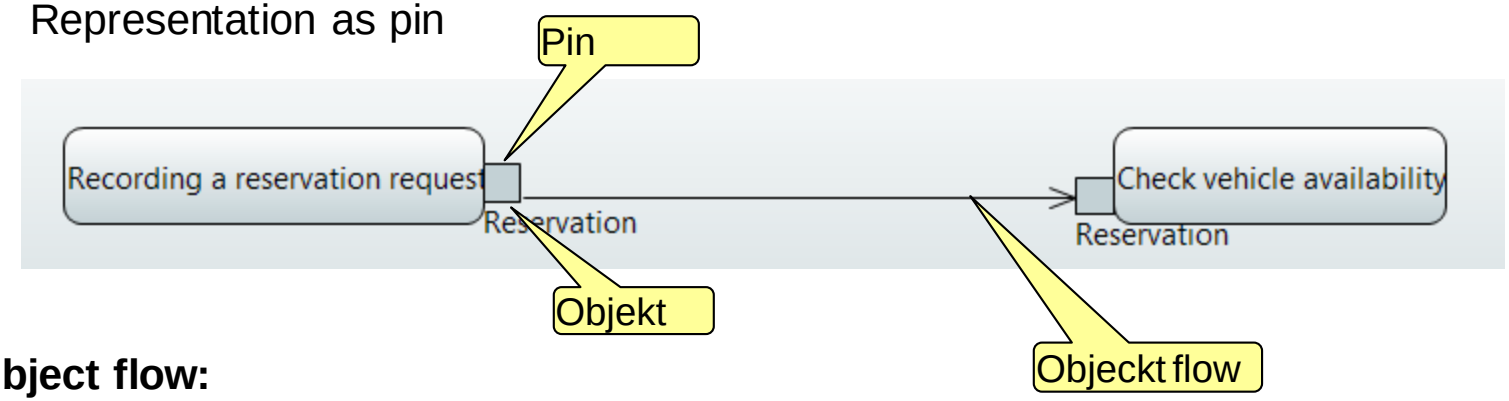


Standard procedure for the "Reserve Car" activity:



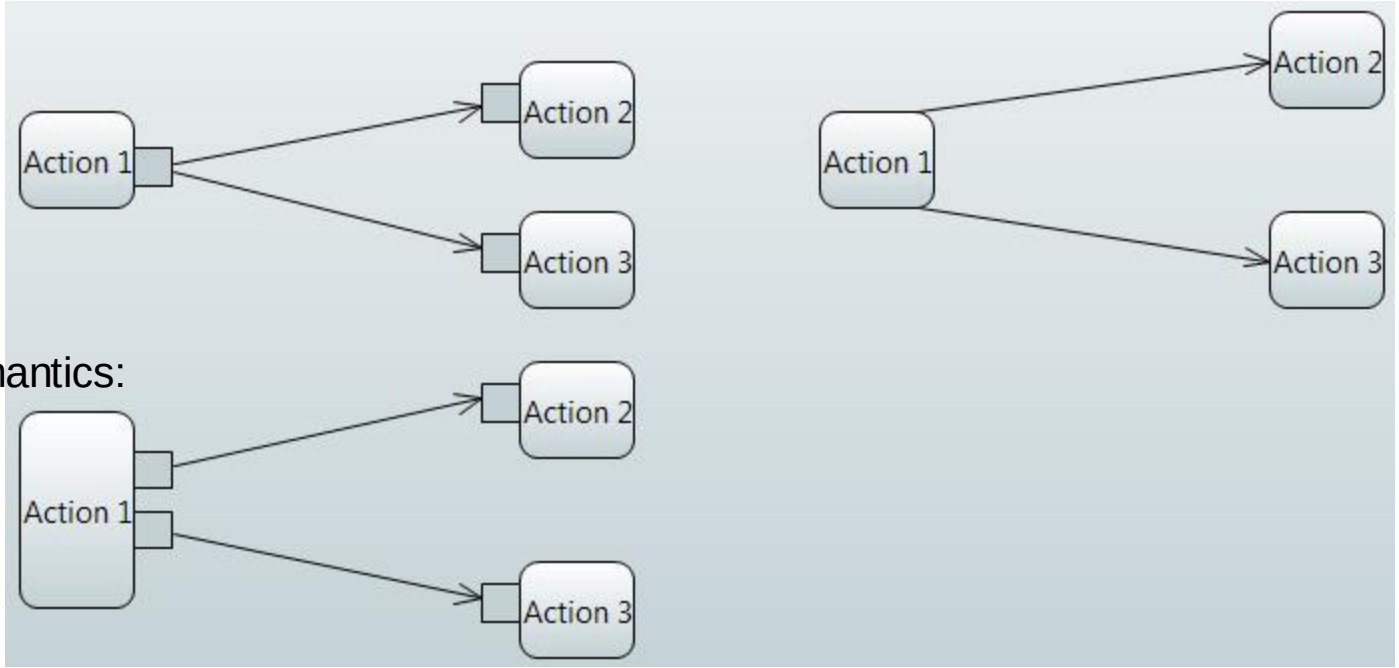
- **Activity:**
  - arranges actions in a network consisting of nodes and edges.
  - Illustration by large rectangle with rounded corners, name top left
  - Possible parameters as object nodes at activity border
- **Action:**
  - is an elementary behavioral component
  - Representation by a rectangle (rounded corners) with name or text for the operation
- **Edges** (control flow and object flow):
  - Display with arrow, optionally with name
  - Actual directional transition between nodes


Enter customer
«LocalPreCondtion»
Customer not in system
Customer
Enter customer data
Check completeness and plausibility
Assign customer number
Create confirmation
Confirmation
Pin notation: object
Input parameter
Output parameter

- **Object nodes:**
  - Representation as pin
  - 

- **Object flow:**
  - Transports objects, so-called object **tokens**
- **Token:**
  - defines the processing position of the flow.
  - token movement through activity means "processing is in progress"
  - several tokens can occur in a processing run
  - Token are not displayed graphically in the activity diagram

- Object flow
  - OR-Semantics:
  - AND-Semantics:

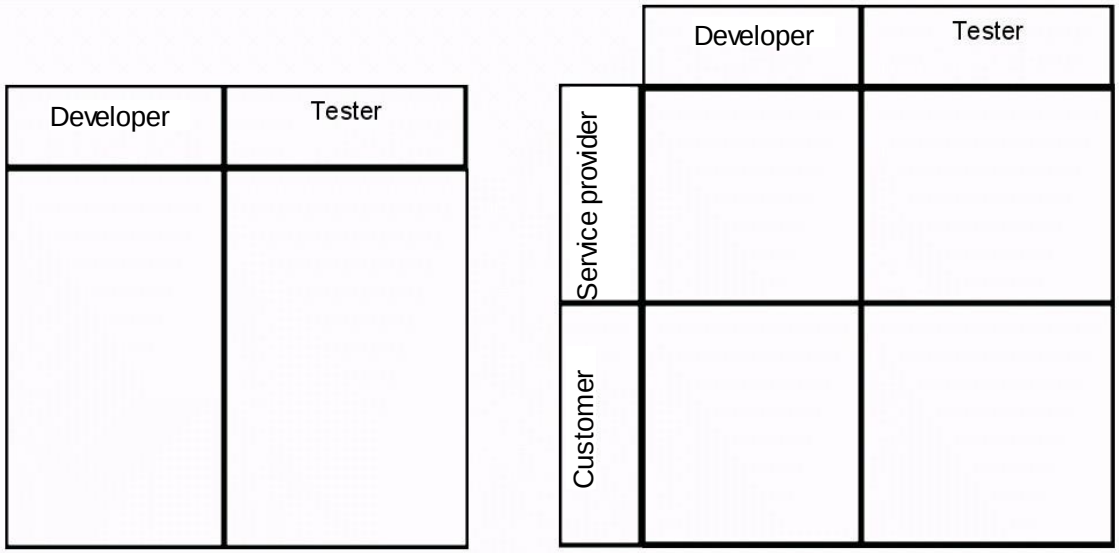


- Control flow
  - AND- Semantics:

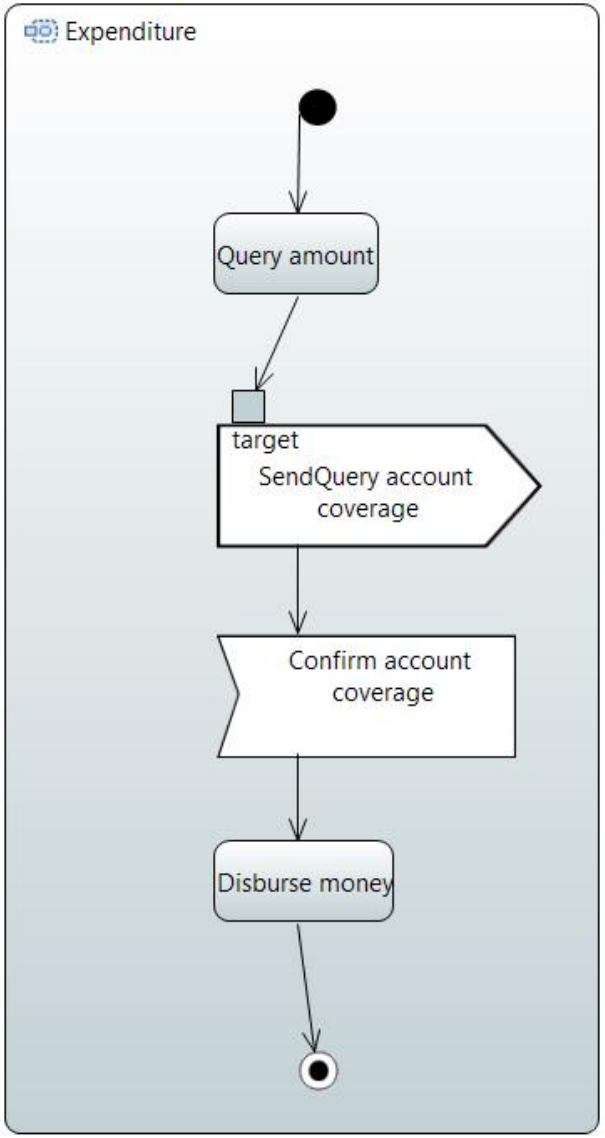
Controls:

- **start node**
  - Starting point of a process in an activity diagram
- **end nodes**
  - ends the described process flow, i.e. all actions and control flows.
- **End of flow**
  - Ends a single control flow
- **Splitting**
  - A step in the process at which an incoming control flow is immediately split up into several outgoing parallel control flows.
- **Synchronization**
  - Control node that waits for all incoming control flows before continuing the control flow.
- **Decision / Branching**
  - Control node with several outbound control flows, at which decisions are made on the basis of conditions that are continued by several inbound control flows.
- **Merging**
  - Control node where each of several control flows immediately leads to a common outbound control footprint.

- **Partition** (swimlane, area of responsibility, properties area)
  - Within an activity model, a partition describes who or what is responsible for certain nodes or which property identifies them.
  - Graphical representation by drawing in lanes.
  - (Alternative: Write partition names in parentheses directly to each node)
  - Nodes must always be exactly within one lane.
  - Multidimensional partitions are possible (since UML 2.0)



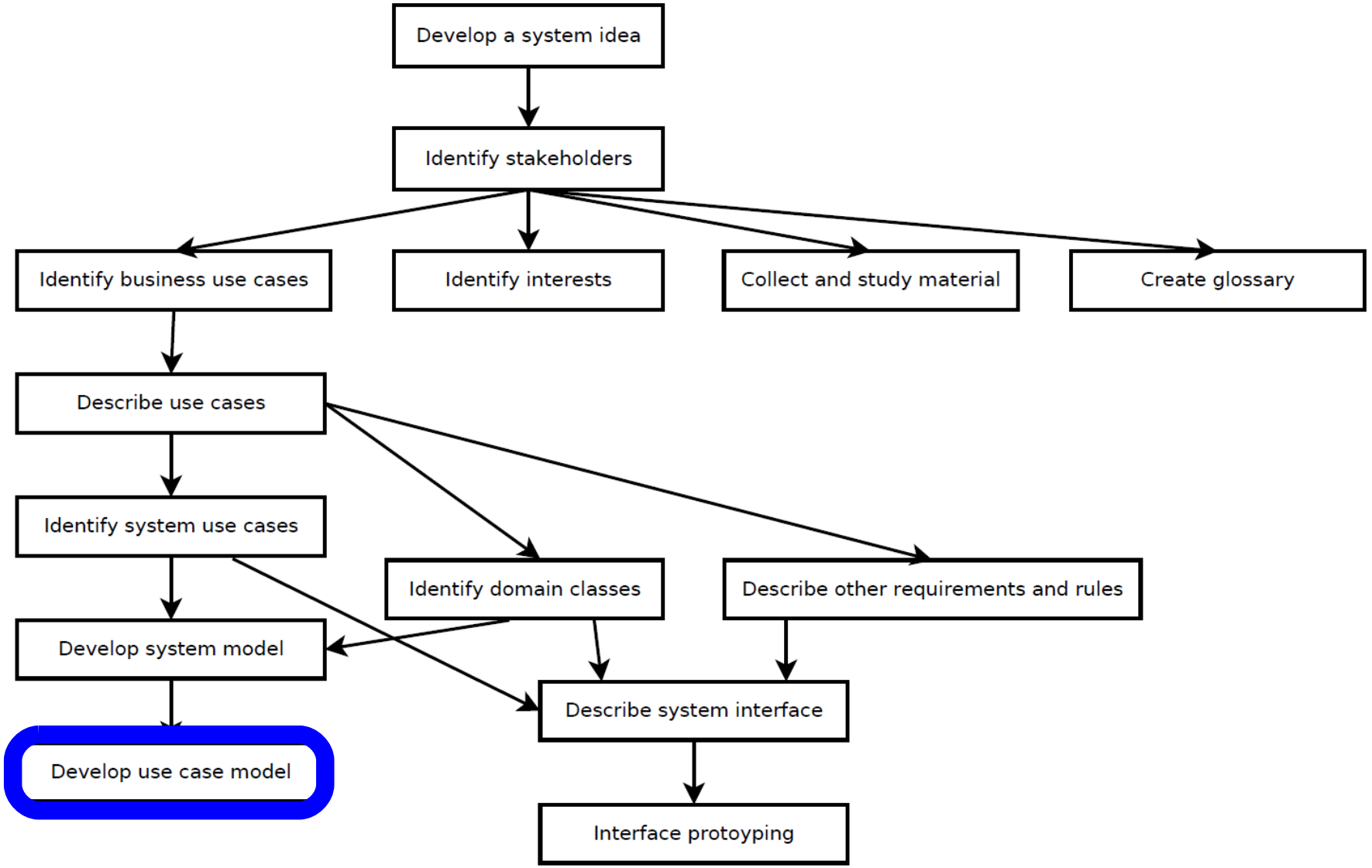
- **Signals**
  - The signal transmitter creates a signal from its input data, which is sent to an event receiver. Once the signal has been sent, this action is considered completed and the activity control flow continues.
  - The counterpart to the signal transmitter is the event receiver. If the control flow reaches an event receiver, the token remains in the action until the expected event occurs. Processing is then continued.



Checklist

- Does an activity diagram exist for each use case?
- Is each use case step represented by at least one activity in the activity diagram?
- Does each diagram have a start state and one or more end states?
- Does each end state have a meaningful name?
- Are all known exceptions and variants noted in the activity diagram?
- Are all outgoing transitions numbered consecutively?
- Are all functional and process-specific exceptions considered, but no technical or general exceptions?
- Are all termination options considered?
- Are the incoming and outgoing object/object statuses noted for each activity?
- Are the corresponding transition numbers noted for all object flows?
- Do the object states used match those in the corresponding state diagram?

Object-Oriented Analysis



Objective

- Create a consistent, redundancy-free model.

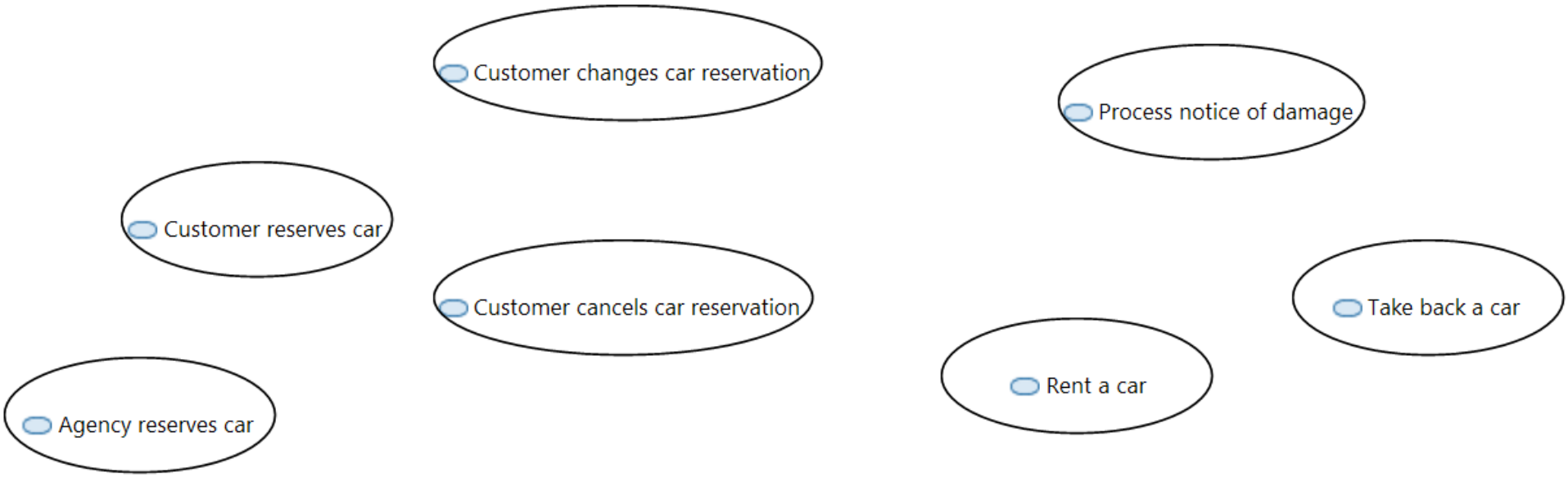
Action

- Representation of all existing system use cases in a use case model.
- Identify identical steps in different system use cases.
- Detaching these steps and transforming them into system use cases.

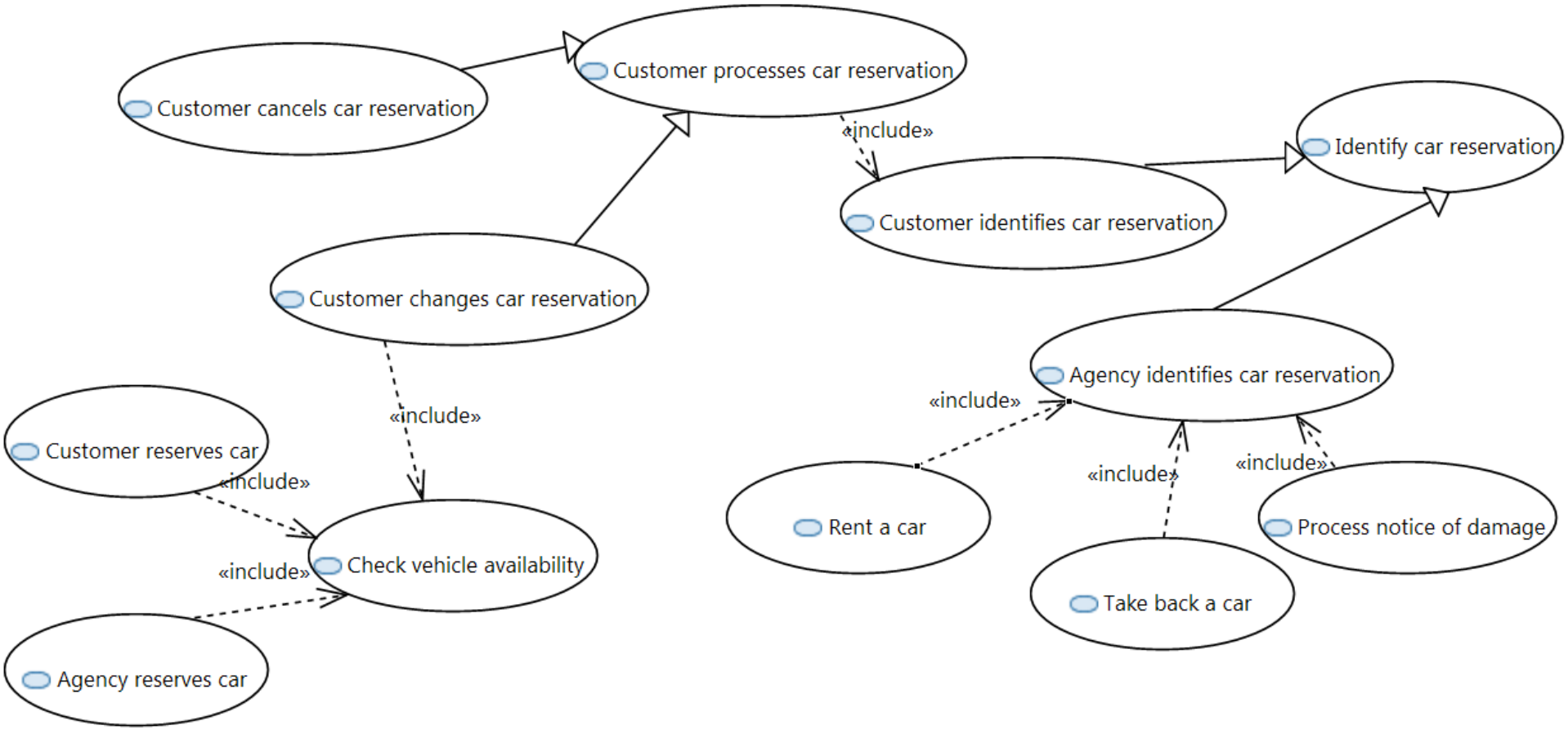
Artifact

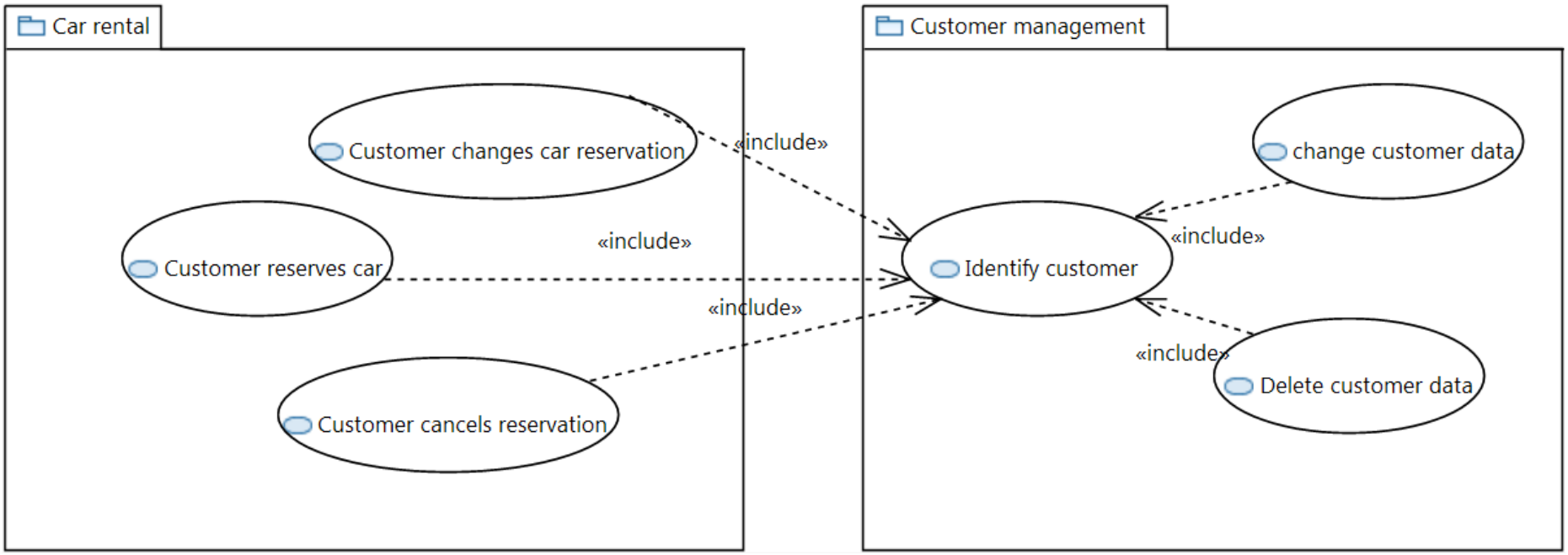
- Analysis model

- Group the use cases in the same way as the business processes or according to functional relationship.
- e.g. car rental:

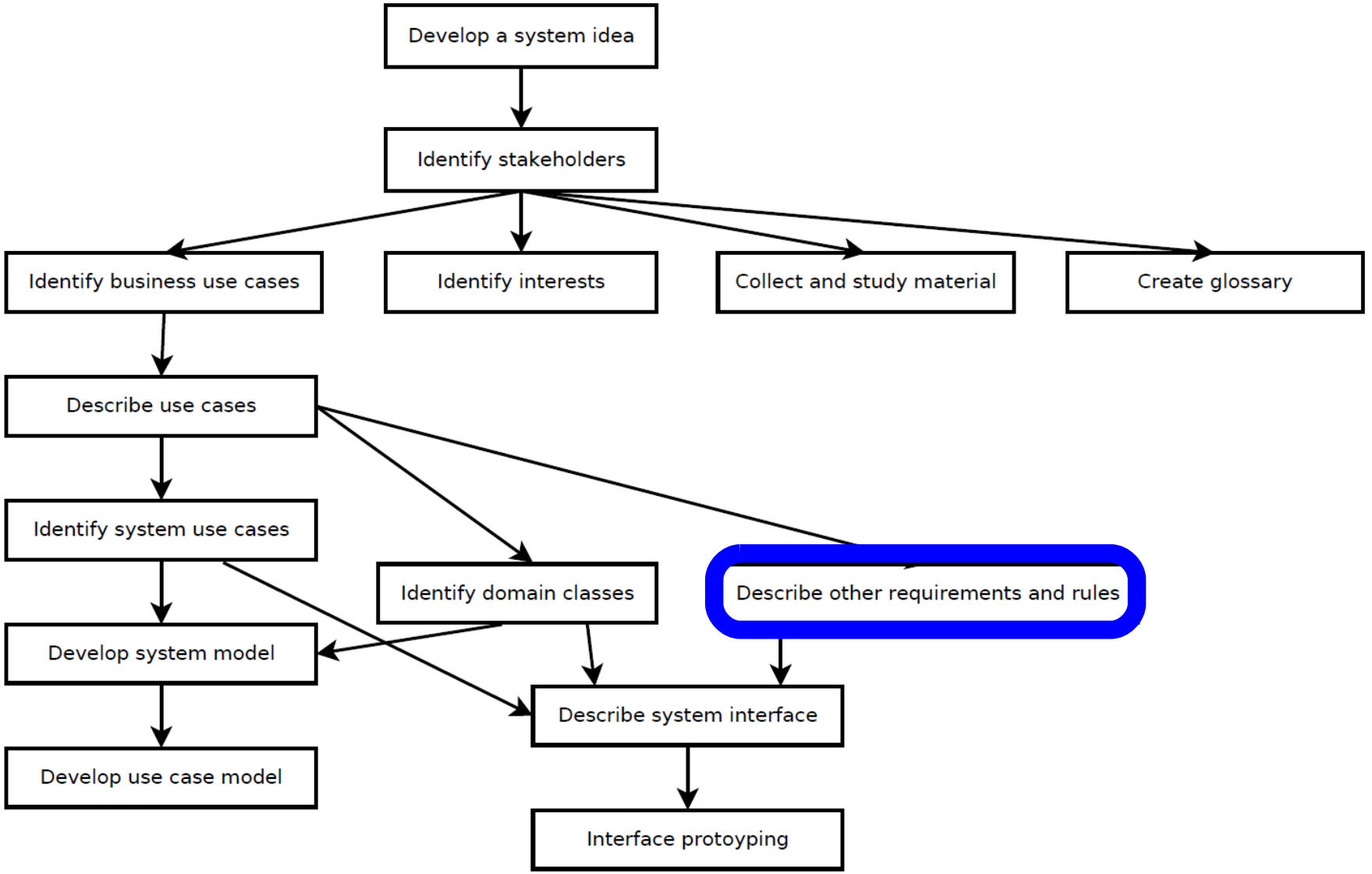


11. Develop system use case model - example




Car rental
Customer changes car reservation
Customer reserves car
Customer cancels reservation
«include»
«include»
«include»
Customer management
change customer data
Identify customer
Delete customer data
«include»
«include»

Object-Oriented Analysis



THU Technische Hochschule Ulm

Objective

- Redundancy-free acquisition of
  - cross-application or cross-business project requirements as well as
  - non-functional requirements
- in a central location with corresponding referencing.

Action

- Describe all requirements and rules outside of use cases in a central requirements catalog.
- Detail these requirements according to a defined description scheme.
- Referencing the requirements in the respective business objects or use cases.
- Registration of existing business concepts and, if necessary, transfer to formal requirements

Artifact

- Supplementary Specification

Why are additional requirement documents required?

- Not all functional requirements are suitable for description in the form of use cases.
  - Especially for platform and middleware products other forms of description are needed, e.g.
    - Features ("Support for EJB EntityBeans1.0")
- Non-functional requirements can often, but not always, be assigned to specific use cases.

Supplementary Specification

- The Supplementary Specification is the artifact of the UP.
- The Software Requirements Specification (SRS) is made up of
  - Use case model (use cases + actor descriptions + diagrams)
  - supplementary specification
- The Supplementary Specification contains all FURPS requirements that are not covered by the use cases.

12. Describe other requirements and rules - Example

Requirements Id	#0815
Name	Minimum rental period and minimum time unit
Type	functional requirement
Description	The minimum rental period is 6 hours. Other rental periods may apply for special types of vehicles. The rental period is always full hours.
Stability	instable
Priority	low
Details	results from current valid tariffs. Contact person is: …
Affected model elements	- Reserve car - Change car reservation

Further examples:

- Minimum age of a driver
- Maximum reservation lead time
- cancellation rules
- Allowed time overruns for car reservations

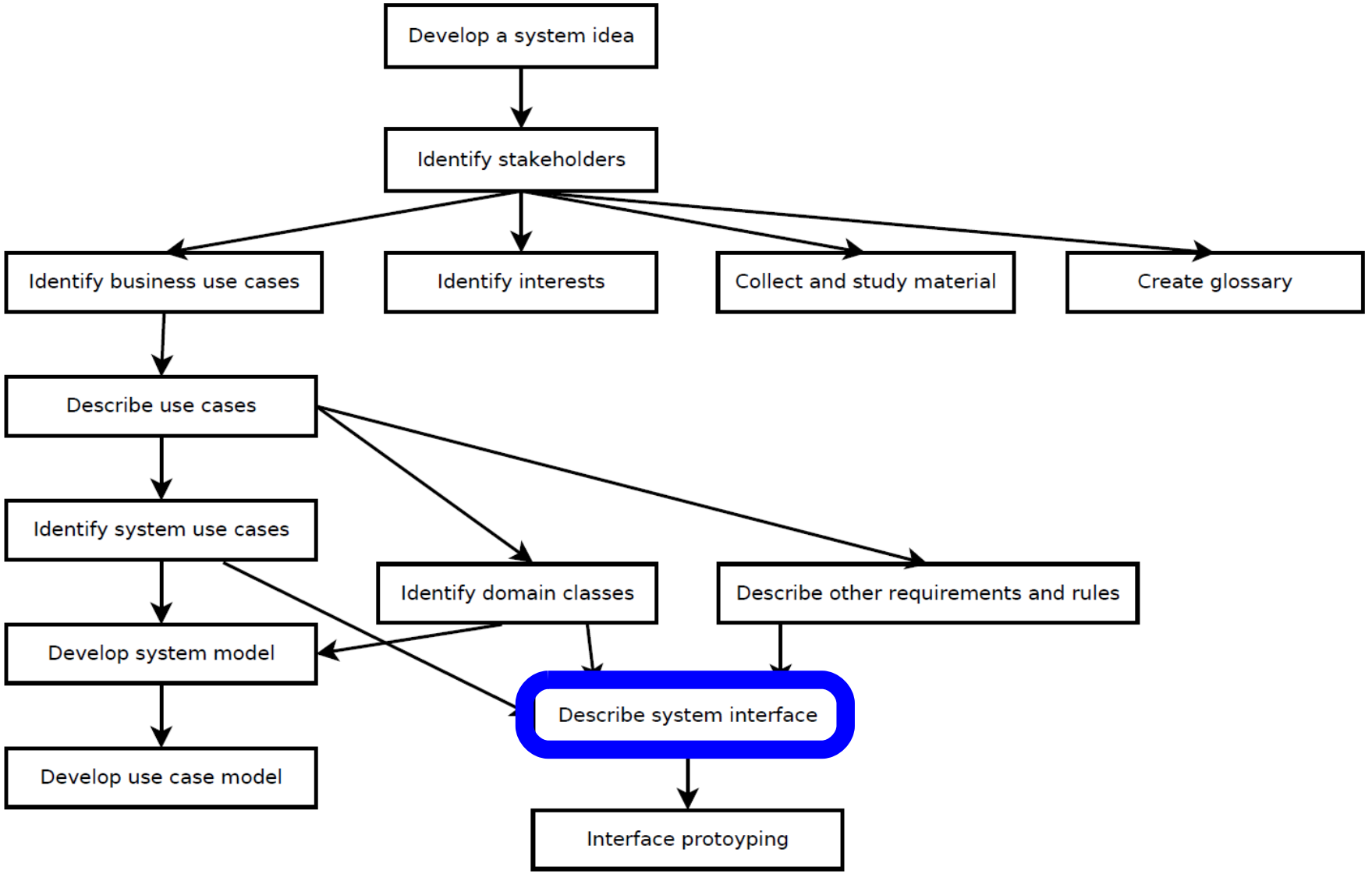
Feature Driven Development (FDD):
→ iterative, incremental software development process, central element: **Feature**

- In FDD, features are small functions with customer value.
- The naming follows the schema
  - <action><result><object>
- Examples:
  - Calculate the total of a sale
  - Assess the performance of a salesman
  - Validate the password of a user.
  - The maximum implementation time for a feature is 2 weeks

Checklist

- Are requirements described independently of use cases?
- Can each individual request be identified by a unique request number?
- Is the obligation, desire, intention, proposal, comment fixed for each requirement?
- Has each requirement formulation been verified with language consolidation tools?

Object-Oriented Analysis



THU Technische Hochschule Ulm

Objective

- Identification and description of all interactions, inputs and outputs of the system to be developed with its environment.

Action

Determine the interface for each use case, that is, summarize the incoming and outgoing data, objects and events in the form of interface descriptions.

so far:

- sequences described with use cases, flow and object flow diagrams.

now:

- Identify structural elements that are located at the outer system boundary
- All interactions, inputs and outputs of the system to be developed are identified and described.
- The interfaces involved can be assigned to each use case step.
- The focus here is on finding the interface elements involved as opposed to the exact assignment to the steps.

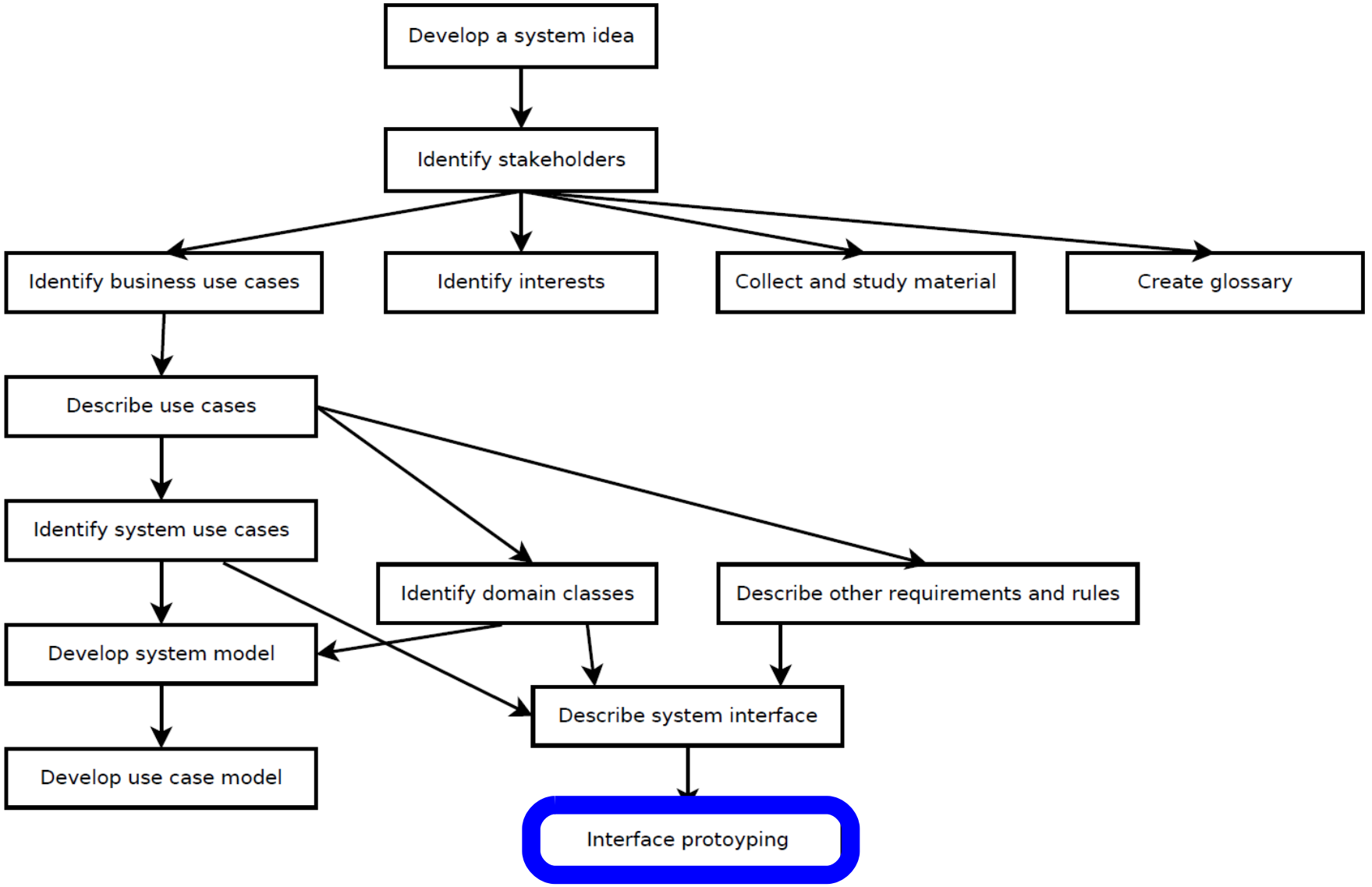
use case step	interface elements
Identify customer Enter customer no. Identify customer	 Customer search dialog
Recording a reservation request	Car Reservation Dialog
Check vehicle availability Determine available vehicles	
Reserve a car Select a car Reserve a car	 Dialog Car Reservation Dialog Car Reservation
Confirm Reservation	Dialog Car Reservation Booking confirmation

Description of dialog	
Name	Search customer
Short description	Search terms can be used to search for an individual customer.
Usage	All users in the branches and in the call center, 40 times daily on average per user
Complexity	simple
Input fields	Keywords: customer number, name, phone
Display fields	Customer data (name, address)
Branching possibilities	Customer file, Cancel, Car reservation
Actions	Start search, select a customer, delete all search terms

Checklist

- Is there a list / assignment of the interfaces involved for each use case?
- Does a short description and complexity rating exist for each interface?
- Is the type of interface described (dialog, output product, functional, data exchange)?
- Are all interfaces described in more detail according to an interface-type-specific schema?

Object-Oriented Analysis



Objective

- Ensure that the content of the dialogs is correct and complete

Action

- Development of explorative prototypes for the identified interface elements.

- Explorative prototypes
  - are executable and usable dialogs in which data can be entered but which have no other functions (help function, error handling, storage in database, performance, robustness, undo function,...).
  - illustrate the use cases and system interfaces
  - are used in the analysis of communication with users.

The objective is to validate the contents of the dialogs.

- The concrete layout is not yet so important
- However, references to this can already be recorded.

- All situations described in use cases can be simulated with the end users dialogue by dialogue

- The user's comments are often aimed to certain attributes (data fields) that are missing or certain courses of action that are missing

- All questions, objections, suggestions,
  - ... which cannot be clarified immediately are recorded and evaluated.
  - should serve as a benchmark for new designs at subsequent meetings

- Dialogue drafts could be
  - simply drawn on paper or
  - can be created with GUI-Builder with relatively little effort (preferable)

Checklist

- Was an exploratory prototype created for at least all interface elements of the complexity classes important and standard?
- Do the exploratory prototypes contain all elements mentioned in the essential use case description?
- Has each exploratory prototype been discussed and agreed with the specialist department in prototyping workshops?
- Is there an agenda and a protocol for each prototyping workshop?

Summary

You should be able to answer these questions:

- Which types of requirements are distinguished?
- Which steps are carried out in the object-oriented analysis according to OEP?
- Which notation elements does UML provide for the representation of use case diagrams?
- What are the five most important attributes that characterize a business use case (natural language)?
- What elements does UML provide to describe activities?
- What is the semantics behind the individual elements?