Use Cases and Scenarios

Concept and diagram

Include and Extend

Scenarios

Templates

Anti-patterns

Agenda

- Use Cases (in UML2)
- Use Cases and Scenarios
- Use Cases and Relationships
- Use Case Diagrams
- More on Scenarios: Templates and Hints

Bibliography

- Silva&Videira

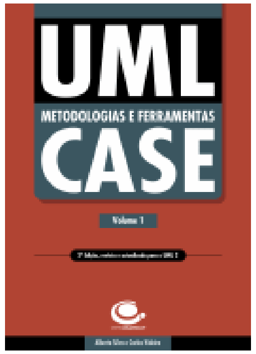
  (UML Use Case Diagrams, Chapter 5)

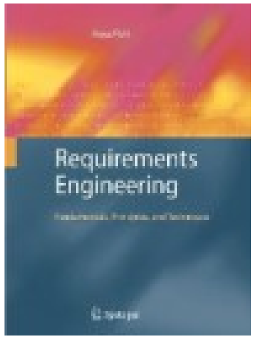
- Pohl

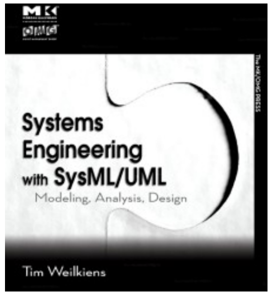
  (Scenarios, Chapters 10 and 11)

- Weilkiens

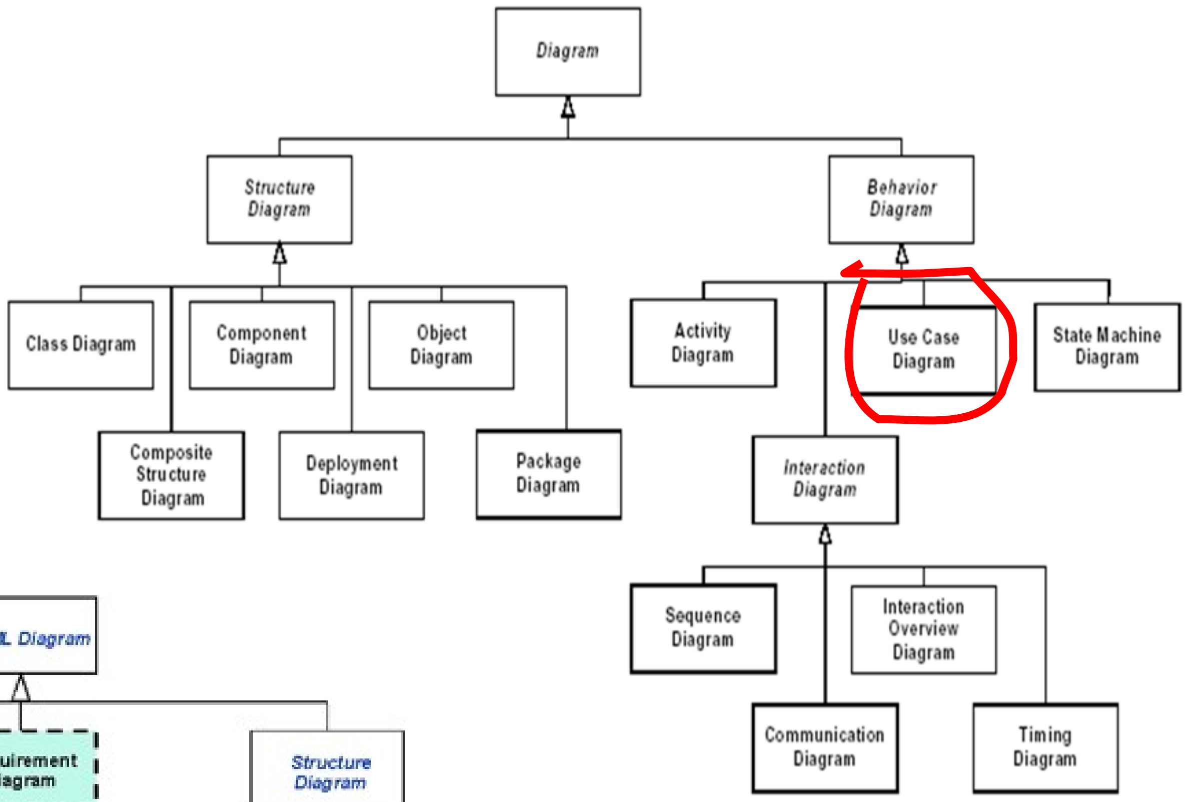
  (Chapters 2 and 3.5)

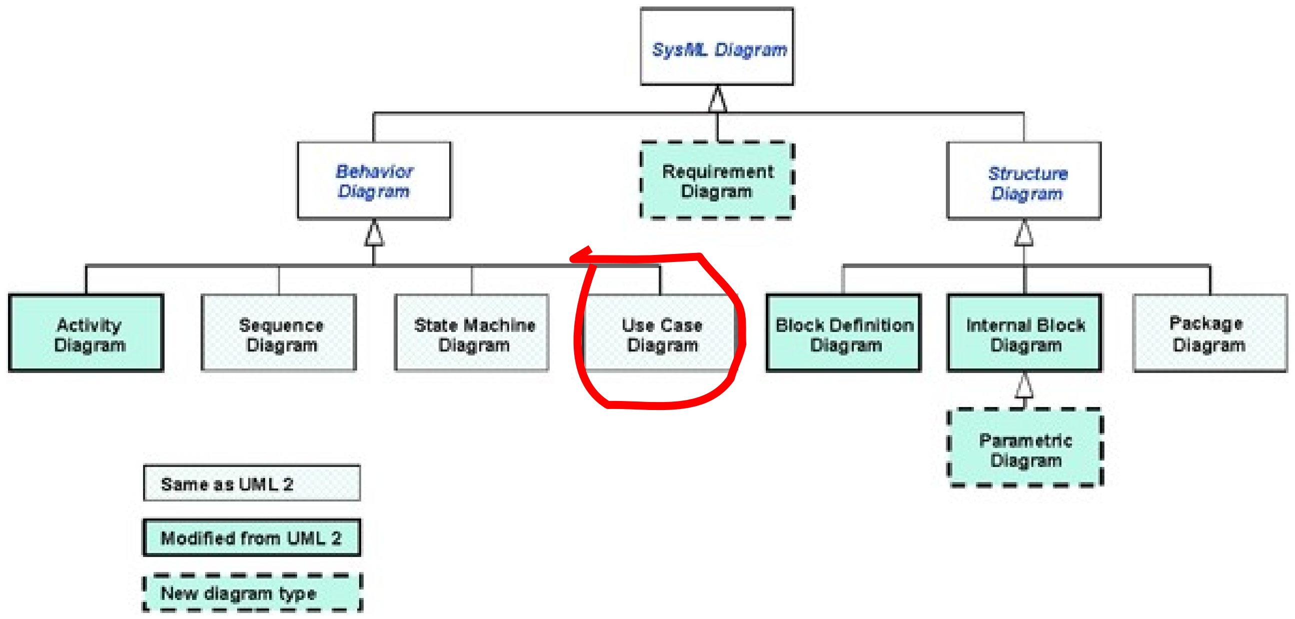






Use Case diagrams in UML and SysML





Use Cases

- **Use cases describe an interaction between the system and an external entity (i.e. an actor).**

- The interaction is always from the **point of view of the actor**.
- Therefore it describes “who can do what” or “who is affected by what”.
- The use case technique is used to elicit **functional requirements** by relating them with usage and behavioral scenarios.
- Use cases are also useful to trace the requirements to the models detailing the system behavior.

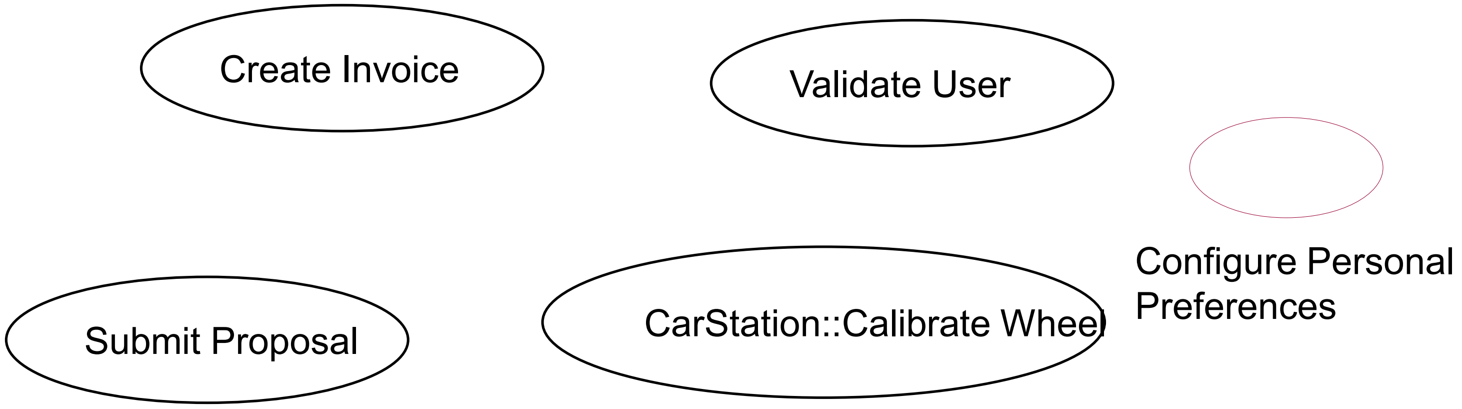
Use Cases

- A **use case** represents a set of **actions** that one or more **actors** request from a **system** in order to obtain a tangible result.
- A use case represents an external view on the system.
  - Describes **what** the system (or part of it) does.
  - Does not describe **how** that behaviour is accomplished.
  - Provides a **black box** (functional) perspective.

Use Cases

Use case is a sequence of actions that one or more actors perform in a system to obtain a particular result.

- It should be represented by a sentence, with a verb in the direct and present tense mode.
- E.g., “Generate reports”, “Create invoice”, “Calibrate wheel”



Use Cases and Scenarios

- Use case describes what a system is (or part of this), but not how this is accomplished – BlackBox model
  - The focus is on the external description of the system's functionality
- Use case can be detailed by a collection of scenarios

- **Scenario** is described by a sequence of actions that partially detail the system's functionality.
  - In abstract, a scenario can be understood as an use case's instance;
  - it is common an use case to be described by different scenarios

Use Cases and Scenarios

- The detail of an use case can be textually specified describing the events flow, so that a nontechnical user can understand it.

- Such specification must include:
  - Assumptions
  - Pre-conditions
  - Initiation: how and when the use case starts
  - Dialogue: when a use case interacts with the system actors
  - Conclusion: how and when the use case finishes
  - Post-conditions

- Other forms:
  - Actors that initiate and benefit…
  - Sequence or Activity Diagrams...

Use Case – ATM Example

Use Case textual description

Validate User

Main Scenario

The use case is initiated when the system presents a screen where the customer is asked for its electronic card. The customer introduces its electronic card and its PIN through a small keyboard. The customer presses the "Enter" button to confirm. The system reads the PIN and the respective electronic card identification, and verifies its validity. If the PIN is valid, the system accepts the input and the use case ends.

Use Case – ATM Example (cont.)

Validate User

Alternative Scenario 1 (Customer cancels operation)

The customer can cancel the transaction at any time by pressing the “Cancel” button, that implies the use case reset. No modification to the customer account is accomplished.

Alternative Scenario 2 (Invalid PIN)

If the customer introduces an invalid PIN, the electronic card is ejected and the use case restarted. If this invalidation occurs 3 consecutive times, the system captures the electronic card and cancels the transaction; and the ATM blocks itself during the following 60 seconds.

Use Case – ATM Example (cont.)

Alternative textual description

Main Scenario

1. *The use case is initiated when the system presents a screen where the user is asked for its electronic card.*
2. *The user introduces its electronic card and its PIN through a small keyboard.*
3. *The user presses the "Enter" button to confirm.*
4. *The system reads the PIN and the respective electronic card identification, and verifies its validity.*
5. *If the PIN is valid, the system recognizes the user as a particular bank customer and the use case ends.*

Alternative Scenario 1 (Customer cancels operation)

3a.1. The customer cancel the transaction at any time by pressing the "Cancel" button

3a.2. The use case ends and the system reset (i.e. presents the initial form). No modification to the customer account is accomplished.

…

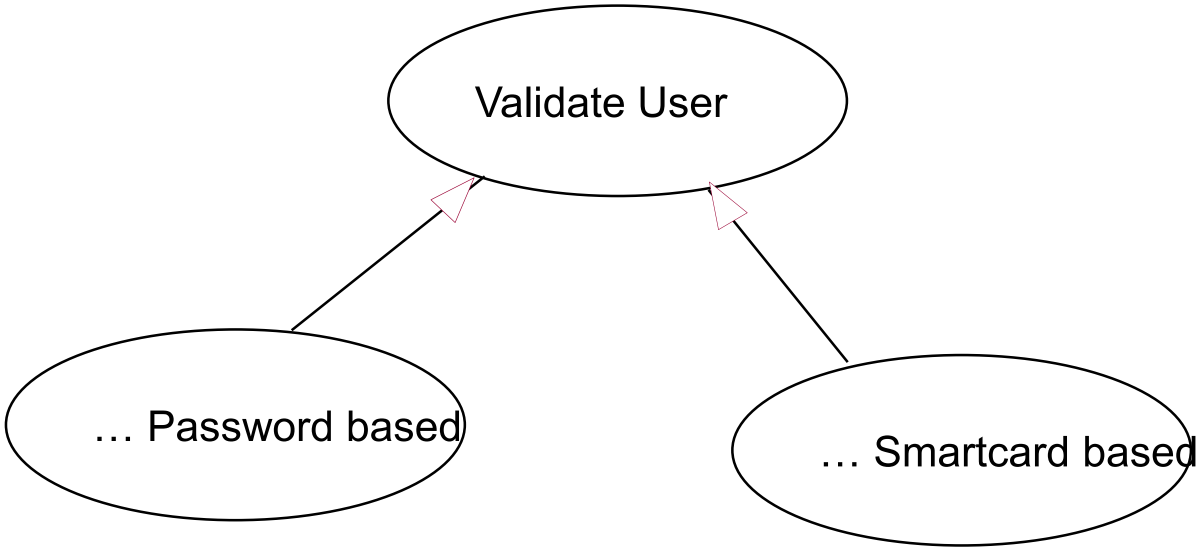
Use Cases and Relationships - Organization

Use cases can be organized in the following ways:

- Grouping in packages
- Specification of interrelations, such as: generalization, include and extend

Use Cases - Generalization

A relation of <u>generalization</u> between use cases <u>allows to refine </u>use cases from others that already exist, using the specialization mechanism, or alternatively, allows to define more abstract use cases from more specific use cases using the reduction or generalization mechanisms.

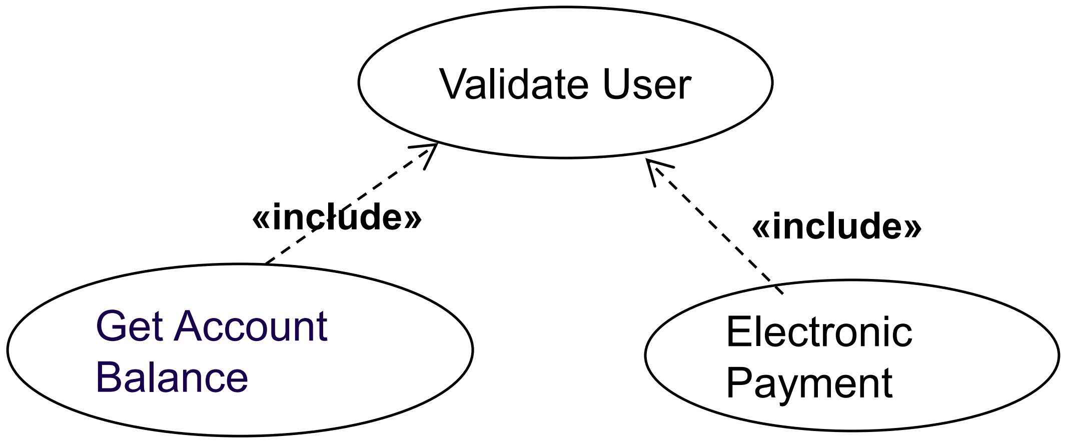


The "child" use case inherits the characteristics of its "father";

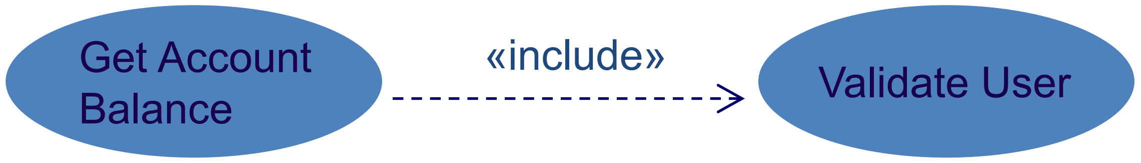
The "child" can refine or extend specifications defined for its "father".

Use Cases - Inclusion

- The include relation between two use cases corresponds to a typical relation of delegation, meaning that the source use case incorporates the behavior of the other target use case.
- The include relation makes possible to avoid describing the same flow of actions multiple times… (therefore, promotes reuse)



Use Cases - Inclusion



Use case “Get Account Balance”

Main Scenario:

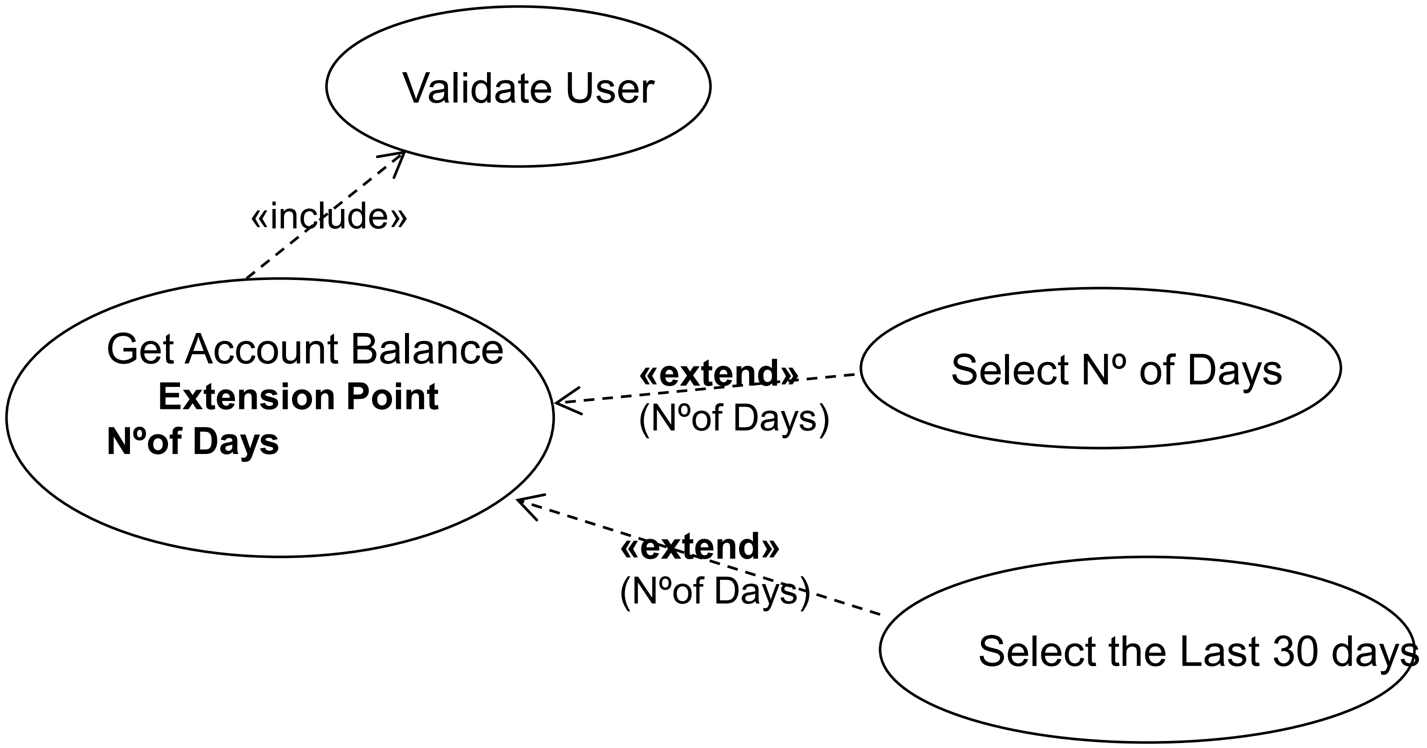
1. Include use case “Validate User”.
2. The System gets and verifies the bank account ID.
3. The System shows the home screen (with all the UI options, including the “Get Account Balance” button).
4. The User presses the “Get Account Balance” button.
5. The System selects all the account transactions accomplished in last 30 days and produces a summary list with these transactions, presenting the date, the type of transaction (debit or credit), a brief description and the value of the transaction. Also include the current balance of the account.
6. The system prints a paper with this information, ejecting it in the ATM terminal.
7. The System presents a message in the ATM terminal’ screen for the user to take that paper.
8. The User takes that paper from the ATM terminal

…

Use Cases - Extension

- A relation of extension between use cases means that the source use case implicitly incorporates its behavior in a point specified indirectly by the target use case. That is, the target use case can be extended with the behavior of other(s) case(s).
- An extension relation allows to represent:
  - The part of a use case that an user sees as optional or as existing in the context of several alternatives.
  - A sub-flow that is executed if only determined conditions are observed.
  - Several flows that can be inserted in one particular extension point, through a explicit interaction with an actor.
- The target Use Case is extended in one or more points, called “extension points”.

Use Cases - Extension



Note: This "Get Account Balance" use case with this Extension Point "Nº of Days" is a very ficcional but simple situation (in real world situations we do not show these variabilities in the model, but defined them in the textual description of the respective scenarios)

Use Cases - Extension

Name: **<u>Get Account Balance</u>**

Extension Points:

<u>Nº of Days (by default is 8 days)</u>

Main Scenario

1. <u>Include use case "Validate User"</u>.
2. Obtain and verify the account number.
3. Select all the account movements accomplished in last <u>"Nº of Days" (Extension Point)</u>.
4. Produce a summary list with these movements, presenting the date, the type of movement (debit or credit), a brief description and the value of the transaction. Include the current balance of the account.
5. Present a document with this information, ejecting it in the multibank terminal.
6. Present a message in the terminal for the customer to remove the trade bill.

Use Cases - Extension

Name: Select Number of Days

Main Scenario

A screen is presented where the user can specify the intended number of days, performing the corresponding selection with numerical buttons (of ‘0' to ‘9'). There is a text box dynamically constructed that displays the current value. Finally, the user uses the “confirm” button and the defined value is returned to the target use case in its “Nº of Days” extension point.

Alternative Scenario 1

Identical to the main scenario. At any point the user can use the button “delete” in order to delete the more recently introduced number.

Alternative Scenario 2

Identical to the main scenario. When the user uses “confirm” and the introduced value is superior than 59 days, a warning is presented, stating that the maximum number is 59, and the use case is restarted.

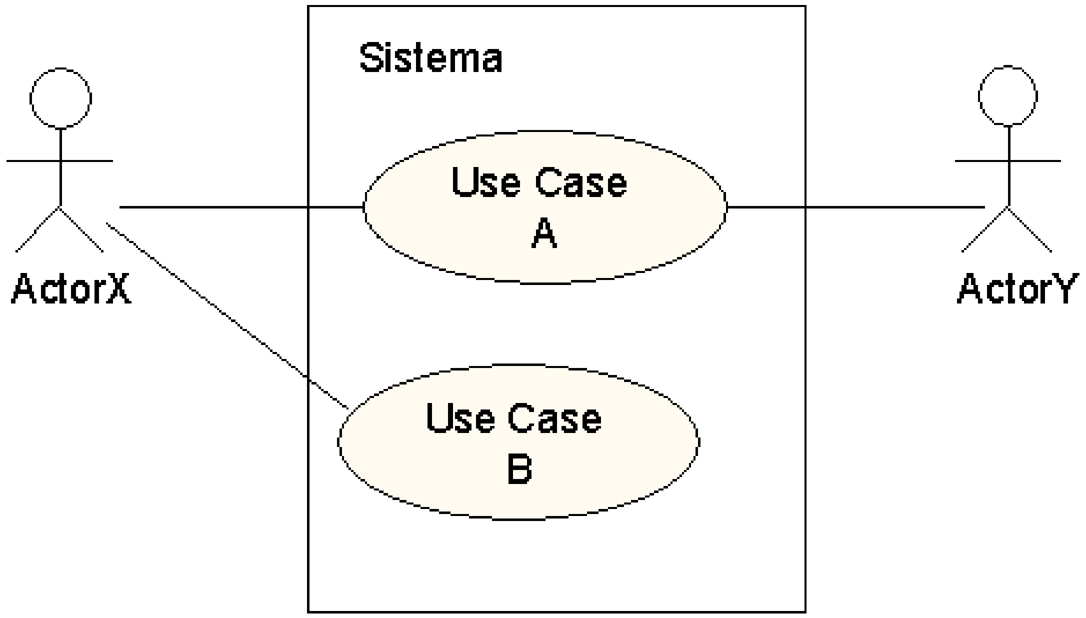
Alternative Scenario 3

Identical to the main scenario. At any point the user can use the “Cancel” button - the use case ends and the value 8 (day) is used, by default.

Use Case Diagrams

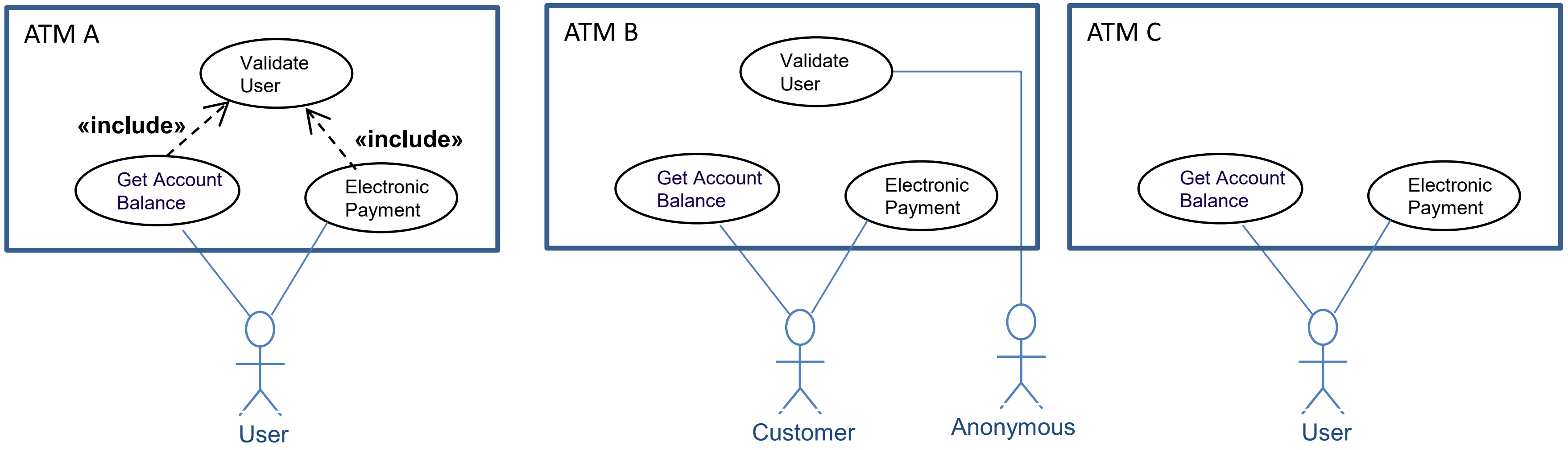
- Illustrate a set of Use Cases, Actors, and its relations.
- These diagrams have two common modeling applications:

- Context System: emphasis in the identification of the system`s boundary, by the identification of its actors and main functionalities.
- Functional Requirements : emphasis in the identification of what the system should perform, independently of the way how this is accomplished.



Use Cases – Inclusion (revisited)

- The include relation between two use cases corresponds to a typical relation of delegation, meaning that the source use case incorporates the behavior of the other target use case.
- The include relation makes possible to avoid describing the same flow of actions multiple times… (therefore, promotes reuse)



Note: “ATM A” is only a possible scenario, implying each time the user executes one of the two possible use cases, the identification is requested. In “ATM B” we conceive a different machine, with two classes of users, so there is a use case to first identify anonymous users, and therefore the two services only must be accessible to recognized costumers. In the example “ATM C” we have one more possible black box conceptualization of the machine where we simply ignore the validation (it can then be a pre-condition of the other two use cases, be an issue modelled in other diagram, etc…).

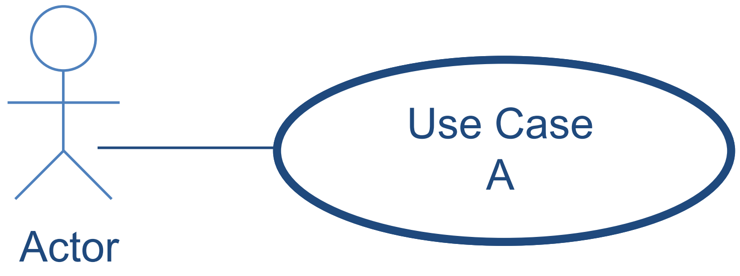
VERY IMPORTANT: Please note that what of these examples can be considered correct or wrong is not possible to decide without more rigorous requirements from the stakeholders!

Use Case Diagrams

- Use-case models describe user requirements, detailing all the scenarios that users can perform.
- The behavior model (or dynamics model) of a system can starts with use-case analysis.
- Use-cases can help or drive the project development.

Use Case Diagrams

- A communication between an actor and a use case implies an interaction between them.
- Each extreme in this relation has a navigation property, that indicates the communication direction.
- If the communication is bidirectional, the direction representation can be neglected.



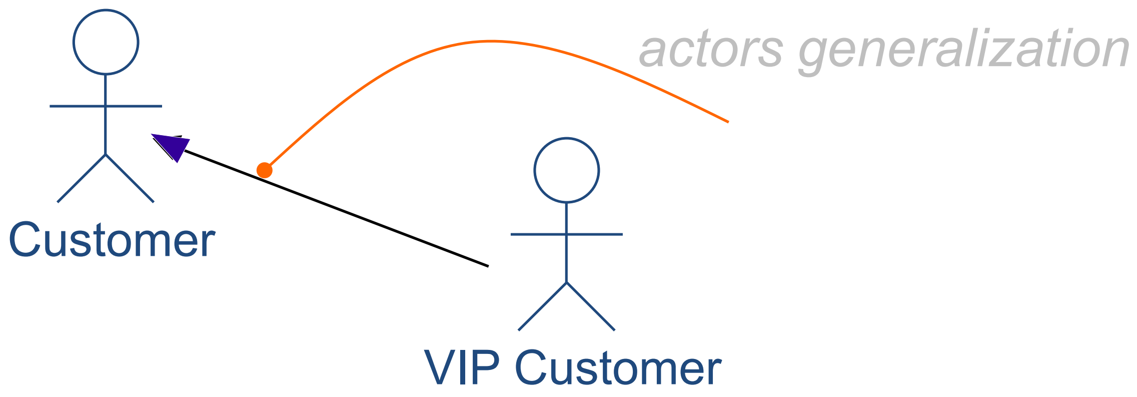
Use Case Diagrams - Actors

Actor (≅ user profile)

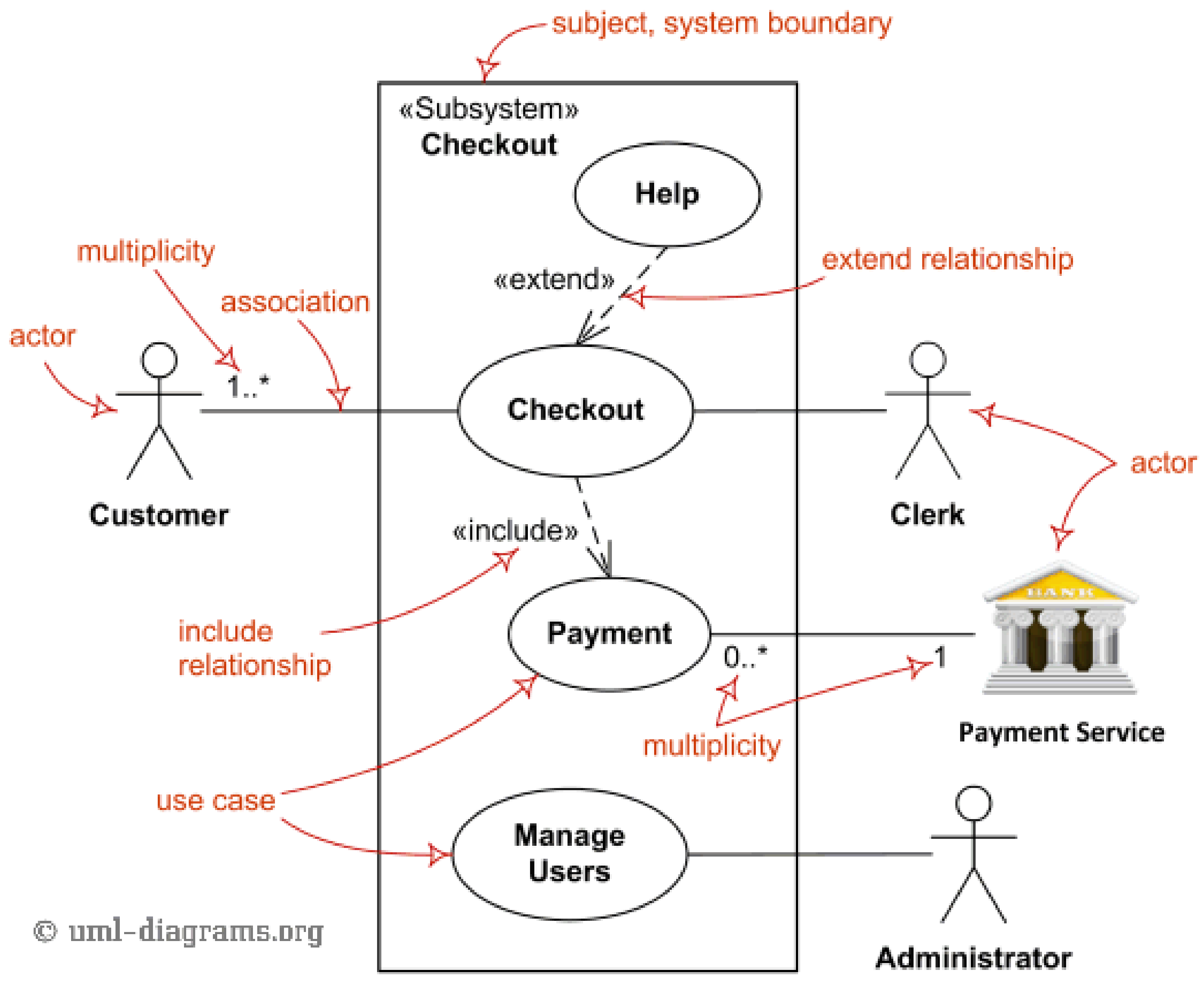
… represents a role that a user perform in the system

…one particular user can play different roles (being able to represent different actors)

…but can also represents external systems, that somehow interact with the system under study

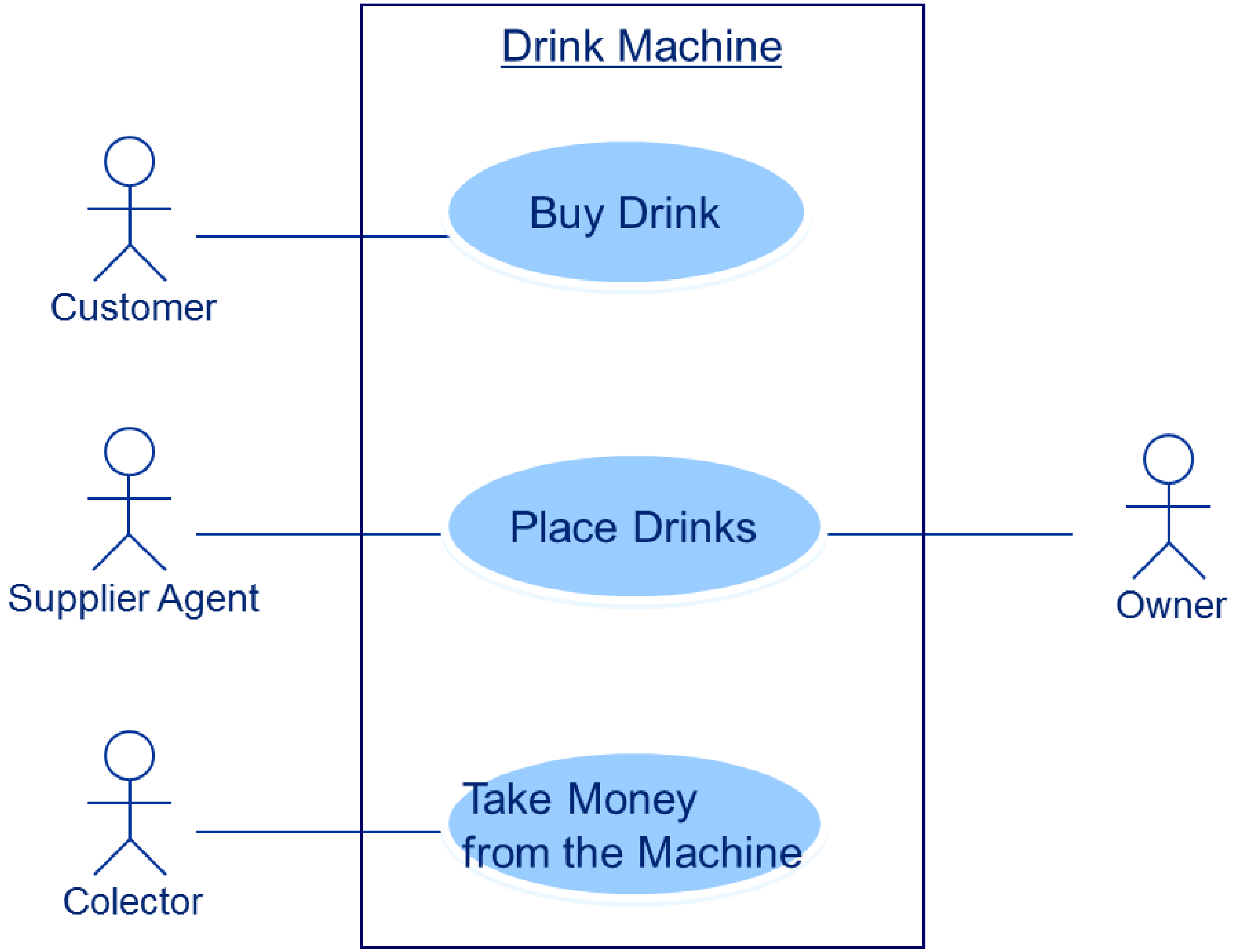


Use Case Diagrams – Summary



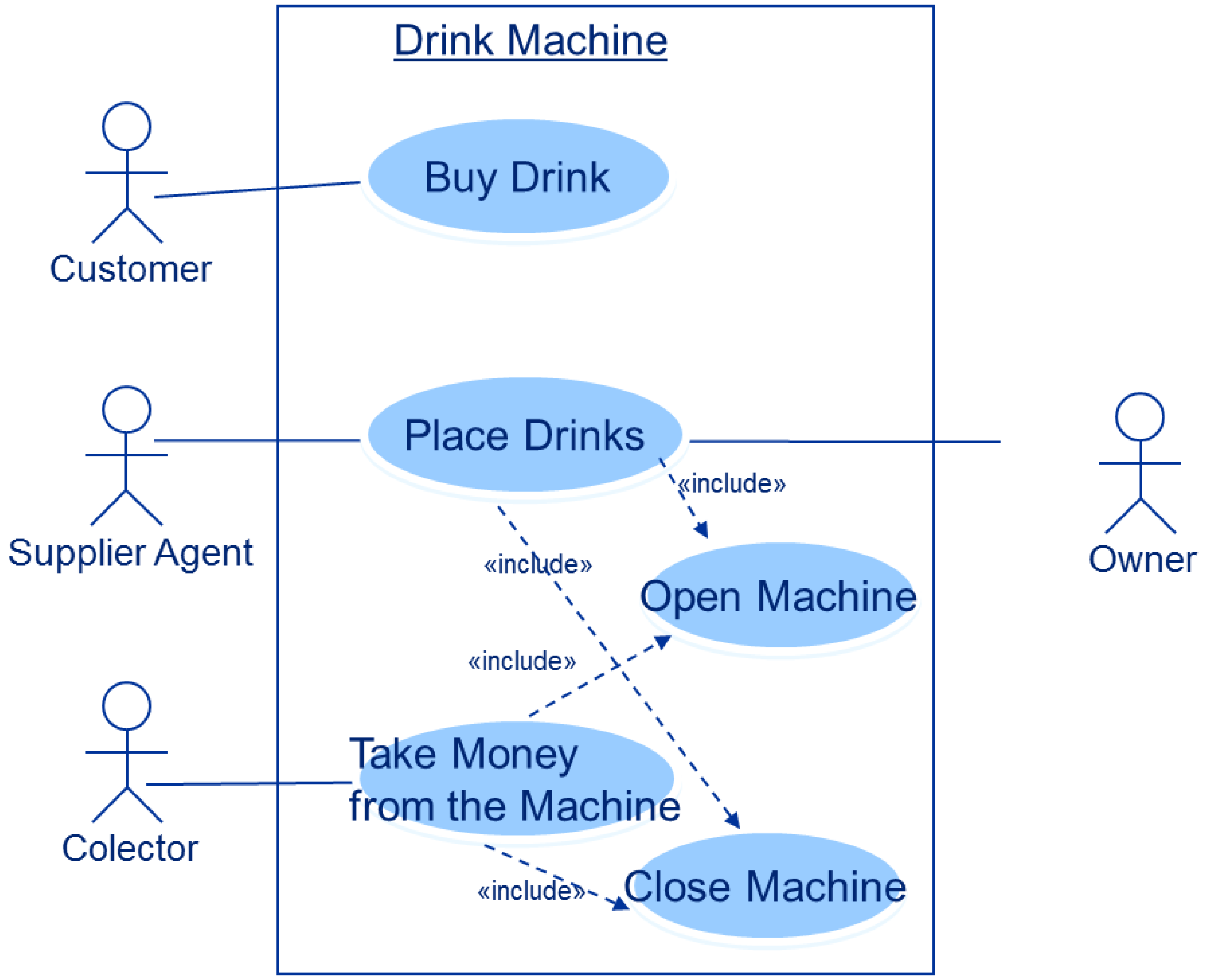
Use Case Diagrams - Example

Use Case Diagrams - Example



Use Case Diagrams - Example

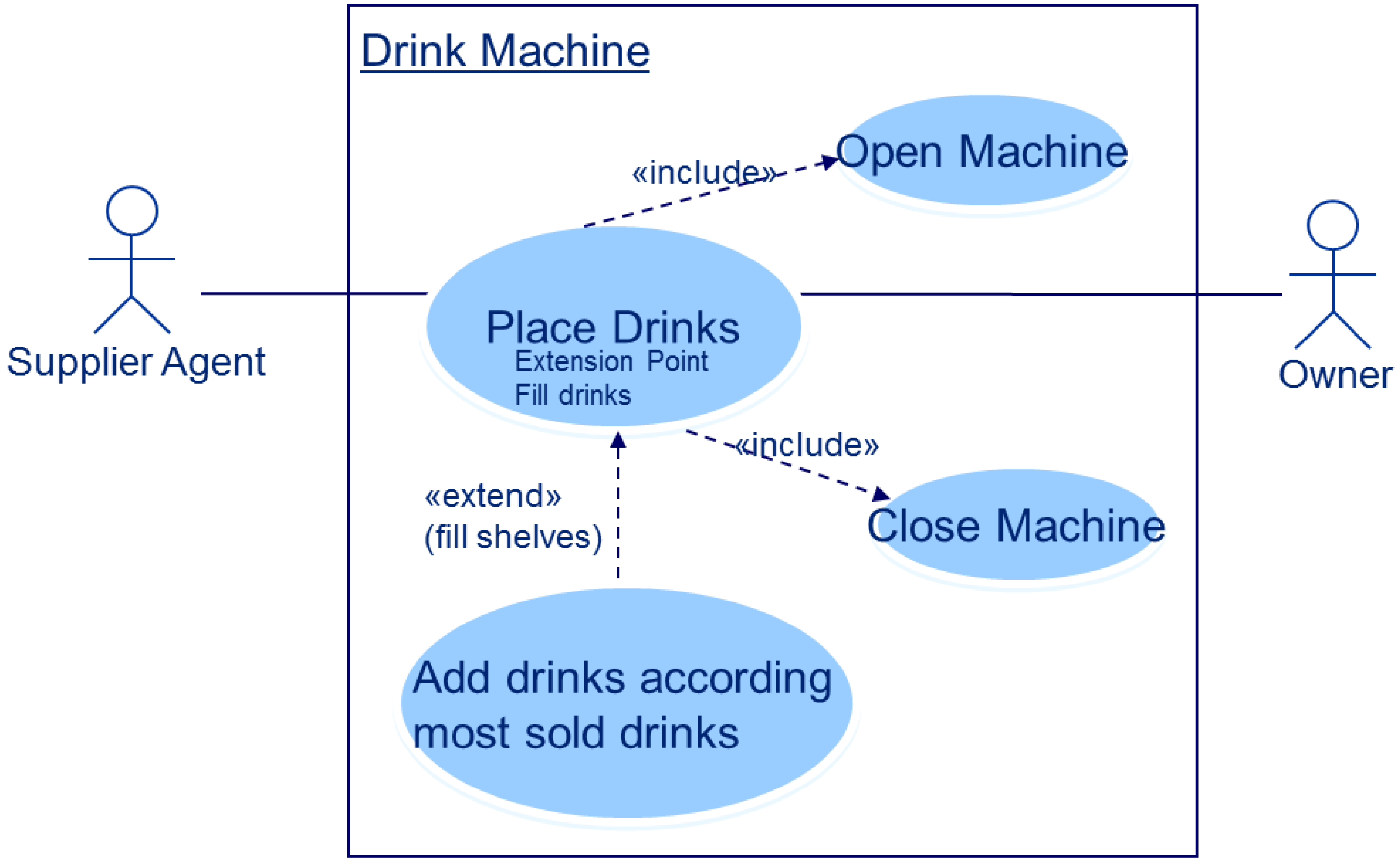
Include...



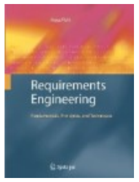
Use Case Diagrams - Example

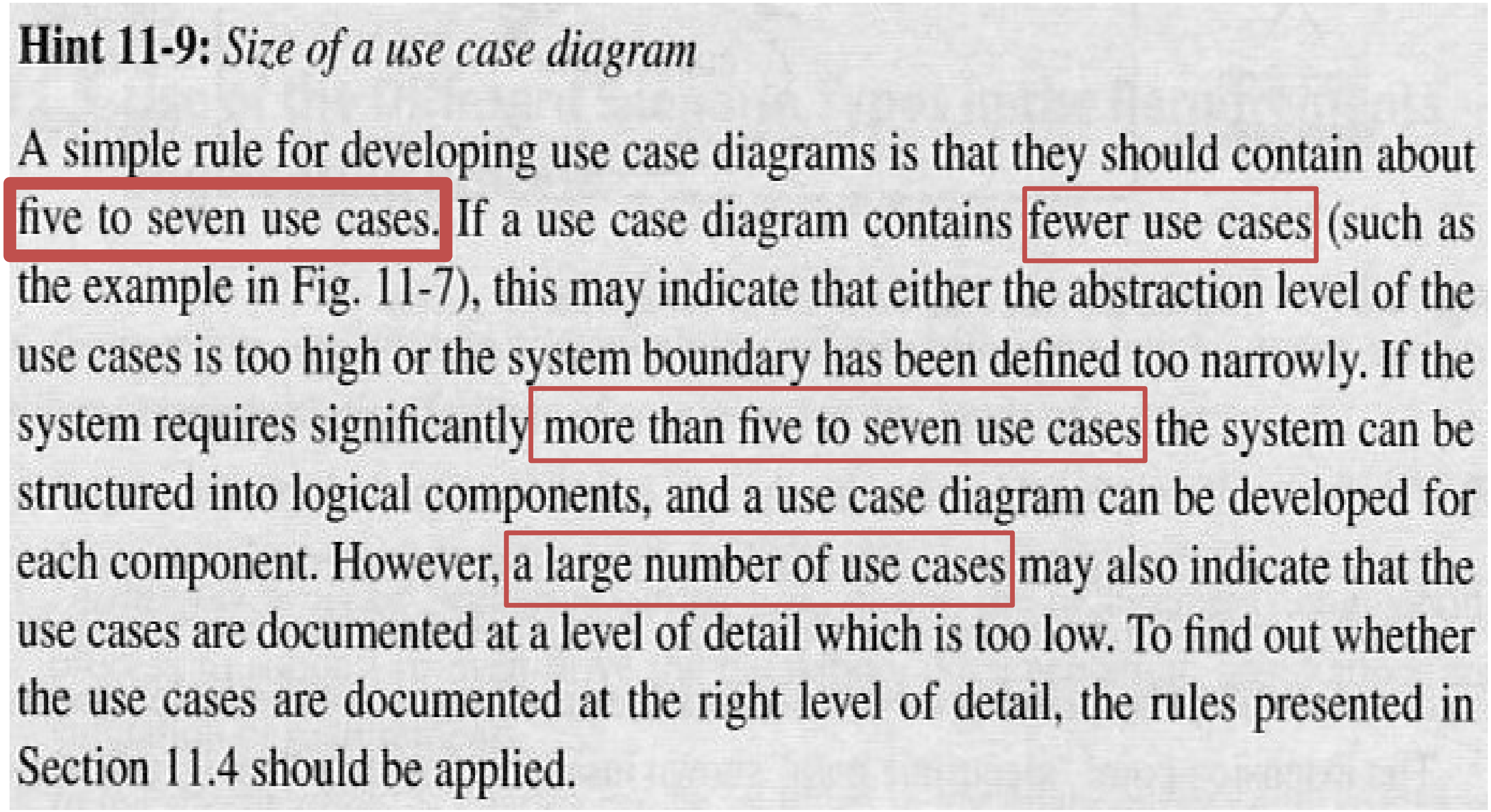
Extend...

The possibility of drinks reposition/placement depends now of different algorithms, e.g. based on the most sold drinks, most profitable drinks, etc…



Use Cases: hints: Size of a UC diags





Hint 11-9: *Size of a use case diagram*

A simple rule for developing use case diagrams is that they should contain about five to seven use cases. If a use case diagram contains fewer use cases (such as the example in Fig. 11-7), this may indicate that either the abstraction level of the use cases is too high or the system boundary has been defined too narrowly. If the system requires significantly more than five to seven use cases the system can be structured into logical components, and a use case diagram can be developed for each component. However, a large number of use cases may also indicate that the use cases are documented at a level of detail which is too low. To find out whether the use cases are documented at the right level of detail, the rules presented in Section 11.4 should be applied.

Use Cases: hints: Use case names

- A Use Case should be written in **active voice** with a **verb** in the **infinitive form.**
- **Subject-Object-Predicate** format.
  - Generate Report
  - Calibrate Wheel
  - Drive Car
  - Handle Costumer Transaction
  - Place Order
  - Take Customer Order
  - Return Faulty Goods

Use Cases: hints: Extend

"Bear in mind that the «extend» relationship is more likely to cause confusion and disagreement than almost any other area of your UML/SysML analysis model. These debates can be time-consuming and pointless so approach it with caution and use it sparingly."

Karona Consulting Ltd. The «include» and «extend» Relationships in Use Case Models. 2010.

Include and Extend – Be careful:

- **These relationships should <u>never</u> be used before the first high level version of all the use cases is complete and fully validated.**
- Using these relationships early is a source of distraction and tends to introduce severe modelling errors,
- These relationships are often improperly used.
- **The aim of these relationships is to simplify the model, not to obscure it!**

Include and Extend – value:

- «include» can be used to replace common sequences of user-system interaction.
- «extend» can be used when:
  - A use case is too big and the scenarios all the alternative scenarios are becoming unmanageable. «extend» allows a use case to be broken up into several small use cases.
  - The base use case is not supposed to be changed. «extend» allows to the describe the new functionality in a separate use case.

Karona Consulting Ltd. The «include» and «extend» Relationships in Use Case Models. 2010.

Include and Extend – simple how to:

1. Do the additional steps constitute a single contiguous set of steps?

Yes, you are adding steps at one place in the flow – use «include».

No, you need to modify the primary Use Case in more than one place – use «extend».

2. Do the additional steps make sense on their own?

Yes – use «include».

No – use «extend».

In an included Use Case the additional steps may or may not be performable in isolation but they are likely to have a degree of coherence on their own. The steps of the extending Use Case are more likely to be an incomplete fragment or sequence of fragments.

3. Do the additional steps occur in more than one primary Use Case?

Yes – use «include».

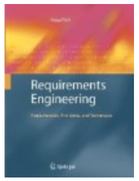
No – use «extend».

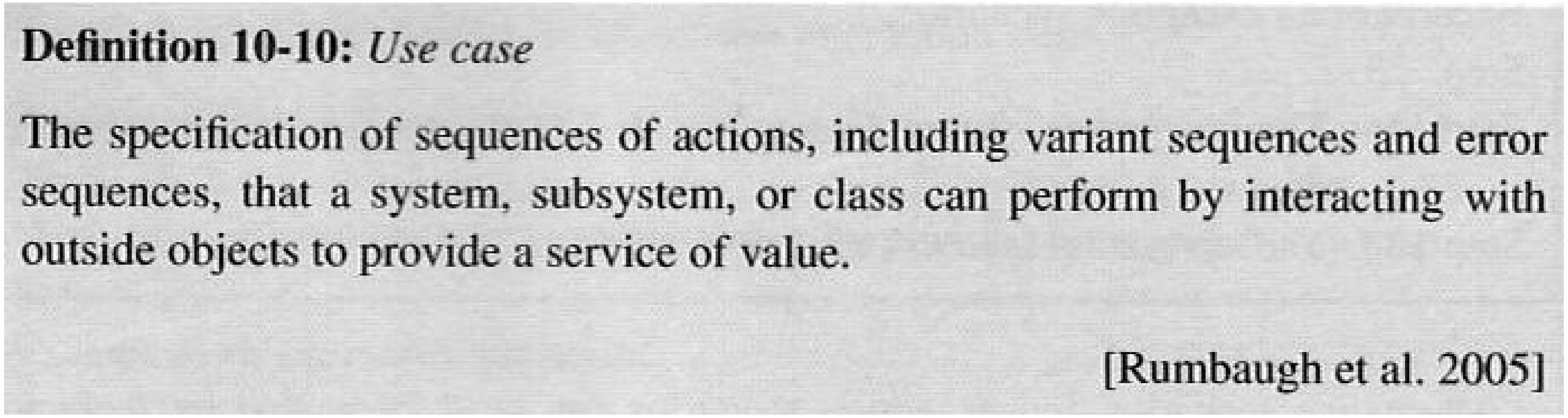
Karona Consulting Ltd. The «include» and «extend» Relationships in Use Case Models. 2010.

Use Case Scenarios

- A **use case scenario** is an instance of a use case.
- It represents a concrete sequence of actions that describes how the system actually behaves.
- A use case is usually described by several scenarios (one for each goal).
- Types of scenarios:
  - **Principal scenario** (main, primary, standard): describes the normal sequence of actions that are required for the use case to achieve its goals or results.
  - **Alternative scenario**: describes a sequence of actions that differs from the main scenario but that achieves exactly the same goals or results.
  - **Exception scenario:** describes the sequence of actions when an error or exception occurs. Some or all of the goals of the use case cannot be achieved.

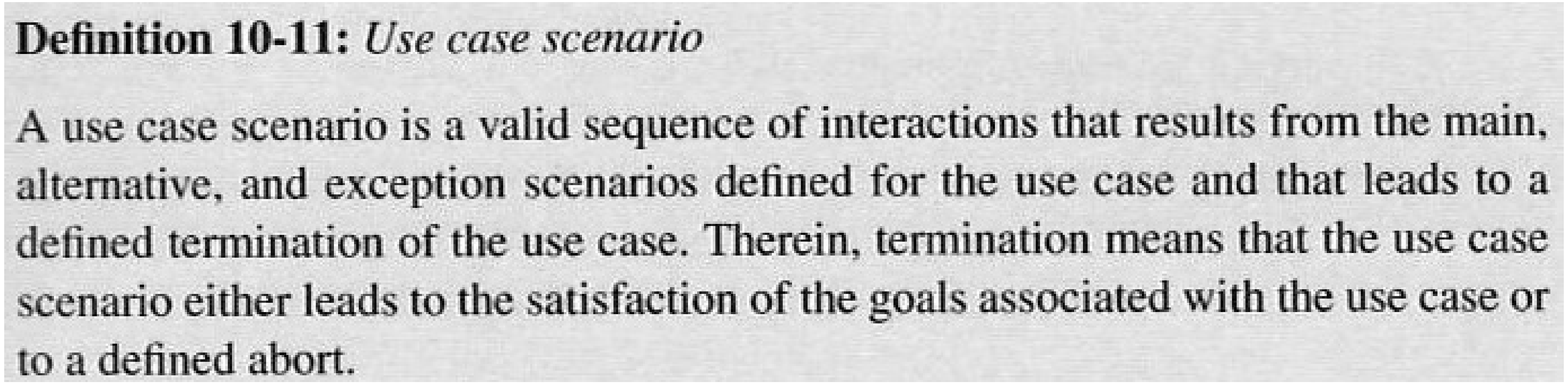
Use Cases and Use Case Scenarios


Requirements Engineering


Definition 10-10: *Use case*

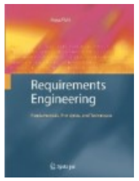
The specification of sequences of actions, including variant sequences and error sequences, that a system, subsystem, or class can perform by interacting with outside objects to provide a service of value.

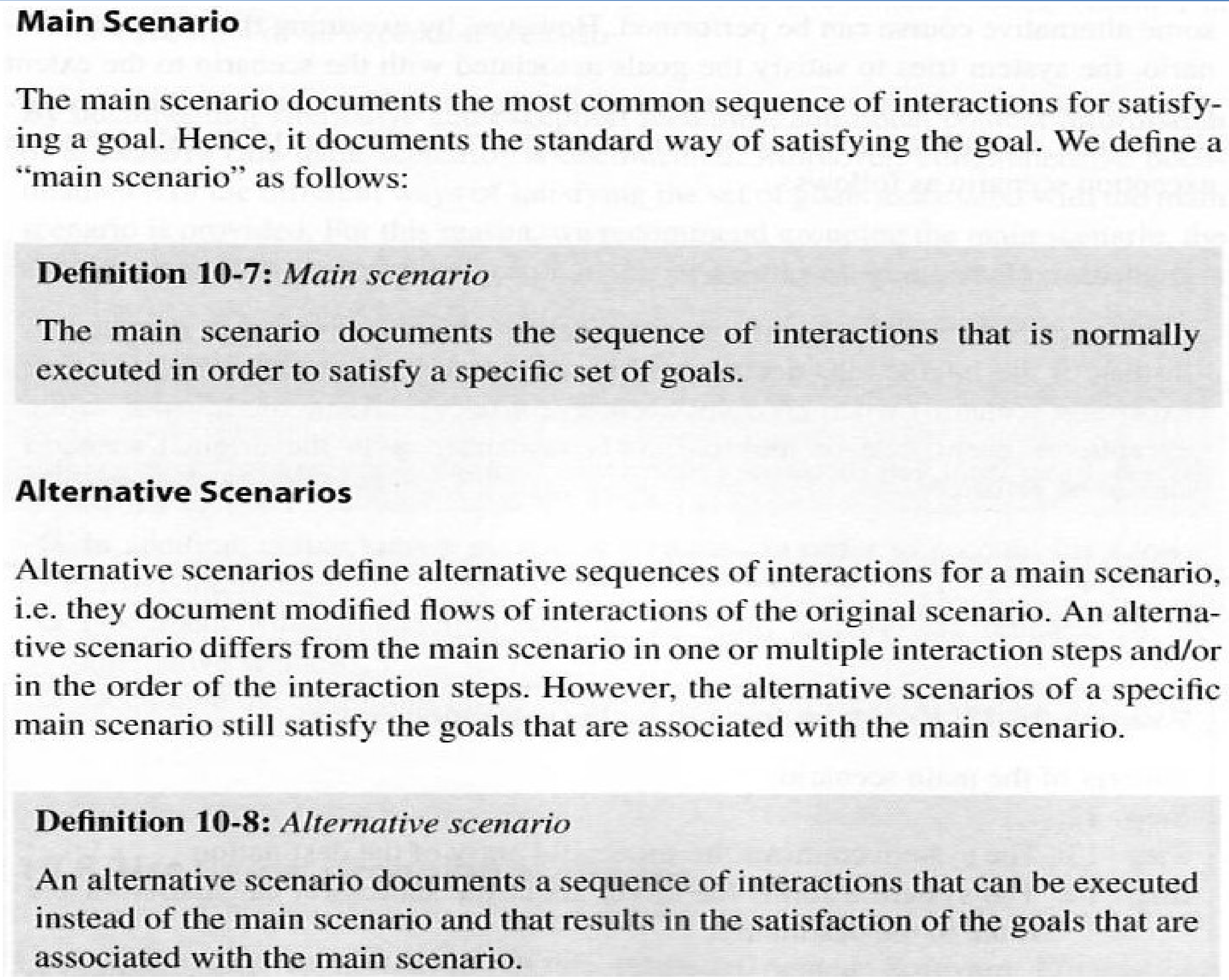
[Rumbaugh et al. 2005]


Definition 10-11: *Use case scenario*

A use case scenario is a valid sequence of interactions that results from the main, alternative, and exception scenarios defined for the use case and that leads to a defined termination of the use case. Therein, termination means that the use case scenario either leads to the satisfaction of the goals associated with the use case or to a defined abort.

Use Case Scenarios


Requirements Engineering


Main Scenario

The main scenario documents the most common sequence of interactions for satisfying a goal. Hence, it documents the standard way of satisfying the goal. We define a "main scenario" as follows:

Definition 10-7: *Main scenario*

The main scenario documents the sequence of interactions that is normally executed in order to satisfy a specific set of goals.

Alternative Scenarios

Alternative scenarios define alternative sequences of interactions for a main scenario, i.e. they document modified flows of interactions of the original scenario. An alternative scenario differs from the main scenario in one or multiple interaction steps and/or in the order of the interaction steps. However, the alternative scenarios of a specific main scenario still satisfy the goals that are associated with the main scenario.

Definition 10-8: *Alternative scenario*

An alternative scenario documents a sequence of interactions that can be executed instead of the main scenario and that results in the satisfaction of the goals that are associated with the main scenario.

An Example of a Use Case Scenario

Main Scenario of the "Order Product" Use Case

1. Use case starts when the *Customer* selects the "Order Product" Option.
2. The *Customer* fills in its name and address.
3. The *Customer* introduces a full or partial product code.
4. The *System* shows the *product description* and *price* that match the *product code.*
5. The *Customer* selects the *products* he wants to order.
6. The *Customer* provides its *credit card details.*
7. The *Customer* finalizes the order.
8. The *System* verifies all the information supplied by the *Customer*.
9. The *System* generates and provides the *Customer* with an order code.

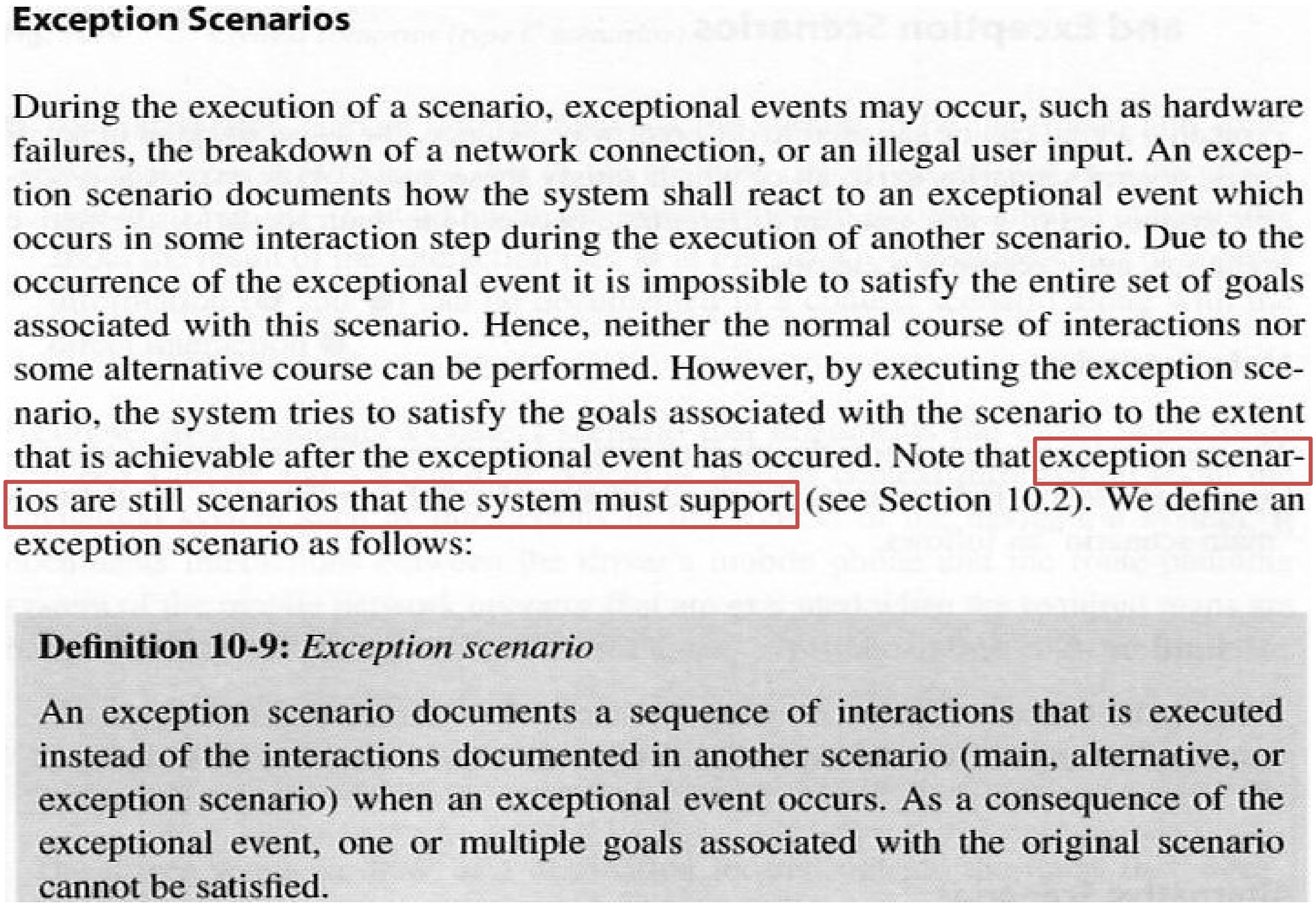
Alternative Scenarios of the "Order Product" Use Case

- Customer pays by Bank Transfer.
  1. …
  2. …

Exception Scenarios of the "Order Product" Use Case

- Credit card is not approved.
- Delivery address unknown or incomplete.
- Product out of stock.
- Product no longer available.

Use Case Scenarios



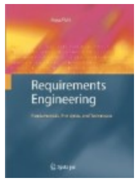
Exception Scenarios

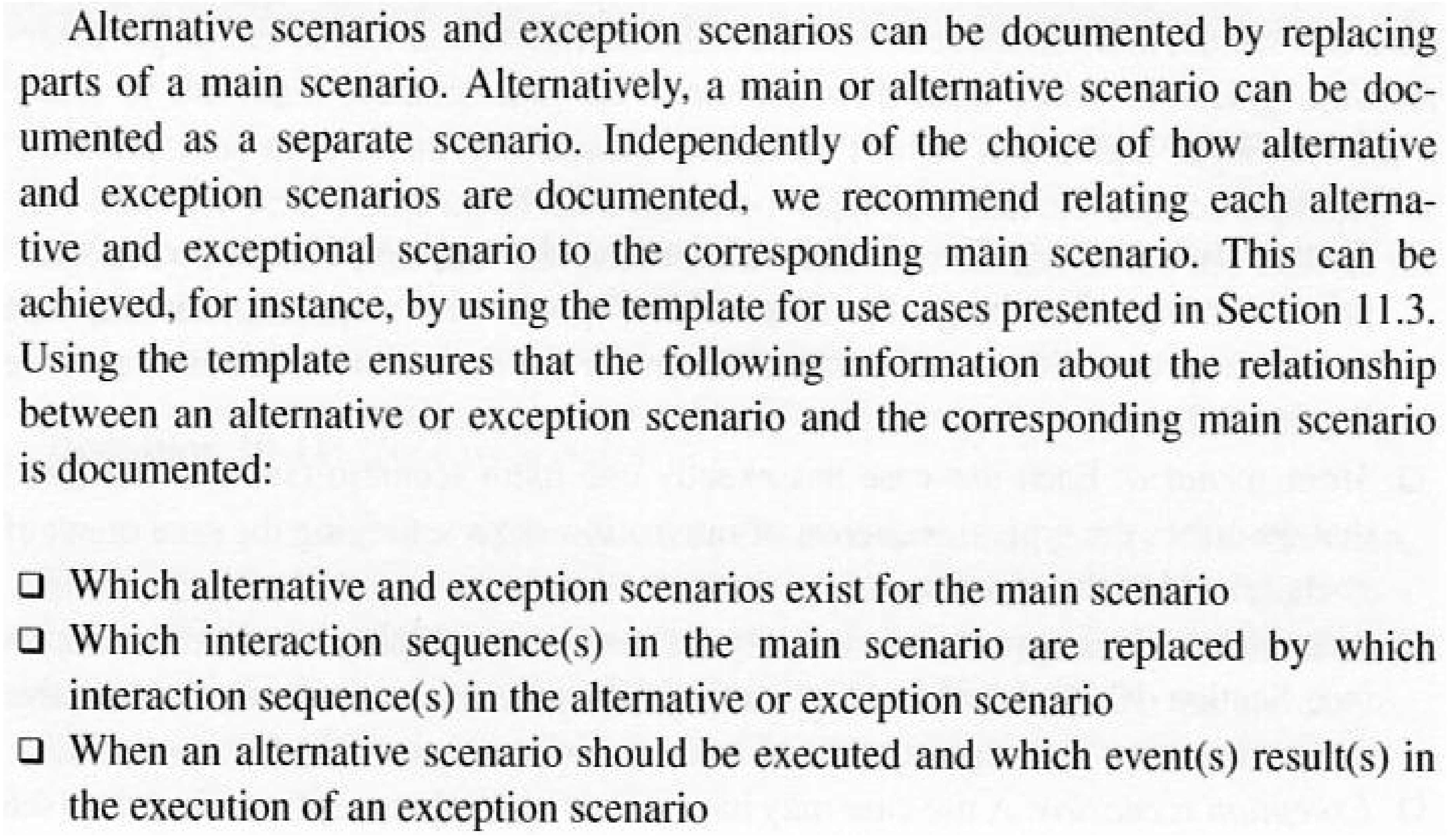
During the execution of a scenario, exceptional events may occur, such as hardware failures, the breakdown of a network connection, or an illegal user input. An exception scenario documents how the system shall react to an exceptional event which occurs in some interaction step during the execution of another scenario. Due to the occurrence of the exceptional event it is impossible to satisfy the entire set of goals associated with this scenario. Hence, neither the normal course of interactions nor some alternative course can be performed. However, by executing the exception scenario, the system tries to satisfy the goals associated with the scenario to the extent that is achievable after the exceptional event has occured. Note that exception scenarios are still scenarios that the system must support (see Section 10.2). We define an exception scenario as follows:

Definition 10-9: *Exception scenario*

An exception scenario documents a sequence of interactions that is executed instead of the interactions documented in another scenario (main, alternative, or exception scenario) when an exceptional event occurs. As a consequence of the exceptional event, one or multiple goals associated with the original scenario cannot be satisfied.

Use Case Scenarios

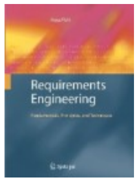


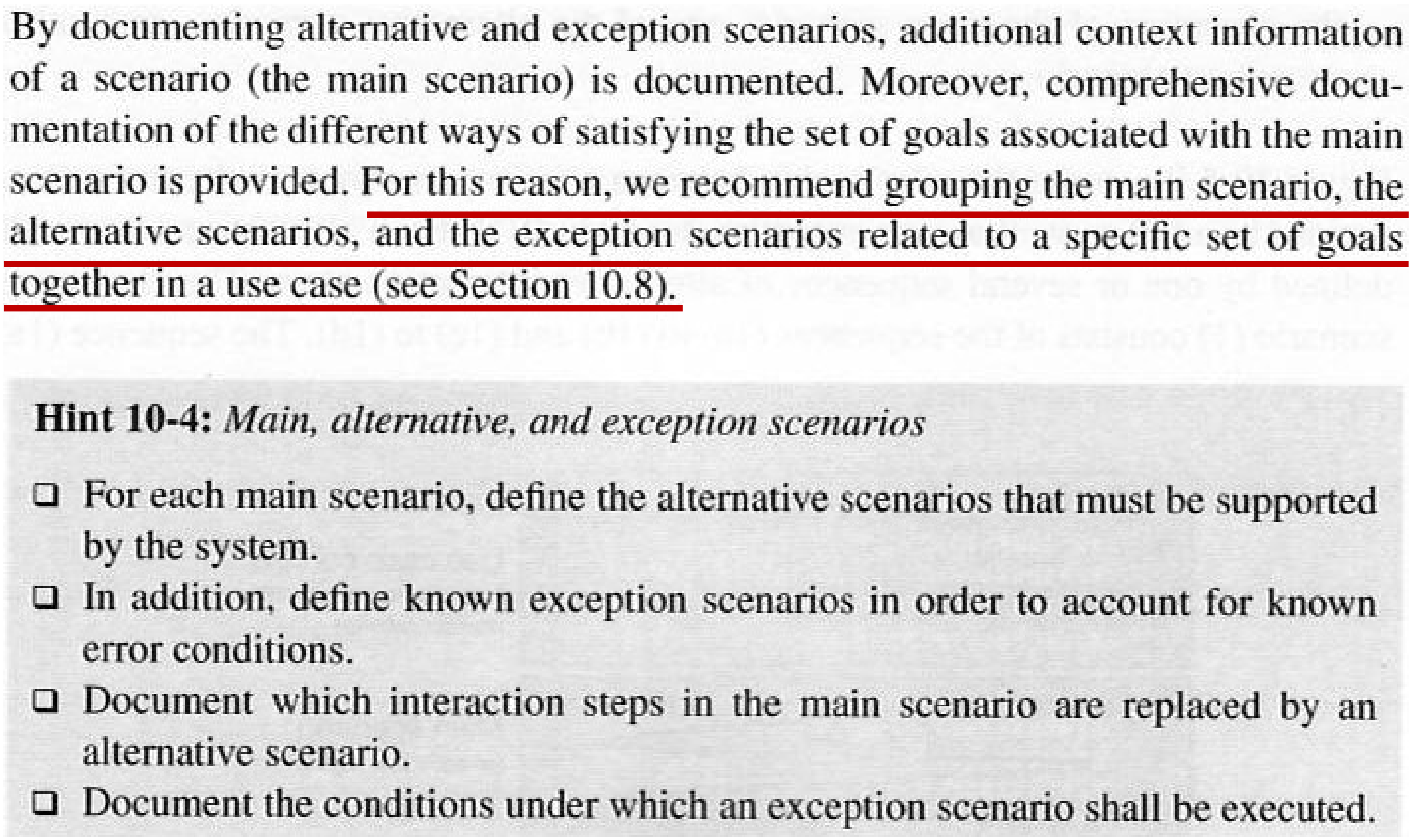


Alternative scenarios and exception scenarios can be documented by replacing parts of a main scenario. Alternatively, a main or alternative scenario can be documented as a separate scenario. Independently of the choice of how alternative and exception scenarios are documented, we recommend relating each alternative and exceptional scenario to the corresponding main scenario. This can be achieved, for instance, by using the template for use cases presented in Section 11.3. Using the template ensures that the following information about the relationship between an alternative or exception scenario and the corresponding main scenario is documented:

- ❑ Which alternative and exception scenarios exist for the main scenario
- ❑ Which interaction sequence(s) in the main scenario are replaced by which interaction sequence(s) in the alternative or exception scenario
- ❑ When an alternative scenario should be executed and which event(s) result(s) in the execution of an exception scenario

Use Case Scenarios

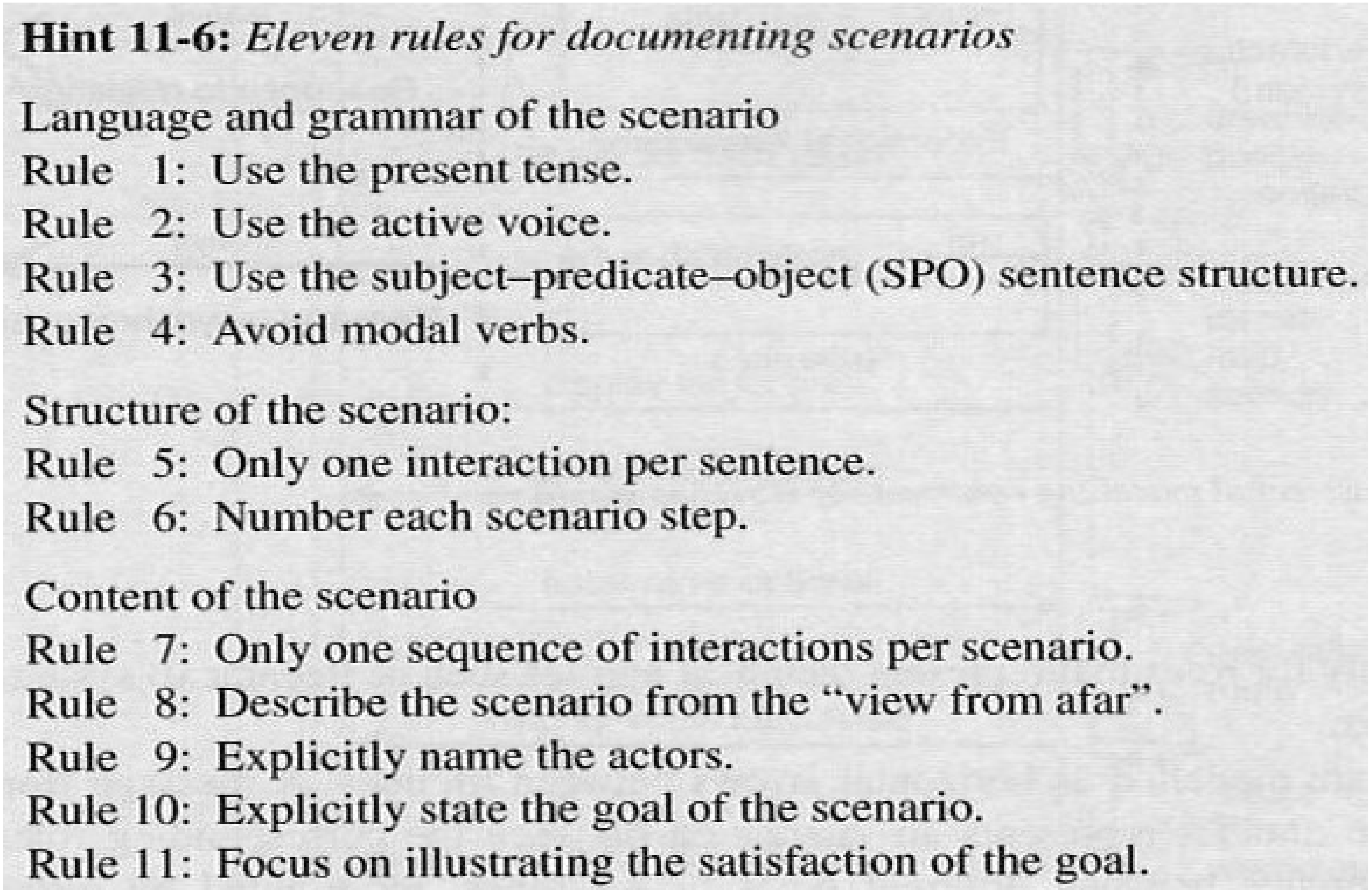

Requirements Engineering


By documenting alternative and exception scenarios, additional context information of a scenario (the main scenario) is documented. Moreover, comprehensive documentation of the different ways of satisfying the set of goals associated with the main scenario is provided. For this reason, we recommend grouping the main scenario, the alternative scenarios, and the exception scenarios related to a specific set of goals together in a use case (see Section 10.8).

Hint 10-4: *Main, alternative, and exception scenarios*

- ❑ For each main scenario, define the alternative scenarios that must be supported by the system.
- ❑ In addition, define known exception scenarios in order to account for known error conditions.
- ❑ Document which interaction steps in the main scenario are replaced by an alternative scenario.
- ❑ Document the conditions under which an exception scenario shall be executed.

Documenting Use Cases



Hint 11-6: *Eleven rules for documenting scenarios*

Language and grammar of the scenario

Rule 1:	Use the present tense.
Rule 2:	Use the active voice.
Rule 3:	Use the subject–predicate–object (SPO) sentence structure.
Rule 4:	Avoid modal verbs.

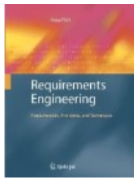
Structure of the scenario:

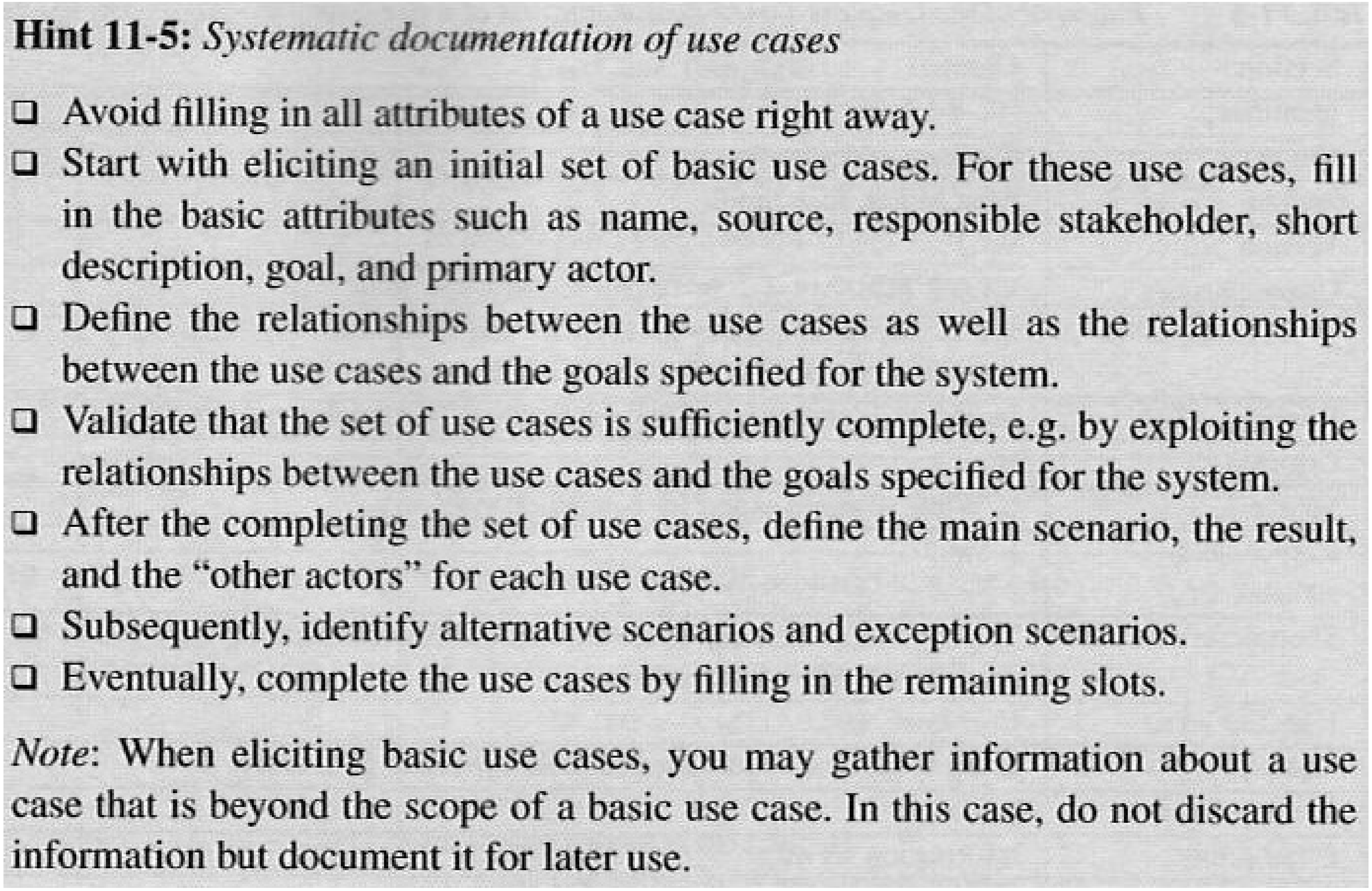
Rule 5:	Only one interaction per sentence.
Rule 6:	Number each scenario step.

Content of the scenario

Rule 7:	Only one sequence of interactions per scenario.
Rule 8:	Describe the scenario from the "view from afar".
Rule 9:	Explicitly name the actors.
Rule 10:	Explicitly state the goal of the scenario.
Rule 11:	Focus on illustrating the satisfaction of the goal.

Documenting Use Cases



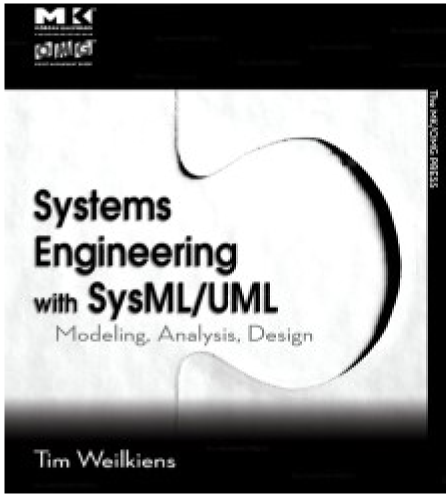


Hint 11-5: *Systematic documentation of use cases*

- ❑ Avoid filling in all attributes of a use case right away.
- ❑ Start with eliciting an initial set of basic use cases. For these use cases, fill in the basic attributes such as name, source, responsible stakeholder, short description, goal, and primary actor.
- ❑ Define the relationships between the use cases as well as the relationships between the use cases and the goals specified for the system.
- ❑ Validate that the set of use cases is sufficiently complete, e.g. by exploiting the relationships between the use cases and the goals specified for the system.
- ❑ After the completing the set of use cases, define the main scenario, the result, and the "other actors" for each use case.
- ❑ Subsequently, identify alternative scenarios and exception scenarios.
- ❑ Eventually, complete the use cases by filling in the remaining slots.

Note: When eliciting basic use cases, you may gather information about a use case that is beyond the scope of a basic use case. In this case, do not discard the information but document it for later use.

Use Cases and Use Case Scenarios

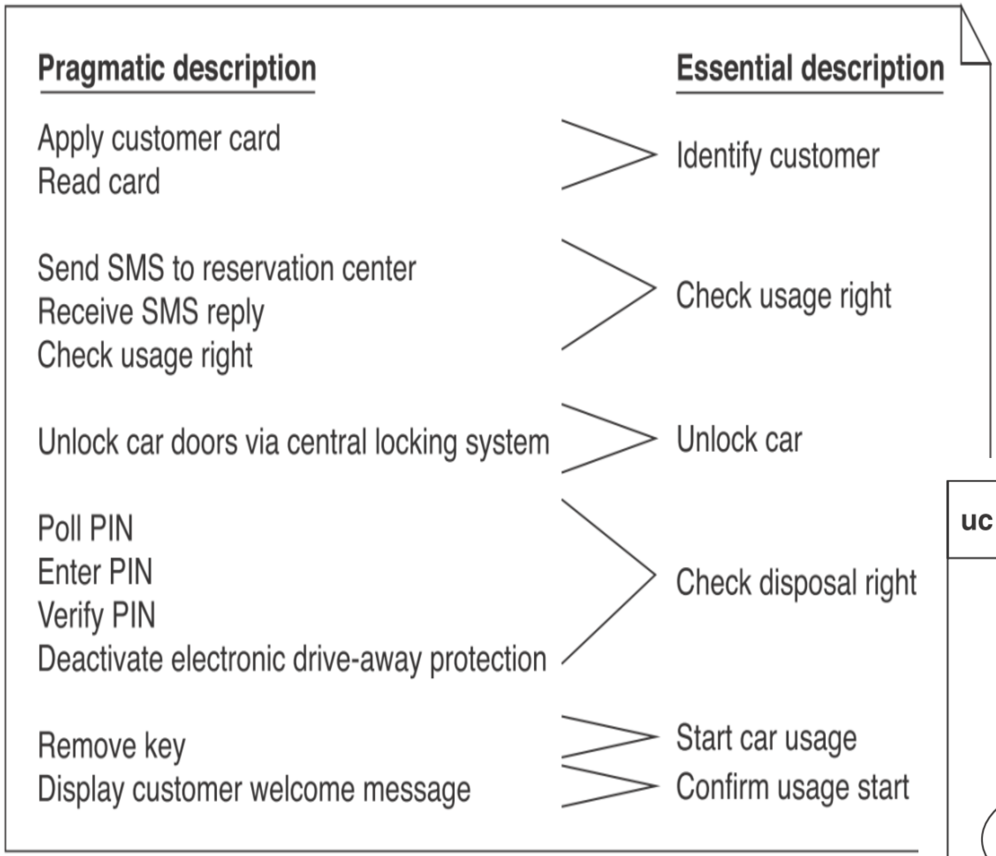


MK OMG

Systems Engineering with SysML/UML

Modeling, Analysis, Design

Tim Weilkiens



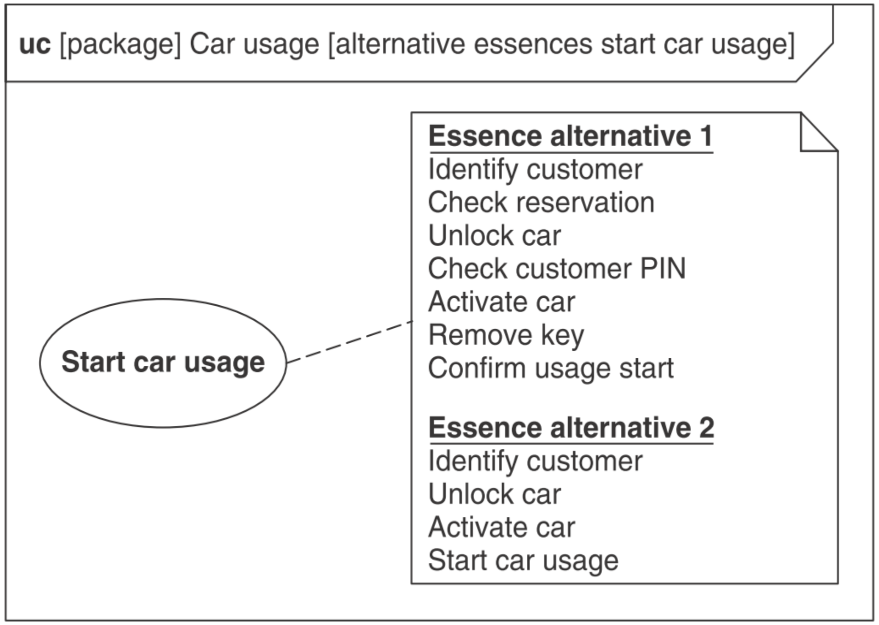


FIGURE 2-46

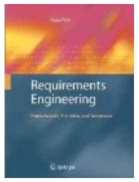
Alternative essences for "start car usage."

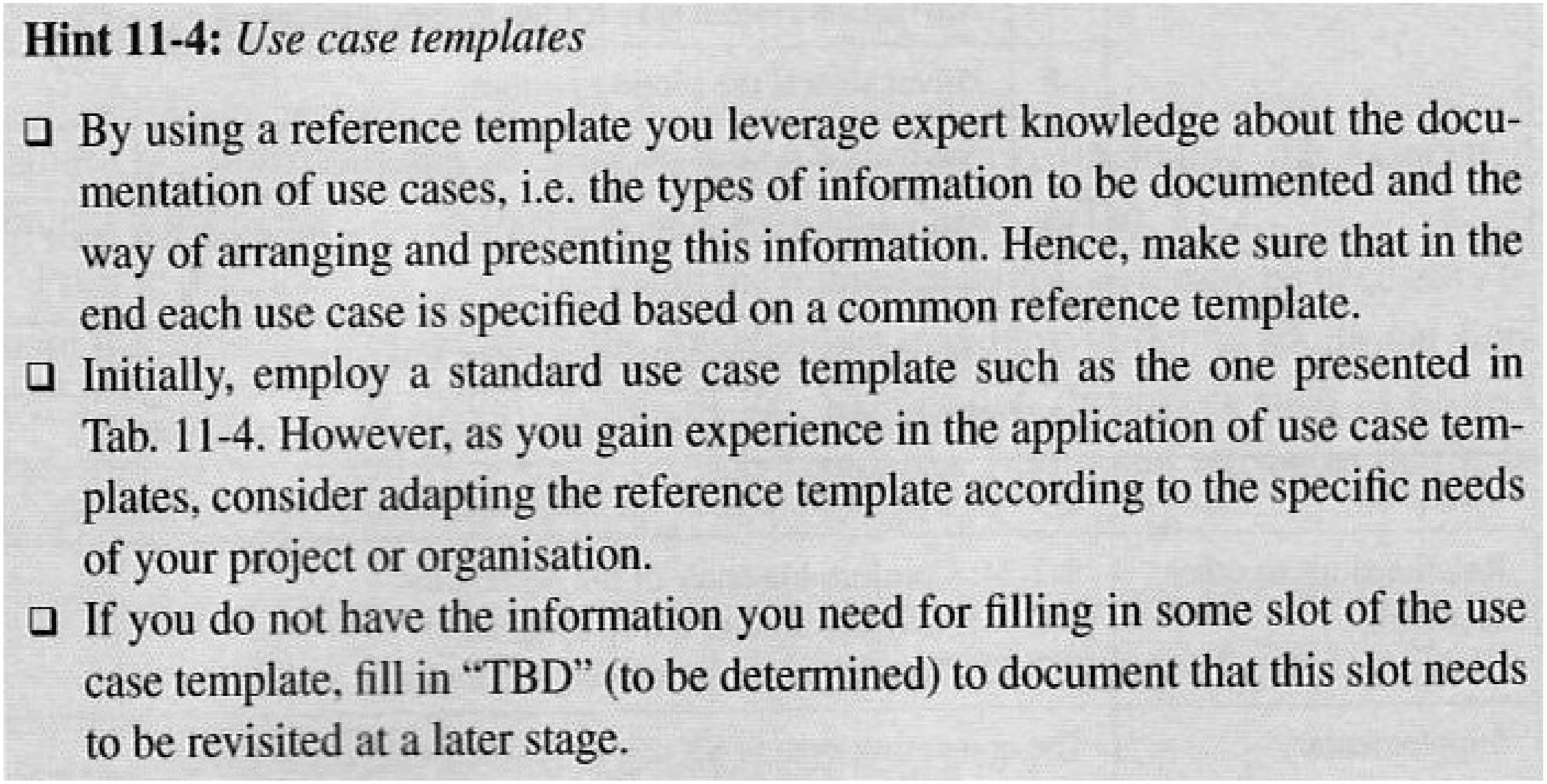
Use Case Templates

- An **use case specification** must include:
  - **Assumptions**
  - **Pre-conditions** to activate the UC
  - **Initialization** how and when the use case starts
  - **Dialog** actor-use case interaction
  - **Conclusion** how and when the
  - **Post-conditions** valid after concluding the UC

- **Use case templates** are a good technique to organize these specifications (each project or team must develop it, according to the problem domain, analysis and design method, etc.).

Use Case Templates



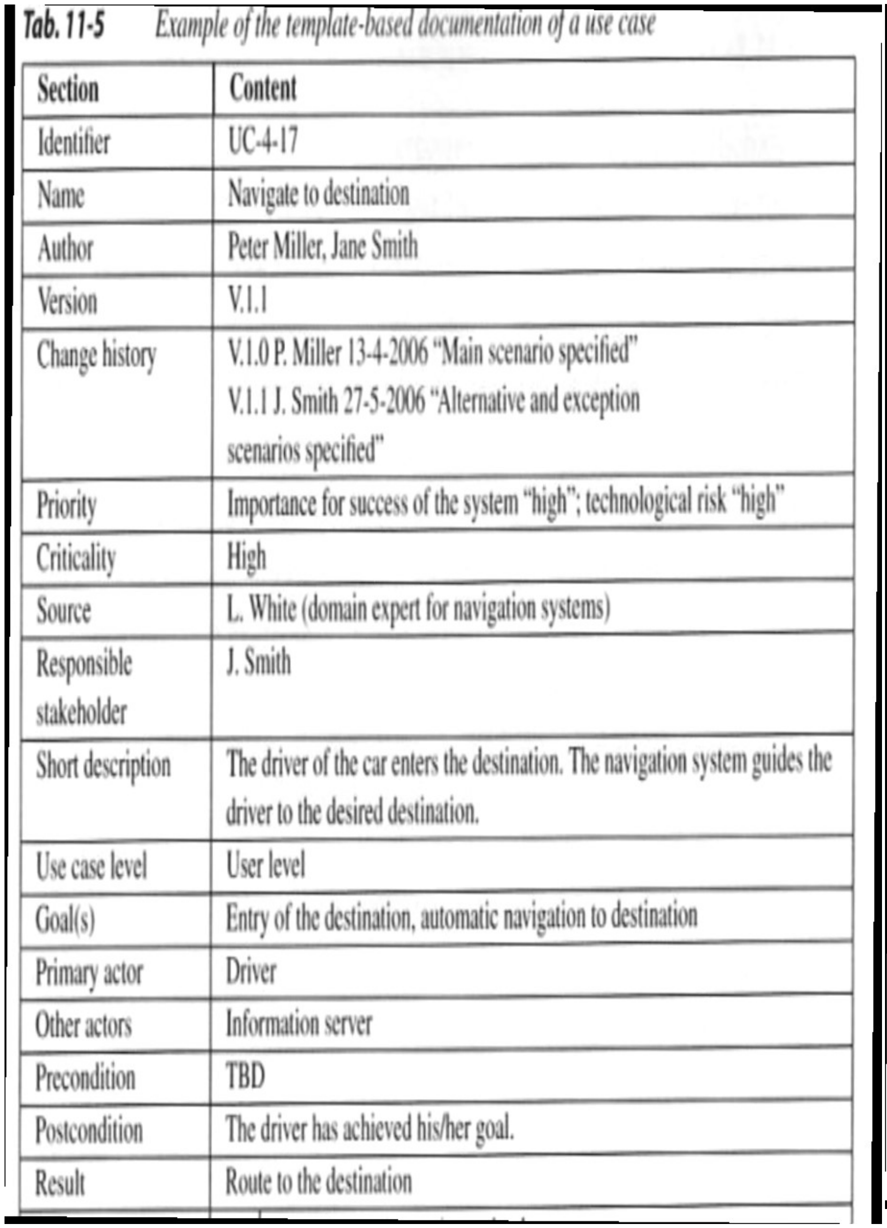
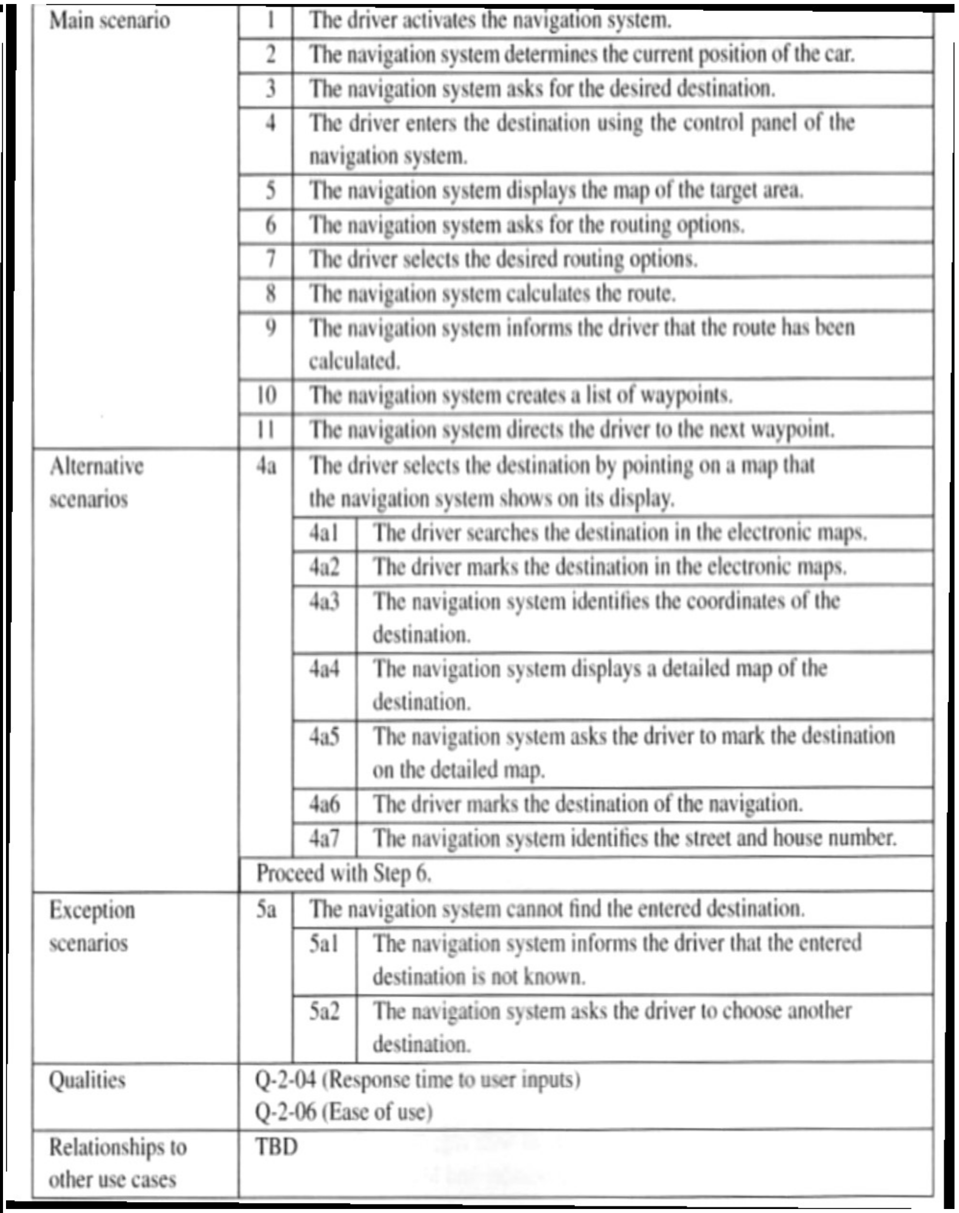


Hint 11-4: *Use case templates*

- ❑ By using a reference template you leverage expert knowledge about the documentation of use cases, i.e. the types of information to be documented and the way of arranging and presenting this information. Hence, make sure that in the end each use case is specified based on a common reference template.
- ❑ Initially, employ a standard use case template such as the one presented in Tab. 11-4. However, as you gain experience in the application of use case templates, consider adapting the reference template according to the specific needs of your project or organisation.
- ❑ If you do not have the information you need for filling in some slot of the use case template, fill in "TBD" (to be determined) to document that this slot needs to be revisited at a later stage.

Use Case Templates

Requirements Engineering

Tab. 11-5 *Example of the template-based documentation of a use case*

Section			Content
Identifier			UC-4-17
Name			Navigate to destination
Author			Peter Miller, Jane Smith
Version			V.1.1
Change history			V.1.0 P. Miller 13-4-2006 "Main scenario specified" V.1.1 J. Smith 27-5-2006 "Alternative and exception scenarios specified"
Priority			Importance for success of the system "high"; technological risk "high"
Criticality			High
Source			L. White (domain expert for navigation systems)
Responsible stakeholder			J. Smith
Short description			The driver of the car enters the destination. The navigation system guides the driver to the desired destination.
Use case level			User level
Goal(s)			Entry of the destination, automatic navigation to destination
Primary actor			Driver
Other actors			Information server
Precondition			TBD
Postcondition			The driver has achieved his/her goal.
Result			Route to the destination
Main scenario	1		The driver activates the navigation system.
	2		The navigation system determines the current position of the car.
	3		The navigation system asks for the desired destination.
	4		The driver enters the destination using the control panel of the navigation system.
	5		The navigation system displays the map of the target area.
	6		The navigation system asks for the routing options.
	7		The driver selects the desired routing options.
	8		The navigation system calculates the route.
	9		The navigation system informs the driver that the route has been calculated.
	10		The navigation system creates a list of waypoints.
	11		The navigation system directs the driver to the next waypoint.
Alternative scenarios	4a		The driver selects the destination by pointing on a map that the navigation system shows on its display.
		4a1	The driver searches the destination in the electronic maps.
		4a2	The driver marks the destination in the electronic maps.
		4a3	The navigation system identifies the coordinates of the destination.
		4a4	The navigation system displays a detailed map of the destination.
		4a5	The navigation system asks the driver to mark the destination on the detailed map.
		4a6	The driver marks the destination of the navigation.
		4a7	The navigation system identifies the street and house number.
	Proceed with Step 6.		
Exception scenarios	5a		The navigation system cannot find the entered destination.
		5a1	The navigation system informs the driver that the entered destination is not known.
		5a2	The navigation system asks the driver to choose another destination.
Qualities			Q-2-04 (Response time to user inputs) Q-2-06 (Ease of use)
Relationships to other use cases			TBD

Use Case Template (Example)

Name	**UC-8: Search and Replace**
Summary	All occurrences of a search term are replaced with replacement text.
Rationale	While editing a document, many users find that there is text somewhere in the file being edited that needs to be replaced, but searching for it manually by looking through the entire document is time-consuming and ineffective. The search-and-replace function allows the user to find it automatically and replace it with specified text. Sometimes this term is repeated in many places and needs to be replaced. Other times, only the first occurrence should be replaced. The user may also wish to simply find the location of that text without replacing it.
Actors	All users
Preconditions	A document is loaded and being edited.
Basic Course of Events	1. The user indicates that the software is to perform a search-and-replace in the document. 2. The software responds by requesting the search term and the replacement text. 3. The user inputs the search term and replacement text and indicates that all occurrences are to be replaced. 4. The software replaces all occurrences of the search term with the replacement text.
Alternative Paths	1. In step 3, the user indicates that only the first occurrence is to be replaced. In this case, the software finds the first occurrence of the search term in the document being edited and replaces it with the replacement text. The postcondition state is identical, except only the first occurrence is replaced, and the replacement text is highlighted. 2. In step 3, the user indicates that the software is only to search and not replace, and does not specify replacement text. In this case, the software highlights the first occurrence of the search term and the use case ends. 3. The user may decide to abort the search-and-replace operation at any time during steps 1, 2 or 3. In this case, the software returns to the precondition state.
Postconditions	All occurrences of the search term have been replaced with the replacement text.

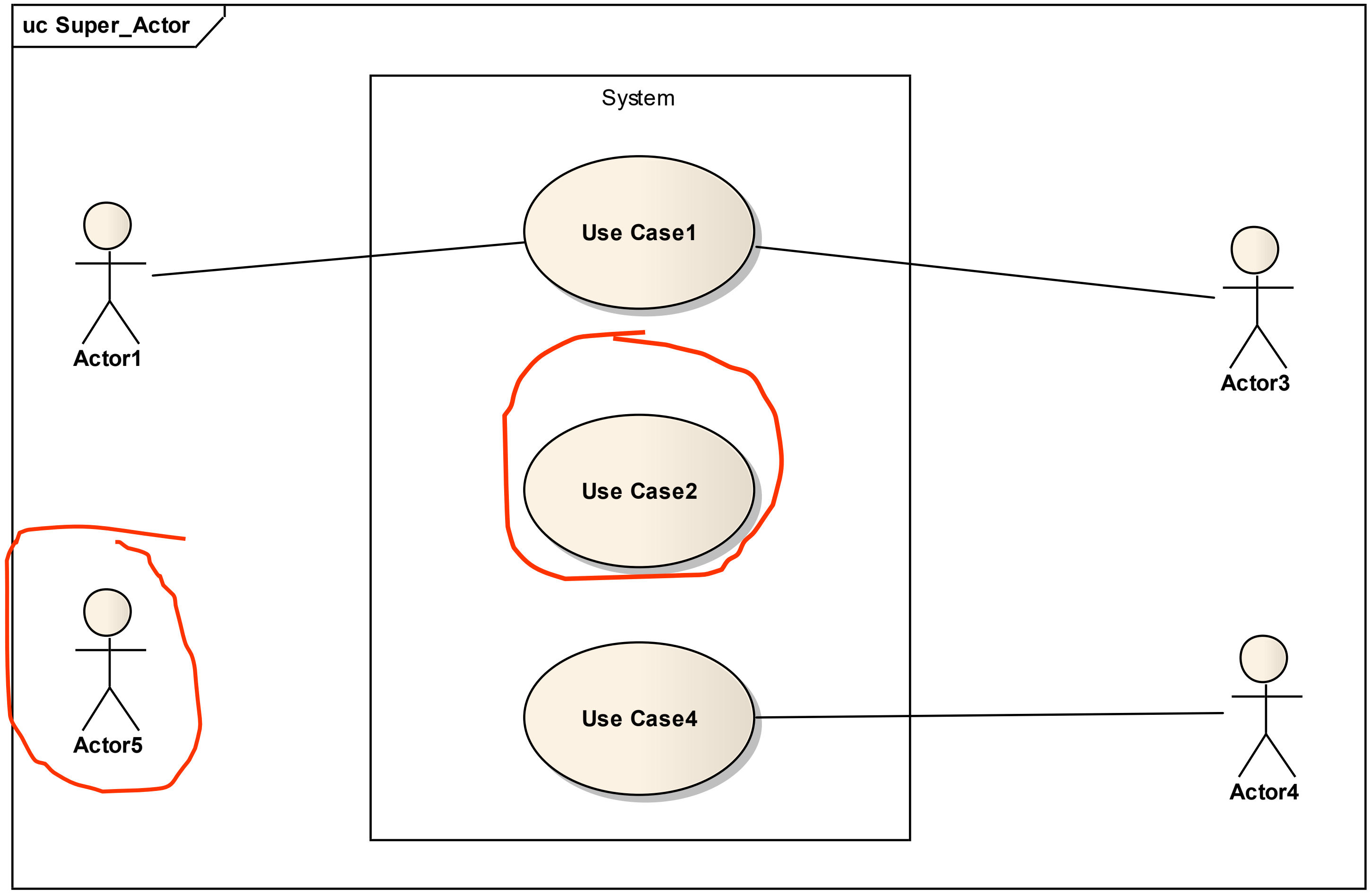
Use Case specification

Use cases are instantiated with scenarios

Name	**UC-8: Search**
Summary	All occurrences of a search term are replaced with replacement text.
Rationale	While editing a document, many users find that there is text somewhere in the fi dited that needs to be replaced, but searching for it manually by looking through the entire ent is time-consuming and ineffective. The search-and-replace function allows the user to f utomatically and replace it with specified text. Sometimes this term is repeated in many place needs to be replaced. Other times, only the first occurrence should be replaced. The user also wish to simply find the location of that text without replacing it.
Users	All users
Preconditions	A document is loaded and being edited.
Basic Course of Events	1. The user indicates that the software is to perform a search-and-replace in the document. 2. The software responds by requesting the search term and the replacement text. 3. The user inputs the search term and replacement text and indicates that all occurrences are to be replaced. 4. The software replaces all occurrences of the search term with the replacement text.
Alternative Paths	1. In step 3, the user indicates that only the first occurrence is to be replaced. In this case, the software finds the first occurrence of the search term in the document being edited and replaces it with the replacement text. The postcondition state is identical, except only the first occurrence is replaced, and the replacement text is highlighted. 2. In step 3, the user indicates that the software is only to search and not replace, and does not specify replacement text. In this case, the software highlights the first occurrence of the search term and the use case ends. 3. The user may decide to abort the search-and-replace operation at any time during steps 1, 2 or 3. In this case, the software returns to the precondition state.
Postconditions	All occurrences of the search term have been replaced with the replacement text.

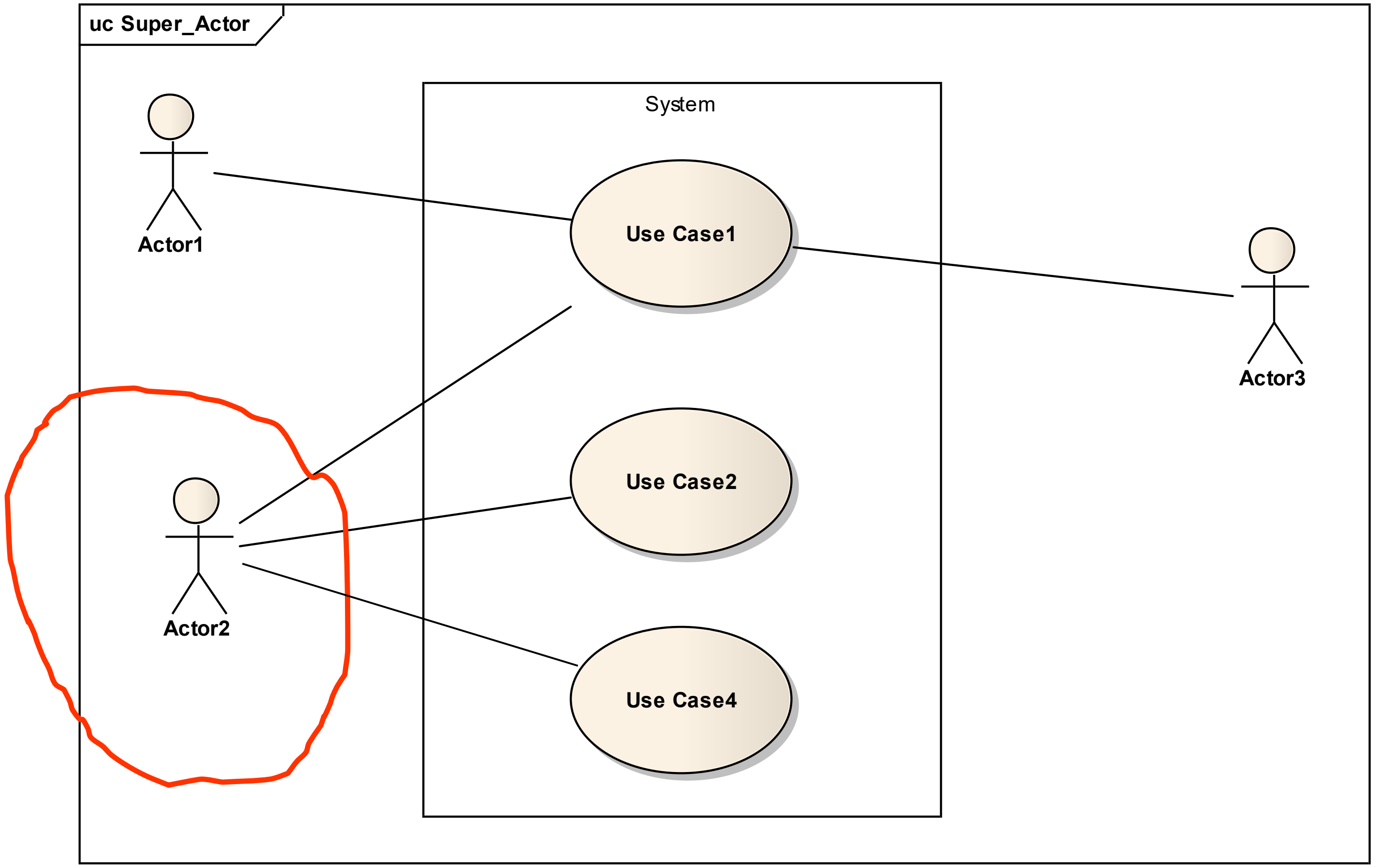
Use Case Anti-Patterns

Actors without Use Cases or Use Cases without Actors.



Use Case Anti-Patterns

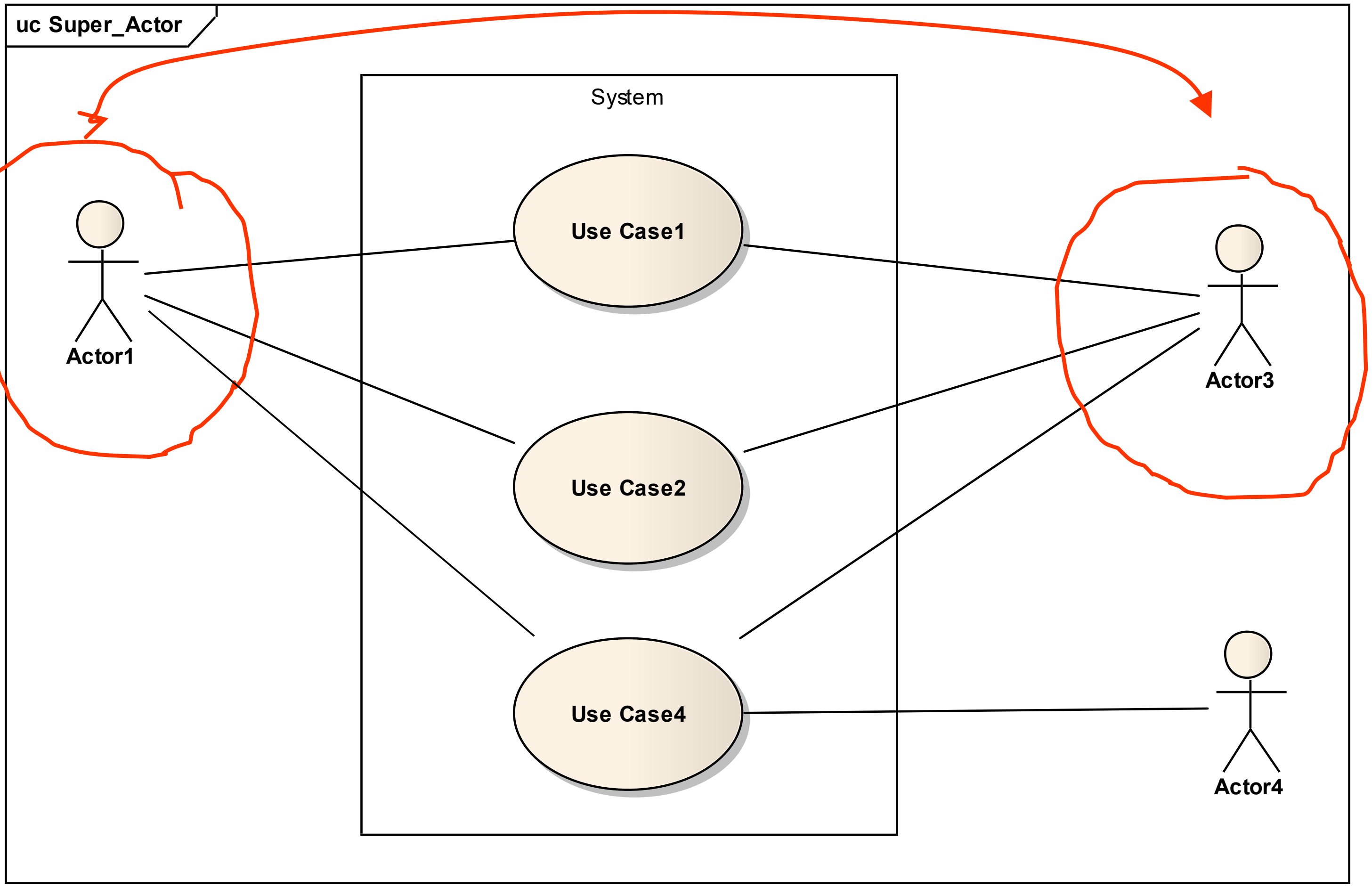
Undefined Actors or actors at a level of abstraction too high (an actor that does it all is a sign of a weak actor's analysis).



Use Case Anti-Patterns

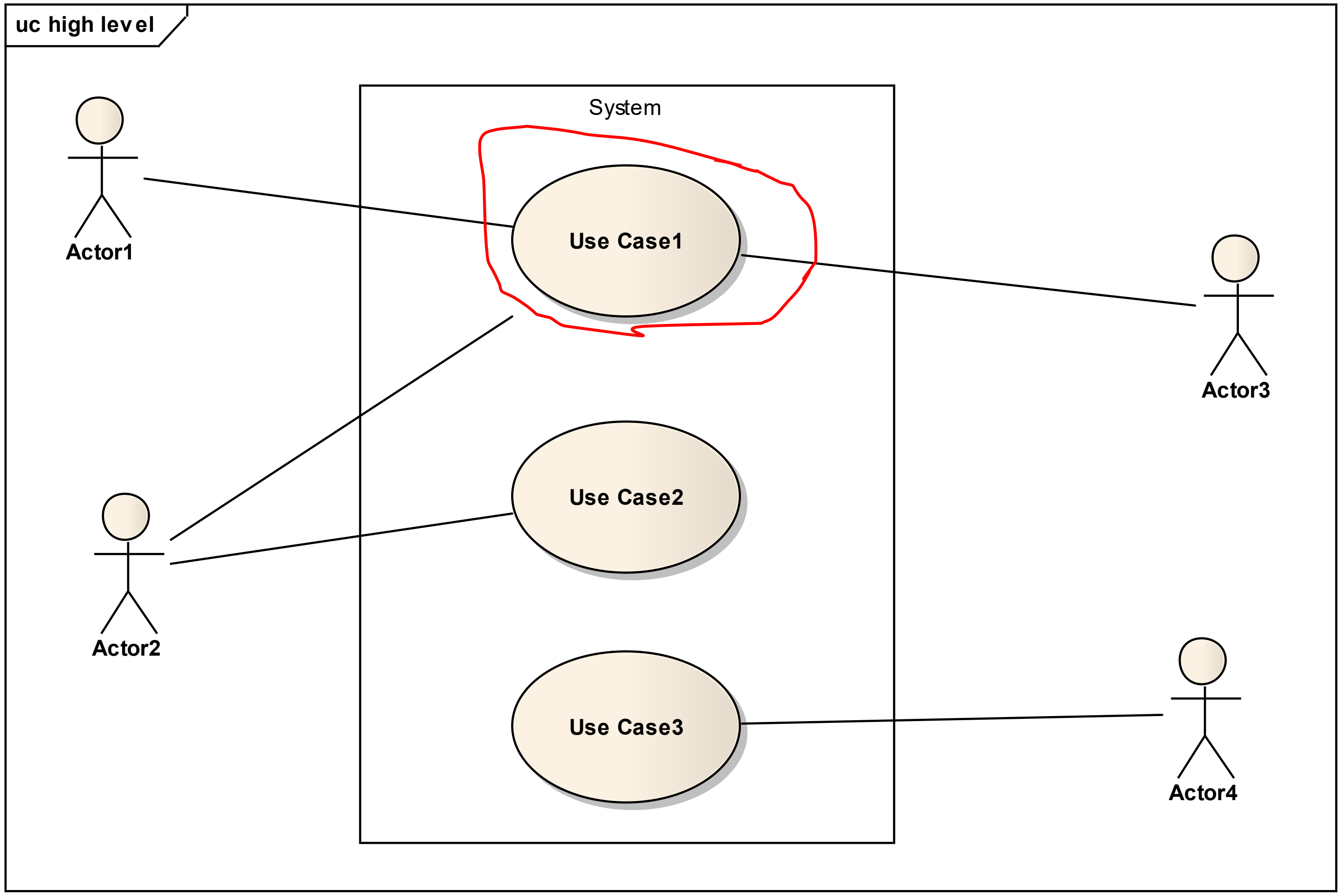
Actors with duplicate interaction patterns

(i.e. Actor 1 ⇔ Actor 3? What's the point?...)



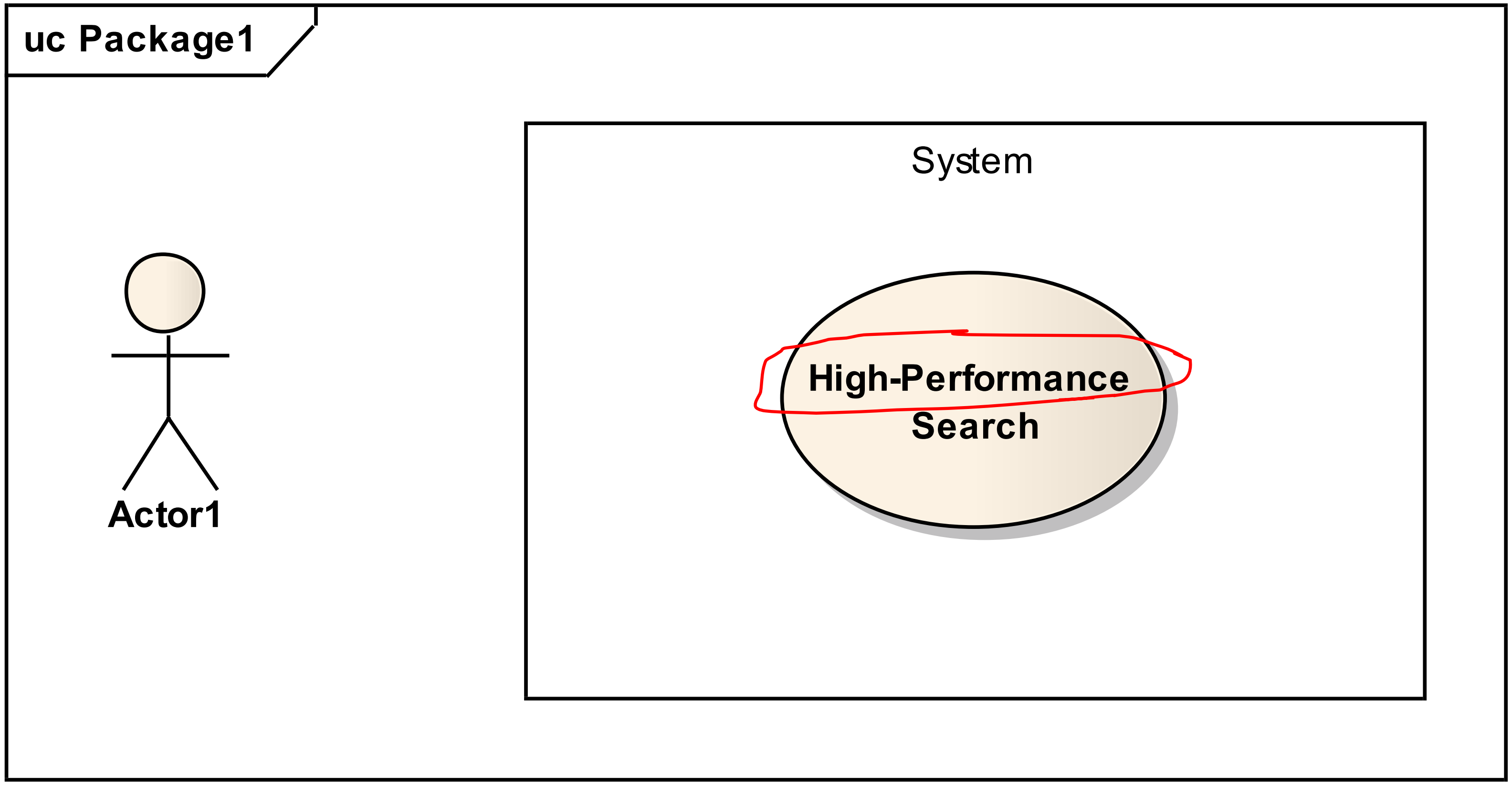
Use Case Anti-Patterns

Too high level Use Cases (a use case that ALL actors execute is a sign of a lazy use case analysis…)?



Use Case Anti-Patterns

"Non-functional" use cases? Makes no sense...



Use Case Anti-Patterns

Don’t confuse use cases with use case scenarios.

“Subway Ticket Vending Machine”

Design #1

- UC1: Buy Ticket

Design #2

- UC1: Select Ticket Type
- UC2: Insert Money
- UC3: Retrieve Ticket
- UC4: Retrieve Change
- UC5: Cancel Purchase

More Use Case Anti-Patterns

1. UC diagrams that do not facilitate communication, analysis, system construction; neglect of guidelines and best practices.
2. Unrefined use cases (UC do not fulfil 1..* FR), no traceability and/or forced traceability.
3. Forced 1:1 correspondence between UC and requirements.
4. Use cases are not end-to-end.
5. UC that fulfil no goals, misinterpreted actor-UC relationships; leads to incomplete and too many UC and artificial flows between UC.
6. Forced primary scenarios (i.e. scenarios artificially describe use cases).
7. Unidentified or forced pre-conditions and post-conditions.
8. Too many actors (no abstraction), too few actors (no roles), too complex actor structures (poor understanding or bad design of the actor roles).
9. UC not oriented toward the fulfilment of the actor's goals; Alternative/exception scenarios not modelled at all, modelled as primary scenarios, modelled without pre- and post-conditions.
10. Use of complex designs (e.g. specialization of actors and UC, «flow», «include», «extend») without first understanding the problem or understanding the modelling construct.
11. Unrefined construction of the system (i.e. functional-oriented construction of the system does not trace back to the UC).
12. Internal (business) processes modelled as flows of UC.