Chapter 11
Documenting Scenarios

In this chapter, we describe different techniques for documenting scenarios and use cases, both using natural language as well as using models. With regard to the documentation of scenarios and use cases using natural language, we present:

- *Narrative scenarios, i.e. scenarios documented in natural language as short narrations*
- *The tabular description of the interaction sequences of scenarios*
- *A reference template for the structured documentation of scenarios*
- *A reference template for the structured documentation of use cases*

With regard to the model-based documentation of scenarios and use cases, we explain:

- *The utilisation of UML sequence diagrams for documenting interaction sequences of scenarios*
- *The utilisation of UML activity diagrams for documenting the control flows of scenarios and use cases*
- *Use case diagrams for documenting relationships between use cases and between use cases and actors*

K. Pohl, *Requirements Engineering*,

11.1 Narrative Scenarios

Documentation in natural language

A narrative scenario documents a sequence of interactions using natural language. It has the form of a short narration. The term "narrative" is rooted in psychology. Narrative psychology is a branch of psychology that deals with the way people work up and communicate their thoughts and experiences by means of stories (see [McAdams et al. 2006]).

Advantages of narrative scenarios

In requirements engineering, narrative scenarios are an important means for supporting the elicitation of knowledge about the system and its context. Due to the narrative form and the use of natural language, narrative scenarios are typically comprehensible to all stakeholders (see e.g. [Holbrook 1990; Erickson 1995]). Within narrative scenarios, information can be communicated at different abstraction levels. For instance, important aspects can be described in a more concrete way and less important aspects in a more abstract way (see Section 10.5). In a narrative scenario, descriptive, exploratory, and explanatory elements, as well as alternative and exceptional steps can be documented together with structural, functional, behavioural, and quality aspects. Example 11-1 presents a fragment of a narrative scenario that contains most of these aspects and elements (underlined in Example 11-1).

Example 11-1: Excerpt from a narrative scenario

The driver assistance system includes a (sub-)system for avoiding rear-end collisions. This system comprises (❶) distance sensors that (❷) permanently check the distance to the vehicle driving ahead (❸) in order to avoid an imminent rear-end collision. (❹) If the system detects that the distance falls below the safety distance yet is still outside the critical range, an acoustic warning signal sounds. (❺) Alternatively, a symbol or message may be displayed on the driver display in the cockpit of the car. If the driver has not reacted to the warning after 2 s and the distance between the two cars still decreases, (❻) the system reduces the speed of the car. (❼) If the distance (in metres) falls below one quarter of the driving speed (in km/h) at any time, the system initiates emergency braking.

Explanation:

(❶) Static/structural aspect
(❷) Functional aspect
(❸) Explanatory aspect
(❹) Behavioural aspect
(❺) Exploratory aspect
(❻) Step of an alternative scenario
(❼) Condition for an exception scenario

Hint 11-1: *Narrative scenarios* !

Use narrative scenarios in order to:

- Elicit information about the context
- Elicit information about goals and requirements

Detail important aspects of narrative scenarios through structured textual scenarios and define, for example, alternative scenarios, exception scenarios, descriptive scenarios, and explanatory scenarios.

11.2 Structured Scenarios

The comprehensibility and readability of scenarios written in natural language can be significantly improved by documenting the interaction steps and/or the context information in a structured manner. The structured documentation of scenarios is described, e.g. in [Rumbaugh et al. 1991; Jacobson et al. 1992; Cockburn 2001].

11.2.1 Structured Documentation of Scenario Steps

Two approaches for the structured textual documentation of interaction sequences of a scenario have proven their worth:

- The enumeration of scenario steps
- The tabular documentation of interaction sequences

Sequential numbering of the interaction steps

Enumerating scenario steps means separating the individual interaction steps of the scenario from each other and numbering the scenario steps sequentially. The numbering of the steps indicates the sequential progression of the interaction steps of the scenario. The number of each step also represents a unique identifier within the scenario. This identifier can be used as a reference to a particular step. Furthermore, each step must state at least the name of the actor who initiates the interaction and a concise description of the interaction. Example 11-2 illustrates the structured documentation of the interaction steps of a scenario. The structured documentation of scenario steps using natural language is further supported by the rules presented in Section 11.4.

Example 11-2: Enumeration of scenario steps

(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.
(5) The navigation system identifies the relevant part of the map.
(6) The navigation system displays the map of the destination area.

(*to be continued*)

Example 11-2 (*continued*)

(7) The navigation system asks for the routing options.
(8) The driver selects the desired routing options.
(9) The navigation system calculates the route.
(10) The navigation system informs the driver that the route has been calculated.
(11) The navigation system creates a list of waypoints.
(12) The navigation system displays the next waypoint of the calculated route.

Tabular documentation of interaction steps

In the case of tabular documentation, the individual scenario steps are documented in separate cells of a table that contains one column for each actor involved in the scenario. The steps of the scenario are described separately, numbered line by line, and entered from top to bottom into the rows of the table according to the progression of the interactions (see the example in Tab. 11-1). Each scenario step is shown in the column corresponding to the actor who initiates the interaction. If necessary, several interactions may be listed in a row to document that these interactions are performed concurrently. In contrast to enumerated documentation of scenario steps, in tabular documentation the initiating actor of each interaction step is directly evident from the table (Example 11-2). Therefore, the actors do not need to be stated in the individual interaction steps of the scenario.

Tab. 11-1 *Tabular documentation of the interaction sequence of a scenario*

Scenario "Navigate to destination"	
Driver	**Navigation system**
1. Activates the navigation system.	
	2. Determines the current position.
	3. Asks for the destination.
4. Enters the destination.	
	5. Identifies the relevant part of the map.
	6. Displays a map of the target area.
	7. Asks for the routing options.
8. Enters the routing options.	
	9. Calculates the route.
	10. Displays that the route has been calculated.
	11. Creates a list of waypoints.
	12. Displays the next waypoint.

! **Hint 11-2:** *Structured documentation of scenarios*

Use the tabular style for the final documentation of main, alternative, and exception scenarios. Number the individual interaction steps and enter them into the column corresponding to the actor who initiates the interaction or who is responsible for the interaction step.

Annotation: However, take care not to use the tabular style at a too early stage. Exploit the benefits of narrative scenarios, first, in order to elicit rich content such as solution-oriented requirements, goals, context information, and further interaction steps. Afterwards, define the interaction sequences of the scenarios more accurately by documenting them in the tabular style. The tabular documentation also simplifies the further use of the scenarios in the development process.

11.2.2 Reference Templates for Scenarios

Documentation of interaction sequences and context information

Reference templates support the structured documentation of the interaction sequence of a scenario (see Section 11.2.1) as well as the context information (see Section 9.2) and the management information (see Section 18.3) of the scenario. Table 11-2 shows a reference template for the structured documentation of a scenario.

Tab. 11-2 *Template for documenting a scenario in natural language*

Scenario Template		
No.	**Section**	**Content/Explanation**
1	Identifier	Unique identifier of the scenario
2	Name	Unique name of the scenario
3	Author	Names of the authors who have worked on the scenario description
4	Version	Current version number of the scenario
5	Change history	List of the changes applied to the scenario including, for each change, the date of the change, the version number, the author, and, if necessary, the reason for and the subject of the change
6	Priority	Indication of the importance of the described scenario according to the prioritisation technique used
7	Criticality	Criticality of the scenario, e.g. for the success of the system
8	Source	Denomination of the source ([stakeholder | document | system]) from which the scenario stems
9	Responsible stakeholder	The stakeholder responsible for the scenario
10	Short description	Concise description of the scenario (approximately 1/4 page)
11	Scenario type	Classification of the scenario based on the scenario types presented in Chapter 10, e.g. context, interaction, or system-internal scenario
12	Goal(s)	Goal(s) that shall be satisfied by executing the scenario including identifiers pointing to the associated goal definitions *Relevant rules* (see Section 11.4): ❑ Name the goal of the scenario explicitly (rule 10) ❑ Focus on the achievement of the goal (rule 11) *Rules for documenting goals* (see Section 8.2): ❑ Document goals concisely (rule 1) ❑ Use the active voice (rule 2) ❑ Document the stakeholder's intention precisely (rule 3) ❑ Decompose high-level goals into more concrete sub-goals (rule 4) ❑ State the additional value of the goal (rule 5) ❑ Document the reasons for introducing a goal (rule 6) ❑ Avoid defining unnecessary restrictions (rule 7)
13	Actors	Indication of the primary actor and other actors *Relevant rules* (see Section 11.4): ❑ Name the actors explicitly (rule 9)
14	Precondition	A list of necessary prerequisites that need to be fulfilled before the execution of the scenario can be initiated
15	Postcondition	A list of conditions that hold after execution of the scenario

Tab. 11-2 (continued)

Scenario Template		
No.	**Section**	**Content/Explanation**
16	Result	Description of the outputs that are created during execution of the scenario
17	Scenario steps	Detailed interaction sequence of the scenario: narrative/sequentially numbered/tabular *Relevant rules* (see Section 11.4): ❑ Use the present tense (rule 1) ❑ Use the active voice (rule 2) ❑ Use the subject–predicate–object (SPO) sentence structure (rule 3) ❑ Avoid modal verbs (rule 4) ❑ Only one interaction per sentence (rule 5) ❑ Number each scenario step (rule 6) ❑ Only one sequence of interactions per scenario (rule 7) ❑ Describe the scenario from the "view from afar" (rule 8) ❑ Explicitly name the actors (rule 9) ❑ Focus on illustrating the satisfaction of the goal (rule 11)
18	Qualities	Cross references to quality requirements
19	Relationships to other scenarios	Relationships of the scenario to other scenarios
20	Supplementary information	Additional information regarding this scenario

Specific and general attributes

The scenario attributes defined in this template comprise:[1]

- ❑ Attributes for the identification of scenarios (rows 1–2)
- ❑ Management attributes (rows 3–7)
- ❑ Attributes for documenting the reference to the system context (rows 8–9)
- ❑ Attributes for documenting general content aspects (rows 10 and 20)
- ❑ Specific scenario attributes (rows 11–19), i.e. the scenario type (row 11), the association to goals (row 12), the actors of the scenario (row 13), pre- and post-conditions (rows 14–15), the result of the scenario (row 16), the scenario steps (row 17), the reference to quality requirements (row 18) as well as the relationships to other scenarios (row 19)[2]

Business- or project-specific adaptation

Table 11-3 illustrates the documentation of a scenario (including the associated context information) using the reference template shown in Tab. 11-2.

Prior to documenting scenarios using the reference template, it may be necessary to adapt the template. For instance, the stakeholders may have to define additional slots (i.e. attributes) due to specific needs of the organisation or project such as project size, project risk, liability laws, or process standards. Additional attributes that may be relevant for documenting scenarios can be found in Section 18.2.

[1] See also the description of requirements attributes in Section 18.3.

[2] The attributes "goal(s)" (row 12), "qualities" (row 18), and "relationships of the scenario to other scenarios" (row 19) refine the general attribute "cross references" described in Section 18.3.4.

Tab. 11-3 Example of the template-based documentation of a scenario

Section	Content	
Identifier	S-2-34	
Name	Navigate to destination	
Author	Peter Miller	
Version	V.1.1	
Change history	V.1.0 12.01.2006 Dan Smith V.1.1 14.02.2006 Peter Miller	
Priority	Medium	
Criticality	High	
Source	William Garland (domain expert for navigation systems)	
Responsible stakeholder	Dan Smith	
Short description	After the driver has entered the destination, the navigation system calculates the route and directs the driver to the desired destination.	
Scenario type	Interaction scenario (type B scenario)	
Goal(s)	G-2-17: Automatic navigation to the destination	
Actors	Driver, navigation system	
Precondition	The navigation system is able to receive GPS signals from at least three GPS satellites.	
Postcondition	The driver has arrived at the desired destination.	
Result	Step-by-step route to the destination	
Scenario steps	1	Driver activates the navigation system.
	2	Navigation system determines the current position of the car.
	3	Navigation system asks for the desired destination.
	4	Driver enters the destination.
	5	Navigation system identifies the relevant part of the map.
	6	Navigation system displays the map of the target area.
	7	Navigation system asks for the routing options.
	8	Driver selects the routing options.
	9	Navigation system calculates the route.
	10	Navigation system informs the driver that the route has been calculated.
	11	Navigation system creates a list of waypoints.
	12	Navigation system directs the driver to the next waypoint.
Qualities	Q-7-42: The calculation of the route shall take at most 3.5 seconds.	
Relationships to other scenarios	S-2-24: Comfortable entry of the destination S-2-54: Navigate around traffic congestion S-3-12: Destination not contained in data base	
Supplementary information	The competing system SX-23-44 realises a similar scenario.	

Hint 11-3: *Systematic documentation of scenarios*

- Avoid filling in all attributes of a scenario right away.
- Elicit an initial set of basic scenarios, i.e. fill in only the basic attributes identifier, name, source, responsible stakeholder, short description, and scenario type.
- Relate each scenario with the goals that are satisfied by the scenario.
- Validate that all relevant scenarios have been elicited, e.g. by checking that each goal defined for the system is satisfied by some scenario.
- Document the missing scenarios as described in the second step.
- Complete the basic scenarios by filling in the remaining slots of the scenario template.

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

11.3 A Reference Template for Use Cases

Reference templates for the documentation of use cases

For documenting use cases, a reference template can be used in a similar way as for the documentation of individual scenarios. Reference templates for use cases are based on expert knowledge about the structured documentation of use cases in natural language. A reference template defines the different types of information that need to be documented for each use case. Furthermore, a reference template defines how to arrange or structure this information. In the following, we explain a comprehensive reference template for the documentation of use cases. Other examples of reference templates for use cases can be found, amongst others, in [Kulak and Guiney 2003; Cockburn 2001; Armour and Miller 2001; Larman 2004; Bittner and Spence 2003; Halmans and Pohl 2003].

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.

We suggest documenting use cases using the reference template presented in Tab. 11-4. This template is based on the template presented in [Halmans and Pohl 2003]). The template shown in Tab. 11-4 contains the following attributes:[3]

- Attributes for the unique identification of use cases (rows 1–2)
- Management attributes (rows 3–7)
- Attributes for documenting the reference to the context (rows 8–9)
- A short description of the use case (row 10)

Tab. 11-4 *Reference template for documenting use cases*

Use Case Template		
No.	**Section**	**Content/Explanation**
1	Identifier	Unique identifier of the use case
2	Name	Unique name for the use case
3	Author	Name of the authors who have worked on the use case description
4	Version	Current version number of the use case
5	Change history	List of the changes applied to the use case including, for each change, the date of the change, the version number, the author, and, if necessary, the reason for and the subject of the change
6	Priority	Importance of the use case according to the prioritisation technique used
7	Criticality	Criticality of the use case for the success of the system
8	Source	Denomination of the source ([stakeholder | document | system]) from which the use case stems
9	Responsible stakeholder	The stakeholder responsible for the use case
10	Short description	Concise description of the use case (approximately 1/4 page)
11	Use case level	Characterisation of the current level of detail of the use case, e.g. through the differentiation between: overview/user level/function group level
12	Goal(s)	Goal(s) that shall be satisfied by executing the use case scenarios including identifiers that point to the associated goal definitions *Relevant rules* (see Section 11.4): ❑ Name the goal of the scenario explicitly (rule 10) ❑ Focus on the achievement of the goal (rule 11) *Rules for documenting goals* (see Section 8.2): ❑ Document goals concisely (rule 1) ❑ Use the active voice (rule 2) ❑ Document the stakeholder's intention precisely (rule 3) ❑ Decompose high-level goals into more concrete sub-goals (rule 4) ❑ State the additional value of the goal (rule 5) ❑ Document the reasons for introducing a goal (rule 6) ❑ Avoid defining unnecessary restrictions (rule 7)
13	Primary actor	Indication of the primary actor (the actor who benefits the most from the execution of the use case). Typically, the primary actor initiates the execution of the use case. *Relevant rules* (see Section 11.4): ❑ Name the actors explicitly (rule 9)

[3] See also the description of requirements attributes in Section 18.3.

Tab. 11-4 *(continued)*

Use Case Template		
No.	**Section**	**Content/Explanation**
14	Other actors	Determination of all other actors involved in the use case *Relevant rules* (see Section 11.4): ❑ Name the actors explicitly (rule 9)
15	Precondition	A list of necessary prerequisites that need to be fulfilled before execution of the use case can be initiated
16	Postcondition	A list of conditions that hold after execution of the use case
17	Result	Description of the outputs that are created during execution of the use case
18	Main scenario	Description of the main scenario of a use case *Relevant rules* (see Section 11.4): ❑ Use the present tense (rule 1) ❑ Use the active voice (rule 2) ❑ Use the subject–predicate–object (SPO) sentence structure (rule 3) ❑ Avoid modal verbs (rule 4) ❑ Only one interaction per sentence (rule 5) ❑ Number each scenario step (rule 6) ❑ Only one sequence of interactions per scenario (rule 7) ❑ Describe the scenario from the "view from afar" (rule 8) ❑ Explicitly name the actors (rule 9) ❑ Focus on illustrating the satisfaction of the goal (rule 11)
19	Alternative scenarios	Description of alternative scenarios of the use case *Relevant rules* (see Section 11.4): ❑ see slot 18 – "Main scenario"
20	Exception scenarios	Description of exception scenarios of the use case *Relevant rules* (see Section 11.4): ❑ see slot 18 – "Main scenario"
21	Qualities	Cross references to quality requirements
22	Relationships to other use cases	Short description of the relationships of the use case (e.g. extend, include, or generalisation relationships) to other use cases
23	Supplementary information	Additional information for this use case

- ❑ Specific use case attributes (rows 11–22), i.e. the use case level (row 11), the goals of the use case or references to the definitions of the goals (row 12), the actors of the use case (rows 13 and 14), pre- and postconditions (rows 15–16), the result of the use case (row 17), the main scenario (row 18), alternative and exception scenarios (rows 19–20), references to quality requirements (row 21) as well as relationships to other use cases (row 22)[4]
- ❑ An attribute for documenting additional, relevant information about a use case (row 23)

Table 11-5 shows the documentation of the use case "navigate to destination" using the reference template presented in Tab. 11-4.

[4] The attributes "goal(s)" (row 12), "qualities" (row 21), and "relationships to other use cases" (row 22) refine the general attribute "cross references" described in Section 18.3.4.

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		

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.

11.4 Eleven Rules for Documenting Scenarios

Adhering to simple rules when documenting scenarios significantly increases the quality of the resulting scenarios. Thereby, the benefit of the scenarios for requirements engineering and the entire development process is increased. In the following, we present 11 rules that have proven useful in supporting the documentation of scenarios in natural language.[5] We subdivide the rules into:

- Rules concerning the language and grammar of scenarios
- Rules concerning the structure of scenarios
- Rules concerning the content of scenarios

Hint 11-6 at the end of this section provides a brief summary of the 11 rules for documenting scenarios.

Language and Grammar Rules

Rule 1: present tense

Rule 1: Use the present tense when documenting scenarios. Scenarios document exemplary courses of interactions. The description of the interactions is more vivid and comprehensible if the present tense is used.

[5] The 11 rules are based on our experiences with regard to the usage of scenarios and on the rules mentioned in [Cockburn 2001] and [Alexander and Maiden 2004].

Example 11-3: Scenarios in the present tense

The user entered his user name and password into the system. The system checked the correctness of the entered data.

Improved definition:

The user enters his user name and password into the system. The system checks the correctness of the entered data.

Rule 2: active voice

Rule 2: Use the active voice when documenting scenarios. Using the active voice makes clear who acts or initiates an interaction. Documenting scenario steps using the active voice guarantees that each scenario step clearly and unambiguously states the responsible actor. Thereby, the documented scenario assigns each action to the responsible person or system and hence avoids ambiguity concerning the initiator of an interaction.

Example 11-4: Scenarios in the active voice

The user name and password are entered and validated.

Improved definition:

The user enters his user name and password into the system. The system validates the correctness of the entered data.

Rule 3: SPO sentence structure

Rule 3: Use the subject–predicate–object (SPO) sentence structure. This simple sentence structure eases reading the scenario and supports focussing on the essential parts of the scenario.

Example 11-5: SPO sentence structure in scenarios

By means of the user database, the system validates the user data.

Improved definition:

The system validates the user data by means of the user database.

Rule 4: no modal verbs

Rule 4: Avoid modal verbs. Modal verbs express that an interaction is possible rather than that it happens. This contradicts the exemplary character of scenarios and should hence be avoided.

Example 11-6: Avoid modal verbs

The system should check the user data.

Improved definition:

The system checks the user data.

Structure Rules

Rule 5: each interaction in a separate sentence

Rule 5: Clearly separate each interaction from other interactions. Avoid describing several interactions within the same sentence. By adhering to this rule, the individual interactions can be numbered, identified, and referenced more easily.

Example 11-7: Each interaction in a separate sentence

The user submits a search query to the online shop, selects an item from the list of search results, and adds the item to the shopping cart.

Improved definition:

See Example 11-8

Rule 6: unique numbering

Rule 6: Number each scenario step. The unique numbering of the individual scenario steps improves the clarity of the scenario and facilitates referencing the scenario steps.

Example 11-8: Numbering scenario steps

(1) The user submits a query to the online shop.
(2) The system displays a list of search results.
(3) The user selects an item from the list.
(4) The user adds the item to his shopping cart.
(5) The user iterates steps 1 to 4 until he has finished shopping.

Content Rules

Rule 7: one interaction sequence per scenario

Rule 7: Only one interaction sequence per scenario. Describe only a single sequence of interactions in each scenario. Avoid merging a main scenario or an alternative scenario with other alternative or exception scenarios. Alternative and exceptional sequences should be described in separate scenarios.

Example 11-9: Only one sequence of interactions per scenario

[...]

(10) The user enters his user data into the system.
(11) The user data is correct → Continue with step (41).
(***incorrect user data***)
(11) System displays "incorrect user data, please retry".
(12) System asks the user to enter his data.
(13) The user enters his user data into the system.
(14) The user data is correct → Continue with step (41).
(***third attempt***)
(21) Perform steps (3) to (5).
(22) The user data is correct → Continue with step (41).
(***incorrect user data entered for the third time, therefore abort***)

(to be continued)

Example 11-9 *(continued)*

(31) System displays "incorrect user data – transaction cancelled".
(32) System returns credit card.
(***correct user data***)
(41) System displays "user data correct".
(42) System asks for the amount to be withdrawn.
[...]

Improved definition:

Main scenario:

[...]
(10) The user enters his user data into the system.
(11) System displays "user data correct".
(12) System asks the user for the amount of money to be withdrawn.
[...]

Alternative scenario: If step (10) is unsuccessful:

(11a.1) System displays "incorrect user data, please retry".
(11a.2) System asks the user for entering his data.
(11a.3) The user enters his user data into the system.

Exception scenario: In step (10), if user data entered incorrectly three times:

(11b.1) System displays "incorrect user data – transaction cancelled".
(11b.2) System returns credit card.

Rule 8: "view from afar"

Rule 8: Describe scenarios from the view of an outsider ("view from afar"). Avoid the description of details that are not necessary for a scenario. For example, abstain from detailed description of internal system workflows (i.e. type A scenarios, see Section 10.6), if the main task is to define the interactions between the user and the system.

Example 11-10: Right perspective for interaction scenarios

(1) The system receives the user name and password of the user.
(2) The system encrypts the user data.
(3) The system logs on to the user server.
(4) The system transfers the user data to the user server.
(5) The user server decrypts the user data.
(6) The user server checks the user data by means of the user database.
(7) The user server transmits that the user data is correct.
(8) The system logs off from the user server.
(9) The system informs the user that the login was successful.

Better wording:

(1) The user logs on with his user data.
(2) The system checks the data.
(3) The system informs the user about the successful login.

Rule 9: explicit actors

Rule 9: Explicitly name the actors involved. The actors involved in a scenario should be named explicitly in the scenario description. In this way, misunderstandings regarding the acting persons or systems are avoided. Sentences in the active voice support the explicit naming of actors in a scenario (see rule 2). The requirements engineers should also make clear which actor initiates the interaction and which actor is affected by the interaction. Adhering to rule 3 supports a clear distinction between the initiating and the affected actors in a scenario.

Example 11-11: Explicit actors

(1) The user logs on to the system.
(2) The system reports that there is no connection to the network.
(3) The system restarts.
(4) The connection to the network is re-established.
(5) The user logs on to the system.

Improved definition:

(1) The user logs on to the system.
(2) The system reports that there is no connection to the network.
(3) The user reboots the system.
(4) The system establishes the connection to the network.
(5) The user logs on to the system.

Rule 10: state the goal

Rule 10: Explicitly state the goal of the scenario. Scenarios describe the satisfaction or failure to satisfy a goal. Document for each scenario, which goal or goals are satisfied by executing the scenario (see Example 11-12).

Example 11-12: State the goal explicitly and focus on illustrating how the goal is satisfied

(1) An unauthorised user boots the system.
(2) The system displays status reports about the boot procedure.
(3) The system displays the successful boot on the screen.
(4) The system asks the user to enter his user name and password.
(5) The unauthorised user enters different, randomly chosen user names and passwords.
(6) After five unsuccessful login attempts, the system locks the login functionality for 30 min.

Improved definition:

Goal of the scenario: "Protecting the system against unauthorised access"

(1) An unauthorised user tries to log on to the system with a randomly chosen user name and password.
(2) After five unsuccessful login attempts, the system locks the login functionality for 30 min.

Rule 11: focus on the satisfaction of the goal

Rule 11: Focus on illustrating how the goal is satisfied by the scenario. Document how the scenario satisfies the goal that is related to this scenario (see rule 10) or fails to satisfy the goal. Avoid documenting unnecessary information, i.e. information that does not contribute to illustrating the satisfaction of the goal (see Example 11-12).

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.

11.5 Sequence Diagrams

Documenting requirements using models has several advantages over documenting requirements in natural language (see Section 20.1). A popular language for the model-based documentation of scenarios is the message sequence chart (MSC) language (see [ITU 1996; ITU 1998; ITU 1999]). Although MSCs were developed for the telecommunication domain, they are well suited for documenting different kinds of interactions such as the interactions between a system and its actors.

UML sequence diagrams

The Unified Modeling Language (UML) supports the documentation of sequences of interactions by means of sequence diagrams. UML sequence diagrams are based on the MSC language. Figure 11-1 shows the essential modelling constructs of UML sequence diagrams. Detailed information about the modelling constructs of sequence diagrams can be found in the UML specification [OMG 2009b] and in [Rumbaugh et al. 2005].

Modelling constructs of UML sequence diagrams

A sequence diagram documents a sequence of message exchanges between a set of roles (see e.g. [Rumbaugh et al. 2005]). A role in a UML sequence diagram can represent the system to be developed, a part of the system, or an external actor (i.e. a person or system interacting with the system to be developed). Each role is shown in the diagram as a lifeline, i.e. a vertical line representing the existence of the role over a period of time. A bar on the lifeline of a role indicates a period of time during which the role is active. A role is active, for instance, during the processing of a message received from another role. The activation of a role is shown in a sequence

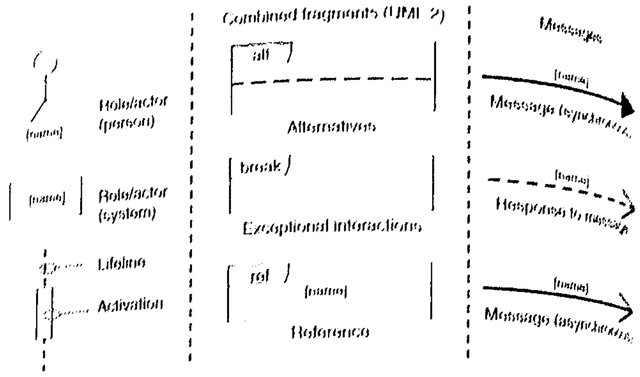

Fig. 11-1 *Essential modelling constructs of sequence diagrams*

diagram mainly for roles that represent technical entities such as (technical) systems or components.

Messages and interactions

Messages are modelled as horizontal arrows between lifelines. A message from one lifeline to another represents an interaction between the corresponding roles. UML differentiates between different types of messages, represented by different arrow styles (see [OMG 2009b]). In the case of a synchronous message (see Fig. 11-1), the sender of the message waits until he receives the response to this message. In contrast, in the case of an asynchronous message (see Fig. 11-1) the sender continues processing immediately after sending the message, i.e. the sender does not wait for the response.

Example

In the sequence diagram shown in Fig. 11-2, two roles are shown: the "driver" and the "navigationSystem" of the car. Since the role "driver" is occupied by a person and the role "navigationSystem" is occupied by a system, two different symbols are used for the two roles in the sequence diagram (corresponding to the notation shown in Fig. 11-1). The rectangular frame of the sequence diagram delimits the scenario. The name of the scenario is stated in the tab in the upper left corner of the frame.

Combined fragments

Combined fragments[6] support the grouping of messages and the assignment of specific semantics to each group. In the following, we briefly explain the combined fragments "ref", "alt", and "break".

Reducing complexity using the "ref" fragment

The combined fragment "ref" refers to the interactions documented in another sequence diagram. The combined fragment "ref" means that the messages documented in the referenced sequence diagram are included in the sequence diagram

[6] The specification of the Unified Modeling Language (UML) has been extended in version 2 with additional modelling constructs for sequence diagrams such as combined fragments [Rumbaugh et al. 2005].

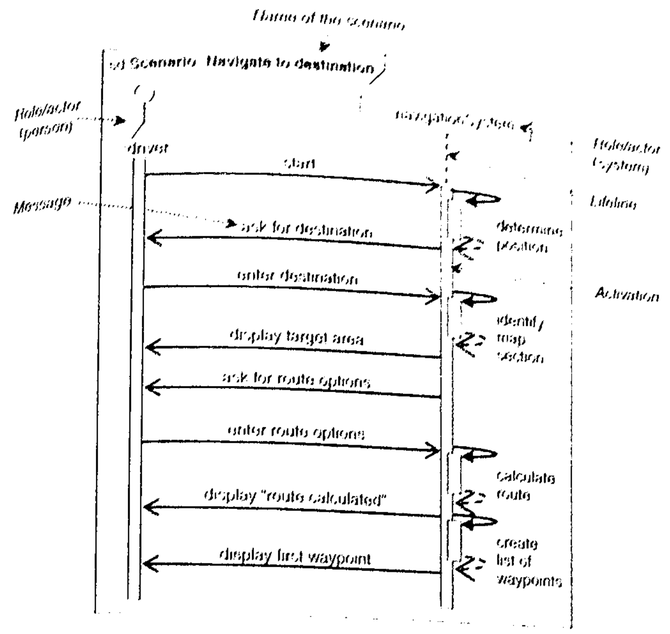

Fig. 11-2 *Documentation of a scenario in a sequence diagram*

containing the combined fragment "ref" at the position of the combined fragment. This type of combined fragment allows abstraction from the sequence of messages documented in the referenced sequence diagram and thus reduction of the complexity of the referring sequence diagram.

The use of the combined fragments "break" and "alt" is illustrated in Fig. 11-3. The combined fragment "break" documents an exception scenario (see Section 10.7). The combined fragment "break" shown in Fig. 11-3 contains two messages: "position coordinates not available" and "route calculation currently not possible", which are sent only if the condition "GPS position coordinates = not available" evaluates to true. If this is the case, the execution of the scenario is aborted after the interactions documented in the "break" fragment have been executed.

Exception scenario

Alternative scenario

The combined fragment "alt" documents an alternative scenario (see Section 10.7) that replaces the interaction step "state destination" in the main scenario. The alternative scenario is executed if voice entry is possible or desired, respectively. If voice entry is not possible or not desired, the destination can also be entered using the control panel. The message "key in destination", together with the other messages documented in the normal course of interactions, defines the alternative scenario.

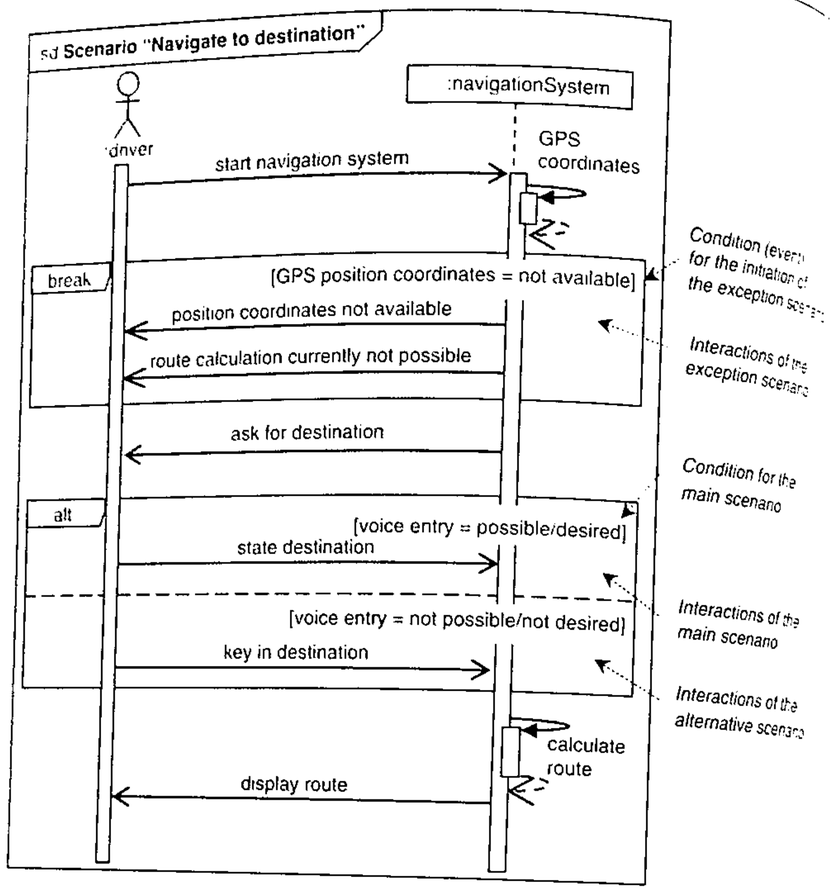

Fig. 11-3 *Alternative and exception scenarios in sequence diagrams*

Hint 11-7: *Documenting scenarios using sequence diagrams*

- Use sequence diagrams for the structured documentation of already well-understood main, alternative, and exception scenarios.
- Use characteristic names for the alternative and exception scenarios.
- Also in the case of sequence diagrams, focus on interactions that contribute to the satisfaction of the goal (or the goals); see rule 11 in Section 11.4.
- Avoid documenting interactions at a too detailed (e.g. algorithmic) level. Rather, take a view from afar, as recommended by rule 8 in Section 11.4.
- If the sequence diagram becomes too complex, use the combined fragment "ref" to divide the sequence diagram into multiple diagrams.
- Avoid extensive use of combined fragments such as "alt" or "break" as this violates the rule "Only one sequence of interactions per scenario" (rule 7 in Section 11.4).
- Deliberate use of comments eases the reading and understanding of sequence diagrams.

11.6 Activity Diagrams

Control flow of several scenarios

While UML sequence diagrams emphasise the sequence of interactions between the system and a set of actors over time, UML activity diagrams allow emphasis of the control flow between multiple scenarios such as the scenarios of a use case. In the following, we sketch the documentation of the control flow of scenarios in UML activity diagrams.

Control flow in the activity diagram

The main focus of an activity diagram is the control flow, i.e. the activities of the different actors and the possible orders of these activities. Figure 11-4 shows the most important modelling constructs of UML activity diagrams. More details on the modelling constructs can be found in [OMG 2009b; Rumbaugh et al. 2005].

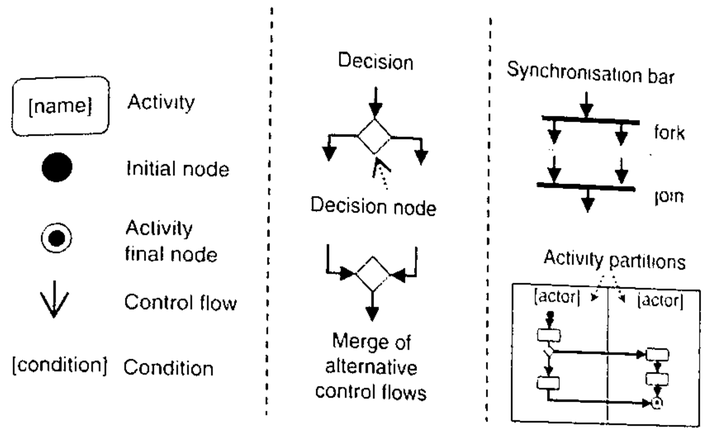

Fig. 11-4 *Important modelling constructs of UML activity diagrams*

Activity nodes

Activity diagrams are control flow graphs. The nodes of an activity diagram represent the execution of activities. The edges represent the flow of control from one activity to another. Activity diagrams have two special activity nodes with a predefined semantics: the initial node and the activity final node. The initial node represents an event that initiates the execution of the activity diagram. Activity final nodes represent the termination of the activity diagram.

Control flows and responsibilities

The representation of alternative control flows in activity diagrams is facilitated through decision nodes. At a decision node, the conditions for choosing one of the alternative control flows are annotated. Synchronisation bars represent concurrent control flows in activity diagrams. By using activity partitions, responsibilities of the system and the different actors for performing actions can be documented.

Main, alternative, and exception scenarios in activity diagrams

Activity diagrams are especially well suited for documenting the interrelations and execution conditions of main, alternative, and exception scenarios. Decision nodes represent the branching of the control flow between the main scenario and

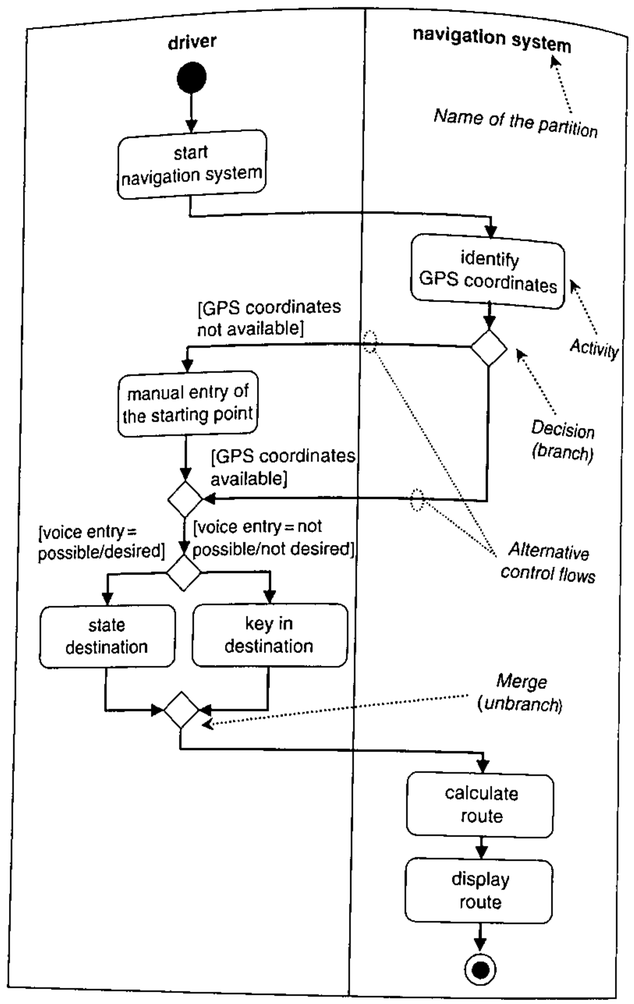

Fig. 11-5 *Documentation of the control flow of scenarios*

Example with two activity partitions

the alternative and exception scenarios. Figure 11-5 shows the activity diagram for the scenarios documented as a sequence diagram in Fig. 11-3.

The activity diagram shown in Fig. 11-5 is subdivided into two activity partitions. The first partition represents the responsibilities for activities and decisions of the actor "driver". The second partition represents the responsibilities of the navigation system. Alternative threads of control that document the alternative and exception scenarios start at the decision nodes.

Hint 11-8: *Documenting the control flow of several scenarios in an activity diagram* !

- Use activity diagrams in order to document:
  - The control flow between multiple scenarios
  - The relationships between main, alternative, and exception scenarios
- Use activity diagrams especially if there are many alternative and/or exception scenarios, i.e. when many valid threads of execution exist.
- Avoid documenting the control flow of several scenarios in activity diagrams if:
  - The interaction sequence of the main scenario is not yet understood;
  - Alternative and exception scenarios are not sufficiently understood and/or their relationships to the main scenario are still unclear.

11.7 Use Case Diagrams

Relationships between use cases

Use case diagrams were suggested for the first time by [Jacobson et al. 1992] in order to support the analysis and documentation of the functionality of a system by means of simple models. Use case diagrams are not suited for documenting (sequences of) interactions between the system and the actors. However, they are well suited for documenting and visualising the relationships between the different use cases of a system as well as the relationships between the actors and the use cases.

Modelling constructs of use case diagrams

Figure 11-6 shows the most important modelling constructs of use case diagrams that are defined in the UML:

- *Actors (systems or persons)*: Actors represent systems or persons outside the system boundary that interact with the system to be developed. If the actor represents a person, the actor is drawn as a small stick figure (see Fig. 11-6). If the actor

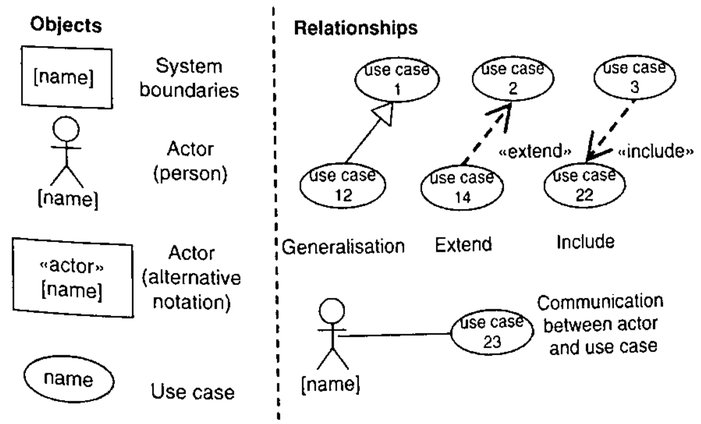

Fig. 11-6 *Important modelling constructs of a use case diagram*

represents a system, it is drawn as a box. Alternatively, a system actor can also be represented by a stick figure that is annotated with the stereotype *system*.

- *Use cases (see also* Definition 10-10*)*: A use case of the system to be developed is represented as an ellipse. The name of the use case is shown inside the ellipse.
- *System boundary*: The system boundary in a use case diagram (visually) delimits the system from its environment. The system boundary is drawn as a rectangle. The use cases for the system are placed within the system boundary. Optionally, the system boundary can be annotated with the name of the system.
- *Relationships between actors and use cases*: A relationship between an actor and a use case expresses that this actor interacts with the system during the execution of the use case. Relationships between actors and use cases are shown in a use case diagram as solid lines.
- *Relationships between use cases*: Use cases can be related to each other by means of the following three types of relationships:
  - *Generalisation relationship*: A generalisation relationship from a use case A to a use case B expresses that use case B is a generalisation of use case A. The specialised use case A inherits the interaction steps contained in the generalising use case B. In addition, the specialised use case A can extend interaction steps inherited from use case B with additional steps, if necessary (see [Rumbaugh et al. 2005]).
  - *Extend relationship*: An "extend" relationship from a use case A to a use case B expresses that the interaction sequence contained in use case A (the extending use case) extends the interaction sequence documented in use case B (the extended use case) at a defined location called the extension point. The extension is modal, i.e. it depends on the occurrence of a defined condition. The extended use case is defined independently of the extending use case and hence represents a meaningful use case event without the extending use case. The extending use case may or may not be meaningful by itself.
  - *Include relationship*: An "include" relationship from a use case A to a use case B expresses that use case A includes the interaction sequence documented in use case B. The "include" relationship is used if two or more use cases have a common part in their sequences of interactions. The common part is extracted into a separate use case and included by the other use cases. The sequence of interactions in an including use case is typically not self-contained and depends on the included use case to be meaningful.

Example illustrating the use of the modelling constructs

Figure 11-7 illustrates the use of some of the modelling constructs for use case diagrams by means of a simplified example. The use case diagram shown in Fig. 11-7 contains a single use case "assist the driver" of the system "driver assistance system". In addition, five actors are shown outside the system boundary. The actors "driver", "customer service employee", and "workshop employee" represent stakeholder roles. The actors "communication system" and "information server" represent external systems that interact with the system to be developed. All actors have a communication relationship to the use case "assist the driver" and hence interact with the system during the execution of this use case.

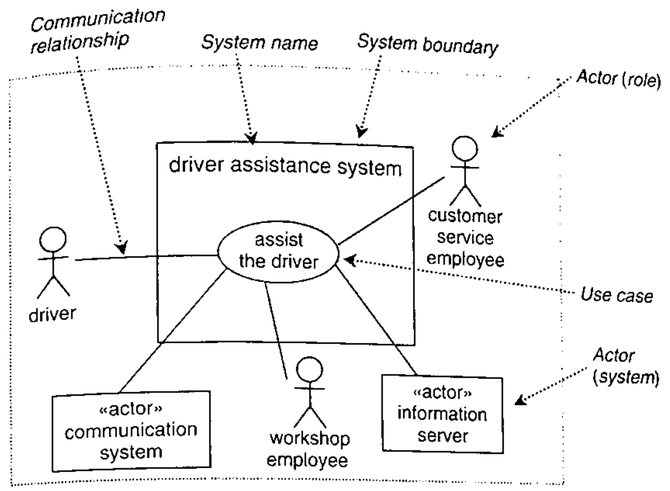

Fig. 11-7 *Modelling constructs of use case diagrams in an example*

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.

Refined example

In the following, we present a more complex example of a use case diagram that can be regarded as a refinement of the example shown in Fig. 11-7. The use case "assist the driver" is refined into the use cases "navigate to destination", "communicate externally", "configure car", and "support maintenance". The refined use cases are shown in Fig. 11-8. The relationships among these use cases are marked with the numbers ❶ to ❸ in Fig. 11-8.

These relationships can be explained as follows:

Include

❶ Each of the use cases "support maintenance", "communicate externally", and "configure car" is related to the use case "authenticate user" by means of an "include" relationship. This relationship expresses that the interaction steps documented in the use case "authenticate user" are also contained in each of the other three use cases.

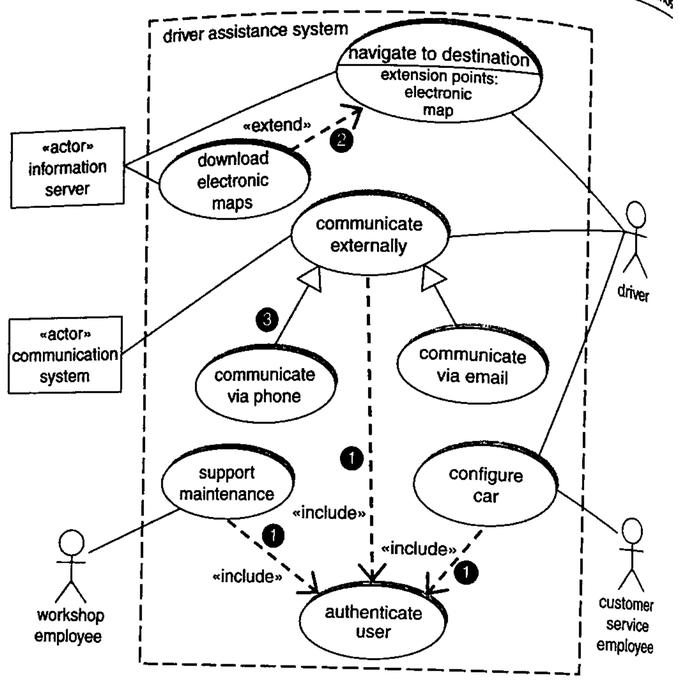

Fig. 11-8 *Refined use case diagram of the driver assistance system*

Extend

❷ The "extend" relationship between the use case "download electronic maps" and the use case "navigate to destination" expresses that the interaction steps documented in the former use case are part of the execution of the use case "navigate to destination" (when a defined event such as "required maps not available" occurs). The extension point "electronic map" shown inside the use case "navigate to destination" names the position in the use case at which the additional interaction steps for the download of electronic maps are executed.

Generalisation

❸ The generalisation relationship shown in Fig. 11-8 documents that the two use cases "communicate via phone" and "communicate via email" inherit the interaction steps documented in the generalising use case. The inherited interaction steps can be adapted, if necessary, in the specialised use cases and extended with additional steps.

Generalisation relationships between actors

The UML also provides a generalisation relationship among actors. A generalisation relationship from an actor A to an actor B expresses that actor A (the specialised actor) inherits the roles and relationships to use cases from actor B (the generalising actor; see e.g. [Rumbaugh et al. 2005]). In Fig. 11-8, the actors "workshop employee" and "customer service employee" could be generalised to an actor "employee". Then, the common aspects of the actors "workshop employee" and "customer service employee" would be assigned to the generalising actor "employee".

Hint 11-10: *Documentation of scenarios and use cases* !

We recommend documenting use cases and scenarios using a combination of natural language documentation and model-based documantion:

- Document context information as well as the relationships of the main scenario to the alternative and exception scenarios by means of use case templates (see Section 11.3).
- Document interaction sequences of a use case within the use case templates using sequence diagrams, if possible (see Section 11.5). If sequence diagrams are not suited due to the stakeholders involved, use the tabular style for documenting the interaction sequences (see Section 11.2.1).
- Document relationships between well-understood main, alternative, and exception scenarios in activity diagrams (see Section 11.6). If this is not a viable option due to the stakeholders involved, use the tabular style instead (see Section 11.2.1).
- Employ use case diagrams in order to provide an overview of the relationships among use cases and the relationships between actors and use cases.

11.8 Use of the Different Scenario Types in the Requirements Engineering Process

In this section we evaluate the suitability of the different scenario types and the different representation formats of scenarios for two kinds of usage:

- *During the process*: Scenarios can be used during the requirements engineering process to support or even drive the elicitation, documentation, negotiation, and validation of requirements.
- *In the specification*: Scenarios can be included in the final requirements specification that is used as the basis for further development activities.

Table 11-6 shows the evaluation of each scenario type (see Chapter 10) and each representation format for these two kinds of usage. The evaluations ("–" not suited, "★" less suited, "★★" suited, "★★★" well suited) should be understood as rough estimations, since the suitability of a specific scenario type or representation format may differ according to the type of project and the specific situation in the requirements engineering process.

Tab. 11-6 *Suitability of the different scenario types and representation formats*

	Scenario type/ Representation format	Suitable for...	
		Requirements engineering process	Requirements specification
Content	Narrative scenarios	***	–
	Current-state scenario	***	*
	Desired-state scenario	***	***
	Positive scenarios	***	***
	Negative scenarios	***	***
	Misuse scenarios	***	***
	Descriptive scenarios	**	***
	Exploratory scenarios	***	–
	Explanatory scenarios	***	*
	Instance scenarios	**	*
	Type scenarios	*	***
	Mixed scenarios	***	**
	System-internal scenarios[7]	*	**
	Interaction scenarios	**	***
	Context scenarios	***	***
	Main scenarios	***	***
	Alternative scenarios	**	***
	Exception scenarios	**	***
	Use cases	***	***
Representation format	Textual (structured) scenarios	***	–
	Template-based scenarios and use cases	**	***
	Sequence diagrams	**	***
	Activity diagrams	–	**
	Use case diagrams	**	**

[7] System-internal scenarios are used in requirements engineering for complex systems where requirements are specified for the system as well as for its sub-systems and components. We explain the use of system-internal scenarios in requirements engineering in Part VII.

Chapter 12
Benefits of Using Goals and Scenarios

In this chapter, we describe the main benefits of using goals and scenarios in requirements engineering. We outline:

- *The benefits of using goals and scenarios in each requirements engineering activity*
- *The benefits of goal–scenario–coupling*

K. Pohl, *Requirements Engineering*,
© Springer-Verlag Berlin Heidelberg 2010

Use of goals and scenarios

This chapter explains the benefits of using goals (see Section 12.1) and scenarios (see Section 12.2) in the three core activities and the two cross-sectional activities of requirements engineering. Furthermore, the chapter describes the essential interdependencies between goals and scenarios. The positive effects of coupling goals and scenarios are illustrated by means of examples (see Section 12.3).

12.1 Benefits of Goal Orientation

The benefits of goal orientation in requirements engineering are presented, for instance, in [Loucopoulos 1994; Nuseibeh et al. 1994; Van Lamsweerde and Letier 2000; Rolland and Salinesi 2005]). In this section, we explain the essential benefits of using goals for each of the five requirements engineering activities outlined in Chapter 4.

12.1.1 Benefits for Documentation

Documentation in requirements engineering aims to document knowledge about the requirements and the context in an appropriate way (see Part IV.a).

Checking the requirements for completeness

During requirements documentation, goals can be used to check the completeness of the documented requirements or the requirements document. This aspect of goals was investigated by, amongst others [Yue 1987]. A set of requirements or a requirements document is complete according to Yue if the documented requirements suffice to satisfy the defined goals. Assuming that the goals are complete, the documented requirements can also be considered to be complete if this condition is met.

Avoiding irrelevant requirements

By explicitly considering goals, irrelevant requirements can be identified. An indication of the irrelevance of a requirement is that the requirement cannot be related to any goal as the requirement does not contribute to satisfying any of the defined goals. Hence, there is no justification for the existence of the requirement. Defining and implementing irrelevant requirements is also referred to as "gold plating".

Structuring of requirements documents

Goals can be used to structure requirements documents. The requirements document is then organised according to the decomposition structure of the goals. For instance, high-level goals can be used as section headings, and, for each high-level goal, the sub-goals decomposing this goal can be used as subsection headings. Eventually, each terminal goal, i.e. each goal that is not refined any further, can be used as the heading of the subsection that documents the requirements needed to satisfy that goal.

Definition of access paths to requirements

Each requirement in a requirements document contributes to the satisfaction of one or multiple goals. Goal models can be used to define logical access paths to the requirements in a requirements document. For this purpose, explicit relationships are defined between each goal and the requirements that contribute to the satisfaction of that goal. In this way, the goals provide access paths to the requirements document. In contrast to using goals for structuring a requirements document, the definition of logical access paths does not require structuring the requirements document according to the decomposition structure of the goals.

Detailed explanation in Part IV.a

The use of goals for documenting requirements is elaborated on in Part IV.a.

12.1.2 Benefits for Elicitation

Foundation for requirements elicitation

Goals document the stakeholders' intentions with respect to the system. Hence, stakeholders often make goals explicit in a conversation with the requirements engineer, e.g. by directly stating the goals. After some consolidation, the stakeholders' goals provide an initial basis for eliciting requirements that lead to the satisfaction of the goals (see [Dardenne et al. 1993; Van Lamsweerde 2001; Potts et al. 1994; Loucopoulos 1994]).

Goal-oriented requirements elicitation

If the goals for the system are known, the application of an elicitation technique can be aligned with a specific goal or set of goals, e.g. the goals documented in a specific branch of an AND/OR goal tree. Using goals for guiding requirements elicitation supports the systematic elicitation of requirements with a clear focus on satisfying the defined goals.

Identification and evaluation of alternative realisations

Furthermore, goal orientation supports the identification of potential alternative realisations. Therefor, a goal is first decomposed into a set of alternative sub-goals. Hence, satisfying one of the sub-goals is sufficient to satisfy the parent goal. In a second step, the stakeholders can sketch a possible realisation of each sub-goal (e.g. in terms of scenarios or solution-oriented requirements). Each such realisation is a possible realisation of the parent goal. The relevant scenarios or solution-oriented requirements identified for each sub-goal support the stakeholders in evaluating each possible realisation of the parent goal, e.g. with respect to cost and risk.

Refinement of the vision

Furthermore, the coupling of goals and scenarios has proven its worth for supporting requirements elicitation, and in particular the elcitation of new and innovative requirements (see Section 12.3). Requirements engineering is also a learning process for the stakeholders. Coupling goals and scenarios supports this learning process. Goals help to refine the system vision at an abstract level. Scenarios provide positive and negative examples of goal satisfaction.

Detailed explanation in Part IV.b

The use of goals during requirements elicitation is detailed in Part IV.b.

12.1.3 Benefits for Negotiation

Consolidation of different views

Stakeholders often have different views on the system to be developed. Due to the different views of the stakeholders, conflicts emerge among the stakeholders with regard to the requirements for the system (see [Nuseibeh et al. 1994]). The negotiation activity aims at consolidating the stakeholders' individual views in order to gain, preferably, a fully consolidated view on the requirements for the system.

Supporting conflict resolution

Conflicts in the solution-oriented requirements often result from conflicts between the intentions of the different stakeholders with regard to the system. Goal models can be used to identify and resolve conflicts between the different stakeholder intentions at an early stage of the requirements engineering process. Thereby, conflicts can be partly resolved before they creep into the solution-oriented requirements. However, requirements conflicts can never be fully avoided. Some conflicts may emerge only when detailed requirements are developed. In these cases, goals can (and should) still be used to support the resolution of conflicts. Requirements engineers should clarify whether the detected requirements conflicts are caused by a conflict in the stakeholders' intensions and goals (see e.g. [Van Lamsweerde and Letier 2000]). If this is the case, the conflicts can be resolved more easily by negotiating about the associated goals rather than by negotiating about a set of detailed requirements.

Detailed explanation in Part IV.c

In addition to supporting the identification and resolution of conflicts, goals can also be used to identify conflict types and select appropriate strategies for conflict resolution. The use of goals for attaining agreement about requirements is described in Part IV.c.

12.1.4 Benefits for Validation

Requirements validation (see Part V) aims at ensuring at an early stage that the right system is developed. The term "right" means, in this regard, that a system is developed that fulfils the consolidated set of wishes and expectations of all relevant stakeholders and meets all its constraints. The validation of requirements is supported by the explicit consideration of goals.

Validity of requirements with respect to a goal model

Based on the goals specified for a system, the requirements for the system can be validated. During the validation, for each goal and the set of requirements related to this goal, the stakeholders check whether the goal is satisfied if the system realises the requirements. If a goal cannot be satisfied by realising the associated requirements, the requirements may be incomplete or have some other type of defect. Note that, in the case of OR-decomposed goals, only one sub-goal needs to be satisfied in order to satisfy the parent goal. Hence the requirements specified for a system may be valid with respect to a goal model even if part of the sub-goals are not satisfied.

Detailed explanation in Part V

We describe the use of goals for supporting the validation of requirements in Part V.

12.1.5 Benefits for Management

Goals support the management activity in requirements engineering. Goals are particularly helpful for prioritising requirements and establishing requirements traceability.

Prioritisation of requirements

The explicit consideration of goals facilitates the prioritisation of requirements. In order to prioritise solution-oriented requirements, it is often possible to start with the prioritisation of the high-level goals specified for the system. The priorities of the goals are then "inherited" along the refinement relationships to the associated solution-oriented requirements (see Chapter 31).

Traceability of requirements

The explicit documentation of goals (together with the documentation of scenarios) helps to establish traceability in a project-specific way and supports the traceability of the requirement sources (see Chapter 30). By explicitly documenting goals, it is possible to establish traceability between the solution-oriented requirements and the stakeholders' wishes and expectations. Traceability is established by explicitly documenting the relationships between goals and the requirements that contribute to the satisfaction of these goals.

Detailed explanation in Part VI

The use of goals to support the management activity in requirements engineering is explained in Part VI.

12.2 Benefits of Using Scenarios

The benefits of using scenarios in requirements engineering are described, e.g. in [Alexander and Maiden 2004; Carroll 2000; Haumer et al. 1998; Haumer et al. 2000; Leite et al. 1997; Rolland et al. 1999; Sindre and Opdahl 2005; Weidenhaupt et al. 1998]. In this section, we sketch the main benefits of using scenarios in the five requirements engineering activities presented in Chapter 4.

12.2.1 Benefits for Documentation

Guiding requirements engineers by means of scenarios

Scenarios can be used to capture context information about the development facet (see Section 9.3). A scenario can describe, for instance, how a requirements engineer uses a specific template, checklist, or modelling language when documenting requirements, or how requirements artefacts of a specific type (e.g. goals) are used when creating requirements artefacts of a different type (e.g. solution-oriented requirements). In such a scenario, hints and instructions can be given, for instance, how to fill in the different slots of a use case template correctly. In this way, scenarios are specified for guiding requirements engineers during the documentation of requirements. In addition, requirements managers can derive from these scenarios, for instance, which attributes are needed for documenting requirements of a specific type.

Structuring requirements documents

Scenarios are also suited for structuring requirements documents. They can be used, for example, for defining views on a requirements document. If doing this, all requirements concerned with a specific scenario are associated with the scenario. This kind of structuring is especially advisable when the system is developed iteratively and a specific subset of scenarios or use cases is implemented in each iteration.

Providing rich context information for requirements

Scenarios facilitate illustrating or explaining solution-oriented requirements by means of examples. In addition, scenarios relate each requirement to relevant context aspects, to other solution-oriented requirements, and to goals. However, a prerequiste for this is that sencerios are interrelated with the corresponding solution-oriented requirements.

Embedding of requirements into a usage context

A scenario embeds requirements into a common usage context. By relating requirements to scenarios, each individual requirement can be traced to at least one scenario that places the requirement in a usage context and hence illustrates the additional value of the requirement for the stakeholders. Furthermore, if a requirement is related to multiple scenarios, stakeholders can obtain a better understanding of the requirement by considering the requirement in its different usage contexts.

Improvement of the comprehensibility of requirements

Since scenarios allow for relating requirements to each other and embedding them into the context, scenarios support not only requirements engineering but also the subsequent development activities as well as future change processes.

Detailed explanation in Part IV.a

The use of scenarios for requirements documentation is detailed in Part IV.a.

12.2.2 Benefits for Elicitation

The use of scenarios is an essential means for supporting the elicitation of requirements. Scenarios support the elicitation of knowledge and the understanding of the stakeholders' needs and intentions during requirements elicitation (see [Karat and Bennett 1991; Jacobson et al. 1992; Dardenne 1993; Potts et al. 1994; Rosson and Carroll 1993]).

Goal refinement

During the development of scenarios, typically, new goals are identified, existing goals are refined, and alternative ways of satisfying the known goals are identified (see Section 12.3).

Explanation of intentions

Scenarios provide an intuitive facility to communicate requirements for the system to be developed. A stakeholder can, for example, explain his or her intentions with the help of narrative scenarios (see Section 11.1). The resulting scenarios convey the goals for the system from the individual perspective of this stakeholder (see [Weidenhaupt et al. 1998; Carroll 2000]).

Communication support

By means of scenarios, poorly understood aspects of the system context can be explained and communicated easily. For example, the stakeholders may develop instance scenarios (see Section 10.5) that document concrete sequences of interactions in the system context of a driver assistance system.

Basis for the development of requirements

Furthermore, scenarios serve as a basis for the development of functional requirements for the system including the associated quality requirements.

Detailed explanation in Part IV.b

The use of scenarios for requirements elicitation is explained in Part IV.b.

12.2.3 Benefits for Negotiation

Scenarios are a simple means of communication and, according to experience, comprehensible to all stakeholders. Thus, scenarios can be used to analyse and resolve requirements conflicts.

Conflict analysis

During conflict analysis, the identification of conflicts, communication about conflicts, and resolution of conflicts can be supported by using scenarios to illustrate conflicts. Furthermore, the creation of scenarios during conflict analysis supports the detection of the actual cause of a conflict.

Conflict resolution

Scenarios can be used during conflict resolution for documenting alternative, possible realisations in a way that is comprehensible to all stakeholders. Hence scenarios contribute to achieving consensus more easily (see [Weidenhaupt et al. 1998]).

Detailed explanation in Part IV.c

The use of scenarios for attaining agreement about the requirements is described in Part IV.c.

12.2.4 Benefits for Validation

Scenarios document concrete examples of intended system usage. For this reason, they are well suited for involving different kinds of stakeholders in the validation of requirements artefacts. By means of scenarios, complex requirements can be presented to the stakeholders who validate the requirements in a comprehensible form.

Validating scenarios from different perspectives

Due to their good comprehensibility to technical and non-technical stakeholders, scenarios can be validated by a variety of stakeholders. Scenarios can be used, for example, to check whether the users of the system agree with the usage workflows. Architects can validate scenarios in order to check whether the scenarios convey sufficient information for developing an adequate architecture. Testers can validate scenarios in order to check whether sufficient information for developing test cases is available.

Validating other requirements artefacts using scenarios

Scenarios can also be used to support the validation of other requirements artefacts. For instance, stakeholders can check solution-oriented requirements for completeness, correctness, and consistency with respect to the scenarios specified for the system. Incompleteness, incorrectness, and inconsistencies in the requirements can be detected by checking whether each positive scenario can be satisfied and each negative or misuse scenarios (see Section 10.1) can be prevented if the requirements are realised.

Detecting irrelevant requirements

Scenarios support, furthermore, the identification of irrelevant requirements in requirements documents. If a documented requirement is not related to a scenario, this is an indication of the irrelevance of the requirement.

Including context information in the validation

Since scenarios document context information, the scenarios related to the requirements can be used to incorporate additional context information during the validation of these requirements. This holds for the validation of solution-oriented requirements as well as goals and alternative realisation possibilities.

Combining prototypes and scenarios

Scenarios are used during validation preferably in combination with prototypes. By employing a prototype, the stakeholders can execute the scenarios interactively. Experimenting with the prototype or the scenarios realised in the prototype enables the stakeholders to detect additional errors, misunderstandings, and conflicts both in the prototypical realisation as well as in the underlying scenario definitions.

Detailed explanation in Part V

The use of scenarios for supporting requirements validation is explained in Part V.

12.2.5 Benefits for Management

Prioritisation support

Within the management activity, scenarios support the prioritisation of requirements. For example, the stakeholders may determine the priorities of solution-oriented requirements based on the priorities that are assigned to the scenarios. For this purpose, the scenarios are prioritised, first. Subsequently, the priorities of the scenarios are used to prioritise the requirements related to the scenarios. According to experience, scenarios facilitate negotiation about the priorities of requirements even in very heterogeneous stakeholder groups.

Determining required traceability information

Additionally, scenarios are well suited for determining what traceability information is needed in a project. For this purpose, scenarios describing the utilisation of the traceability information are developed. Based on these usage scenarios, requirements engineers determine which traceability relationships are needed to support these usage scenarios (see [Dömges and Pohl 1998; Pohl et al. 1997]).

Link between goals and solution-oriented requirements

Scenarios support the management of solution-orientied requirements by acting as a bridge between solution-oriented requirements and goals. Scenarios illustrate goal satisfaction (or the failure to satisfy goals). Therefore, each scenario is related to one or multiple goals. In addition, scenarios put solution-oriented requirements into a usage context (see Section 12.2.1). Therefore, each scenario is also related to a set of solution-oriented requirements. Since scenarios are related to goals as well as to solution-oriented requirements, they also relate the stakeholders' goals and the solution-oriented requirements for the system to each other.

Support of change management

Scenarios support change management insofar as the context information contained in the scenarios supports an evaluation of the planned changes. Furthermore,

one can exploit the scenarios associated with a requirement that needs to be changed to assess which other requirements may be affected by the change as well.

Detailed explanation in Part VI

The use of scenarios for supporting the management activity in requirements engineering is explained in more detail in Part VI.

12.3 Benefits of Goal–Scenario–Coupling

Figure 12-1 illustrates the most important interdependencies between goals and scenarios. These interdependencies are the key motivation for coupling goals and scenarios in requirements engineering (see e.g. [Antón and Potts 1998; Antón et al. 2000; Haumer et al. 1998; Haumer et al. 1999; Potts 1995; Rolland et al. 1998b; Sutcliffe et al. 1998]). In the following sections, we explain each interdependency and illustrate by means of examples how the interdependencies can be exploited to support the requirements engineering activities.

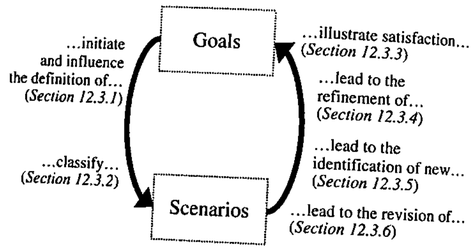

Fig. 12-1 *Interdependencies between goals and scenarios*

12.3.1 Goals Initiate the Definition of Scenarios

Better understanding of goals through scenarios

In order to improve the stakeholders' understanding of a goal, scenarios are developed for this goal. These scenarios describe exemplary interaction sequences that lead either to the satisfaction of the goal or to failure to satisfy the goal. Hence, a goal initiates the definition of scenarios (see Fig. 12-2).

Example of a goal initiating the definition of scenarios

The left-hand side of Fig. 12-2 shows an extract of a goal tree that documents the decomposition of the goal "high efficiency of the car". The sub-goal "protection against theft" is refined by the goals "car-theft protection through alarm system" and "ability to localise car position".

In order to gain additional knowledge about the goals "car-theft protection through alarm system" and "ability to localise car position", the stakeholders develop scenarios for these two goals. Thus, the two goals initiate the definition of new scenarios. The resulting scenarios describe exemplary workflows for the satisfaction of these goals (main and alternative scenarios) as well as sequences of interactions that are

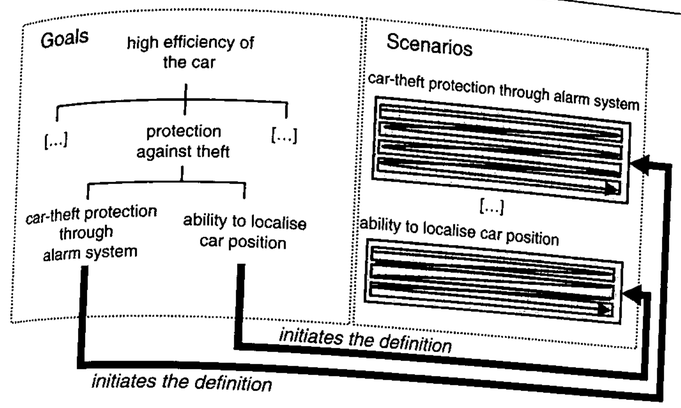

Fig. 12-2 *Goals initiate the definition of scenarios*

executed when defined exceptional events occur (exception scenarios). If goals in the goal model change, this leads to a change of the associated scenarios in the majority of cases.

12.3.2 Goals Classify Scenarios

Scenarios illustrate the satisfaction of goals or the failure to satisfy some goals. Hence, each scenario can be related to the goals that are (partly or fully) satisfied by the scenario. These goals can be used to classify the scenarios (see e.g. [Jacobson et al. 1992; Rolland et al. 1998b]). Based on the relationships between the goals and scenarios, the following classes of scenarios can be identified for each goal (see Section 10.7):

- *Scenarios documenting the satisfaction of the goal*: This set or class of scenarios can be determined by considering the scenarios of types "main" and "alternative" that are related to a specific goal. *Satisfaction of a goal*
- *Scenarios documenting the failure to satisfy the goal*: This set of scenarios can be determined by considering the scenarios of type "exception" that are related to a specific goal. *Failure to satisfy a goal*
- *Scenarios documenting system usage violating the goal*: This set of scenarios can be determined by considering the scenarios of type "misuse" that are related to a specific goal. *System usage to be prevented*

Figure 12-3 illustrates the classification by means of the goal "ability to localise car position". Three scenarios are associated with this goal: a main, an alternative, and an exception scenario.

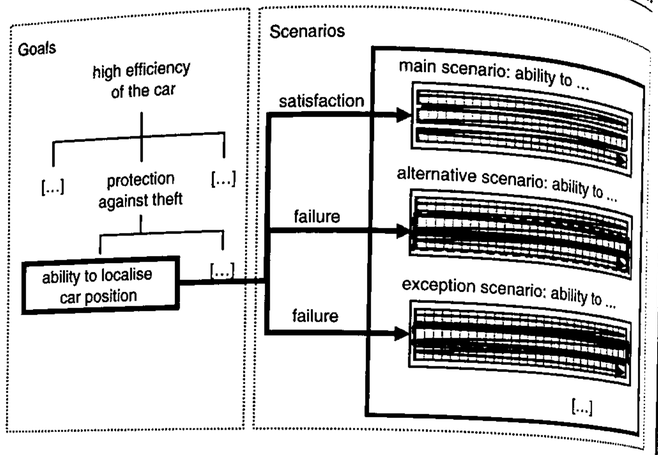

Fig. 12-3 *Classification of scenarios by means of goals*

12.3.3 Scenarios Illustrate Goal Satisfaction

Interaction sequences illustrate goal satisfaction

Scenarios illustrate the satisfaction of goals by means of exemplary interaction sequences. An exemplary interaction sequence provides additional information about a goal (see e.g. [Potts 1995; Pohl and Haumer 1997; Haumer et al. 1998]) such as:

- An example of satisifying the goal
- An example of failing to satisfy the goal
- An example of a purposeful violation of the goal, i.e. the misuse of the system (see Section 10.3)

Example of goal satisfaction

Figure 12-4 shows a main and an alternative scenario (right-hand side of Fig. 12-4) that illustrate the satisfaction of the goal "automatic navigation" (left-hand side of Fig. 12-4). The scenarios are represented using a sequence diagram with an "alt" fragment (see Section 11.5). Each of the two scenarios documents a sequence of interactions satisfying the associated goal.

12.3.4 Scenarios Initiate the Elaboration of Goals

Interplay between scenario definition and goal elaboration

Illustrating the satisfaction of goals by means of scenarios often leads to the elaboration of the goals. The elaboration of scenarios and the associated elaboration of goals support the stakeholders' learning process and improve the understanding of the requirements. By iterating between goal-oriented scenario definition and

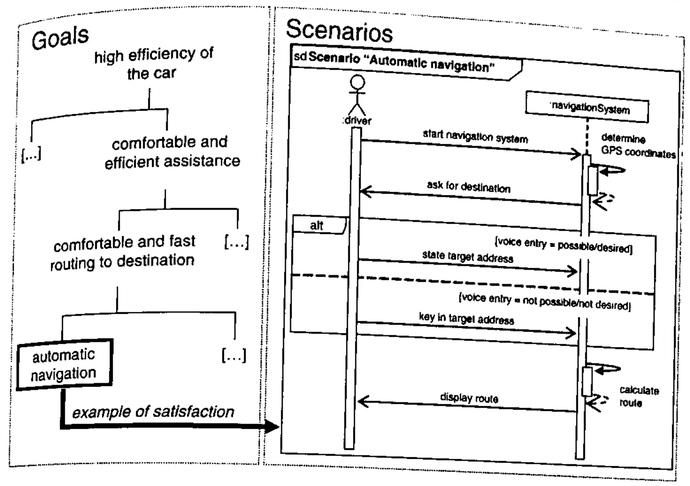

Fig. 12-4 *Scenarios illustrate goal satisfaction*

scenario-based goal elaboration, additional knowledge is successively gained about the stakeholders' goals and the intended ways of satisfying these goals. Thereby the understanding of the system to be developed is increased.

The elaboration of goals includes the following cases (see e.g. [Potts 1995; Haumer et al. 1998; Antón et al. 2000]):

- Goal decomposition, i.e. the identification of new sub-goals
- Identification of new, independent goals
- Revision or removal of existing goals

Goal Decomposition

A scenario initiates the definition of new sub-goals

When concretising a goal by means of a scenario, new sub-goals of this goal may be identified. Defining sub-goals for an existing goal is called goal decomposition (see Section 7.3).

Example of goal decomposition

Figure 12-5 shows an example of goal decomposition based on the development of a scenario. The satisfaction of the goal "automatic navigation" (left-hand side of Fig. 12-5) is illustrated by a scenario (right-hand side of Fig. 12-5). This step is marked with ❶ in Fig. 12-5. The first interaction in the presented scenario is the activation of the navigation system by the driver. The scenario documents that, in a second step, the navigation system determines the current position of the car. The definition of this interaction initiates the identification of a new sub-goal. The new sub-goal "determination of car position via GPS" of the goal "automatic navigation" is added to the goal model. This step is marked with ❷ in Fig. 12-5.

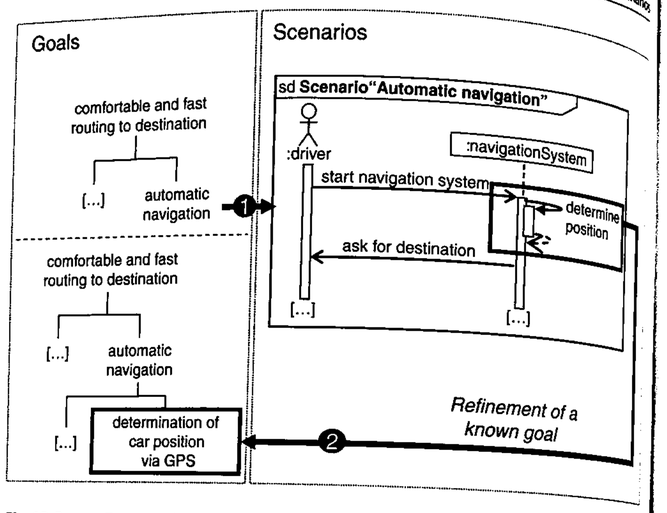

Fig. 12-5 *Decomposition of a goal initiated by a scenario*

Identification of New, Independent Goals

A scenario leads to the identification of independent goals

When the satisfaction of a goal is illustrated by a scenario, entirely new goals may be identified, i.e. goals that are independent goals, not sub-goals of some goal that is associated with the scenario (see [Antón et al. 2000; Haumer et al. 1998; Rolland et al. 1998b]).

Example of identifying new goals

Figure 12-6 illustrates the identification of a new goal with the help of a scenario. As shown in Fig. 12-6, first, the satisfaction of the goal "automatic navigation" is illustrated by a scenario (Fig. 12-6, ❶). The scenario documents that the driver of the

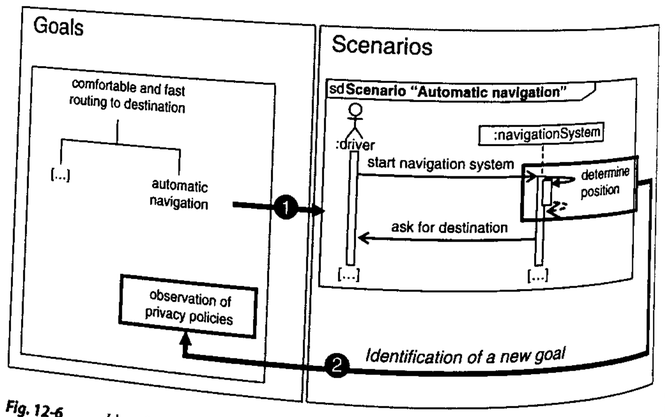

Fig. 12-6 *Identification of an independent goal initiated by a scenario*

car first activates the navigation system. Subsequently, the position of the car is determined automatically. With regard to the interaction step "determine position", the data security engineer remarks that all privacy policies definitely have to be adhered to when realising this functionality. Hence, a new goal "observation of privacy policies" is identified (Fig. 12-6, ❷) that has not been considered before. Unlike the goal identified in Section 12.3.4, this goal is not a sub-goal of some goal that is already associated with the scenario. Therefore, the new goal is specified as an independent goal.

Revision/Removal of Goals

A scenario initiates revision/removal of a goal

Concretising goals by means of scenarios can also lead to the revision or even the removal of goals associated with the scenario (see e.g. [Haumer et al. 1998; Haumer et al. 1999]). This interaction between goals and scenarios is especially important for the validation of goal models (see e.g. [Sutcliffe et al. 1998]).

Example of revising/removing a goal

Figure 12-7 illustrates a revision of a goal that is initiated by a scenario. In the example, the satisfaction of the goal "manual entry of traffic blocks during the trip" is illustrated by a scenario (Fig. 12-7, ❶). In the scenario, the driver enters a traffic block manually during the trip by entering the number of the road and the blocked section of the present route. The validation of the goal reveals that the manual entry during the trip contradicts the legal regulations for driving safety (Fig. 12-7, ❷). Therefore, it is necessary to revise the goal "manual entry of traffic blocks during the trip" (Fig. 12-7, ❸). The revision of the goal can, for example, lead to the removal of the goal, i.e. the driver is not provided with the functionality to enter traffic blocks manually during the trip. Alternatively, the goal could be revised so that manual entry is only allowed for a stationary car or for a car driving at walking speed.

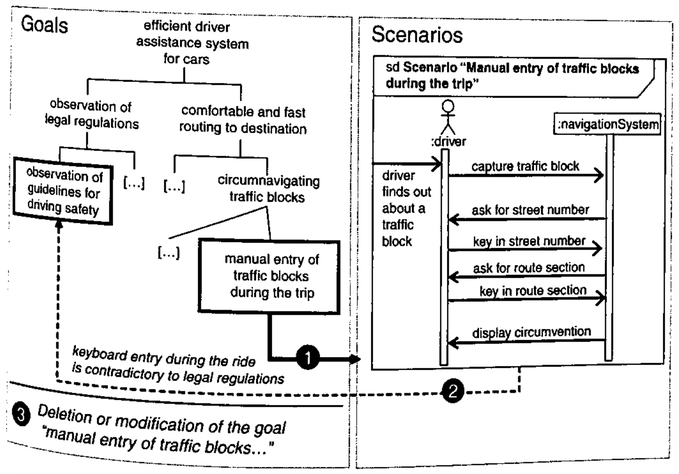

Fig. 12-7 *Revision of a goal initiated by a scenario*