UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Requirements Engineering & Management

Scenarios III – Message Sequence Charts

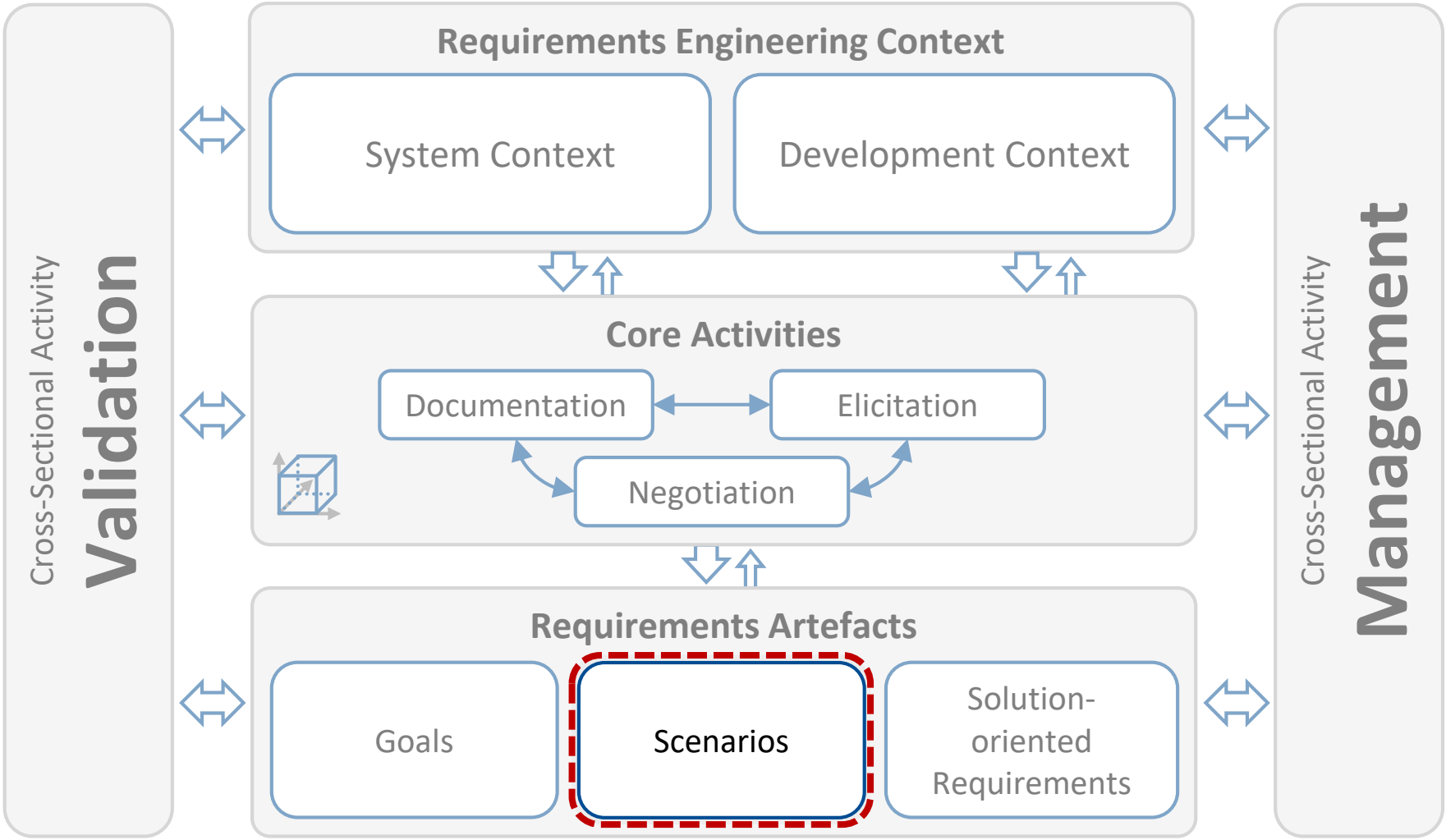
Prof. Dr. Klaus Pohl

PALUNO
The Ruhr Institute for Software Technology

Agenda

1. Scenario Modelling
2. Structuring of Scenarios

Framework for Requirements Engineering



1. Scenario Modelling

Modelling Languages of Scenarios

Commonly used languages:

- ITU Message Sequence Charts
- UML Sequence Diagrams
- UML Communication Diagrams
- UML Activity Diagrams

ITU Message Sequence Charts

Consists of three parts:

- **Message Sequence Chart Document:**

  Container document, to specify all relevant scenarios
- **Basic Message Sequence Chart:**

  Specify scenarios by means of interaction sequences
- **High-Level Message Sequence Chart:**

  Define possible execution orders of scenarios

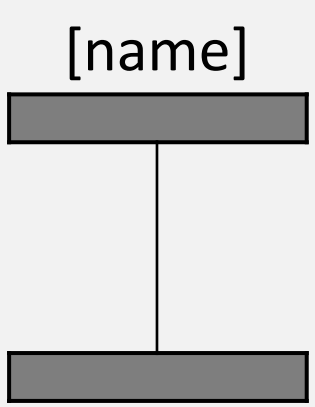
Basic Message Sequence Charts (bMSC) - Overview

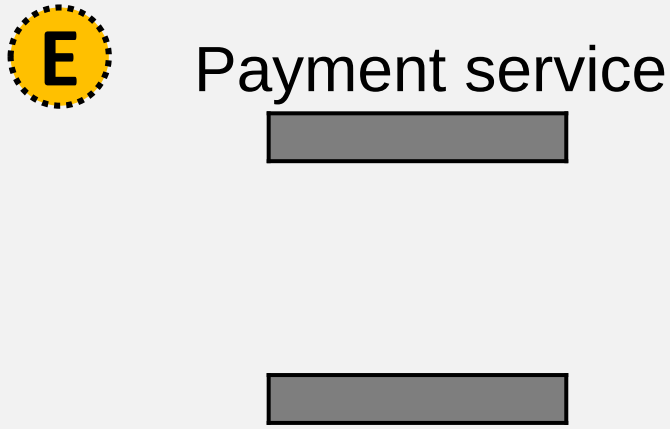
- Can be used to specify single scenarios.
- Define interaction-based system **behavior**.
- Specify **interactions** between objects:
  - Between actors
  - Between actor and system
  - Between system instances
- Can be used for **instance level** and for **type level descriptions**.

Modelling Construct: Instance

- Represents the **existence of an object** (e.g., a role of an actor or a
- system instance) in a scenario.
- The lifeline of an instance represents **time usage** from top to bottom.
  - Note: The time axis is particular to each object and not true to scale.

Notation

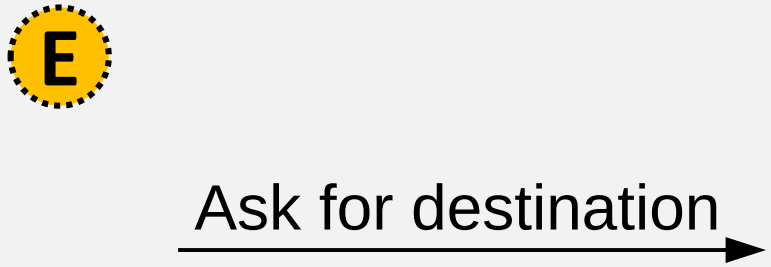




Modelling Construct: Message

- A message represents a **signal**, **data flow**, or **procedure call**.
- Two possible interpretations:
  - **Synchronous** messages: sending and receiving events take place in the same given order.
  - **Asynchronous** messages: sending and receiving events are not coupled.

Notation



Modelling Construct: Action

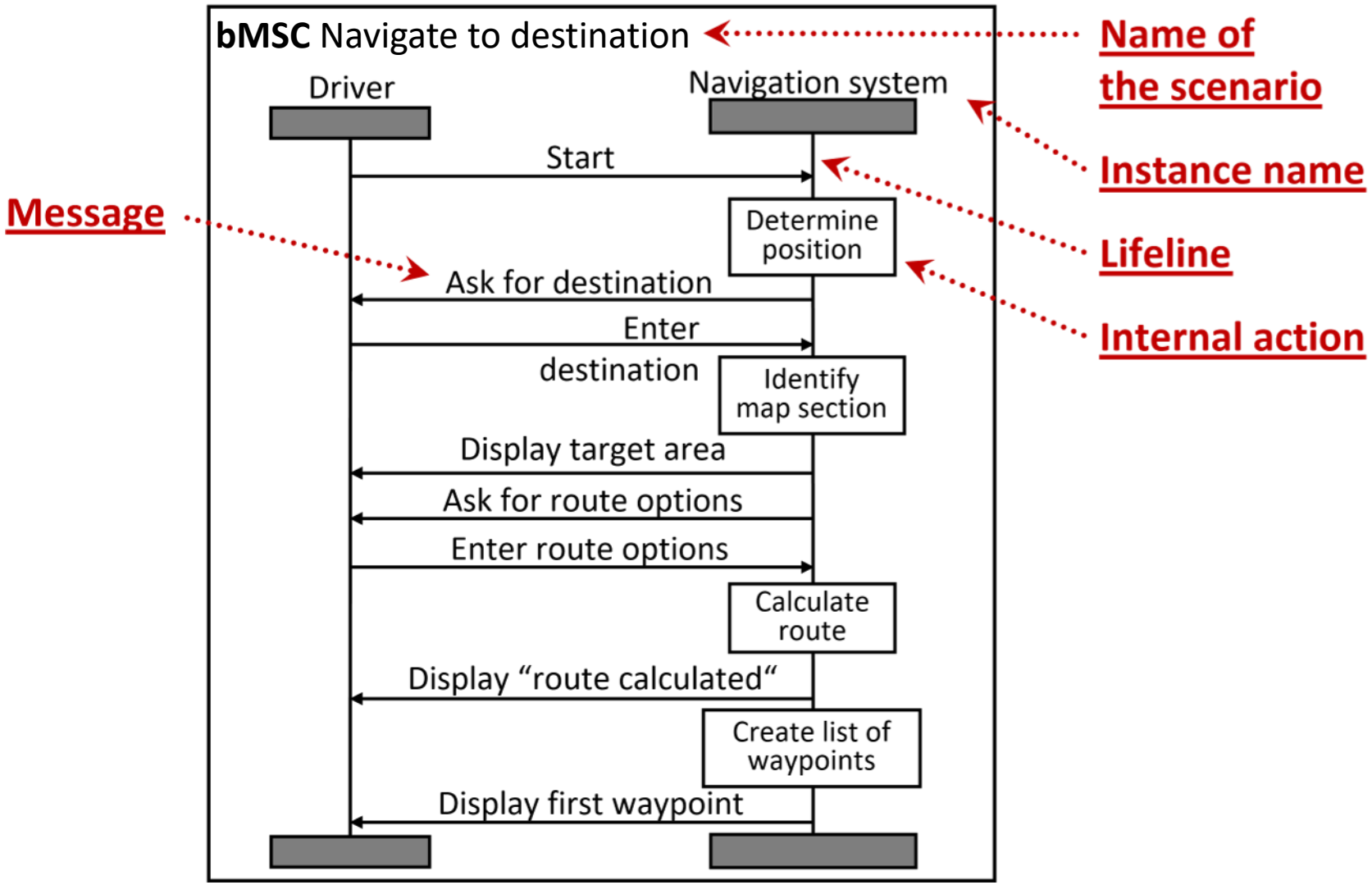
- An action represents an **atomic activity of an instance**.

Notation

[name]

determine position

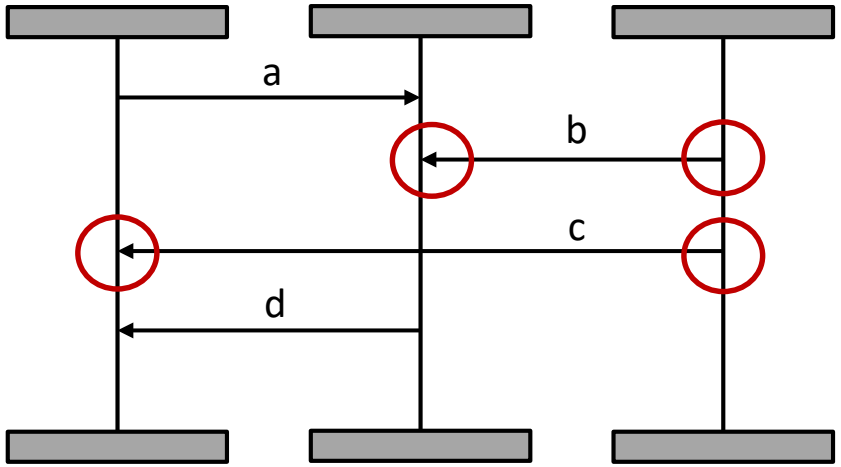
Main Scenario in a bMSC



Synchronous vs. Asynchronous Communication

Main assumptions of asynchronous data exchange:

Sending and receiving of a message are different events.



Asynchronous Communication

Message b is sent before message c, but message c can be received before message b is received. Thus, there are several possible orders of events:

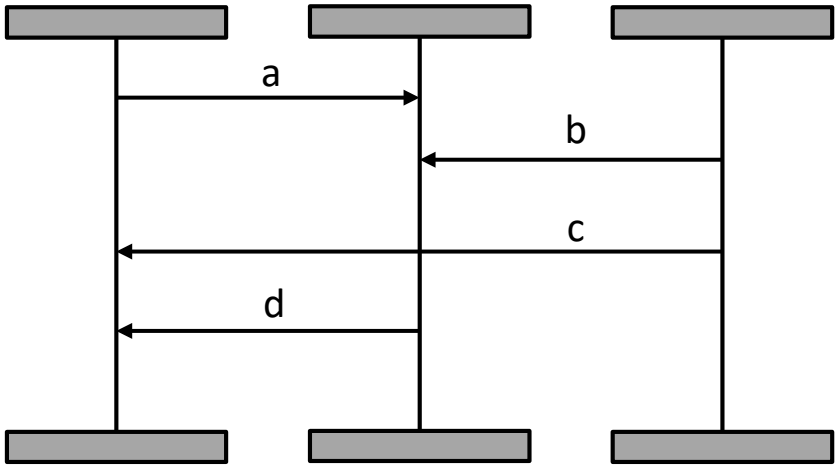
1.) s(b) < r(b) < s(c) < r(c)

2.) s(b) < s(c) < r(b) < r(c)

3.) s(b) < s(c) < r(c) < r(b)

Visual vs. Causal Order

In contrast to UML sequence diagrams, ITU Message Sequence Charts use a causal order. This means, **events** are **not ordered according to their visual arrangement**, **but according to their logical occurrence**.



Visual Order:

a < b < c < d

Causal Order:

(a < d) & (b < d) & (b < c)

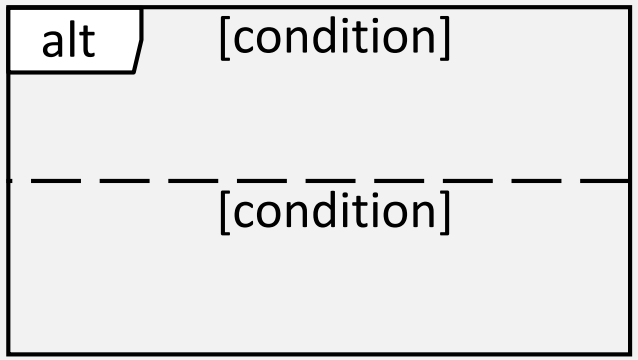
Following possibilities arise:

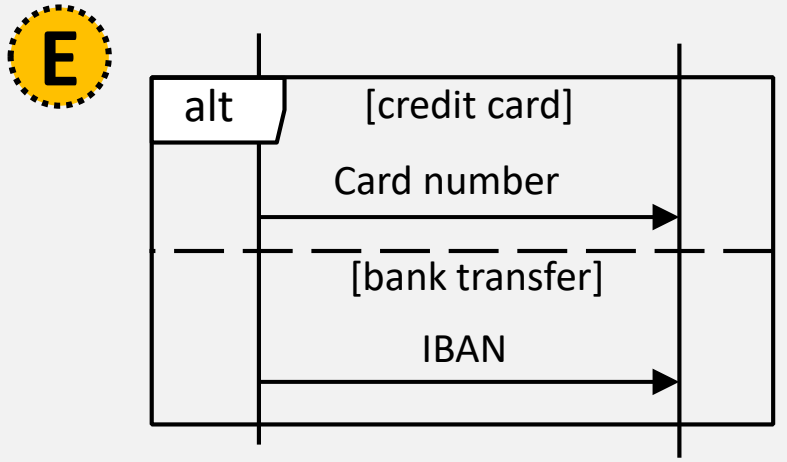
1.) a < b < c < d
2.) b < a < c < d
3.) a < b < d < c
4.) b < a < d < c
5.) b < c < a < d

Modelling Construct: Inline Expression “alt”

- Inline Expressions (in general):
  - A **region in a bMSC**, to which a certain rule applies.
  - Can be nested.
- **“alt”** separates **parts** of an interaction that are **executed alternatively**, based on a certain **condition**.

Notation

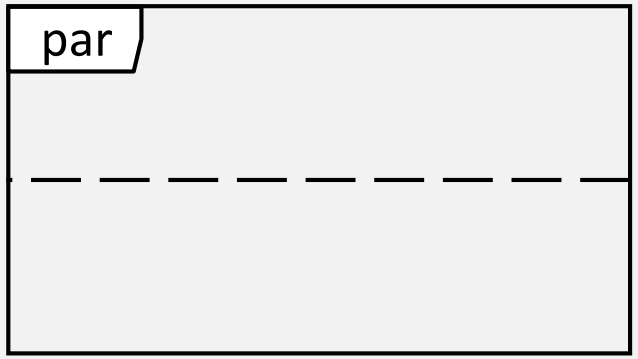


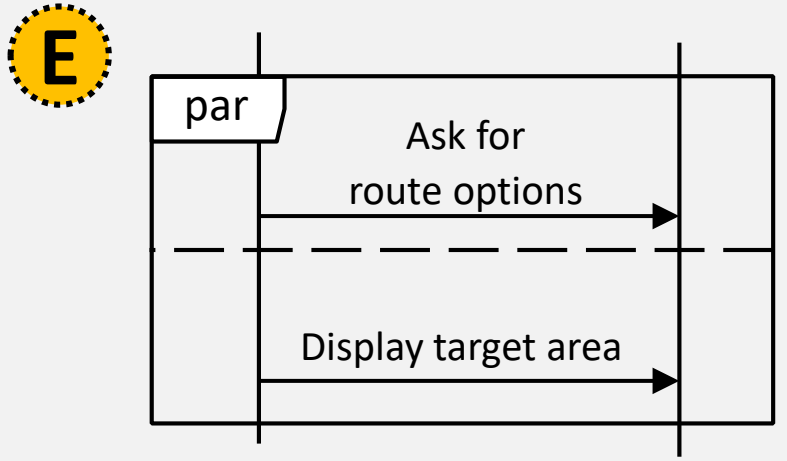


Modelling Construct: Inline Expression “par”

- Inline Expressions (in general):
  - A **region in a bMSC**, to which a certain rule applies.
  - Can be bested.
- **“par”** defines interaction parts that are **executed in parallel**.
- Parallel does not mean simultaneously, but in an **arbitrary order**.

Notation

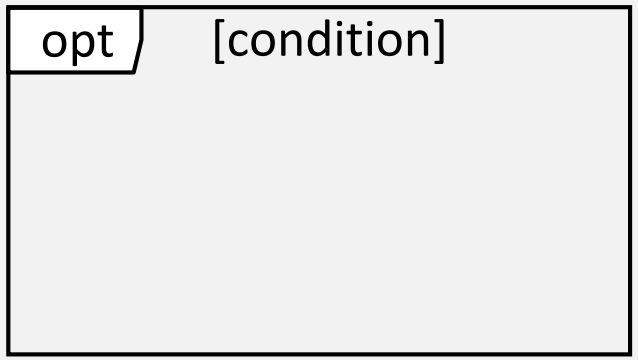


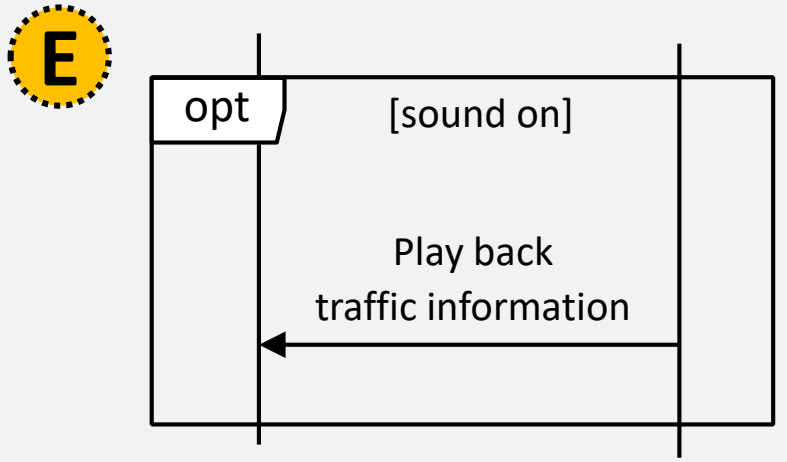


Modelling Construct: Inline Expression “opt”

- Inline Expressions (in general):
  - A **region in a bMSC**, to which a certain rule applies.
  - Can be bested.
- **“opt”** defines interaction parts, which are only **executed if a certain condition holds**.

Notation

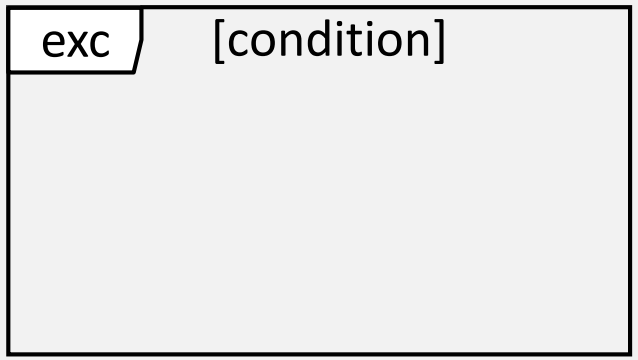


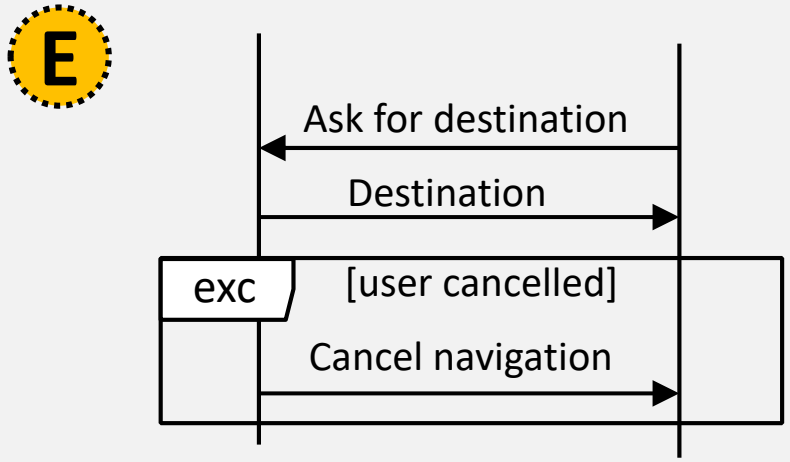


Modelling Construct: Inline Expression “exc”

- Inline Expressions (in general):
  - A **region in a bMSC**, to which a certain rule applies.
  - Can be bested.
- **“exc”** defines an interaction part **executed in case of an exception**.
- **“exc”** part is executed and the **surrounding parts are cancelled**, i.e. the scenario terminates after the execution of the break interaction.

Notation

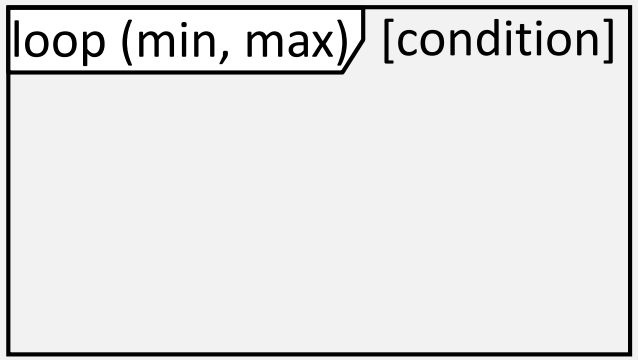


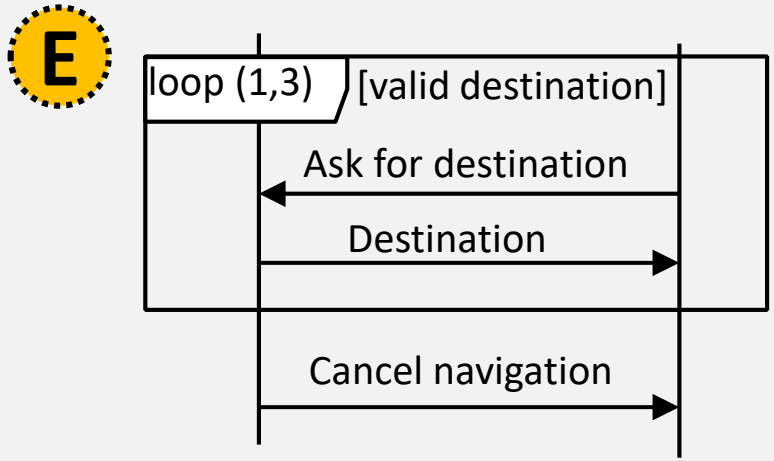


Modelling Construct: Inline Expression “loop”

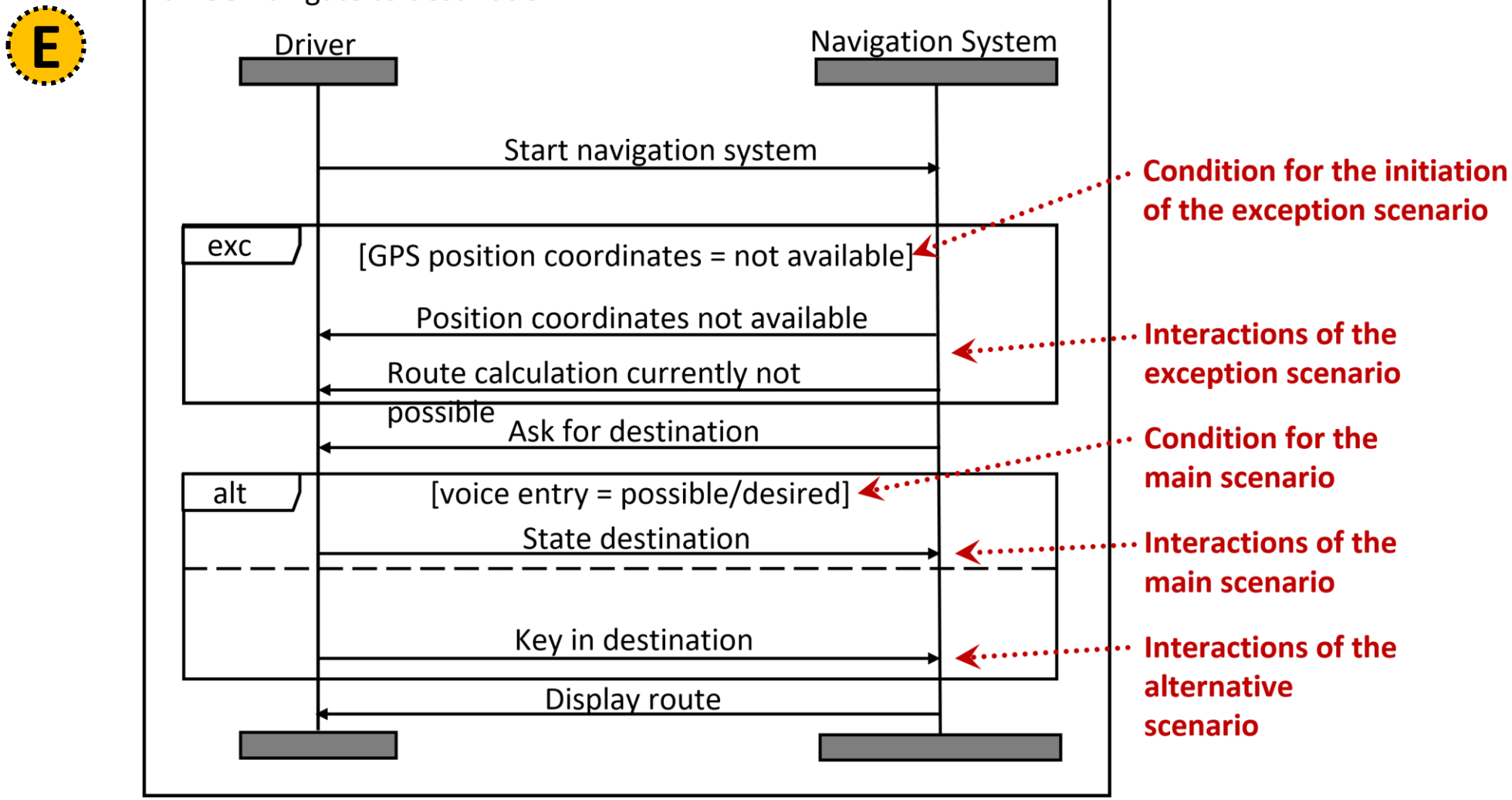
- Inline Expressions (in general):
  - A **region in a bMSC**, to which a certain rule applies.
  - Can be bested.
- **“loop”** defines the iteration of a **particular sequence of interaction steps** within a scenario,
- Interactions within the loop fragment are **iterated** min to max times or until a specific (optional) **loop condition** becomes valid.

Notation

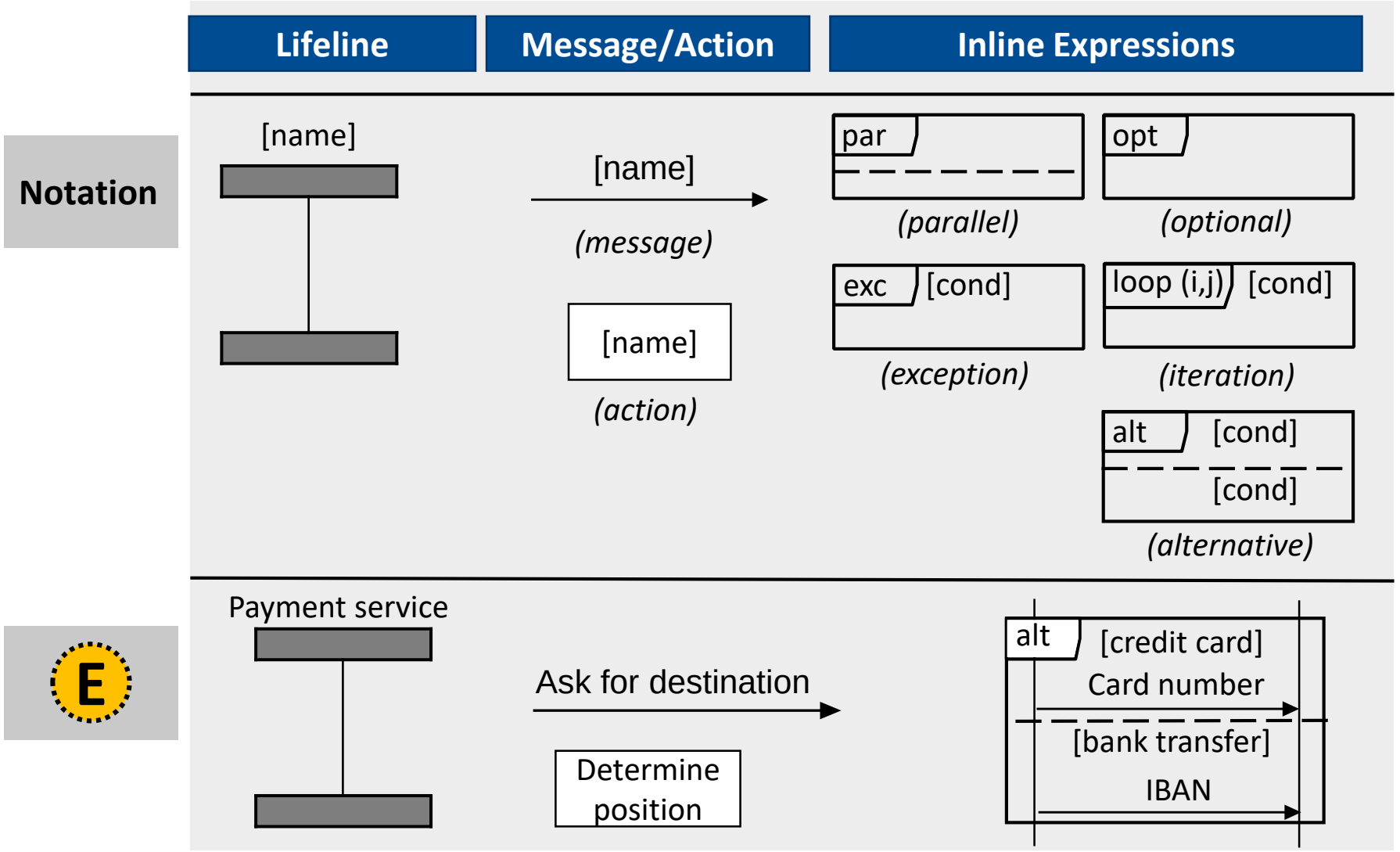




Alternative and Exception Scenarios



Modelling Constructs: bMSCs



2. Structuring of Scenarios

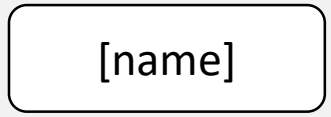
High-Level Message Sequence Charts (hMSC)

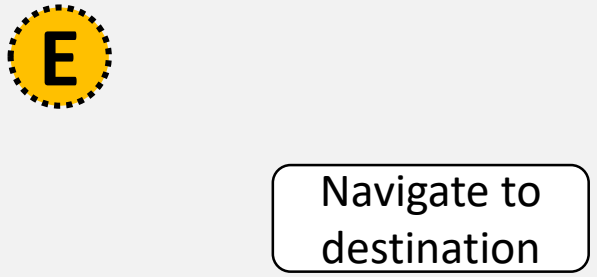
- **hMSCs**
  - Show ordering of the single scenarios.
  - Can be nested.
- **Advantages and disadvantages of hMSCs**
  - hMSCs provide a clear structure and aid in differentiation between distinct scenarios.
  - Extensive use of hMSCs transfers all concepts of bMSC inline expressions to the hMSC:
    - This aids in easy to understand bMSCs and reduces the complexity of a specification.
    - Single bMSCs can easily tend to become trivial and already simple scenario must be investigated across multiple hMSCs and bMSCs.

Modelling Construct: MSC Reference

- The nodes (MSC references) are represented in forms of **rectangles with rounded edges**.
- Each node **references** either a **bMSC** (describing a single scenario) or **another hMSC**.

Notation

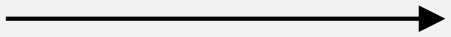


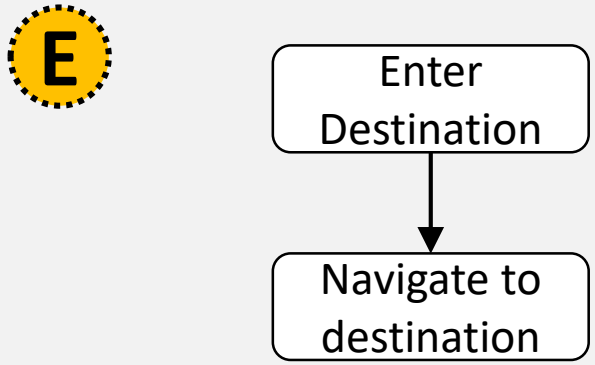


Modelling Construct: Flow Lines

- Flow lines indicate the **logical temporal order** in which the nodes (i.e. the corresponding scenarios) are **passed through**.

Notation

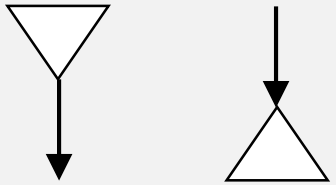


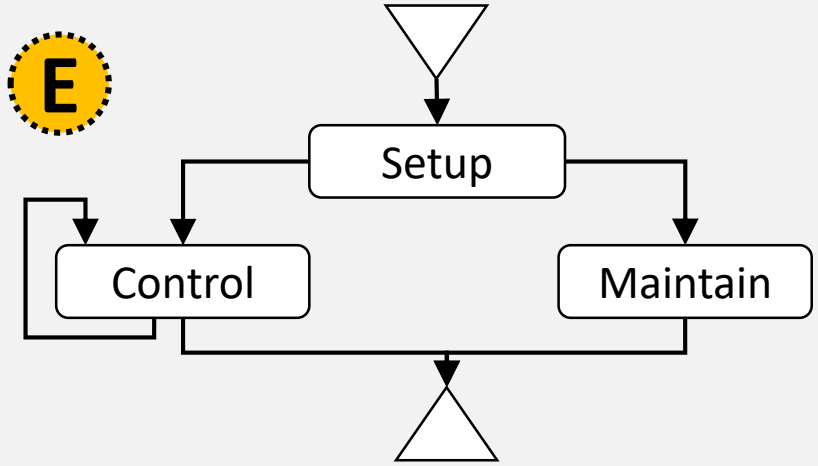


Modelling Construct: Start/End Nodes

- Start and end nodes are represented in form of a **triangle**.
- There is always just **one start node**.
- There can be **multiple end nodes**.

Notation

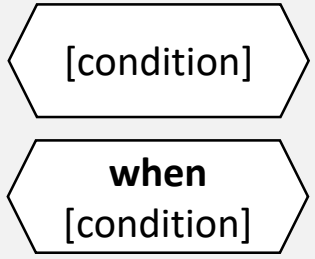


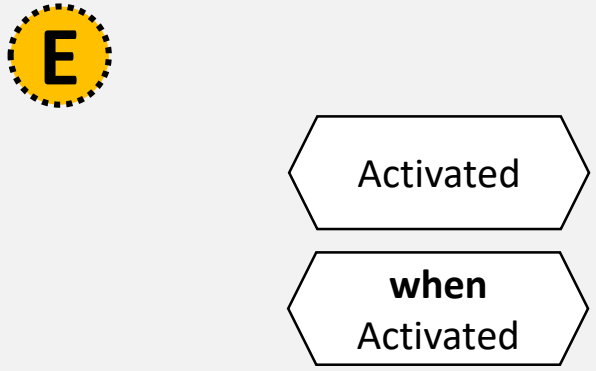


Modelling Construct: Condition

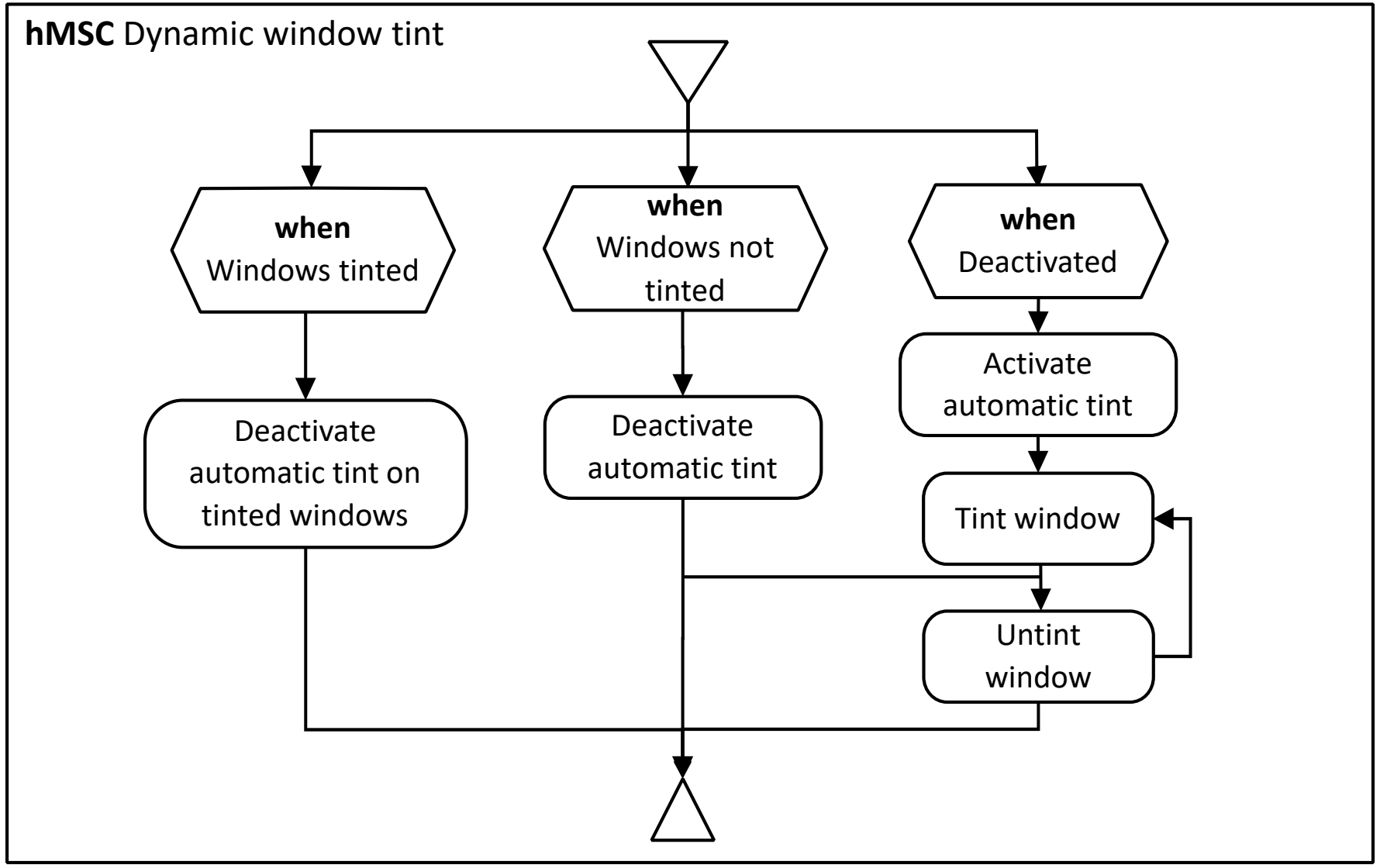
- Conditions are modelled using **hexagons**.
- Conditions **limit the execution paths** of a scenario specification.
- Conditions can also be used in bMSCs.
- It is to distinguish between **guarding conditions** (i.e. checking whether a condition is true) and corresponding **setting conditions**.

Notation

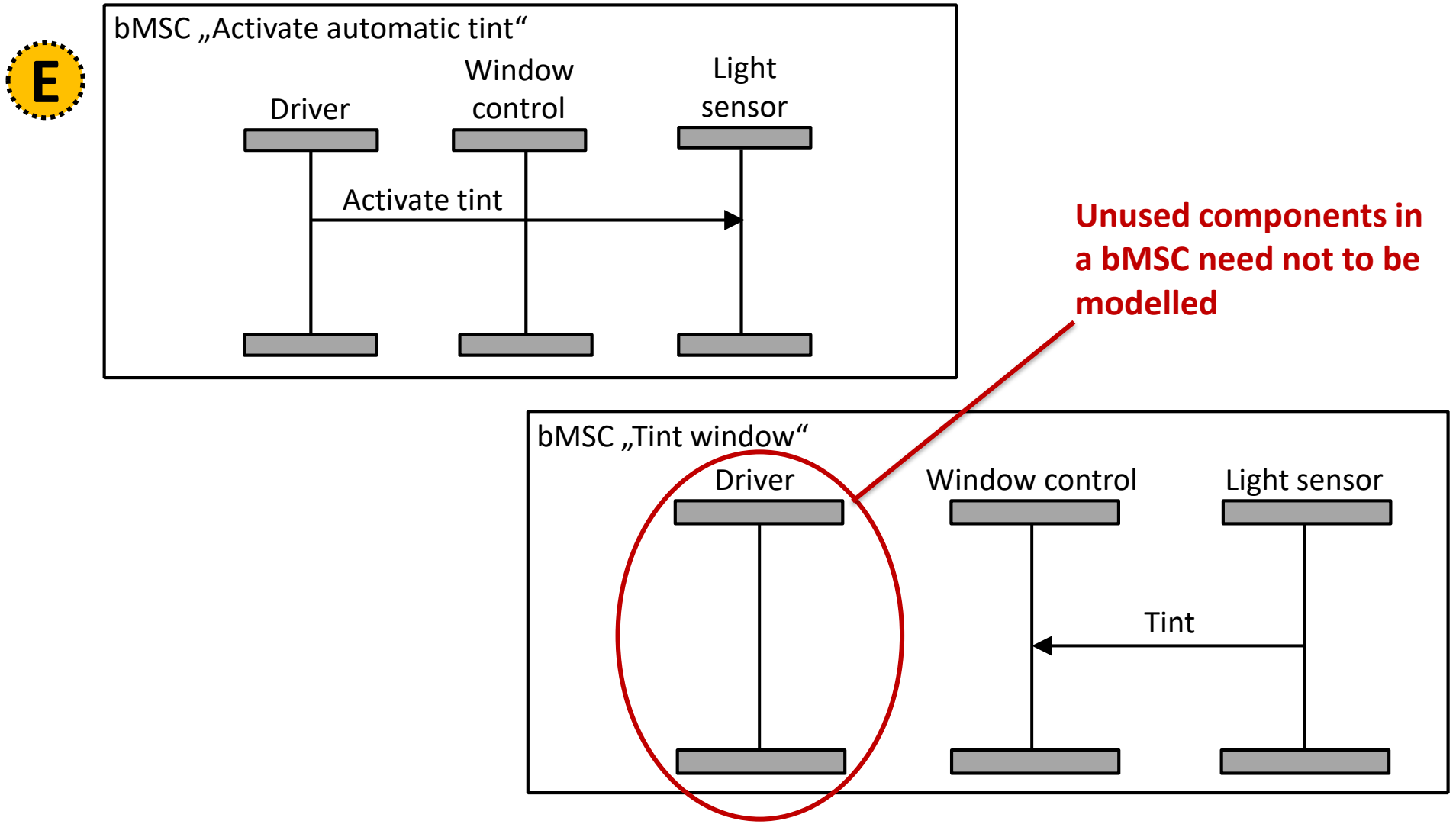




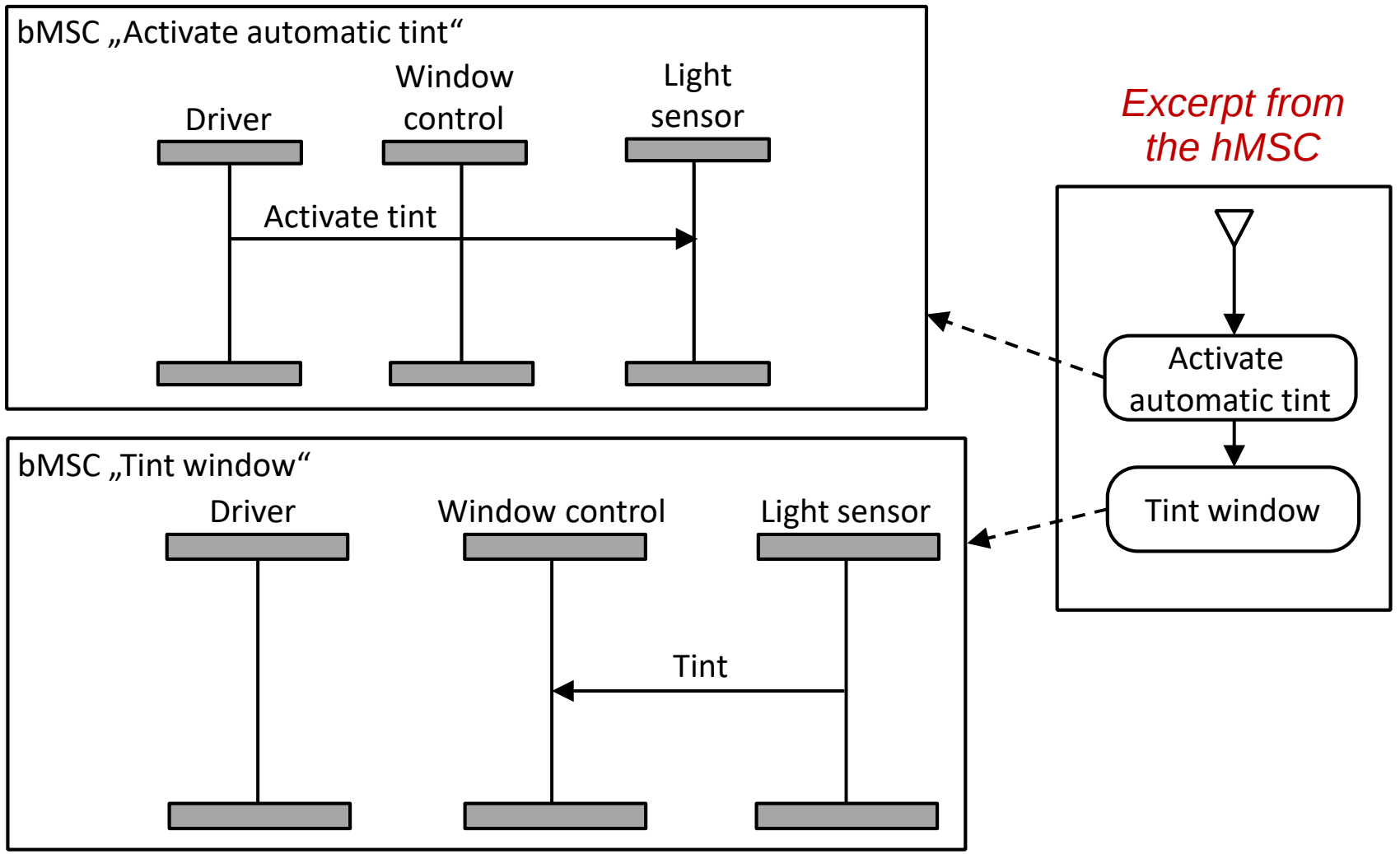
hMSC of a Dynamic Window Tint



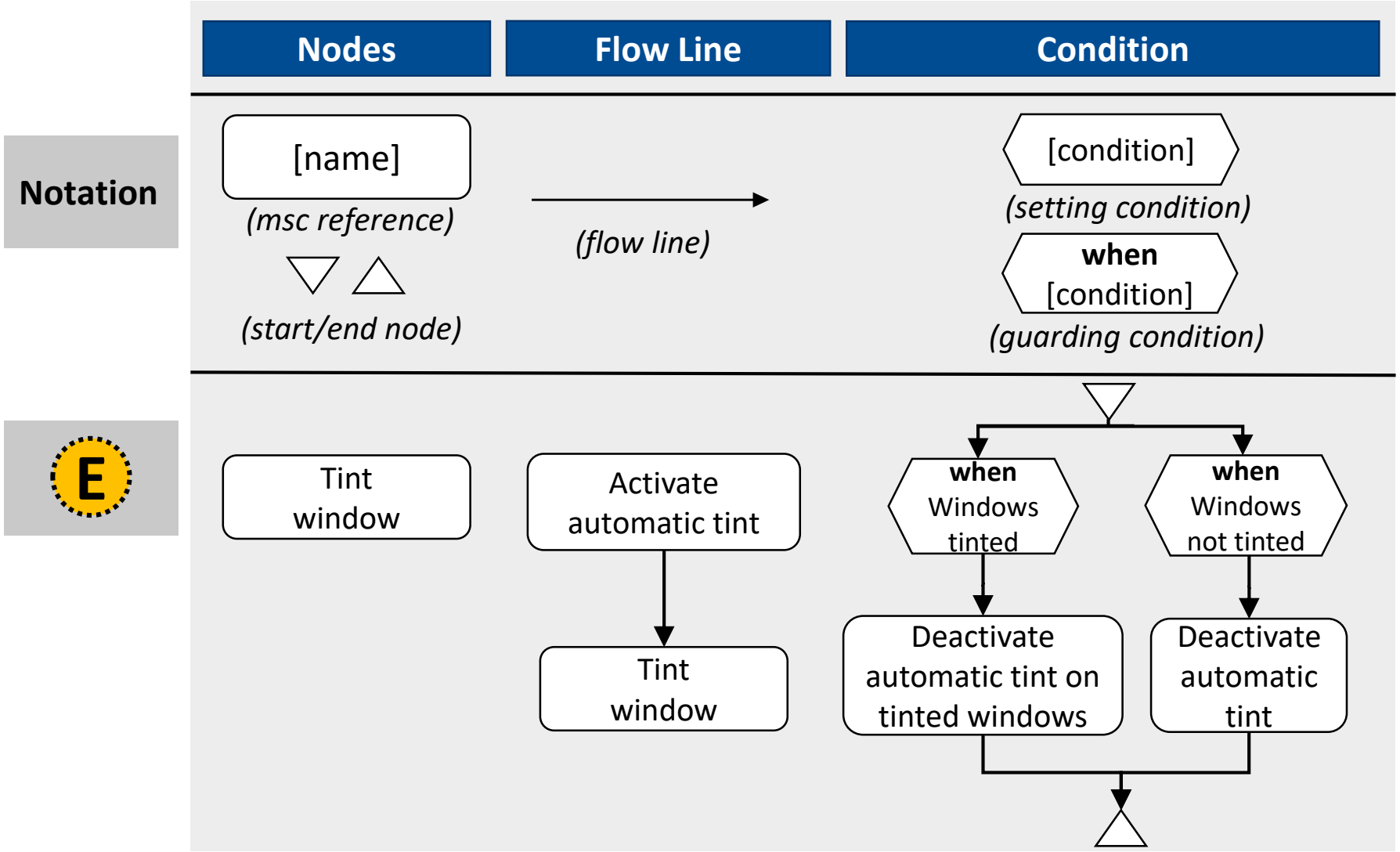
Corresponding bMSCs of the Dynamic Window Tint



Integration of the bMSCs



Modelling Constructs: hMSCs



Summary

- A MSC document is a container document, to specify all relevant scenarios.
- Basic MSCs define interaction-based system behavior by means of interaction sequences between actors, actors and the system or between system instances.
- bMSCs can be used to specify single scenarios for instance level and for type level descriptions.
- High-Level MSCs structure scenarios by specifying control flow between bMSCs.

Literature

[Caroll 2000] J.M. Caroll (Ed.): Making Use – Scenario-Based Design of Human Computer Interactions. MIT Press, Cambridge, 2000.

[Cockburn 2001] A. Cockburn: Writing effective Use Cases. Addison-Wesley, Bosten, 2001.

[ITU 2011] International Telecommunication Union: Recommendation Z.120 – Message Sequence Chart (MSC). International Standard, 2011.

[Jacobson et al. 1992] I. Jacobson, M. Christerson, P. Jonsson, G. Oevergaard: Object-Oriented Software Engineering – A Use Case Driven Approach. Addison-Wesley, Reading, 1992.

[Weidenhaupt et al. 1998] K. Weidenhaupt, K. Pohl, M. Jarke, P. Haumer: Scenario Usage in System Development – A Report on Current Practice. IEEE Software, Vol. 15, No. 2, IEEE Press, Los Alamitos, pp. 22-45.

Literature for Further Reading

[Alexander and Maiden 2004] I. Alexander, N. Maiden (Eds.): Scenarios, Stories, Use Cases – Through the system Development Life-Cycle. Wiley, Chichester, 2004.

[Haumer et al. 1998] P. Haumer, K. Pohl, K. Weidenhaupt: Requirements Elicitation and Validation with Real World Scenes. IEEE Transactions on Software Engineering, Vol. 24, No. 12, 1998, pp. 1036-1054.

[Rolland et al. 1998a] C. Rolland, C. Ben Achour, C. Cauvet, J. Raylt, A. Sutcliffe, N. Maiden, M. Jarke, P. Haumer, K. Pohl, E. Dubois, P. Heymans: A Proposal for a Scenario Classification Framework. Requirements Engineering Journal, Vol. 3, No. 1, Springer, Berlin, Heidelberg, 1998, pp. 23-47.

[Rolland et al. 1998b] C. Rolland, C. Souveyet, C. Ben Achour: Guiding Goal Modelling Using Scenarios. IEEE Transactions on Software Engineering, Vol. 24, No 1., 1998, pp. 1055-1071.

[Sutcliffe et al. 1998] A. Sutcliffe, N. Maiden, S. Minocha, M. Darrel: Supporting scenario-Based Requirements Engineering. IEEE Transactions on Software Engineering, Vol. 24, No. 12, 1998, pp. 1072-1088.

Image References

[1] Licensed by http://www.iconshock.com/

[2] Provided by Microsoft Office

Legend

Definition

Example

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Requirements Engineering & Management

Vielen Dank für Ihre Aufmerksamkeit

PALUNO
The Ruhr Institute for Software Technology