Embedded Systems Software Engineering

Jarosław Kuchta

Interaction modeling
Basic concepts

- **Interaction**
  - Message interchange between objects cooperating to realize some task in the context of collaboration

- **Collaboration**
  - Connection set between objects creating ways to interchange messages in an interaction

- **Message**
  - An information passed between objects as the consequence of some event

- **Event**
  - occurring of some situation which is important in some context; time of an event is atomic
Types of interaction diagrams

• **Collaboration diagrams**
  – graph-like connections between nodes modeling objects
  – message flow along connections

• **Sequence diagrams**
  – objects modeled in a row at a top of the diagram
  – vertical object lifelines going down
  – horizontal message passing between lifelines
Interaction diagrams usage

• System overall description
• Use case scenario description
• Class operation description
Collaboration diagrams elements

Collaboration Diagram:

- **User**
- **actor**
- **anObject:aClass**
- **connection**
- **message**
- **direction**
- **tag**
- **ordering number**

1: DoSomething() → anObject:aClass

Diagram elements:
- **Actor**
- **Message**
- **Direction**
- **Tag**
- **Ordering Number**
- **Connection**
Collaboration diagrams guidelines (1)

• Actors – user roles
• Objects – system components
• Connections – logical means of information passing (they do not have to be physical connections)
• Message – information passing and also operation call; operation must be defined in the message receiver class
Collaboration diagrams guidelines (2)

• One connection allows many various messages passed in various directions and in various time
• Message passing direction determined with an arrow
• Ordinal numbers determine message passing order
• Advanced specification can be used instead of the ordinal number
• If a time aspect is more important, then a sequence diagram should be used
Message signature (1)

• Ordinal number
  – can consist of many segments; each segment is a number determining an order at a specific embedding level: e.g. 1.2.1 precedes 1.2.2 and 1.3
  – alternate and concurrent messages are marked with letters, e.g.: 1.2a and 1.2b

• Preceding message determination instead of an ordinal number
  – Preceding message ordinal number (or a list of preceding message numbers) ending with a “/” sign
  – This message will be passed only if all the preceding messages are passed
Message signature (2)

• Guard condition
  – A condition checked when the message can be passed
  – Specified in brackets, e.g.: [n<10]

• Iteration clause
  – Iteration condition in brackets preceded by a “*” sign, e.g.: *[n<10] – means “while (n<10)”
  – Condition can be expressed in some language or pseudocode, e.g.: *[i:=1..10] – means “for (int i=1; i<=10; i++)”
Message signature (3)

• Result of an operation
  \[ \text{result} := \text{operation name (argument list)} \]

• Argument list
  – A list of any values assigned to operation parameters
  – Values can be expressed in some language or pseudocode
  – Result values or preceding arguments can be used
Message examples

• 1.2a: DoSomething()
• 3, 5/7: DoSomething()
• 4: [action=new] NewDocument(SelectType())
Sequence diagram

actor
User

object
anObject:aClass

life line

message
DoSomething()

message passing vector
focus of control

answer message passing
Sequence diagram guidelines

• Sequence diagram and collaboration diagram are semantically equivalent
• Only one message along message passing vector
• Diagram passing vector determines the direction of message passing
• Sequence numbers can be used but are optional
• Message passing ordered in time by vertical message vectors positions along life lines
• Guard conditions and iteration clauses can be used
• Answer vector are not described – if an operation result is important it should be described in the request vector
• An ordinal message passing vector can be used instead of an answer vector
• Focus of control is represents time when an object is active (can receive and send messages)
Object creation and destruction

Create() \rightarrow \text{anObject:aClass}
Concurrent threads of control

anObject:aClass
Recurrent

anObject:aClass
Precise timing

\( t_1 \) \hspace{1cm} \{t_2-t_1<1\ \text{min}\} \hspace{1cm} t_2 \hspace{1cm} t_3 \hspace{1cm} \{t_3'-t_3<10\ \text{ms}\} \hspace{1cm} t_3'
Sequence diagram example

1: Login() -> Active Directory
2: ok:= Authorization() -> Controller
3a:[not ok] Invalid login
4: Get Status() -> Controller
5: Menu
Collaboration diagram example

User

1: Login() \(\rightarrow\)

2: ok:= Authorization() \(\rightarrow\) Active Directory

3a: [not ok] \(\leftarrow\) Invalid login

4: Get Status() \(\rightarrow\) Controller

5: Menu \(\leftarrow\)
Bibliography