Refining UML specifications

- the STAIRS method

Ragnhild Kobro Runde
Joint work with Ketil Stølen and Øystein Haugen
Outline

• Introduction to sequence diagrams.
• What is refinement?
• The pragmatics of creating sequence diagrams.
• The pragmatics of refining sequence diagrams.
• Summary.
Example: Appointment system

• MakeAppointment
• CancelAppointment
• DecideAppTime
**UML sequence diagrams**

- **Partial ordering of events:**
  - The send event is ordered before the corresponding receive event.
  - Events on the same lifeline are ordered from the top and downwards.
- **CancelAppointment** specifies two traces.
Referencing

sd CancelAppointment

:Client  

:AppSystem

ref

DecideAppTime

cancel(appointment)

appointmentCancelled()

sd DecideAppTime

:Client  

:AppSystem

appointmentSuggestion(time)

yes()

appointmentMade()
UML 2.x operators

sd D1

: L1

alt

: L2

a

b

c

sd D2

: L1

alt

: L2

a

loop {0...4}

b
Sequence diagrams as example runs

- A sequence diagram is a partial description!
- Positive traces represent valid system behaviour.
- Negative traces represent invalid system behaviour.
- All remaining traces are called inconclusive.

```
sd CancelAppointment

:Client

  cancel(appointment)
  appointmentCancelled()
  appointmentSuggestion(time)
  yes()
  appointmentMade()

:AppSystem

```
Outline

• Introduction to sequence diagrams.
• What is refinement?
• The pragmatics of creating sequence diagrams.
• The pragmatics of refining sequence diagrams.
• Summary.
Refinement

- Refinement is a development step, adding more information to the model.
- Step-wise refinement.
- Compositional refinement.
Other changes

• Not all changes are refinements!
  - Correcting errors.
  - Handling changed requirements.
  - Describing the system addressing a different concern, i.e. from a different viewpoint.
Outline

• Introduction to sequence diagrams.
• What is refinement?

The pragmatics of creating sequence diagrams.
  - The pragmatics of alternative behaviour
  - The pragmatics of guards
  - The pragmatics of negative behaviour

• The pragmatics of refining sequence diagrams.
• Summary.
Alternative behaviour

- alt \(\approx\) similar traces, giving implementation freedom
- xalt \(\approx\) traces that must all be present in the system
Alternative behaviour (2)

```
sd CancelAppointment

:Client
  cancel(appointment)

:xalt
 errorMessage()
  appointmentCancelled()

:AppSystem

:ref
  DecideAppTime
```
- Use guards to constrain the situations in which the different alternatives are positive.
- Traces with a false guard are negative. Always make sure that for each alternative, the guard is sufficiently general to capture all possible situations in which the described traces are positive.
Guards (2)

- In an alt-construct, make sure that the guards are exhaustive.
- If doing nothing is valid, specify this by using the empty diagram, skip.
Negative behaviour

- To effectively constrain the implementation, the specification should include a reasonable set of negative traces.
- Use veto when the empty trace (i.e. doing nothing) should be positive, as when specifying a negative message in an otherwise positive scenario.
- Use refuse when specifying that one of the alternatives in an alt-construct represents negative traces.
- Use assert on an interaction fragment when all possible positive traces for that fragment have been described.
Outline

• Introduction to sequence diagrams.
• What is refinement?
• The pragmatics of creating sequence diagrams.
• The pragmatics of refining sequence diagrams.
• Summary.
Refinement in STAIRS

- Positive
- Inconclusive
- Negative

Supplementing → Inconclusive → Narrowing
Supplementing

- Use supplementing to add positive or negative traces to the model.
- All of the original positive traces must remain positive and all of the original negative traces must remain negative.
- Do not use supplementing on the operand of an assert.
Narrowing

- Use narrowing to remove underspecification by redefining positive traces as negative.
- All original negative traces must remain negative.
• Guards may be added to an alt/xalt-construct as a legal narrowing step.
• Guards may be narrowed, i.e. the refined condition must imply the original one.
Outline

• Introduction to sequence diagrams.
• What is refinement?
• The pragmatics of creating sequence diagrams.
• The pragmatics of refining sequence diagrams.

Summary.
Summary

• Take special care when specifying
  - alternative behaviours
  - negative behaviours

• Refinement means adding more information
  - supplementing: new positive/negative traces
  - narrowing: positive traces redefined as negative

• These refinement notions enable
  - step-wise refinement
  - compositional refinement
Evaluation

• Formal proofs
  - Transitivity.
  - Monotonicity.

• Other examples/cases
  - Automatic teller machine.
  - Vending machine with tea/coffee.
  - Network communication.
  - Games: tic-tac-toe, flipping a coin, throwing a dice.
  - Gambling machine.
  - BuddySync: automatically matching service providers with users of those services.
Thank you!

ragnhild.runde@ifi.uio.no

http://heim.ifi.uio.no/~ragnhilk/