INF5150
Unassailable IT-systems
Exam Preparation Notes

Brian Elvesæter
(briane@ifi.uio.no)

Table of contents

1 Agile modelling with UML................................................................................................................................. 3
  1.1 Concepts ......................................................................................................................................................... 3
  1.1.1 Sequence diagram ................................................................................................................................. 3
  1.1.2 State machines ....................................................................................................................................... 4
  1.1.3 Consistency in UML diagrams ............................................................................................................... 4
  1.2 Modelling patterns ........................................................................................................................................ 6
  1.2.1 Decomposition ....................................................................................................................................... 6
  1.2.2 Composite structure ............................................................................................................................. 7
  1.2.3 Services as substatemachines ............................................................................................................... 8
  1.2.4 Sessions as concurrent parts ............................................................................................................... 9
  1.2.5 Sessions and routing ........................................................................................................................... 11
  1.2.6 Classes and signals ............................................................................................................................. 15
  1.2.7 Archive ................................................................................................................................................ 16
  1.2.8 Testing ................................................................................................................................................ 17
  1.2.9 Robustification ..................................................................................................................................... 18
  1.2.10 Intra process communication ......................................................................................................... 20

2 STAIRS refinements ........................................................................................................................................ 22
  2.1 Concepts ..................................................................................................................................................... 22
  2.1.1 Sequence diagram ................................................................................................................................. 22
  2.1.2 Traces .................................................................................................................................................... 22
  2.1.3 Causality and weak sequencing ........................................................................................................... 22
  2.1.4 Example ................................................................................................................................................ 23
  2.1.5 Executions ............................................................................................................................................. 23
  2.1.6 Underspecification and non-determinism ........................................................................................... 24
  2.2 Refinement .................................................................................................................................................. 24
  2.2.1 Supplementing ...................................................................................................................................... 24
  2.2.2 Narrowing .......................................................................................................................................... 24
  2.2.3 Detailing .............................................................................................................................................. 24
  2.2.4 General refinement .............................................................................................................................. 25
  2.2.5 Limited refinement ............................................................................................................................... 25
  2.3 Pragmatics ................................................................................................................................................. 26
  2.3.1 Alt vs. xalt .......................................................................................................................................... 26
  2.3.2 Guards ................................................................................................................................................. 26
  2.3.3 Negations .......................................................................................................................................... 26
  2.3.4 Seq ...................................................................................................................................................... 26

3 CORAS security (risk) analysis ..................................................................................................................... 28
  3.1 Concepts ................................................................................................................................................... 28
<table>
<thead>
<tr>
<th>Section</th>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.1</td>
<td>Conceptual model</td>
<td>28</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Modelling language</td>
<td>28</td>
</tr>
<tr>
<td>3.2</td>
<td>Methodology</td>
<td>29</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Step 1: Introduction</td>
<td>29</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Step 2: High-level analysis</td>
<td>29</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Step 3: Approval</td>
<td>30</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Step 4: Risk identification workshop</td>
<td>31</td>
</tr>
<tr>
<td>3.2.5</td>
<td>Step 5: Risk estimation workshop</td>
<td>31</td>
</tr>
<tr>
<td>3.2.6</td>
<td>Step 6: Risk evaluation workshop</td>
<td>32</td>
</tr>
<tr>
<td>3.2.7</td>
<td>Step 7: Risk treatment workshop</td>
<td>33</td>
</tr>
<tr>
<td>3.3</td>
<td>Calculus</td>
<td>34</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Initiate rule</td>
<td>34</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Leads-to rule</td>
<td>34</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Mutually exclusive vertices rule</td>
<td>35</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Independent vertices rule</td>
<td>35</td>
</tr>
</tbody>
</table>
1 Agile modelling with UML

1.1 Concepts

1.1.1 Sequence diagram

![Sequence Diagram](image)

**Figure 1: Sequence diagram overview**

![Sequence Diagram Elements](image)

**Figure 2: Sequence diagram elements**

- a) Interaction [Frame, Sequence Diagram]
- b) Lifeline
- c) Combined fragment [alt-fragment]
- d) Message
- e) Gate
1.1.2 State machines

![State Machine Diagram]

**Figure 3: State machine overview**

1.1.3 Consistency in UML diagrams

- Interactions are incomplete specifications.
- State machines are complete specifications and may model (some of) the behaviour in the interaction.
- The order of OccurrenceSpecifications along a Lifeline is significant denoting the order in which these OccurrenceSpecifications will occur.

![State Machine Diagram]

**Figure 4: State machine elements**

a) Initial state (pseudo state)
b) Choice (pseudo state)
c) Final state
d) Submachine state
e) Exit-point
f) Trigger
g) Effect
h) State Machine
Figure 5: Model-time consistency

Figure 6: Run-time consistency
1.2 Modelling patterns

1.2.1 Decomposition

Figure 7: The essence of decomposition

Figure 8: Decomposition covering refs and combined fragments
1.2.2 Composite structure

Create a composite structure with three parts:

- **[SystemName]Controller** part instance of **[SystemName]Controller** state machine
- **[SystemName]Process** part instance of **[SystemName]Process** state machine
- **database** part instance of **Archive** state machine
  - Responsible for persistent storage

Write down any assumptions when describing the composite structure.
1.2.3 Services as substatemachines

Figure 10: ICU services
1.2.4 Sessions as concurrent parts

- Each initiative by a user is represented by an instantiation of a state machine (a session)
  - With all the temporary data associated with that user
  - Taking care of all the communication related to that user
- The session is generated when the user initiates a service
- The session is terminated when the service is finished
Figure 12: Session generator

Figure 13: Enhancing the behaviour
1.2.5 Sessions and routing

- One session per concurrent user initiative
  - The state machine type ICUprocess describes the session
- One receptionist state machine creates the sessions
  - When the session initiation arrives
  - here: Sms-message
- Centralized routing through the receptionistcontr
  - One routing port (SimpleIdRouterMediator)
  - all signals aiming for a session are sent through contr
- Terminating the session by reaching the final state
  - and the runtime system machinery takes care of the rest

![Diagram](image.png)

Figure 14: Simple routing
Figure 15: One-to-many port

Figure 16: Generate session and adding the ID
Figure 17: Connecting connectors

Figure 18: Forwarding from port
Figure 19: Terminating a session

At the final state the ICUprocess representing the session will terminate

The compiler and JavaFrame make sure that the implementation gets rid of the session

Figure 20: Routing in TaskSolver
1.2.6 Classes and signals

Figure 21: Classes and the signals

Figure 22: Routing using the internal signal
1.2.7 Archive

Figure 23: Adding a new service

Figure 24: Enhancing the archive for the new service
1.2.8 Testing

Figure 25: Test execution

Figure 26: Test context
1.2.9 Robustification

- Data may have strange syntax or values
  - Checking the data invariant
  - We have looked at data checks for ICUcontroller

- An unexpected signal arrives
  - We explicitly describe every conceivable transition (Explicit transitions please!)
  - UML State Machines also define default transitions
    - where the signal is just discarded/consumed
    - We believe that default transitions are a warning of design flaw
  - Not all signals can be properly handled at any time
    - We may defer a signal to a state where the signal can be dealt with
  - We have looked at this for ICUprocess’KML

- No signal arrives
  - We guard our protocols/services with timers (ICUprocess’KML)
- Security issues
  - Authentication + logging + statistics
  - Check for registration in ICUprocess' Hotpos

- Availability issues
  - Self tests (we shall improve the Archive)
1.2.10 Intra process communication

Figure 29: Introducing new requirement that users must accept being positioned

Figure 30: Revised state machine
Figure 31: New internal signals

Figure 32: New communication path must be added
2 STAIRS refinements

2.1 Concepts

2.1.1 Sequence diagram

![Sequence Diagram]

Figure 33: Sequence diagram

2.1.2 Traces

- Traces are used to represent system runs mathematically.
- Traces are sequences of events:
  - <e1, e2, e3, e4, e1, e2, e5, …………..>
- Two kinds of events:
  - transmission events <!event>
  - reception events <?event>

2.1.3 Causality and weak sequencing

- Causality:
  - a message can never be received before it has been transmitted
  - the transmission event for a message is therefore always ordered before the reception event for the same message
- Weak sequencing:
  - events from the same lifeline are ordered in the trace in the same order as on the lifeline
2.1.4 Example

There are six possible traces if time information is ignored:

```
<!a, ?a, !b, ?b, !c, ?c, !d, ?d>
<!a, ?a, !b, ?b, !c, !d, ?c, ?d>
<!a, ?a, !b, ?b, !d, !c, ?c, ?d>
<!a, ?a, !b, !c, ?b, ?c, !d, ?d>
<!a, ?a, !b, !c, !d, !c, !d, ?d>
<!a, ?a, !b, !c, !d, ?c, ?b, !d, ?d>
```

Figure 34: Example

2.1.5 Executions

- Each positive execution is represented by a trace

Figure 35: Trace tree for the example

Figure 36: Positive, inconclusive and negative executions
Each negative execution is represented by a trace
The semantics of a sequence diagram is a pair of sets of traces (Positive, Negative)
All other traces over the actual alphabet of events are inconclusive

2.1.6 Underspecification and non-determinism
- Underspecification: Several alternative behaviours are considered equivalent (serve the same purpose).
- Inherent non-determinism: Alternative behaviours that must all be possible for the implementation.

2.2 Refinement
A sequence diagram B is a refinement of a sequence diagram A if
- every trace classified as negative by A is also classified as negative by B
- every trace classified as positive by A is classified as either positive or negative by B

Supplementing means to add new behaviours to the specification, i.e., re-categorising inconclusive traces as either positive or negative.
- Use supplementing to add positive or negative traces to the specification.
- When supplementing, 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.

2.2.2 Narrowing
Narrowing which means to reduce the set of possible system behaviours, i.e., reducing underspecification by redefining positive traces as negative.
- Use narrowing to remove underspecification by redefining positive traces as negative.
- In cases of narrowing, all of the original negative traces must remain negative.
- Guards may be added to an alt-construct as a legal narrowing step.
- Guards may be added to an xalt-construct as a legal narrowing step.
- Guards may be narrowed, i.e. the refined condition must imply the original one.

2.2.3 Detailing
Detailing which means to add more details to the specification by decomposing lifelines, maintaining positive traces as positive and negative traces as negative in relation to detailing.
- Use detailing to increase the level of granularity of the specification by decomposing lifelines.
- When detailing, document the decomposition by creating a mapping \( L \) from the concrete to the abstract lifelines.
- When detailing, make sure that the refined traces are equal to the original ones when abstracting away internal communication and taking the lifeline mapping into account.

### 2.2.4 General refinement

Supplementing, narrowing and detailing are all important refinement steps when developing interactions. Often, it is useful to combine two or three of these activities into a single refinement step. We therefore define a **general refinement** notion, of which supplementing, narrowing and detailing are all special cases.

- Use general refinement to perform a combination of supplementing, narrowing and detailing in a single step.
- To define that a particular trace must be present in an implementation use `xalt` and `assert` to characterize an obligation with this trace as the only positive one and all other traces as negative.

#### General refinement (of sets of interaction obligations):

- \( d \sim d' \) if \( \forall o \in [[d]] \exists o' \in [[d']] : o \sim o' \)
- \( d' \) is a general refinement of \( d \) if
  - for every interaction obligation \( o \) in \([[[d]]]\) there is at least one interaction obligation \( o' \) in \([[[d']]\)
    such that \( o' \) is a general refinement of \( o \)
- New interaction obligations may also be added
  - that do not refine any obligation at the abstract level

\[
[[d]]: \quad \begin{array}{c}
\mathcal{H}(p_1 \cup n_1) \\
\mathcal{H}(p_1 \cup n_2) \\
\mathcal{H}(p_2 \cup n_1) \\
\mathcal{H}(p_2 \cup n_2)
\end{array}
\quad \begin{array}{c}
\mathcal{H}(p_2 \cup n_3) \\
\mathcal{H}(p_2 \cup n_2)
\end{array}
\]

\[
[[d']] : \quad \begin{array}{c}
\mathcal{H}(p_1 \cup n_1) \\
\mathcal{H}(p_1 \cup n_2) \\
\mathcal{H}(p_2 \cup n_1) \\
\mathcal{H}(p_2 \cup n_2)
\end{array}
\quad \begin{array}{c}
\mathcal{H}(p_2 \cup n_3) \\
\mathcal{H}(p_2 \cup n_2)
\end{array}
\]

#### Figure 38: General refinement

### 2.2.5 Limited refinement

**Limited refinement** is a special case of general refinement, with less possibilities for adding new interactions obligations.

- Use `assert` and `switch` to limited refinement in order to avoid fundamentally new traces being added to the specification.
- To specify globally negative traces, define these as negative in all operands of `xalt`, and switch to limited refinement.

**Limited refinement:**

- Limits the possibility of adding new interaction obligations
  - Typically used at a later stage
  - \( d' \) is a limited refinement of \( d \) if
  - \( d' \) is a general refinement of \( d \), and
    - every interaction obligation in \([[[d']]\) is a general refinement of at least one interaction obligation in \([[[d]]\)
2.3 Pragmatics

2.3.1 alt vs. xalt
- Use `alt` to specify alternatives that represent similar traces, i.e. to model underspecification.
- Use `xalt` to specify alternatives that must all be present in an implementation, i.e. to model
  - inherent nondeterminism, as in the specification of a coin toss.
  - alternative traces due to different inputs that the system must be able to handle (as in Fig. 4);
  - alternative traces where the conditions for these being positive are abstracted away (as in Fig. 5).

2.3.2 Guards
- Use guards in an alt/xalt-construct to constrain the situations in which the different alternatives are positive.
- Always make sure that for each alternative, the guard is sufficiently general to capture all possible situations in which the described traces are positive.
- In an alt-construct, make sure that the guards are exhaustive. If doing nothing is valid, specify this by using the empty diagram, skip.

2.3.3 Negations
STAIRS defines three negation operators:
- `refuse` that can be used to specify that one of the alternatives in an `alt` construct represents negative traces.
- `veto` that can be used to specify that the empty trace should be positive.
- `assert` on an interaction fragment that can be used to specify that all possible traces for that fragment have been described.

The pragmatics of negations:
- To effectively constrain the implementation, the specification should include a reasonable set of negative traces.
- Use `refuse` when specifying that one of the alternatives in an alt-construct represents 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 `assert` on an interaction fragment when all possible positive traces for that fragment have been described.

2.3.4 `seq`
- Be aware that by weak sequencing,
- A positive sub-trace followed by a positive sub-trace is positive.
- A positive sub-trace followed by a negative sub-trace is negative.
- A negative sub-trace followed by a positive sub-trace is negative.
- A negative sub-trace followed by a negative sub-trace is negative.
- The remaining trace combinations are inconclusive.
3 CORAS security (risk) analysis

3.1 Concepts

3.1.1 Conceptual model

![Conceptual model diagram]

Figure 40: Conceptual model

3.1.2 Modelling language

![Graphical symbols]

Figure 41: Graphical symbols
3.2 Methodology

![Figure 42: Seven steps](image)

3.2.1 Step 1: Introduction

Tasks:
- The security analysis method is introduced.
- The client presents the goals and the target of the analysis.
- The focus and scope of the analysis is set.
- The meetings and workshops are planned.

Modelling guidelines:
- At this early stage of the analysis it can be useful to describe the target with informal like drawings, pictures or sketches on a blackboard.
- The presentation can later be supplemented with more formal modelling techniques such as UML or data flow-diagram.

3.2.2 Step 2: High-level analysis

Tasks:
- The target as understood by the analysts is presented.
- The assets are identified.
- A high-level analysis is conducted.

Modelling guidelines:
- Target description:
  - Use a formal or standardised notation such as UML, but ensure that the notation is explained thoroughly so that the participants understand it.
  - Create models of both the static and the dynamic features of the target.
  - Static maybe hardware configurations, network design etc., while dynamic may be work processes, information etc.
  - For the static parts of the description UML class diagrams and UML collaboration diagrams (or similar notations) are recommended.
  - For the dynamic parts we recommend UML activity diagrams and UML sequence diagrams (or similar notations).
• Asset diagrams:
  o Draw a region that logically or physically represents the target of analysis.
  o Place the direct assets within the region.
  o Place the indirect assets outside the region. Indirect assets are harmed as a consequence of a direct asset being harmed first.
  o Indicate with arrows how assets may affect other assets.
  o Assets may be ranked according to their importance. (1= very important, 5= minor importance).
  o If the analysis has more than one client, the clients should be associated with their assets.

Example:

<table>
<thead>
<tr>
<th>Asset</th>
<th>Importance</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Health records</td>
<td>2</td>
<td>Direct asset</td>
</tr>
<tr>
<td>Provision of telecardiology service</td>
<td>3</td>
<td>Direct asset</td>
</tr>
<tr>
<td>Public’s trust in system</td>
<td>3</td>
<td>Indirect asset</td>
</tr>
<tr>
<td>Patient’s health</td>
<td>1</td>
<td>Indirect asset</td>
</tr>
</tbody>
</table>

Figure 43: Asset table

3.2.3 Step 3: Approval

Tasks:
• The client approves target descriptions and asset descriptions.
• The assets should be ranked according to importance.
• Consequence scales must be set for each asset within the scope of the analysis.
• A likelihood scale must be defined.
• The client must decide risk evaluation criteria for each asset within the scope of the analysis.

Example:

<table>
<thead>
<tr>
<th>Consequence value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catastrophic</td>
<td>1000+ health records (HRs) are affected</td>
</tr>
<tr>
<td>Major</td>
<td>100-1000 HRs are affected</td>
</tr>
<tr>
<td>Moderate</td>
<td>10-100 HRs</td>
</tr>
<tr>
<td>Minor</td>
<td>1-10 HRs are affected</td>
</tr>
<tr>
<td>Insignificant</td>
<td>No HR is affected</td>
</tr>
</tbody>
</table>

Figure 44: Consequence scale for each asset (should be different for each asset)

<table>
<thead>
<tr>
<th>Likelihood value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certain</td>
<td>Five times or more per year (50-<em>: 10y = 5-</em>: 1y)</td>
</tr>
<tr>
<td>Likely</td>
<td>Two to five times per year (21-49: 10y = 2,1-4,9: 1y)</td>
</tr>
<tr>
<td>Possibly</td>
<td>Once a year (6-20: 10y = 0,6-2: 1y)</td>
</tr>
<tr>
<td>Unlikely</td>
<td>Less than once per year (2-5: 10y = 0,2-0,5: 1y)</td>
</tr>
<tr>
<td>Rare</td>
<td>Less than once per ten years (0-1:10y = 0-0,1:1y)</td>
</tr>
</tbody>
</table>

Figure 45: Likelihood scale (typically for all assets – may be different for each asset)
3.2.4 Step 4: Risk identification workshop

Tasks:
- The initial threat diagrams should be completed with identified threats, vulnerabilities, threat scenarios and unwanted incidents.

Modelling guidelines for threat diagrams:
- Use the region from the asset diagram and add more regions if necessary,
- Model different kinds of threats in separate diagrams, e.g. deliberate sabotage in one diagram, mistakes in another, environmental in a third, etc.
- Assets are listed to the right, outside the region,
- Threats are placed to the left in the region, while threats that can be classified as external are placed outside the region,
- Unwanted incidents are placed within the region in relation to the assets on which they have an impact,
- Assets that are not harmed by any incidents are removed from the diagram,
- Add threat scenarios between the threats and the unwanted incidents in the same order as they occur in real time (i.e. in a logical sequence),
- Insert the vulnerabilities before the threat scenario or unwanted incident they lead to.

Example:

3.2.5 Step 5: Risk estimation workshop

Tasks:
- Every threat scenario must be given a likelihood estimate and unwanted incident likelihoods are based on these.
- Every relation between an unwanted incident and an asset must be given a consequence estimate.

**Modelling guideline for risk estimation on threat diagrams:**
- Add likelihood estimates to the threat scenarios.
- Add likelihood estimates to the unwanted incidents, based on the threat scenarios.
- Annotate each unwanted incident-asset relation with a consequence taken from the respective asset's consequence scale.

**Example:**

![Threat diagram with likelihood and consequence estimates](image)

**Figure 48: Threat diagram with likelihood and consequence estimates**

### 3.2.6 Step 6: Risk evaluation workshop

**Tasks:**
- Likelihood and consequence estimates should be confirmed or adjusted.
- The final adjustments of the acceptable area in the risk matrices should be made (if needed).
- An overview of the risks maybe given in a risk diagram.

**Modelling guidelines for the risk diagrams:**
- Use the threat diagram and replace all unwanted incidents with risk symbols, showing a short risk description and whether the risk is acceptable or not.
- Remove threat scenarios and vulnerabilities, but keep the relations between the threats and the risks.
- If useful, split the risk diagrams into several diagrams according to type of threat, part of the target or asset importance (e.g. show all risks related to network, all risks for specific assets etc.).

**Example:**
3.2.7 Step 7: Risk treatment workshop

Tasks:
- Add treatments to threat diagrams.
- Estimate the cost/benefit of each treatment and decide which ones to use.
- Show treatments in risk overview diagrams.

Modelling guidelines:
- Treatment diagrams:
  - Use the threat diagrams as a basis and annotate all arrows from unwanted incidents to assets with risk icons. Show only the unacceptable risks.
  - Annotate the diagram with treatments, pointing to where they will be applied.
- Treatment overview diagrams:
  - Use the risk diagrams as a basis, remove the acceptable risks.
  - Add treatments according to the treatment diagram(s).

Example:
3.3 Calculus

3.3.1 Initiate rule

If the vertices t and v are related by initiate:

\[
\frac{t \xrightarrow{l} v}{(t \parallel v)(l)}
\]

3.3.2 Leads-to rule

If the vertices \(v_1\) and \(v_2\) are related by leads-to:
3.3.3 Mutually exclusive vertices rule

If the vertices $v_1$ and $v_2$ are mutually exclusive:

\[
\frac{v_1(f_1) \cdot v_2(f_2)}{(v_1 \cap v_2)(f \cdot l)}
\]

3.3.4 Independent vertices rule

If the vertices $v_1$ and $v_2$ are statistically independent:

\[
\frac{v_1(f_1) \cdot v_2(f_2)}{(v_1 \cup v_2)(f_1 + f_2)} = 1 - ((1-f_1) \cdot (1-f_2))
\]