Modeling Requirements

Critical Embedded Systems

*Dr. Ákos Horváth*

Prepared by
Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Dept. of Measurement and Information Systems
© All rights reserved.

This material can only be used by participants of the course.
Overview

- Modeling Textual Requirements
- Modeling Requirements with Use Cases
- Modeling Flow Based Behavior with Activities
Modeling Textual Requirements

Context of the Modeling Aspect
Sample Requirements (Cyber-physical Agricultural System)
Modeling Elements & Notation
Summary
Roots & Relations

- Document based system development
  - Formulated requirements textually (e.g. in Word)
  - Handled by Req. management tools (e.g. DOORS)
  - Challenge: complexity
System Modeling Process

1. Collect Textual Requirements
2. Model System Context
3. Model Functional Requirements with Use Cases
4. Elaborate Functional Requirements with Activities
5. Create Data Model
6. Model Interactions between System and Environment
7. Derive System State Model
8. Derive System Interfaces
9. Model System Structures
What are the main requirements formulated textually and what are their hierarchy?
Objectives

- Provides linkage between traditional textual and model based requirements specifications
- Helps establishing relations between requirements
  - Containment hierarchy
  - Derivation
  - Reusing between projects
- Provides traceability of requirements
Modeling Textual Requirements

Context of the Modeling Aspect

Sample Requirements (Cyber-physical Agricultural System)

Modeling Elements & Notation

Summary
Cyber-physical system

- American terminology
  - Novel buzz-word for embedded system
  - In EU it is ~ „Internet of things”

„Cyber-Physical Systems (CPS) are engineered systems comprising interacting physical and computational components. In CPS, computation and communication are deeply embedded in and interacting with physical processes to add new capabilities and characteristics to physical systems.”

- E.g., acoustic sniper detection system
Example requirements

Design a simple Cyber-physical agricultural system (CPAS), which helps a farmer with his/her everyday life using sensors to measure the environment and react to its changes by using automated operations like irrigation, mowing and spraying.

Requirements

- The CPAS system is capable of measuring the environment through its sensors.
- The CPAS uses the following sensors: temperature, humidity, luminance, rain.
- The CPAS can execute operations to change its surrounding environment.
- These operations can be mowing, irrigation and spraying.
- The mowing operation signals the robot mower to execute its programmed task.
- If the mower robot executes its task without any problem it returns to its refueling station.
- If the mower robot fails to complete its task, it sends a notification about its status.
Example requirements (con’t)

- The irrigation operation simply activates the pre-installed irrigation-system.
- If the irrigation-system fails, it sends a notification about its status.
- Whenever a notification arrives the CPAS signals the farmer based on the configured communication mean.
- The spraying operation signals the laborers to execute the spraying task.
- The laborers report to the CPAS when they finished their task.
- In case an error occurs during the spraying the laborers submit a form to the CPAS and it notifies the farmer.
- The farmer can configure the system, when to activate its operations based on its sensor inputs.
- The farmer can shut down the CPAS system that immediately stops all of its active operations.
- The system shall provide diagnostic information about its components for maintenance.
# Modeling Textual Requirements

<table>
<thead>
<tr>
<th>Context of the Modeling Aspect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample Requirements (Cyber-physical Agricultural System)</td>
</tr>
<tr>
<td><strong>Modeling Elements &amp; Notation</strong></td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
Example – Top Level Requirements

- **Name:** Performing agricultural tasks
- **Id:** 1
- **Text:** The system shall support execution of autonomous and manual agricultural tasks according to predefined rules taking into account environmental effects.

- **Environment measurement**
  - **Id:** 1.1
  - **Text:** The CPAS system shall be capable of measuring the environment through its sensors.

- **Behavior configuration**
  - **Id:** 1.2
  - **Text:** The farmer can configure the system, when to activate its operations based on its sensor inputs.

- **Operation execution**
  - **Id:** 1.3
  - **Text:** The CPAS shall execute operations to change its surrounding environment.

- **Sensors**
  - **Id:** 1.1.1
  - **Text:** The CPAS uses the following sensors: temperature, humidity, luminance, rain (gauge).

- **Farmer notification**
  - **Id:** 1.3.1
  - **Text:** Whenever a notification arrives the CPAS signals the farmer based on the configured communication mean.

- **System shutdown**
  - **Id:** 1.3.2
  - **Text:** The farmer can shut down the system that immediately stops all of its active operations.

- **System Maintenance**
  - **Id:** 2
  - **Text:** The system shall provide diagnostic information about its components for maintenance.
Example – Further Decomposed

- **Supported operations**
  - Id = 1.4
  - Text = "Supported operations are mowing, irrigation and spraying."

- **Mowing**
  - Id = 1.4.1
  - Text = "The mowing operation signals the robot lawn-mower to execute its programmed task."

- **Irrigation**
  - Id = 1.4.2
  - Text = "The irrigation operation simply activates the pre-installed irrigation system."

- **Spraying**
  - Id = 1.4.3
  - Text = "The spraying operation signals the laborers to execute the spraying task."

- **Execution of mowing**
  - Id = 1.4.1.1
  - Text = "The mower robot executes its task without any problem it returns to its refueling station."

- **Failure of mowing**
  - Id = 1.4.1.2
  - Text = "If the mower robot fails to complete its task, it sends a notification about its status."

- **Execution of irrigation**
  - Id = 1.4.2.1
  - Text = "The irrigation operation simply activates the pre-installed irrigation system."

- **Failure of irrigation**
  - Id = 1.4.2.2
  - Text = "If the irrigation system fails, it sends a notification about its status."

- **Execution of spraying**
  - Id = 1.4.3.1
  - Text = "The spraying operation signals the laborers to execute the spraying task."

- **Failure of spraying**
  - Id = 1.4.3.2
  - Text = "In case an error occurs during the spraying the laborers submit a form to the CPAS and it notifies the farmer.”
Example – Full Hierarchy

[Diagram showing a hierarchical structure of functional requirements, including nodes for Environment measurement, Farmer notification, System shutdown, Mowing, Irrigation, Spraying, etc., with associated text for each node describing the task or requirement.]

Example:
- **Functional Requirement**: Performing agricultural tasks
  - **Environment measurement**: The CPAS system shall be capable of measuring the environment through its sensors.
  - **Behavior configuration**: The farmer can configure the system, when to activate its operations based on its sensor inputs.
  - **Operation execution**: The CPAS shall execute operations to change its surrounding environment.
- **Function Requirement**: Mowing
  - **Execution of mowing**: The mowing operation signals the robot lawn-mower to execute its programmed task.
  - **Configuration of mowing**: The mowing operation simply activates the pre-installed irrigation system.
- **Function Requirement**: Irrigation
  - **Execution of irrigation**: The irrigation operation signals the irrigation system to execute the spraying task.
  - **Configuration of irrigation**: The irrigation system signals the irrigation system to execute the spraying task.

Example:
- **Function Requirement**: Reporting of spraying
  - **Successful spraying**: The farmers report the system when they finished their task.
  - **Failed spraying**: In case an error occurs during the spraying, the farmers submit a form to the CPAS and it notifies the farmer.
<table>
<thead>
<tr>
<th>#</th>
<th>Id</th>
<th>Name</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Performing agricultural tasks</td>
<td>The CPAS system shall be capable of measuring the environment through its sensors.</td>
</tr>
<tr>
<td>2</td>
<td>1.1</td>
<td>Environment measurement</td>
<td>The CPAS uses the following sensors: temperature, humidity, luminance, rain (gauge).</td>
</tr>
<tr>
<td>3</td>
<td>1.1.1</td>
<td>Sensors</td>
<td>The farmer can configure the system, when to activate its operations based on its sensor inputs.</td>
</tr>
<tr>
<td>4</td>
<td>1.2</td>
<td>Behavior configuration</td>
<td>The CPAS shall execute operations to change its surrounding environment.</td>
</tr>
<tr>
<td>5</td>
<td>1.3</td>
<td>System shutdown</td>
<td>Whenever a notification arrives the CPAS signals the farmer based on the configured communication mean.</td>
</tr>
<tr>
<td>6</td>
<td>1.3.1</td>
<td>Supported operations</td>
<td>The farmer can shut down the system that immediately stops all of its active operations.</td>
</tr>
<tr>
<td>7</td>
<td>1.4</td>
<td>Supported operations</td>
<td>Supported operations are mowing, irrigation and spraying.</td>
</tr>
<tr>
<td>8</td>
<td>1.4.1</td>
<td>Mowing</td>
<td>The mowing operation signals the robot lawn-mower to execute its programmed task.</td>
</tr>
<tr>
<td>9</td>
<td>1.4.1.1</td>
<td>Successful mowing</td>
<td>If the mower robot executes its task without any problem it returns to its refueling station.</td>
</tr>
<tr>
<td>10</td>
<td>1.4.1.1</td>
<td>Failed mowing</td>
<td>If the mower robot fails to complete its task, it sends a notification about its status.</td>
</tr>
<tr>
<td>11</td>
<td>1.4.1.2</td>
<td>Configuration of mowing</td>
<td>The irrigation operation simply activates the pre-installed irrigation-system.</td>
</tr>
<tr>
<td>12</td>
<td>1.4.1.2</td>
<td>Failed irrigation</td>
<td>If the irrigation-system fails, it sends a notification about its status.</td>
</tr>
<tr>
<td>13</td>
<td>1.4.2</td>
<td>Irrigation</td>
<td>The irrigation operation simply activates the pre-installed irrigation-system.</td>
</tr>
<tr>
<td>14</td>
<td>1.4.2.1</td>
<td>Execution of irrigation</td>
<td>If the irrigation-system fails, it sends a notification about its status.</td>
</tr>
<tr>
<td>15</td>
<td>1.4.2.2</td>
<td>Failed irrigation</td>
<td>The system shall support the control of manual spraying.</td>
</tr>
<tr>
<td>16</td>
<td>1.4.3</td>
<td>Spraying</td>
<td>The spraying operation signals the laborers to execute the spraying task.</td>
</tr>
<tr>
<td>17</td>
<td>1.4.3.1</td>
<td>Reporting of spraying</td>
<td>The laborers report to the system when they finished their task.</td>
</tr>
<tr>
<td>18</td>
<td>1.4.3.2</td>
<td>Successful spraying</td>
<td>The laborers report to the system when they finished their task.</td>
</tr>
<tr>
<td>19</td>
<td>1.4.3.2</td>
<td>Failed spraying</td>
<td>In case an error occurs during the spraying the laborers submit a form to the CPAS and it notifies the farmer.</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>System Maintenance</td>
<td>The system shall provide diagnostic information about its components for maintenance.</td>
</tr>
</tbody>
</table>
Requirements Trace Relations

- **Refine**
  - Depicts a model element that clarifies a requirement
  - Typically a use case or a behavior

- **Satisfy**
  - Depicts a design or implementation model element that satisfies the requirement

- **Verify**
  - Used to depict a test case that is used to verify a requirement

- **Derive**
  - Used when a requirement is derived from another requirement based on analysis
  - Typically at the next level of the system hierarchy

- **Copy**
  - Supports reuse by copying requirements to other namespaces
  - Master-slave relation between requirements

- **Trace**
  - General trace relationship
  - Between requirement and any other model element
Example refine relationship

```mermaid
diagram refine
  start as «functionalRequirement»
  label Performing agricultural tasks
  id = "1"
  text = "The system shall support execution of autonomous and manual agricultural tasks according to predefined rules taking into account environmental effects."

  refine[«refine»] system use case
  label Perform agricultural tasks

  refine[«refine»] activity
  label Perform agricultural tasks

  «functionalRequirement»
  label Performing agricultural tasks
  refine[«refine»] requirement
  label RefinedBy
  text = "The system shall support execution of autonomous and manual agricultural tasks according to predefined rules taking into account environmental effects."
```

Direct notation

Compartment notation

Callout notation
Example trace relationships

req [Package] Functional Requirements [ Functional requirement traces ]

- **Behavior configuration**
  - Id = "1.2"

- **Configuration of mowing**
  - Id = "1.4.1.2"

- **Configuration of irrigation**
  - Id = "1.4.2.2"

- **Operation execution**
  - Id = "1.3"

- **Execution of mowing**
  - Id = "1.4.1.1"

- **Execution of irrigation**
  - Id = "1.4.2.1"

- **Initialization of spraying**
  - Id = "1.4.3.1"

- **Reporting of spraying**
  - Id = "1.4.3.2"
## Requirements Relations in Table

<table>
<thead>
<tr>
<th>#</th>
<th>Id</th>
<th>Name</th>
<th>Text</th>
<th>Traced To</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Performing agricultural tasks</td>
<td>The CPAS system shall be capable of measuring the environment through its sensors.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1.1</td>
<td>Environment measurement</td>
<td>The CPAS uses the following sensors: temperature, humidity, luminance, rain (gauge).</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1.1.1</td>
<td>Sensors</td>
<td>The farmer can configure the system, when to activate its operations based on its sensor inputs.</td>
<td>1.4.2.2 Configuration of irrigation, 1.4.1.2 Configuration of mowing</td>
</tr>
<tr>
<td>4</td>
<td>1.2</td>
<td>Behavior configuration</td>
<td>The CPAS shall execute operations to change its surrounding environment.</td>
<td>1.4.2.1 Execution of irrigation, 1.4.3.1 Initialization of spraying, 1.4.3.2 Reporting of spraying, 1.4.1.1 Execution of mowing</td>
</tr>
<tr>
<td>5</td>
<td>1.3</td>
<td>Operation execution</td>
<td></td>
<td>30 Shutdown speed</td>
</tr>
<tr>
<td>6</td>
<td>1.3.1</td>
<td>Farmer notification</td>
<td>Whenever a notification arrives the CPAS signals the farmer based on the configured communication mean.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1.3.2</td>
<td>System shutdown</td>
<td>The farmer can shut down the system that immediately stops all of its active operations.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1.4</td>
<td>Supported operations</td>
<td>Supported operations are mowing, irrigation and spraying.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1.4.1</td>
<td>Mowing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.4.1.1</td>
<td>Execution of mowing</td>
<td>The mowing operation signals the robot lawn-mower to execute its programmed task.</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1.4.1....</td>
<td>Successful mowing</td>
<td>If the mower robot executes its task without any problem it returns to its refueling station.</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1.4.1....</td>
<td>Failed mowing</td>
<td>If the mower robot fails to complete its task, it sends a notification about its status.</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1.4.1.2</td>
<td>Configuration of mowing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1.4.2</td>
<td>Irrigation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1.4.2.1</td>
<td>Execution of irrigation</td>
<td>The irrigation operation simply activates the pre-installed irrigation-system.</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>1.4.2....</td>
<td>Failed irrigation</td>
<td>If the irrigation-system fails, it sends a notification about its status.</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>1.4.2.2</td>
<td>Configuration of irrigation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>1.4.3</td>
<td>Spraying</td>
<td>The system shall support the control of manual spraying.</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>1.4.3.1</td>
<td>Initialization of spraying</td>
<td>The spraying operation signals the laborers to execute the spraying task.</td>
<td></td>
</tr>
</tbody>
</table>
Modeling Textual Requirements

Context of the Modeling Aspect
Sample Requirements (Cyber-physical Agricultural System)
Modeling Elements & Notation

Summary
Summary

- **Goal**
  - Bridge the gap between textual requirements and requirement and design models
    - Handles textual req.s as model elements
    - Provides support for requirements traceability

- **Modeling aspect**
  - *What are the main requirements formulated textually and what are their hierarchy?*

- **Relation of requirements to other aspects**
  - Refined by model elements (e.g. use case, activity)
  - Satisfied by blocks
  - Verified by test cases
Modeling Requirements with Use Cases

Context of the Modeling Aspect

Elements of Use Case Diagrams by Example

Relations between UC elements

Summary
System Modeling Process

1. Collect Textual Requirements
2. Model System Context
3. Model Functional Requirements with Use Cases
4. Elaborate Functional Requirements with Activities
5. Create Data Model
6. Model Interactions between System and Environment
7. Derive System State Model
8. Derive System Interfaces
9. Model System Structures
Who will use the system and for what?

- Farmer
- Laborer
- Maintainer

Cyber Physical Agricultural System

- Perform agricultural tasks
- Maintain the system

- Rain
- Luminance
- Humidity
- Temperature
Use cases (használati eset) capture the functional requirements of a system.

UCs describe:
- the typical interactions
- between the users of a system and
- the system itself,
- by providing a narrative of how a system is used.

A set of scenarios tied together by a common user goal.

Its definition comes from:
- Either directly from the written requirement
  → **Verb + Noun (Unique)!**
- Based on the Requirement diagram + System context definition
  → **refinement**

Relations to other aspects

- Refines textual requirements
- Can be further refined by behaviors (e.g. activity)
Modeling Requirements with Use Cases

Context of the Modeling Aspect

Elements of Use Case Diagrams by Example

Relations between UC elements

Summary
Example requirements

- Design a simple Cyber-physical agricultural system (CPAS), which helps a farmer with his/her everyday life using sensors to measure the environment and react to its changes by using automated operations like irrigation, mowing and spraying.

Requirements

- The CPAS system is capable of measuring the environment through its sensors.
- The CPAS uses the following sensors: temperature, humidity, luminance, rain.
- The CPAS can execute operations to change its surrounding environment.
- These operations can be mowing, irrigation and spraying.
- The mowing operation signals the robot mower to execute its programmed task.
- If the mower robot executes its task without any problem, it returns to its refueling station.
- If the mower robot fails to complete its task, it sends a notification about its status.
Example requirements (con’t)

- **Requirements**
  - The irrigation operation simply activates the pre-installed irrigation system.
  - If the irrigation-system fails, it sends a notification about its status.
  - Whenever a notification arrives the CPAS signals the farmer based on the configured communication mean.
  - The spraying operation signals the laborers to execute the spraying task.
  - The laborers report to the CPAS when they finished their task.
  - In case an error occurs during the spraying the laborers submit a form to the CPAS and it notifies the farmer.
  - The farmer can configure the system, when to activate its operations based on its sensor inputs.
  - The farmer can shut down the CPAS system that immediately stops all of its active operations.
  - The system shall provide diagnostic information about its components for maintenance.
Initial set of Activities

- Configure behavior
- Activate irrigation system
  {UseCase Number = 1}
- Execute operation
- Manage spraying
- Report to CPAS
  {UseCase Number = 1}
- Signal mower
  {UseCase Number = 1}
- Send spraying operation
- Shut down system
- Operate irrigation system
- Submit Spraying Error Form
  {UseCase Number = 1}
- Return to fuel station
  {UseCase Number = 1}
- Receive spraying report
- Perform mowing
- Send irrigation error information
  {UseCase Number = 1}
- Measure environment
Definition of Actors

- **Actor** (aktor) is a **role** that a user plays with respect to the system.
  - **Primary actor**: calls the system to deliver a service
  - **Secondary actor**: the system communicates with them while carrying out the service

- Relationship of UCs and Actors
  - A single actor may perform many use cases;
  - A use case may have several actors performing it.

- One person may act as more than one actor,
  - Example: The farmer may also act as a laborer who performs the spraying

- An actor is outside the boundary of the system

- Its definition comes from
  - Directly from the written requirements
  - Based on the System context definition
(Initial) Collection of Primary Actors

11 Laborer  10 Farmer  2 Maintainer
(Initial) Collection of Secondary Actors

Rain  Humidity
Temperature  Luminance

Customized representation

Could use the UML actor visualization instead!
Modeling Requirements with Use Cases

Context of the Modeling Aspect
Elements of Use Case Diagrams by Example

Relations between UC elements
Summary
System-level overview (User)

Cyber Physical Agricultural System

- Farmer
- Laborer
- Maintainer

- Perform agricultural tasks
- Maintain the system

- Rain
- Luminance
- Humidity
- Temperature
System-level overview (User)

Cyber Physical Agricultural System

- **Actor**: Farmer, Laborer, Maintainer
- **System boundary**: Perform agricultural tasks, Maintain the system
- **Association**: actor initiates or participates in interaction
- **Same behavior!**

- **System use case**: Perform agricultural tasks
- **Environment**: Rain, Luminance, Humidity, Temperature
Generalization of Actors (abstraction)

Actor Generalization (Inheritance)

Environmental effect

Rain
Temperature
Humidity
Luminance

Abstraction/refinement
System-level overview (User)

10 Farmer

11 Laborer

2 Maintainer

Cyber Physical Agricultural System

- Perform agricultural tasks
- Maintain the system

Environmental effect:
- Temperature
- Humidity
- Luminance
- Rain
How to handle complex functionality?

Perform agr. tasks =
- Measure the environment
- Configure the system
- Execute agr. operations
- Shut down the system
The included UC breaks down the complex core functionality into more elementary steps.
Generalization of UCs

What happens if
• the irrigation operation fails?
Extend relationship

Execute operation

Operate irrigation system

Base UC

Activate irrigation system

Send irrigation error information

Extension UC

The extension UC extends core functionality by handling unusual (exceptional) situation
Summary: UC Relations

- **Association**
  - actor – use case
  - the actor initiates (or participates in) the use of the system

- **Extend**
  - use case – use case
  - a UC may be extended by another UC (typically solutions for exceptional situations)
Summary: UC Relations

- **Generalization**
  - actor – actor
  - use case – use case
  - a UC or actor is more general / specific than another UC or actor

- **Include**
  - use case – use case
  - a complex step is divided into elementary steps
  - a functionality is used in multiple UCs
Example: Complete ‘Perform agr. task’ UC refinement

Diagram showing the refinement of the system use case 'Perform agricultural tasks'.
Modeling Requirements with Use Cases

Context of the Modeling Aspect
Elements of Use Case Diagrams by Example
Relations between UC elements

Summary
Summary

- **Goal**
  - Identify top level functional requirements
  - Identify involved actors

- **Modeling Aspect**
  - *Who will use the system and for what?*

- **Relations to other aspects**
  - Refines textual requirements
  - Can be refined by other behaviors (e.g. activity)
Modeling Flow Based Behavior with Activities

Context of the Modeling Aspect
Modeling Elements & Notation
Semantics of the Model
Summary
Roots & Relations

- Flow-sheets and flow-charts are used everywhere...
  - Brainstorming
  - Computer algorithms
  - Business processes
System Modeling Process

act [Activity] Simplified System Modeling Process

- Collect Textual Requirements
- Model System Context
- Model Functional Requirements with Use Cases
- Elaborate Functional Requirements with Activities
- Create Data Model
- Model Interactions between System and Environment
- Derive System State Model
- Derive System Interfaces
- Model System Structures
What are the steps in a process?
What data flows in the process?
Objectives

- Modeling behavior that specifies the *transformation of inputs to outputs* through a sequence of actions
- Combined modeling of *control flow* and *data flow* in a process or workflow
- Supporting the definition of *high level processes*
  - Elaboration of use cases, i.e. helps to define functional requirements that system components or actors will perform
  - Providing *functional decomposition* of the system
- Supporting the definition of *low level activities*
  - Elaboration of behavior executed at given ‘points’ of the system (e.g. reaction to an event)
Elaborates use cases
Modeling Flow Based Behavior with Activities

Context of the Modeling Aspect

**Modeling Elements & Notation**

Semantics of the Model

Summary
Control flow

Initial node

Fork

Join

Activity final

Action in activity

Control flow
Control flow with flow final
Fork, join, decision, merge

Fork

Decision

Merge

Join
# Action types

<table>
<thead>
<tr>
<th>Action type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Primitive action node</strong></td>
<td>Primitive actions include object access, update and manipulation actions.</td>
</tr>
<tr>
<td><img src="image" alt="Primitive action node" /></td>
<td></td>
</tr>
<tr>
<td><strong>Send signal node</strong></td>
<td>Send signal data to target.</td>
</tr>
<tr>
<td><img src="image" alt="Send signal node" /></td>
<td></td>
</tr>
<tr>
<td><strong>Accept event node</strong></td>
<td>Accept events, typically has output pins for received data.</td>
</tr>
<tr>
<td><img src="image" alt="Accept event node" /></td>
<td></td>
</tr>
<tr>
<td><strong>Accept time event node</strong></td>
<td>Time event corresponds to an expiration of an (implicit) timer.</td>
</tr>
<tr>
<td><img src="image" alt="Accept time event node" /></td>
<td></td>
</tr>
<tr>
<td><strong>Call behavior node</strong></td>
<td>Call other behavior (e.g. another activity).</td>
</tr>
<tr>
<td><img src="image" alt="Call behavior node" /></td>
<td></td>
</tr>
</tbody>
</table>
Combined control and data flow

**Object flow**

**Object**

**Parameter of activity**

**Pin**

**Measure temperature**
- : Temperature Data

**Measure humidity**
- : Humidity Data

**Measure luminance**
- : Luminance Data

**Detect rain**
- : Rain Data

**act** [Activity] Measure environment [Measure environment - with object flow]
Activity decomposition
Allocating actions
Modelling Streams (SysML)

Activity parameter

Discrete flow

Flow rate

Stream

Pin

Discrete flow
Interruptible Activity Region

Interruptible activity region

Signal reception

Interrupting edge
Modeling Flow Based Behavior with Activities

Context of the Modeling Aspect
Modeling Elements & Notation

Semantics of the Model
Summary
Data flow and Control flow

- **Combined control and data flow model**
  - semantics ≈ dataflow networks

- **Data Flow: data token**
  - Object node ⇒ Action node
    - An object node is a channel / queue
    - An object may be linked to multiple action nodes
    - Output actions are competing for the data token (i.e. the object)
  - Type conformance: object type < input type of action

- **Control flow: control token**
  (ordering constraint between two actions)
  - All predecessor actions should be terminated prior to starting the current action
  - The current action should terminate prior to starting any of the successor actions
Tokens: control + several data

Channel: object node
    Stores the tokens

Node: action node
    Processing tokens

Edges:
    Flow of tokens
    weights: how many tokens are in the flow at a time?

Firing rule:
    Behaviour of a node
Firing rule (cont.):

- precondition:
  - input tokens + current state

- postcondition
  - output tokens + new state
Semantics: Dataflow Networks

- **Firing rule (cont.):**
  - **precondition:**
    - input tokens + current state
  - **postcondition**
    - output tokens + new state

- **Execution of a firing:**
  - Is there token on all inputs with
    - Right amount?
    - Right type?

```
Firing rule (cont.):
- precondition:
  - input tokens + current state
- postcondition
  - output tokens + new state

Execution of a firing:
- Is there token on all inputs with
  - Right amount?
  - Right type?
```
Firing rule (cont.):
- precondition:
  - input tokens + current state
- postcondition
  - output tokens + new state

Execution of a firing:
- Is there token on all inputs with
  - Right amount?
  - Right type?
- Execution of action
Firing rule (cont.):
- precondition:
  - input tokens + current state
- postcondition
  - output tokens + new state

Execution of a firing:
- Is there token on all inputs with
  - Right amount?
  - Right type?
- Execution of action
- Sending the output tokens
Example: Shutdown system
Example: Shutdown system
Example: Shutdown system
Example: Shutdown system
Example: Shutdown system
Example: Shutdown system
Example: Shutdown system
Example: Shutdown system
Example: Shutdown system
Example: Shutdown system
Example: Shutdown system
Example: Shutdown system
Example: Shutdown system
Example: Shutdown system
Example: Shutdown system
Example: Shutdown system
Example: Shutdown system
Example: Shutdown system

- Check mower status
  - Moving
    - Return Mower
  - Not moving

- Check irrigation system status
  - On
    - Stop Irrigation
  - Off
    - Spraying Status
    - Stop Spraying

- Check spraying status
  - Spraying
    - Spraying Status
  - Not spraying

- Determine overall status
  - System Status
  - Notification
Modeling Flow Based Behavior with Activities

Context of the Modeling Aspect
Modeling Elements & Notation
Semantics of the Model

Summary
Summary

- **Goal**
  - Model transformation of input to output in processes
  - Combined modeling of control and data flow

- **Modeling aspect**
  - *What are the steps in a process?*
  - *What data flows in the process?*

- **Relations to other aspects**
  - Refines requirements, use cases and interactions
  - Allocates activities to blocks
  - Defines behavior of blocks, operations or in state machines