Modeling Requirements

Critical Embedded Systems

Dr. Balázs Polgár

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

This material can only used by participants of the course.

Overview

- UML & SysML Overview
- Modeling Textual Requirements
- Modeling Requirements with Use Cases
- Modeling Flow Based Behavior with Activities

UML & SysML Overview

UML Overview

SysML Overview

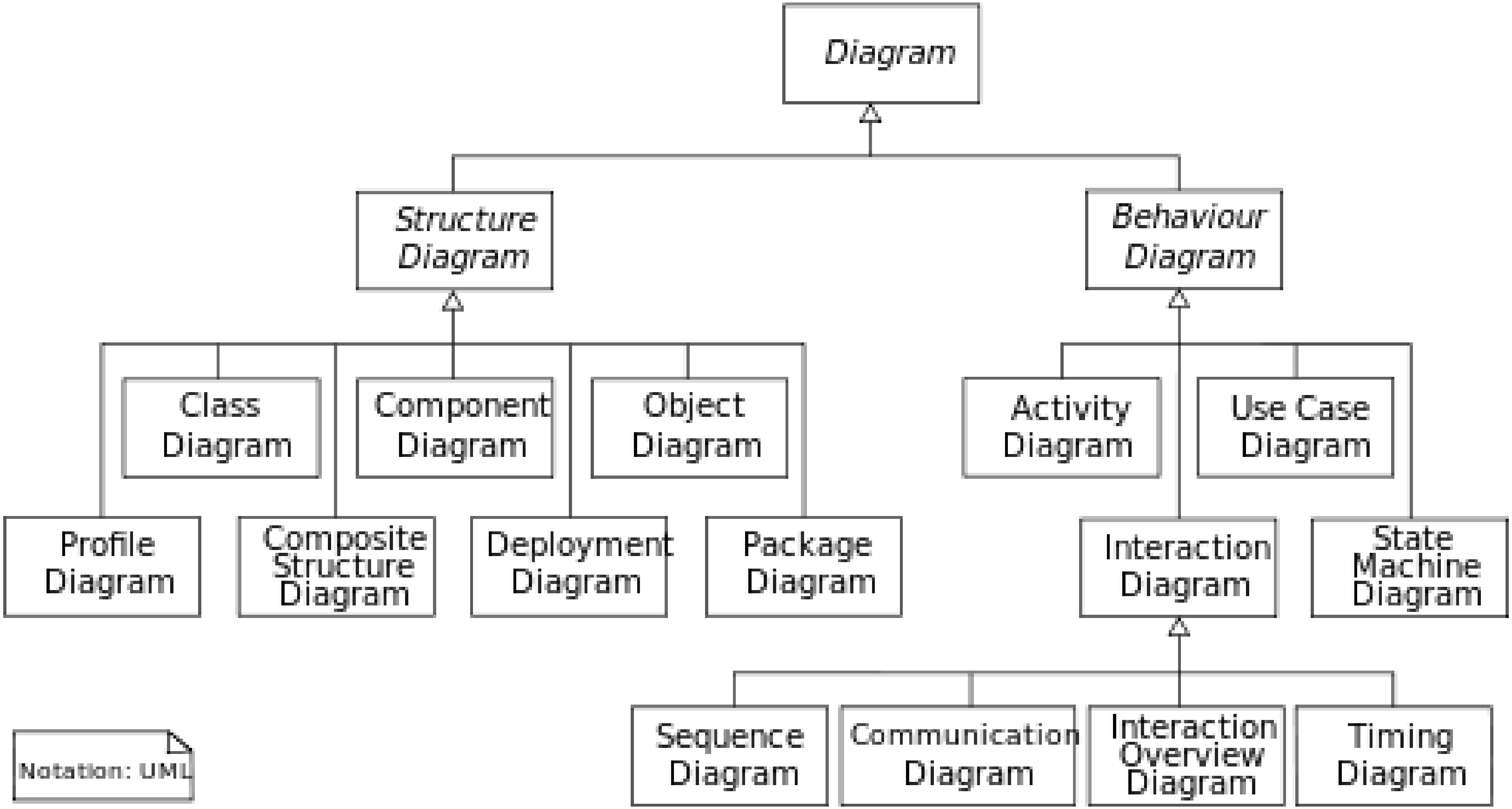
UML Overview

- Unified Modeling Language
  - An OMG (Object Management Group) standard
- 1.x series
  - 1997 – Initial version (v1.1 – first adopted version)
    - by James Rumbaugh, Grady Booch, Ivar Jacobson at Rational
  - 2000 – v1.3, v1.4
  - 2003 – v1.5
- 2.x series
  - 2005 – v2.0
  - 2007 – v2.1.2
  - 2009 – v2.2
  - 2010 – v2.3
  - 2011 – v2.4.1
  - 2012 – v2.5 – „In Process”

Related Standards

- MOF – Meta Object Facility Core
  - 2011 – v2.4.1
  - Modeling language for defining modeling languages
- OCL – Object Constraint Language
  - 2012 – v2.3.1
  - Textual language for formulating constraints and queries over models
- fUML – Foundational UML
  - 2013 – v1.1
  - Semantics of a Foundational Subset for Executable UML Models
- ALF – Action Language for Foundational UML
  - 2012 – v1.0.1 Beta3
  - Concrete Syntax for a UML Action Language
- XMI – XML Metadata Interchange
  - 2011 – v2.4.1
  - XML representation of models
- DD – Diagram Definition
  - 2012 – v1.0
  - for modeling and interchanging graphical notations

UML Diagram Taxonomy



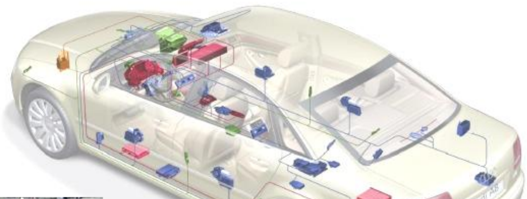
UML & SysML Overview

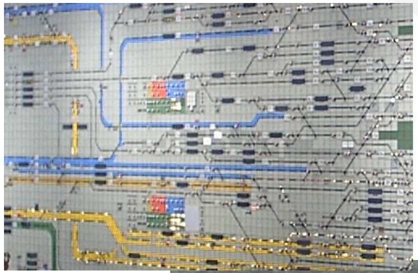
UML Overview

SysML Overview

Systems Engineering

- Systems Engineering is a multidisciplinary approach to develop balanced system solutions in response to diverse stakeholder needs
- ~ Integration Engineering
  - Software engineering
  - Hardware engineering
  - Mechanical engineering
  - Safety engineering
  - Security engineering
  - ...
- ~ Process Engineering
- System
  - Military, airplane, car, aviation, railway interlocking, notebook, etc.

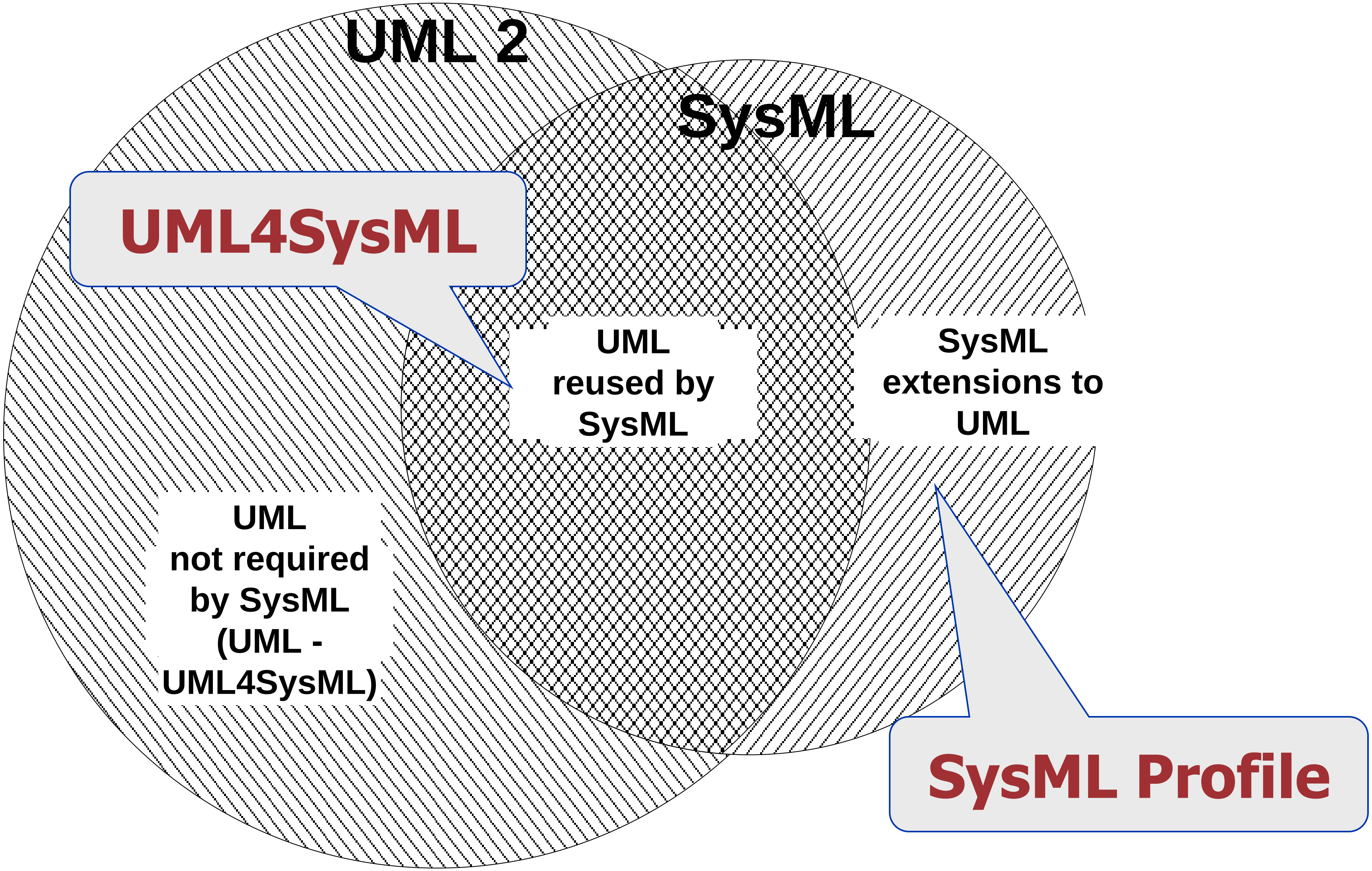




SysML overview

- „UML for Systems Engineering”
  - Supports the specification, analysis, design, verification and validation of systems that include hardware, software, data, personnel, procedures, and facilities
- Developed by OMG and International Council on Systems Engineering (INCOSE)
- OMG SysML™ (http://www.omgsysml.org)
  - RFP – March 2003
  - Version 1.0 – September 2007
  - Version 1.1 – November 2008
  - Version 1.2 – June 2010
  - Version 1.3 – June 2012

Relationship Between SysML and UML



UML 2

SysML

UML4SysML

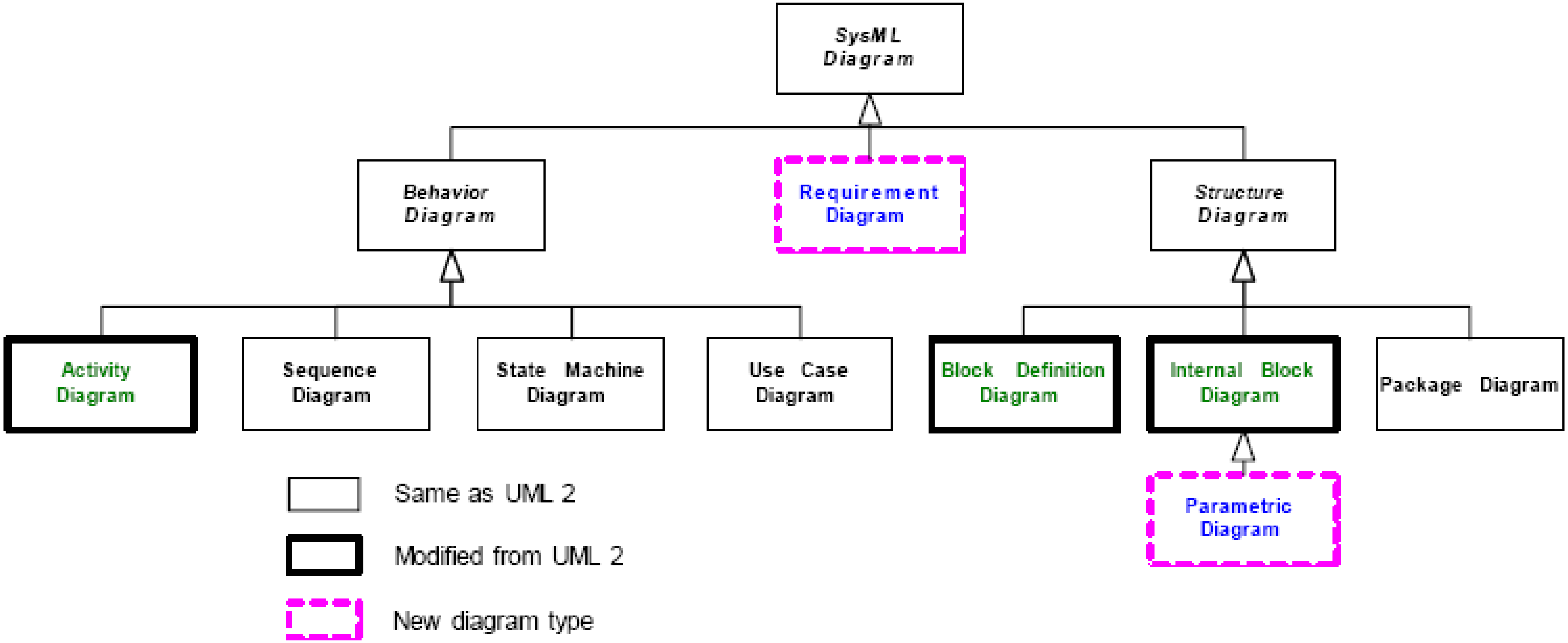
UML reused by SysML

SysML extensions to UML

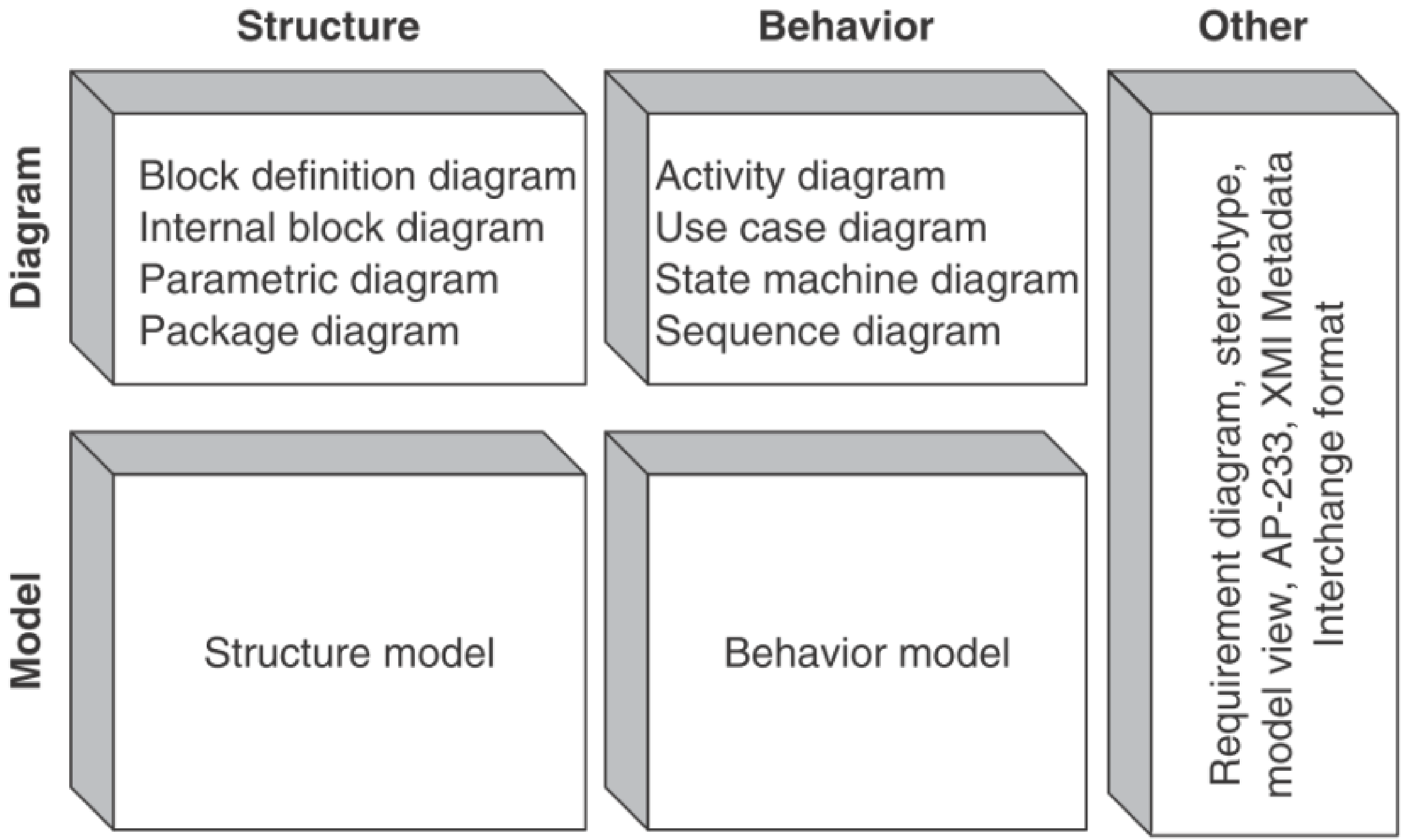
UML not required by SysML (UML - UML4SysML)

SysML Profile

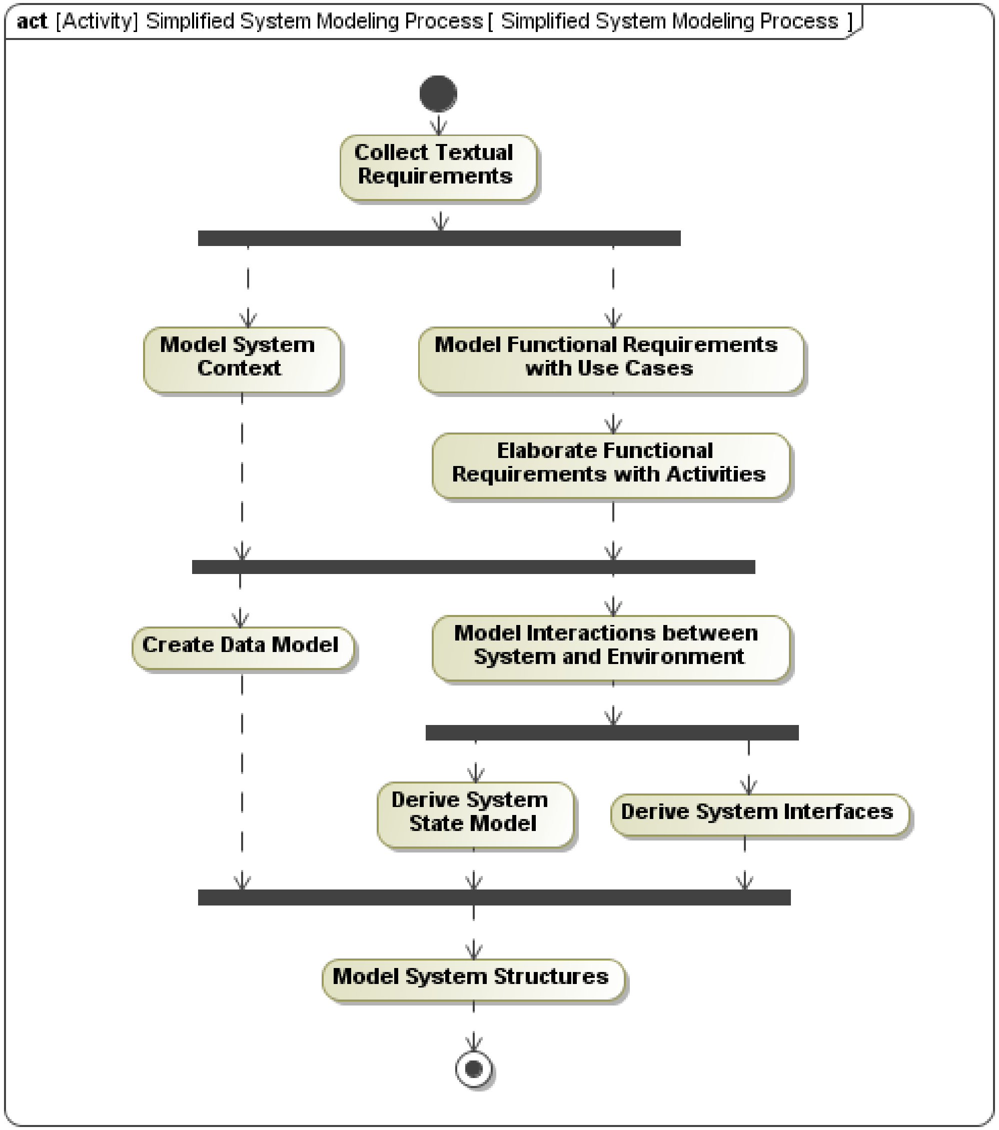
SysML Diagram Taxonomy



Aspects of SysML



A Simplified System Modeling Process


act [Activity] Simplified System Modeling Process [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

Modeling Textual Requirements

Context of the Modeling Aspect

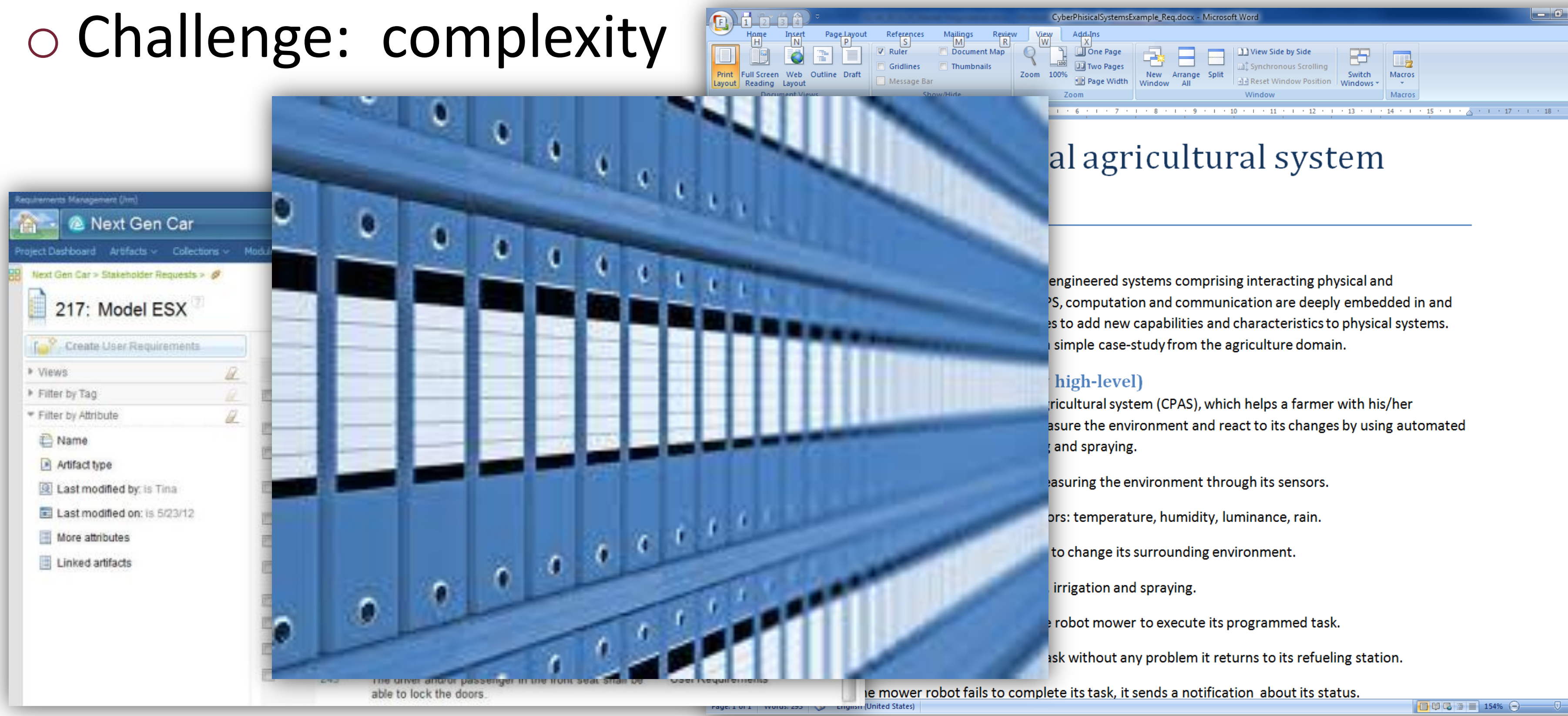
Sample Requirements (Cyber-physical Agricultural System)

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



217: Model ESX

Last modified by: is Tina

Last modified on: is 5/23/12

243 The driver and/or passenger in the front seat shall be able to lock the doors.

al agricultural system

engineered systems comprising interacting physical and
PS, computation and communication are deeply embedded in and
es to add new capabilities and characteristics to physical systems.
simple case-study from the agriculture domain.

high-level)

ricultural system (CPAS), which helps a farmer with his/her
asure the environment and react to its changes by using automated
and spraying.

asuring the environment through its sensors.

ors: temperature, humidity, luminance, rain.

to change its surrounding environment.

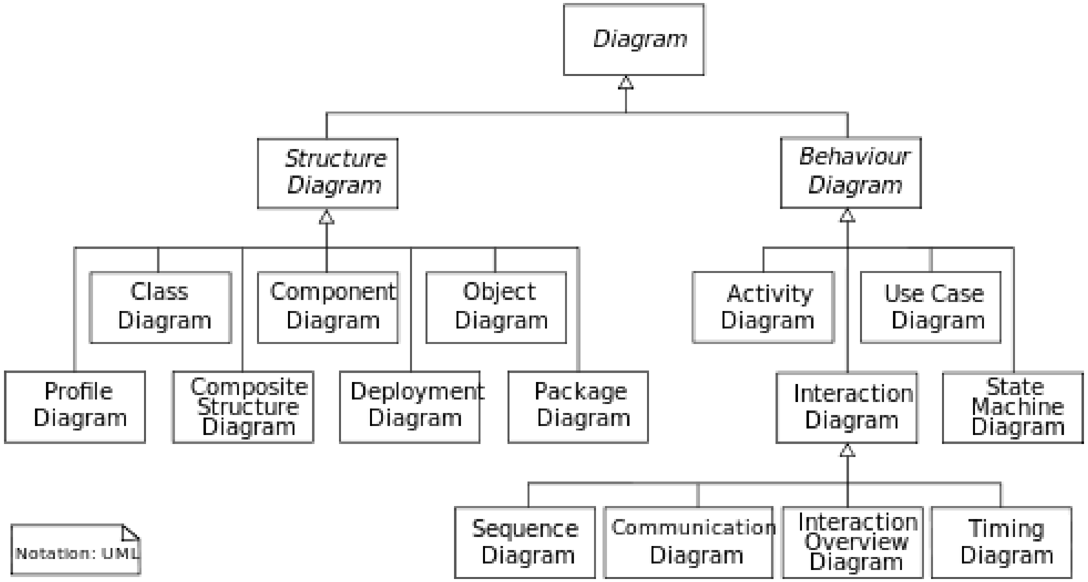
irrigation and spraying.

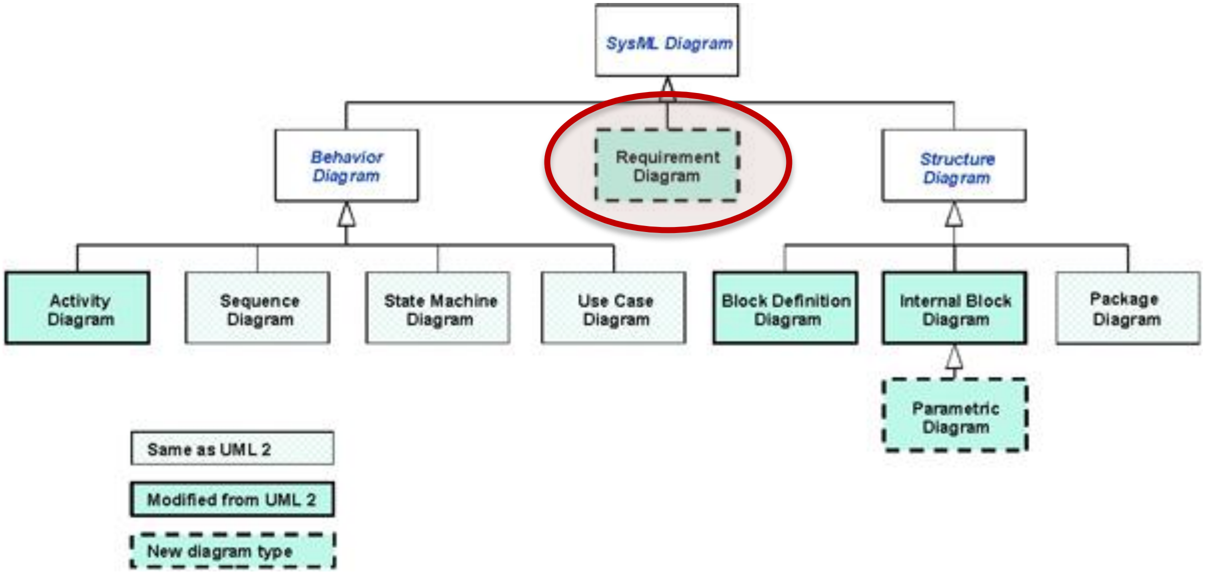
robot mower to execute its programmed task.

sk without any problem it returns to its refueling station.

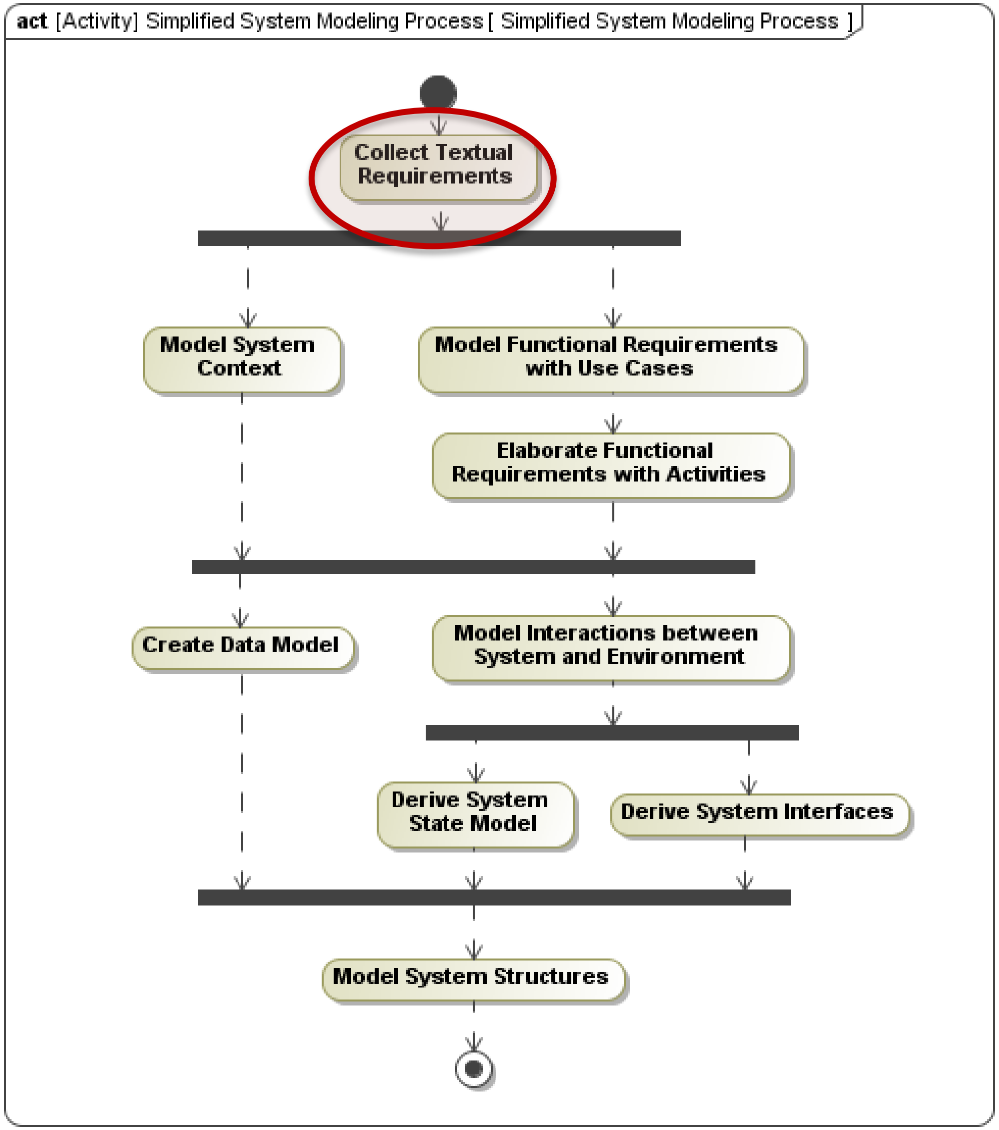
e mower robot fails to complete its task, it sends a notification about its status.

Requirements Diagram



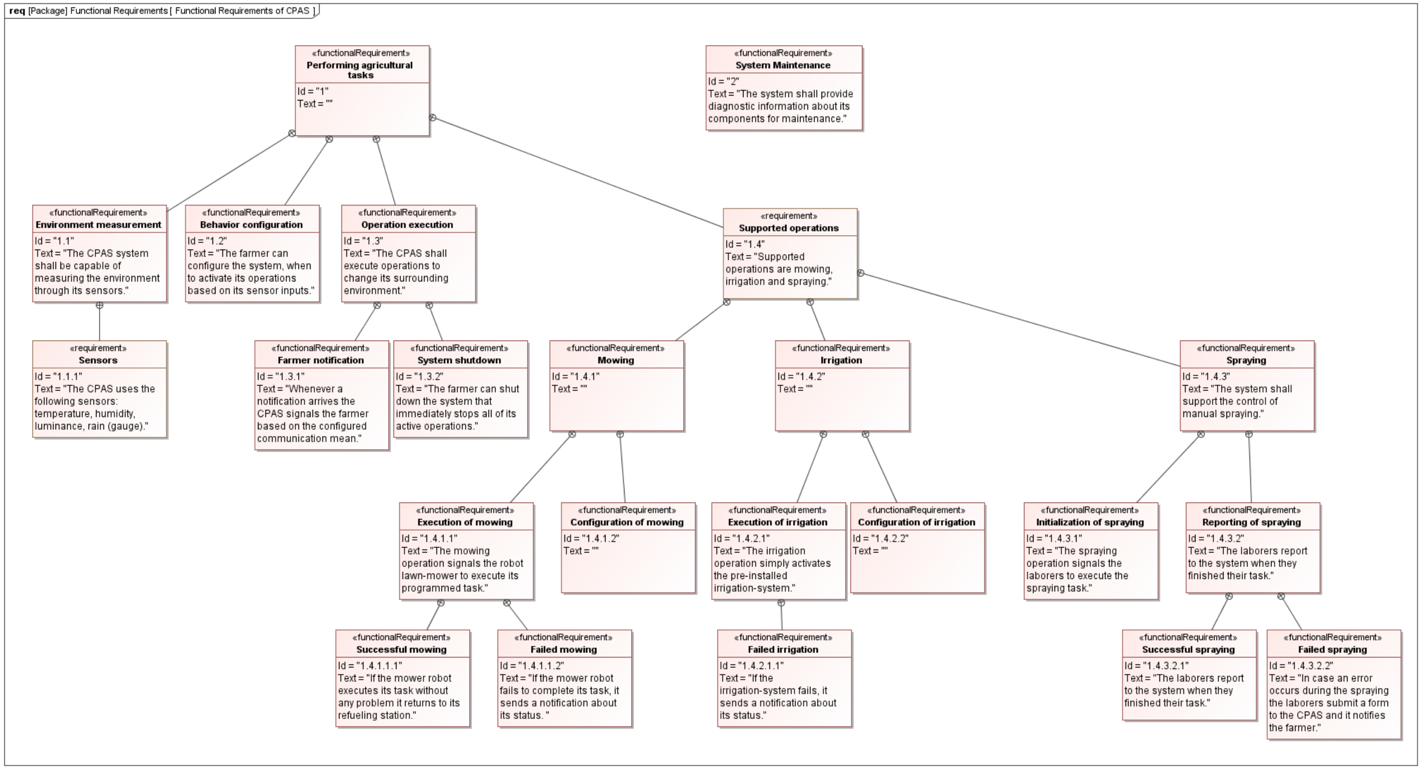


System Modeling Process


act [Activity] Simplified System Modeling Process [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

Modeling Aspect

What are the main requirements formulated textually and what are their hierarchy?



req [Package] Functional Requirements [Functional Requirements of CPAS]

«functionalRequirement»
Performing agricultural tasks
Id = "1"
Text = ""

«functionalRequirement»
System Maintenance
Id = "2"
Text = "The system shall provide diagnostic information about its components for maintenance."

«functionalRequirement»
Environment measurement
Id = "1.1"
Text = "The CPAS system shall be capable of measuring the environment through its sensors."

«functionalRequirement»
Behavior configuration
Id = "1.2"
Text = "The farmer can configure the system, when to activate its operations based on its sensor inputs."

«functionalRequirement»
Operation execution
Id = "1.3"
Text = "The CPAS shall execute operations to change its surrounding environment."

«requirement»
Supported operations
Id = "1.4"
Text = "Supported operations are mowing, irrigation and spraying."

«requirement»
Sensors
Id = "1.1.1"
Text = "The CPAS uses the following sensors: temperature, humidity, luminance, rain (gauge)."

«functionalRequirement»
Farmer notification
Id = "1.3.1"
Text = "Whenever a notification arrives the CPAS signals the farmer based on the configured communication mean."

«functionalRequirement»
System shutdown
Id = "1.3.2"
Text = "The farmer can shut down the system that immediately stops all of its active operations."

«functionalRequirement»
Mowing
Id = "1.4.1"
Text = ""

«functionalRequirement»
Irrigation
Id = "1.4.2"
Text = ""

«functionalRequirement»
Spraying
Id = "1.4.3"
Text = "The system shall support the control of manual spraying."

«functionalRequirement»
Execution of mowing
Id = "1.4.1.1"
Text = "The mowing operation signals the robot lawn-mower to execute its programmed task."

«functionalRequirement»
Configuration of mowing
Id = "1.4.1.2"
Text = ""

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

«functionalRequirement»
Configuration of irrigation
Id = "1.4.2.2"
Text = ""

«functionalRequirement»
Initialization of spraying
Id = "1.4.3.1"
Text = "The spraying operation signals the laborers to execute the spraying task."

«functionalRequirement»
Reporting of spraying
Id = "1.4.3.2"
Text = "The laborers report to the system when they finished their task."

«functionalRequirement»
Successful mowing
Id = "1.4.1.1.1"
Text = "If the mower robot executes its task without any problem it returns to its refueling station."

«functionalRequirement»
Failed mowing
Id = "1.4.1.1.2"
Text = "If the mower robot fails to complete its task, it sends a notification about its status. "

«functionalRequirement»
Failed irrigation
Id = "1.4.2.1.1"
Text = "If the irrigation-system fails, it sends a notification about its status."

«functionalRequirement»
Successful spraying
Id = "1.4.3.2.1"
Text = "The laborers report to the system when they finished their task."

«functionalRequirement»
Failed spraying
Id = "1.4.3.2.2"
Text = "In case an error occurs during the spraying the laborers submit a form to the CPAS and it notifies the farmer."

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

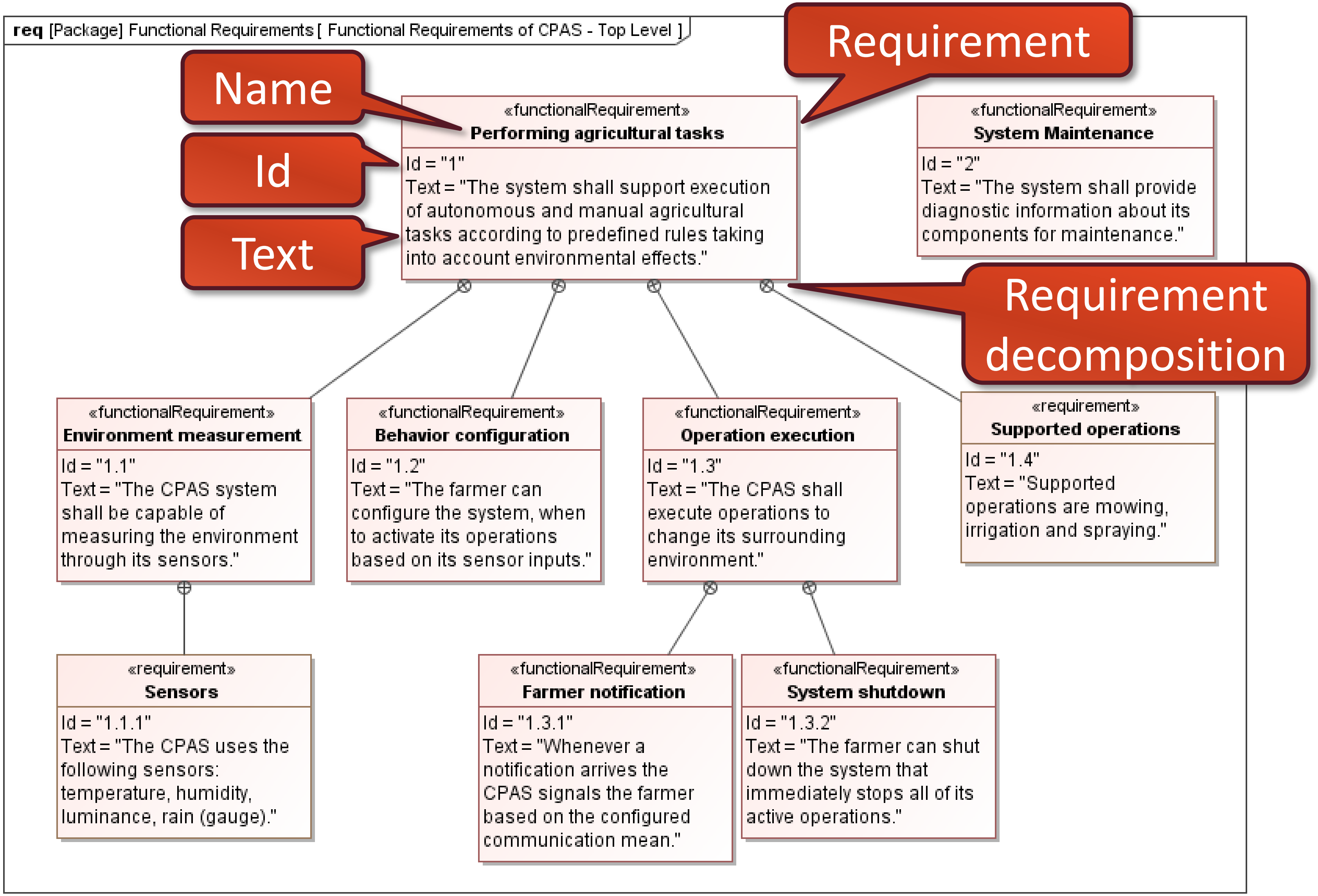
Context of the Modeling Aspect

Sample Requirements (Cyber-physical Agricultural System)

Modeling Elements & Notation

Summary

Example – Top Level Requirements



req [Package] Functional Requirements [Functional Requirements of CPAS - Top Level]

Name

Id

Text

Requirement

Requirement decomposition

«functionalRequirement»
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."

«functionalRequirement»
System Maintenance
Id = "2"
Text = "The system shall provide diagnostic information about its components for maintenance."

«functionalRequirement»
Environment measurement
Id = "1.1"
Text = "The CPAS system shall be capable of measuring the environment through its sensors."

«functionalRequirement»
Behavior configuration
Id = "1.2"
Text = "The farmer can configure the system, when to activate its operations based on its sensor inputs."

«functionalRequirement»
Operation execution
Id = "1.3"
Text = "The CPAS shall execute operations to change its surrounding environment."

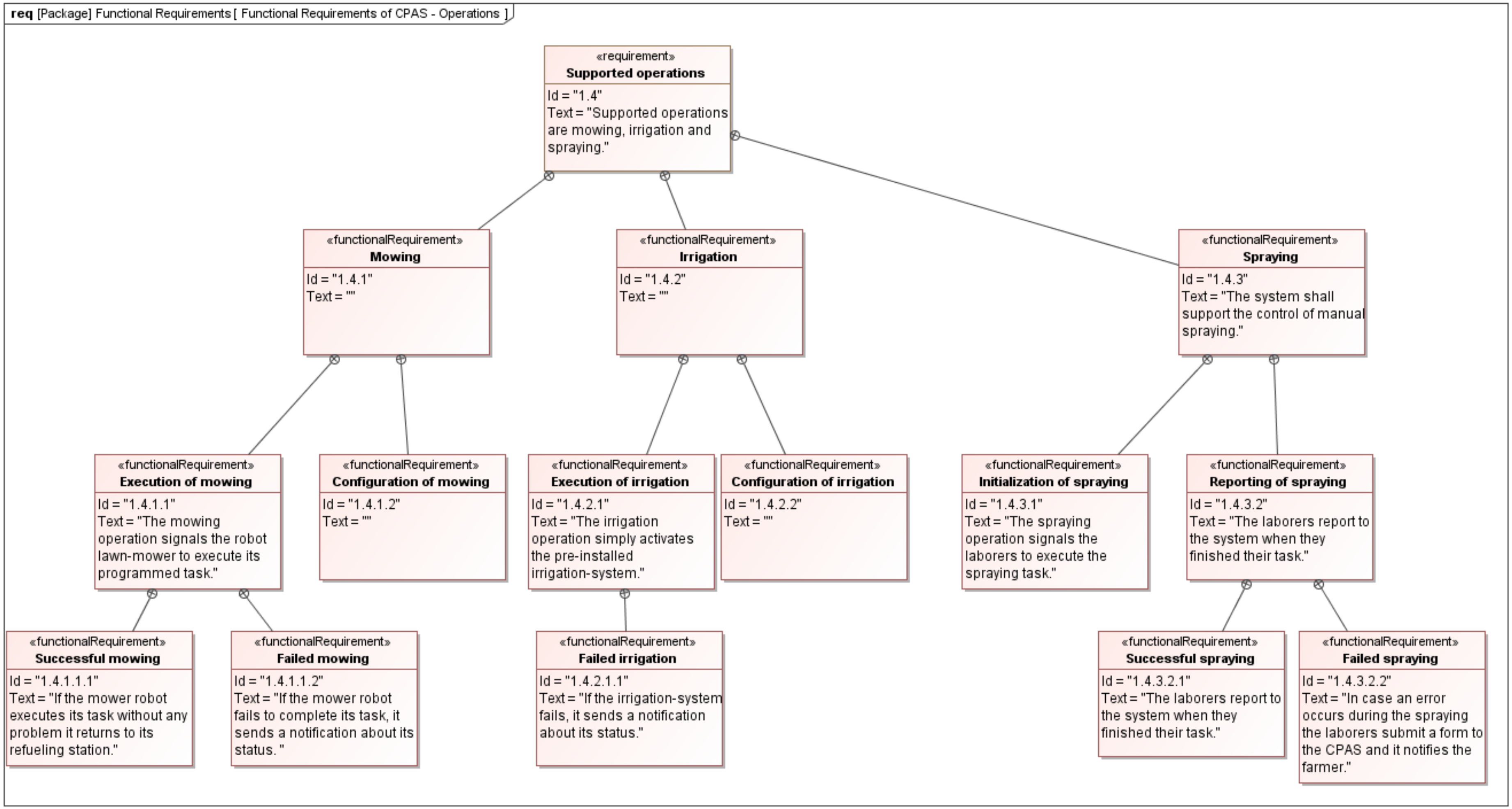
«requirement»
Supported operations
Id = "1.4"
Text = "Supported operations are mowing, irrigation and spraying."

«requirement»
Sensors
Id = "1.1.1"
Text = "The CPAS uses the following sensors: temperature, humidity, luminance, rain (gauge)."

«functionalRequirement»
Farmer notification
Id = "1.3.1"
Text = "Whenever a notification arrives the CPAS signals the farmer based on the configured communication mean."

«functionalRequirement»
System shutdown
Id = "1.3.2"
Text = "The farmer can shut down the system that immediately stops all of its active operations."

Example – Further Decomposed



req [Package] Functional Requirements [Functional Requirements of CPAS - Operations]

«requirement»
Supported operations
Id = "1.4"
Text = "Supported operations are mowing, irrigation and spraying."

«functionalRequirement»
Mowing
Id = "1.4.1"
Text = ""

«functionalRequirement»
Irrigation
Id = "1.4.2"
Text = ""

«functionalRequirement»
Spraying
Id = "1.4.3"
Text = "The system shall support the control of manual spraying."

«functionalRequirement»
Execution of mowing
Id = "1.4.1.1"
Text = "The mowing operation signals the robot lawn-mower to execute its programmed task."

«functionalRequirement»
Configuration of mowing
Id = "1.4.1.2"
Text = ""

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

«functionalRequirement»
Configuration of irrigation
Id = "1.4.2.2"
Text = ""

«functionalRequirement»
Initialization of spraying
Id = "1.4.3.1"
Text = "The spraying operation signals the laborers to execute the spraying task."

«functionalRequirement»
Reporting of spraying
Id = "1.4.3.2"
Text = "The laborers report to the system when they finished their task."

«functionalRequirement»
Successful mowing
Id = "1.4.1.1.1"
Text = "If the mower robot executes its task without any problem it returns to its refueling station."

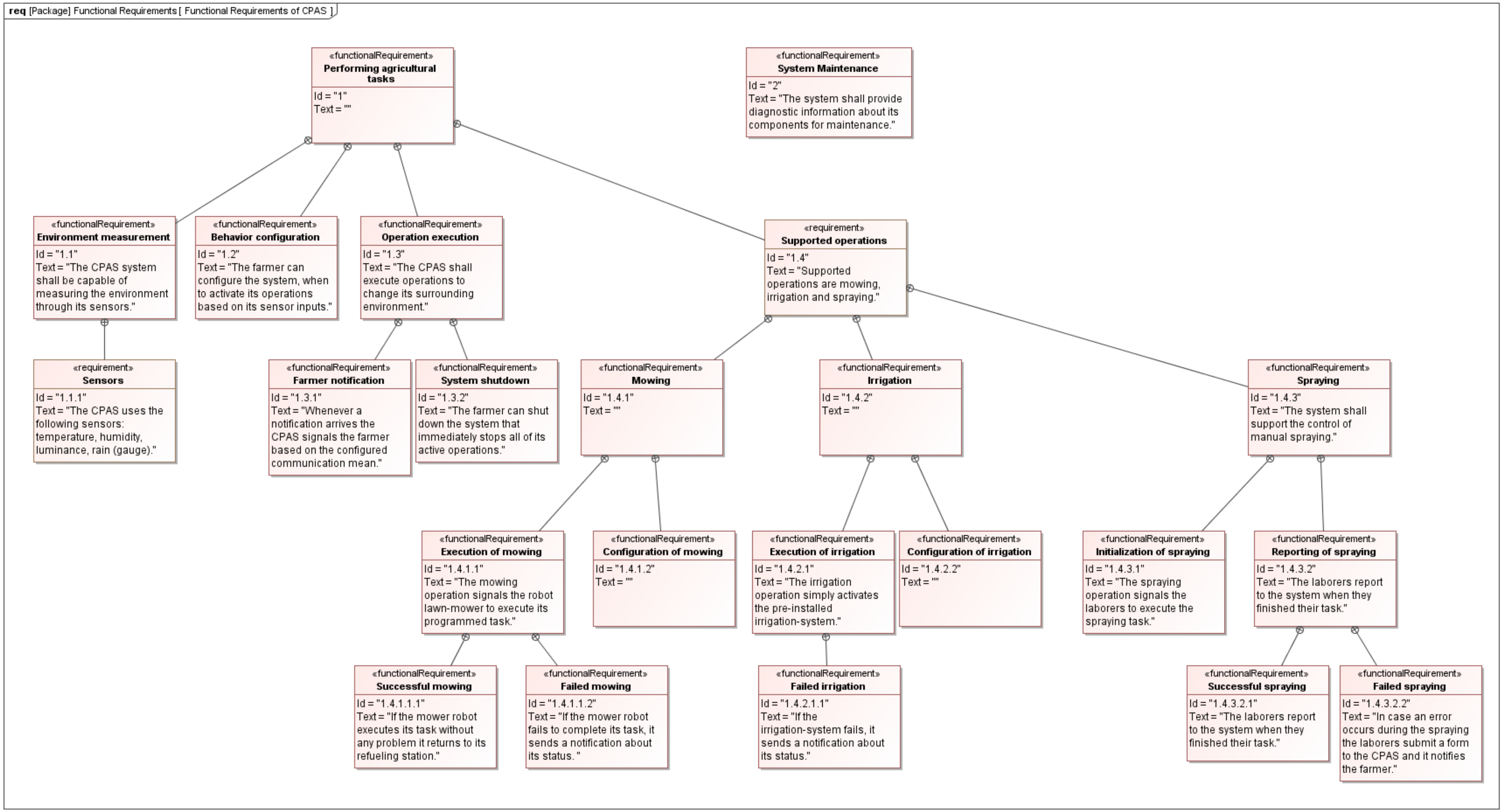
«functionalRequirement»
Failed mowing
Id = "1.4.1.1.2"
Text = "If the mower robot fails to complete its task, it sends a notification about its status. "

«functionalRequirement»
Failed irrigation
Id = "1.4.2.1.1"
Text = "If the irrigation-system fails, it sends a notification about its status."

«functionalRequirement»
Successful spraying
Id = "1.4.3.2.1"
Text = "The laborers report to the system when they finished their task."

«functionalRequirement»
Failed spraying
Id = "1.4.3.2.2"
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



req [Package] Functional Requirements [Functional Requirements of CPAS]

«functionalRequirement»
Performing agricultural tasks
Id = "1"
Text = ""

«functionalRequirement»
System Maintenance
Id = "2"
Text = "The system shall provide diagnostic information about its components for maintenance."

«functionalRequirement»
Environment measurement
Id = "1.1"
Text = "The CPAS system shall be capable of measuring the environment through its sensors."

«functionalRequirement»
Behavior configuration
Id = "1.2"
Text = "The farmer can configure the system, when to activate its operations based on its sensor inputs."

«functionalRequirement»
Operation execution
Id = "1.3"
Text = "The CPAS shall execute operations to change its surrounding environment."

«requirement»
Supported operations
Id = "1.4"
Text = "Supported operations are mowing, irrigation and spraying."

«requirement»
Sensors
Id = "1.1.1"
Text = "The CPAS uses the following sensors: temperature, humidity, luminance, rain (gauge)."

«functionalRequirement»
Farmer notification
Id = "1.3.1"
Text = "Whenever a notification arrives the CPAS signals the farmer based on the configured communication mean."

«functionalRequirement»
System shutdown
Id = "1.3.2"
Text = "The farmer can shut down the system that immediately stops all of its active operations."

«functionalRequirement»
Mowing
Id = "1.4.1"
Text = ""

«functionalRequirement»
Irrigation
Id = "1.4.2"
Text = ""

«functionalRequirement»
Spraying
Id = "1.4.3"
Text = "The system shall support the control of manual spraying."

«functionalRequirement»
Execution of mowing
Id = "1.4.1.1"
Text = "The mowing operation signals the robot lawn-mower to execute its programmed task."

«functionalRequirement»
Configuration of mowing
Id = "1.4.1.2"
Text = ""

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

«functionalRequirement»
Configuration of irrigation
Id = "1.4.2.2"
Text = ""

«functionalRequirement»
Initialization of spraying
Id = "1.4.3.1"
Text = "The spraying operation signals the laborers to execute the spraying task."

«functionalRequirement»
Reporting of spraying
Id = "1.4.3.2"
Text = "The laborers report to the system when they finished their task."

«functionalRequirement»
Successful mowing
Id = "1.4.1.1.1"
Text = "If the mower robot executes its task without any problem it returns to its refueling station."

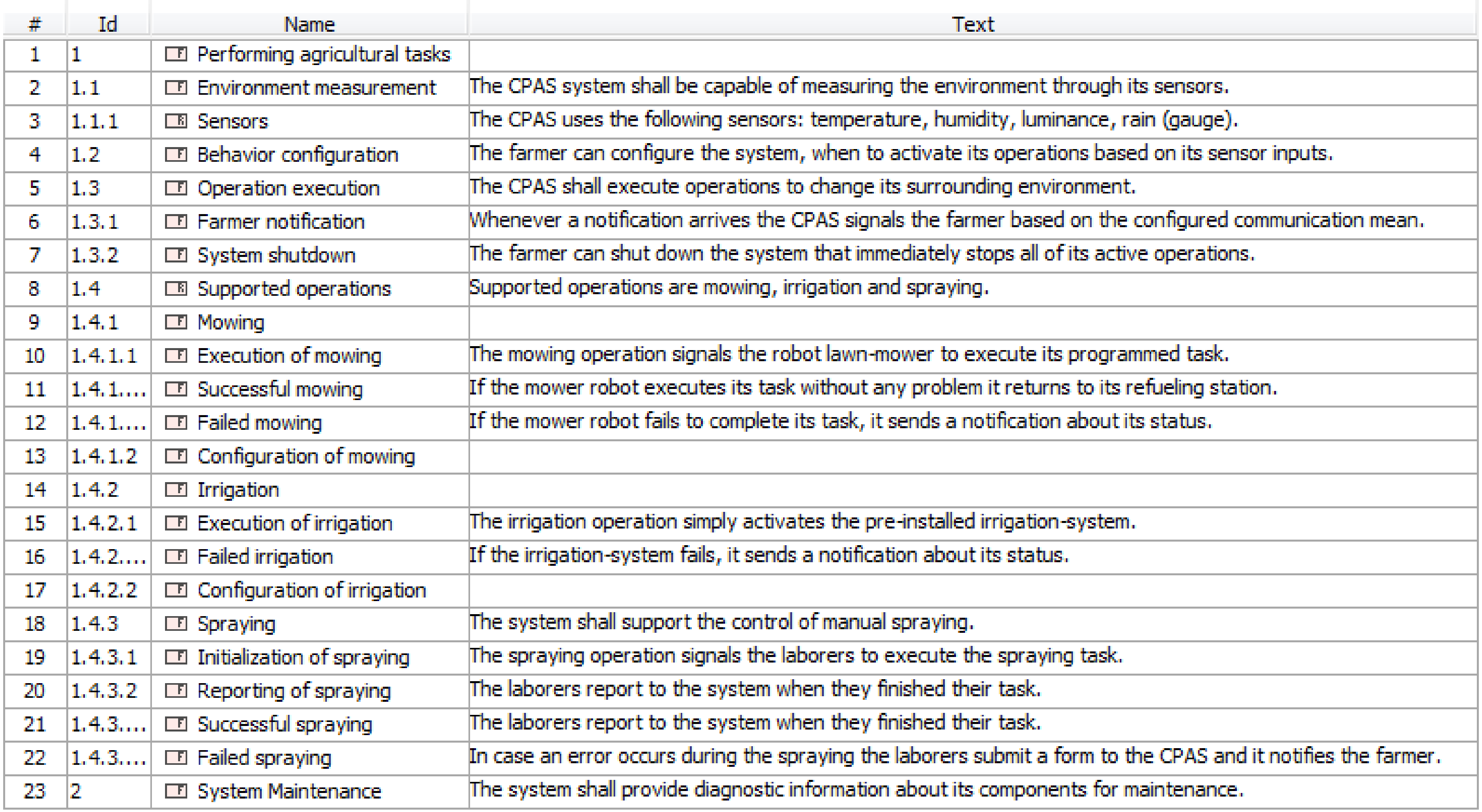
«functionalRequirement»
Failed mowing
Id = "1.4.1.1.2"
Text = "If the mower robot fails to complete its task, it sends a notification about its status. "

«functionalRequirement»
Failed irrigation
Id = "1.4.2.1.1"
Text = "If the irrigation-system fails, it sends a notification about its status."

«functionalRequirement»
Successful spraying
Id = "1.4.3.2.1"
Text = "The laborers report to the system when they finished their task."

«functionalRequirement»
Failed spraying
Id = "1.4.3.2.2"
Text = "In case an error occurs during the spraying the laborers submit a form to the CPAS and it notifies the farmer."

Requirements Table

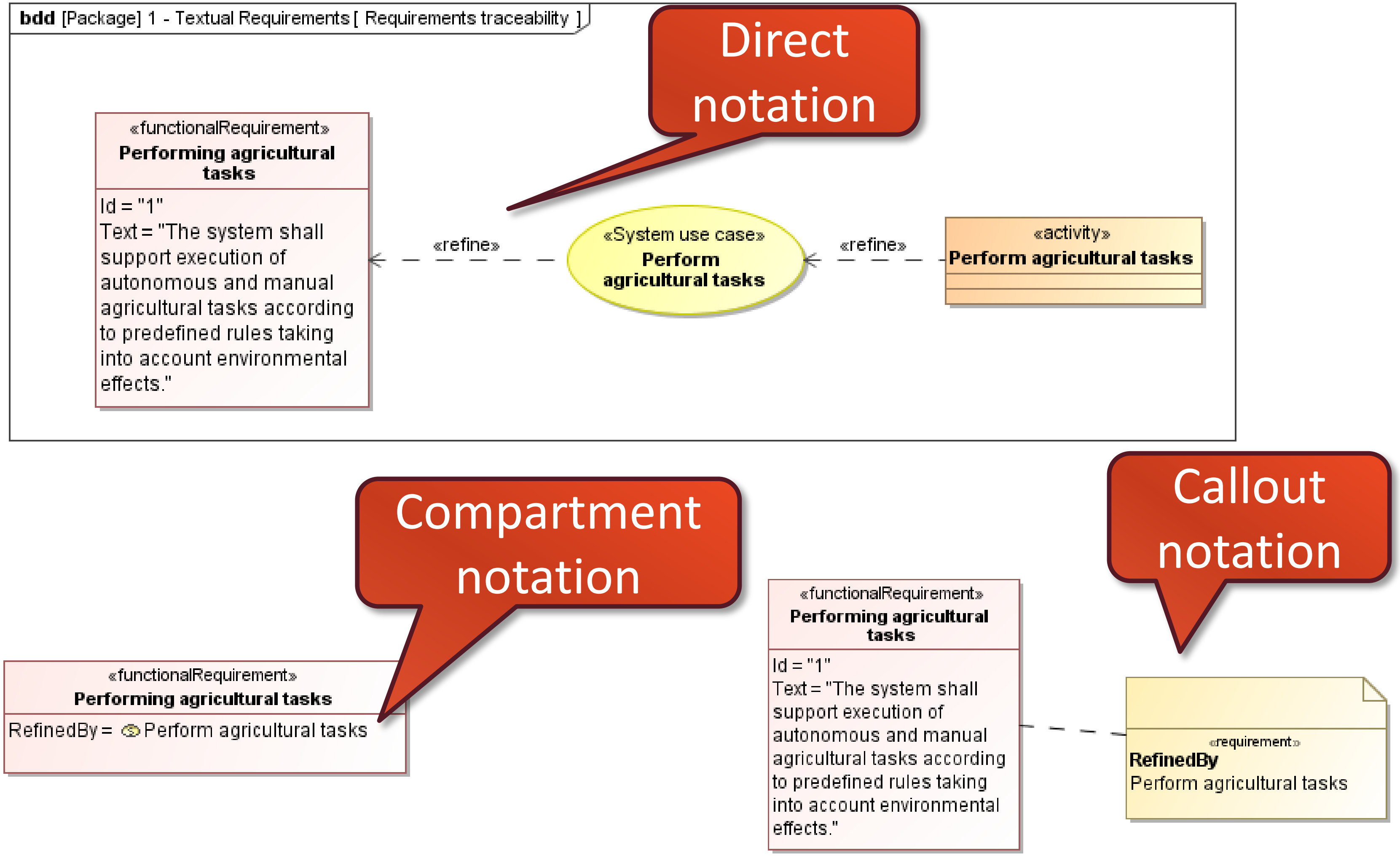


#	Id	Name	Text
1	1	Performing agricultural tasks	
2	1.1	Environment measurement	The CPAS system shall be capable of measuring the environment through its sensors.
3	1.1.1	Sensors	The CPAS uses the following sensors: temperature, humidity, luminance, rain (gauge).
4	1.2	Behavior configuration	The farmer can configure the system, when to activate its operations based on its sensor inputs.
5	1.3	Operation execution	The CPAS shall execute operations to change its surrounding environment.
6	1.3.1	Farmer notification	Whenever a notification arrives the CPAS signals the farmer based on the configured communication mean.
7	1.3.2	System shutdown	The farmer can shut down the system that immediately stops all of its active operations.
8	1.4	Supported operations	Supported operations are mowing, irrigation and spraying.
9	1.4.1	Mowing	
10	1.4.1.1	Execution of mowing	The mowing operation signals the robot lawn-mower to execute its programmed task.
11	1.4.1....	Successful mowing	If the mower robot executes its task without any problem it returns to its refueling station.
12	1.4.1....	Failed mowing	If the mower robot fails to complete its task, it sends a notification about its status.
13	1.4.1.2	Configuration of mowing	
14	1.4.2	Irrigation	
15	1.4.2.1	Execution of irrigation	The irrigation operation simply activates the pre-installed irrigation-system.
16	1.4.2....	Failed irrigation	If the irrigation-system fails, it sends a notification about its status.
17	1.4.2.2	Configuration of irrigation	
18	1.4.3	Spraying	The system shall support the control of manual spraying.
19	1.4.3.1	Initialization of spraying	The spraying operation signals the laborers to execute the spraying task.
20	1.4.3.2	Reporting of spraying	The laborers report to the system when they finished their task.
21	1.4.3....	Successful spraying	The laborers report to the system when they finished their task.
22	1.4.3....	Failed spraying	In case an error occurs during the spraying the laborers submit a form to the CPAS and it notifies the farmer.
23	2	System Maintenance	The system shall provide diagnostic information about its components for maintenance.

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



bdd [Package] 1 - Textual Requirements [Requirements traceability]

Direct notation

«functionalRequirement»
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»

«System use case»
Perform agricultural tasks

«refine»

«activity»
Perform agricultural tasks

Compartment notation

«functionalRequirement»
Performing agricultural tasks

RefinedBy = Perform agricultural tasks

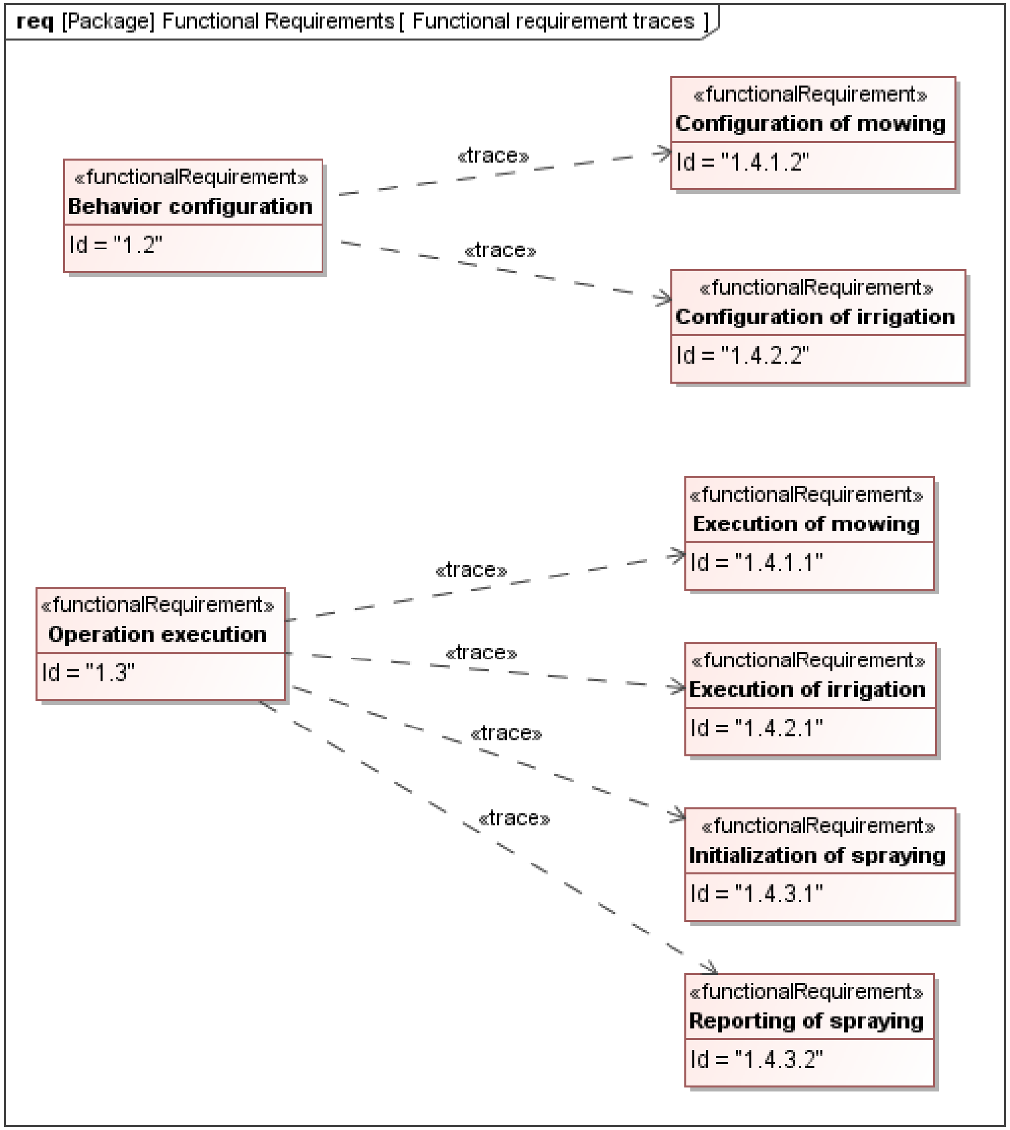
Callout notation

«functionalRequirement»
Performing agricultural tasks

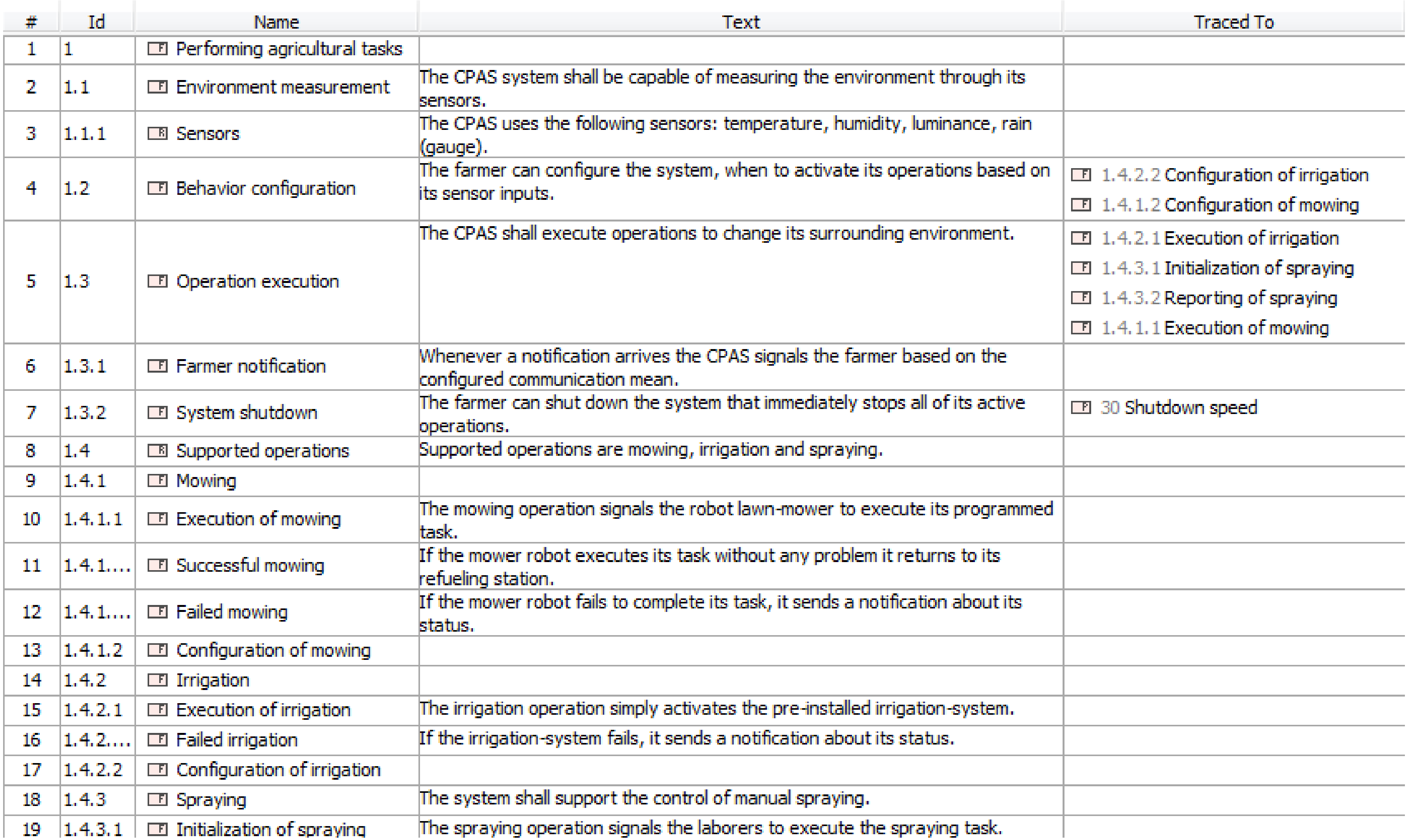
Id = "1"
Text = "The system shall support execution of autonomous and manual agricultural tasks according to predefined rules taking into account environmental effects."

«requirement»
RefinedBy
Perform agricultural tasks

Example trace relationships



Requirements Relations in Table



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

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

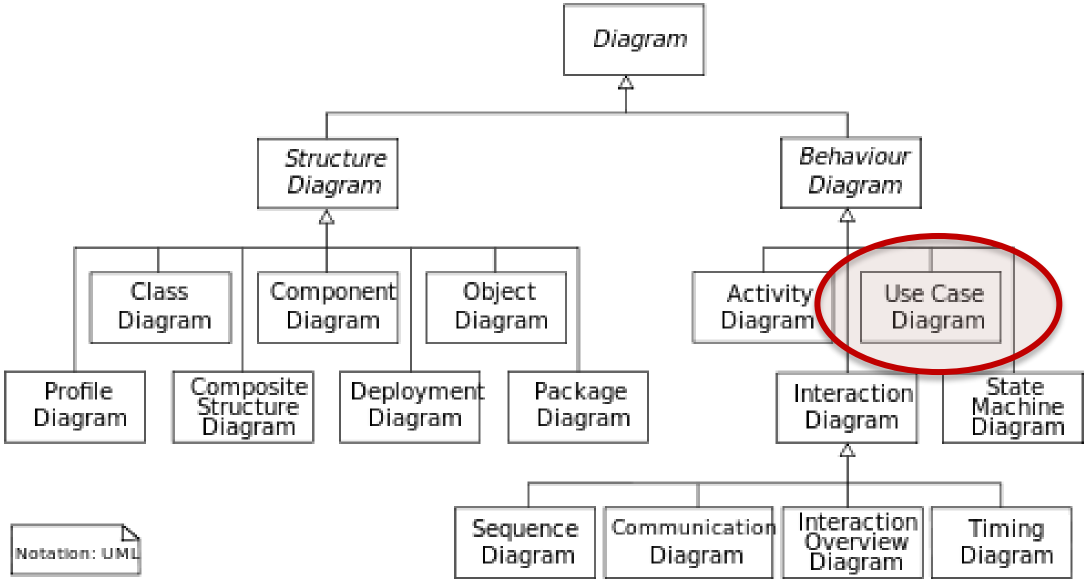
Context of the Modeling Aspect

Elements of Use Case Diagrams by Example

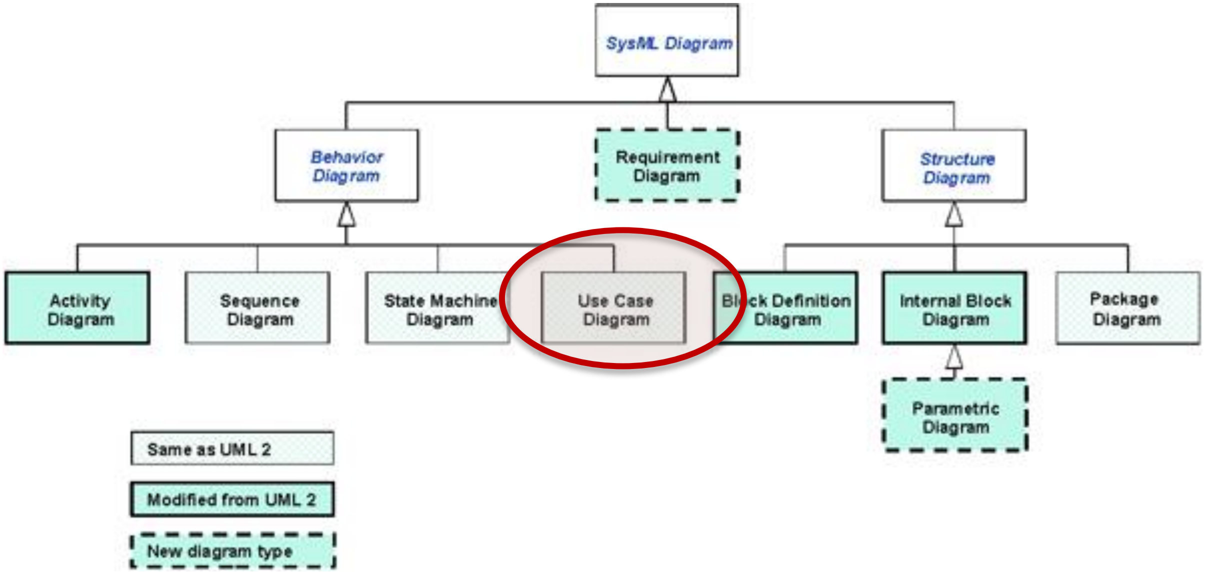
Relations between UC elements

Summary

Use Case Diagrams

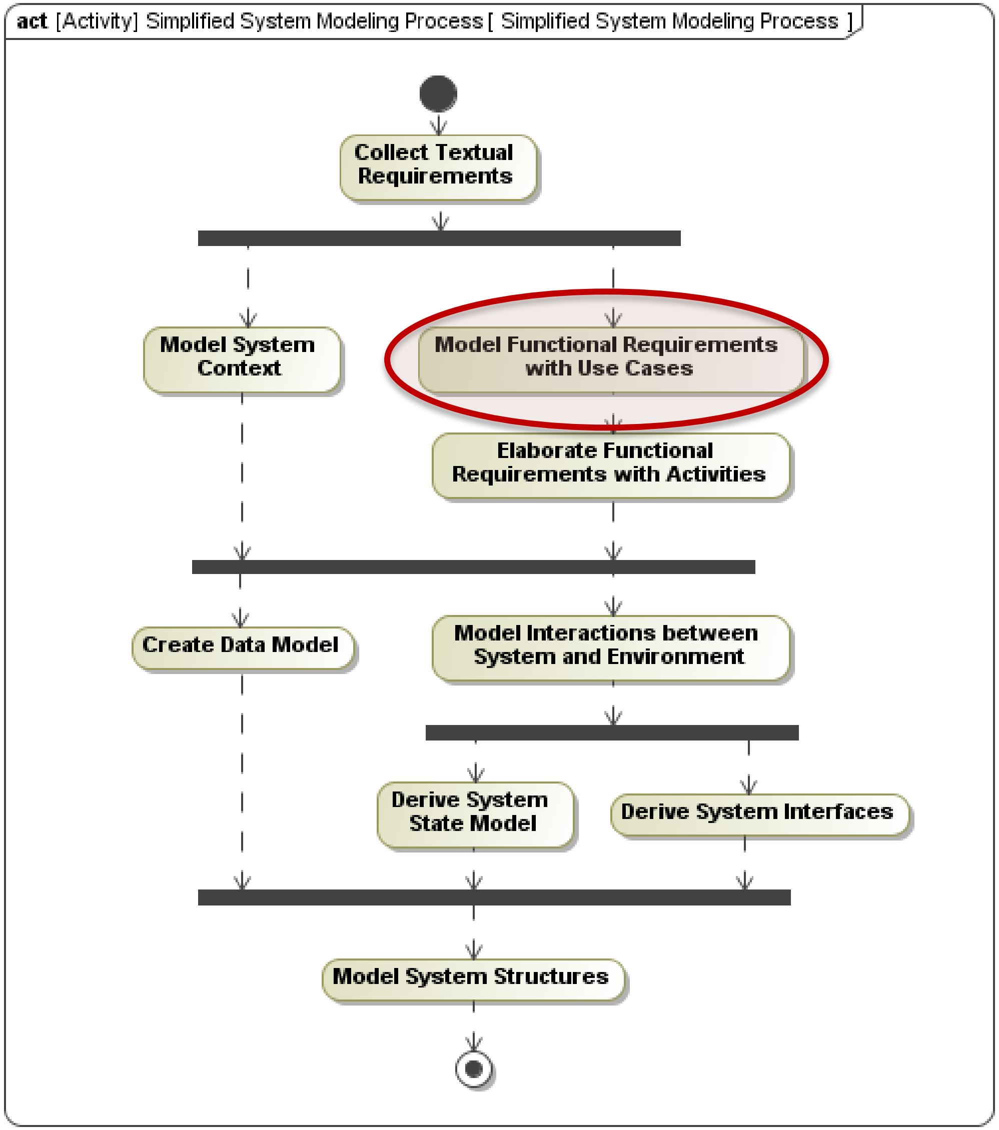


UNIFIED MODELING LANGUAGE ™



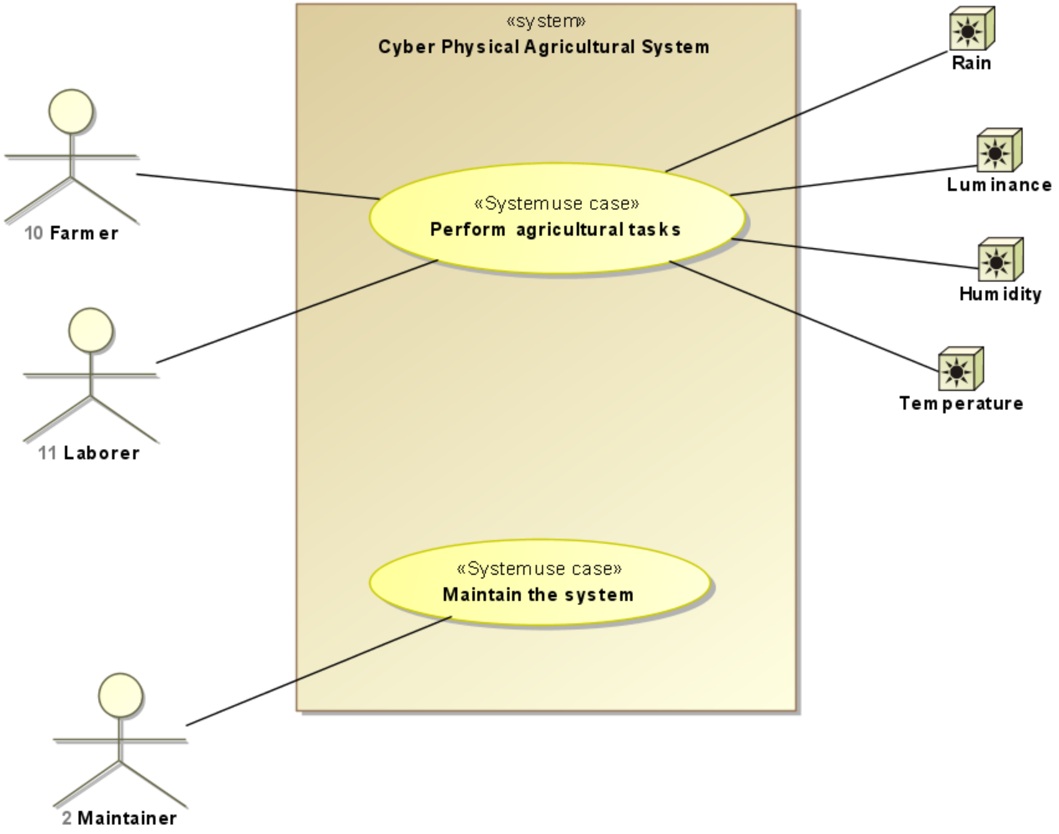
OMG SYSTEMS MODELING LANGUAGE ™

System Modeling Process


act [Activity] Simplified System Modeling Process [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

Modeling Aspect

Who will use the system and for what?



Definition of Use Cases

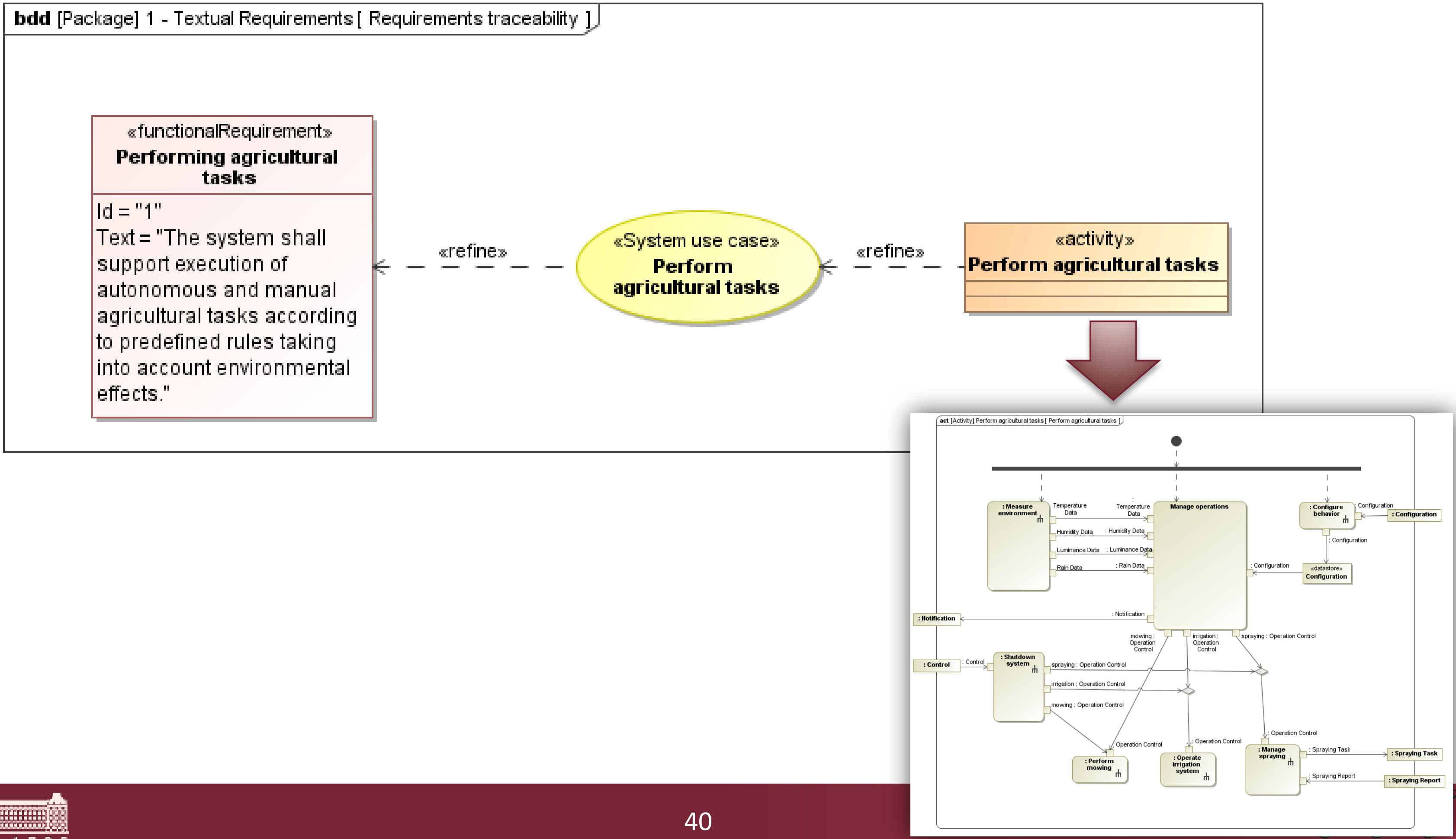
- **Use cases (használati eset)** capture the functional requirements of a system
- UCs describe
  - the typical interactions
  - between the *users* of a *system* and
  - **the system itself,**
  - by providing a narrative of how a system is used

M. Fowler: UML Distilled.
3rd Edition. Addison-Wesley

- 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)



bdd [Package] 1 - Textual Requirements [Requirements traceability]

«functionalRequirement»
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»

«System use case»
Perform agricultural tasks

«refine»

«activity»
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

Example requirements

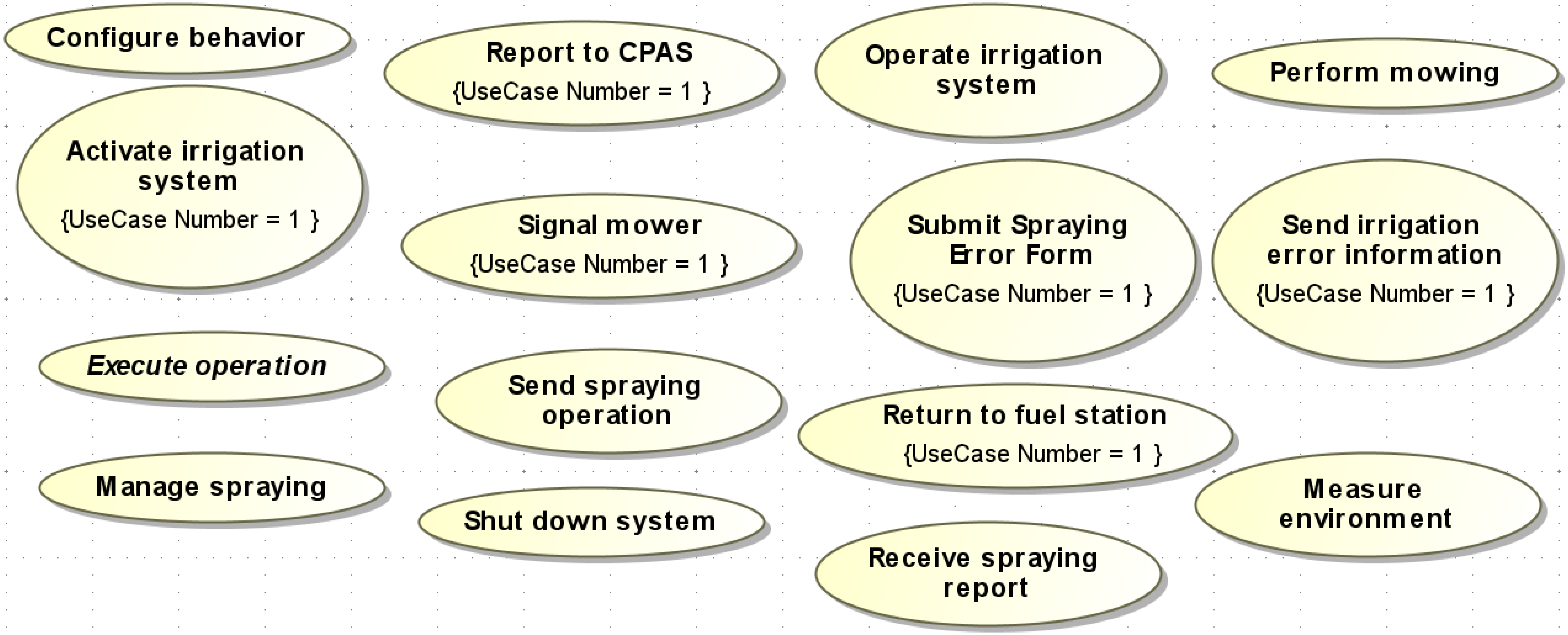
- Design a simple Cyber-physical agricultural system (CPAS), which helps a farmer with his/her everyday life using sensors to measure the environment and react to its changes by using automated operations like irrigation, mowing and spraying.

- 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

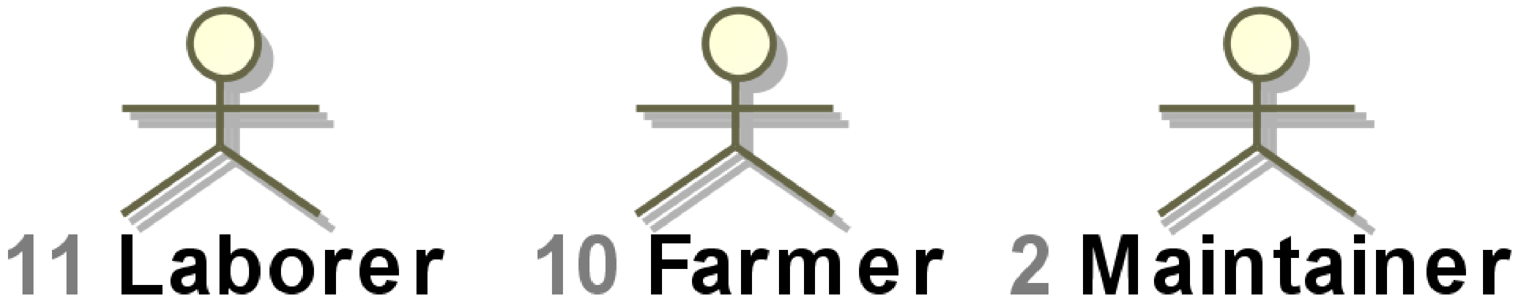
Send irrigation error information
{UseCase Number = 1 }

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

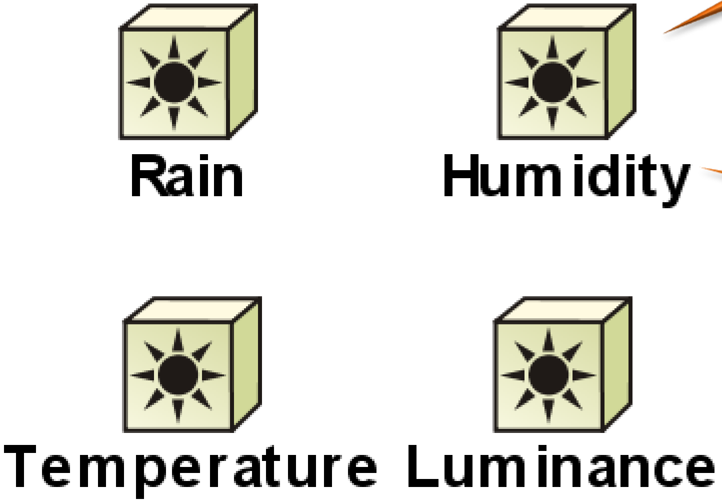


11 Laborer

10 Farmer

2 Maintainer

(Initial) Collection of Secondary Actors



Customized representation

Could use the UML actor visualization instead!

Modeling Requirements with Use Cases

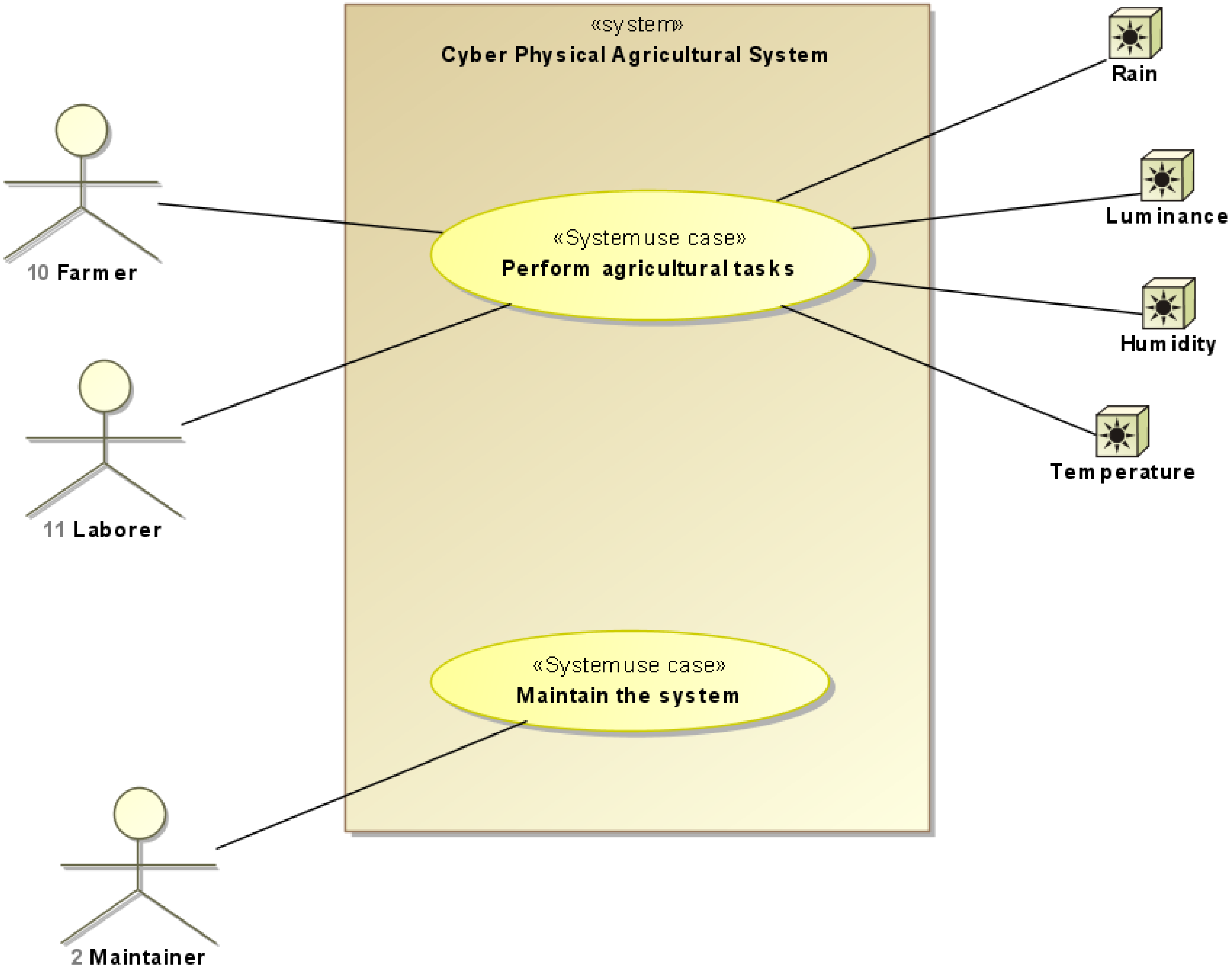
Context of the Modeling Aspect

Elements of Use Case Diagrams by Example

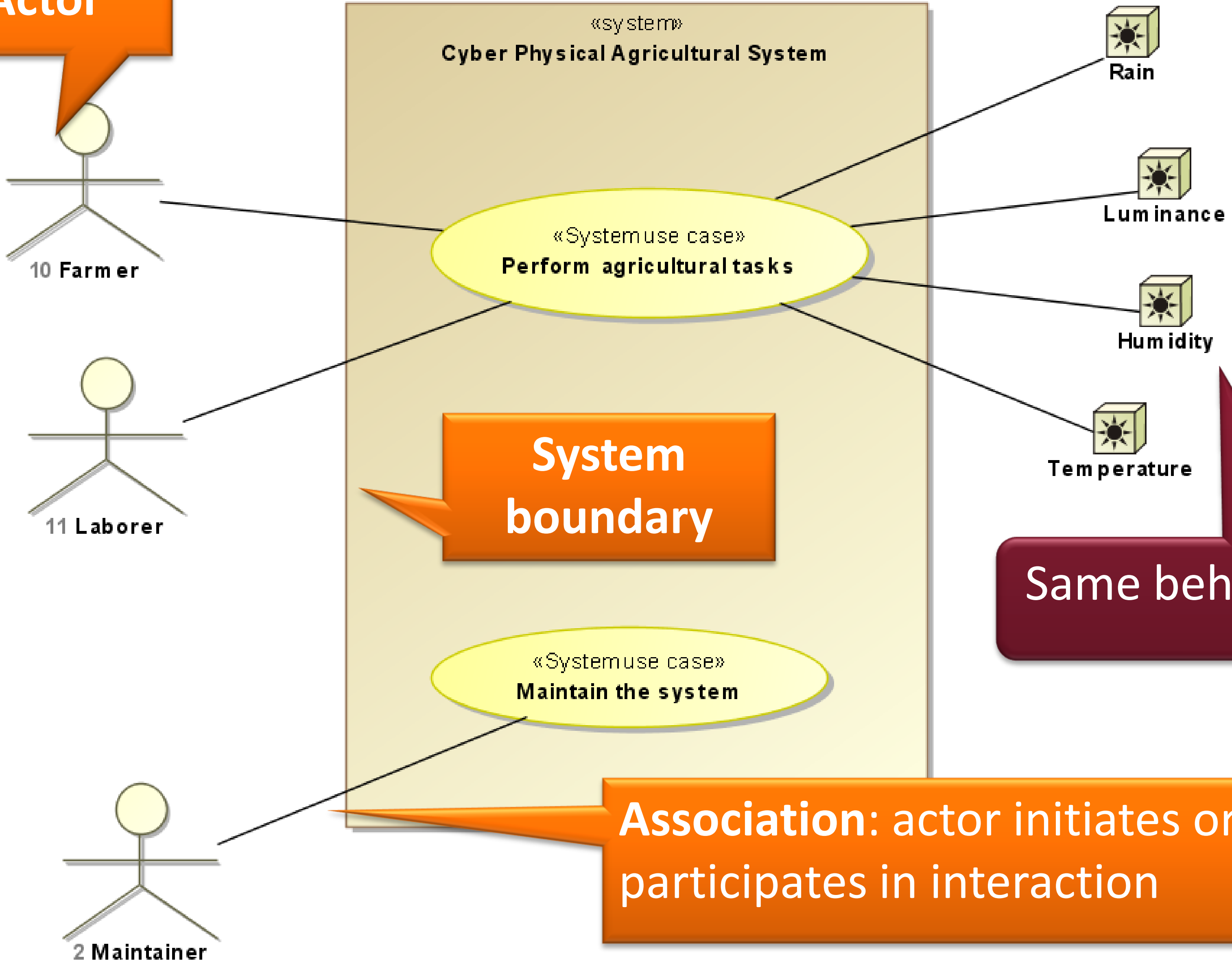
Relations between UC elements

Summary

System-level overview (User)



System-level overview (User)



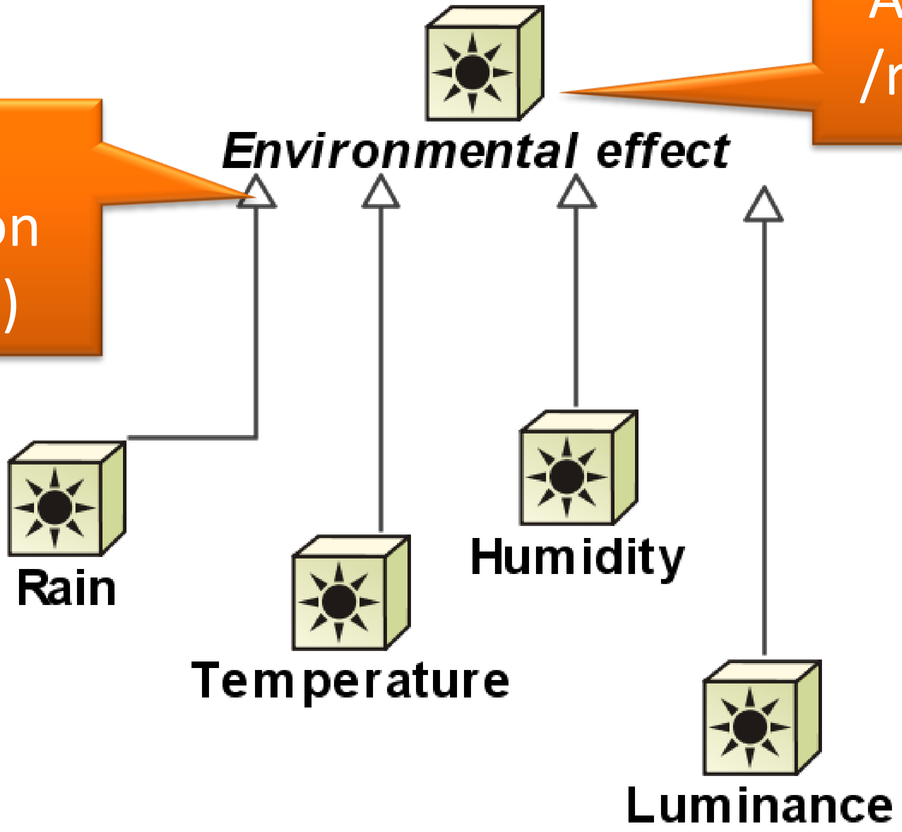
Actor

System boundary

Same behavior!

Association: actor initiates or participates in interaction

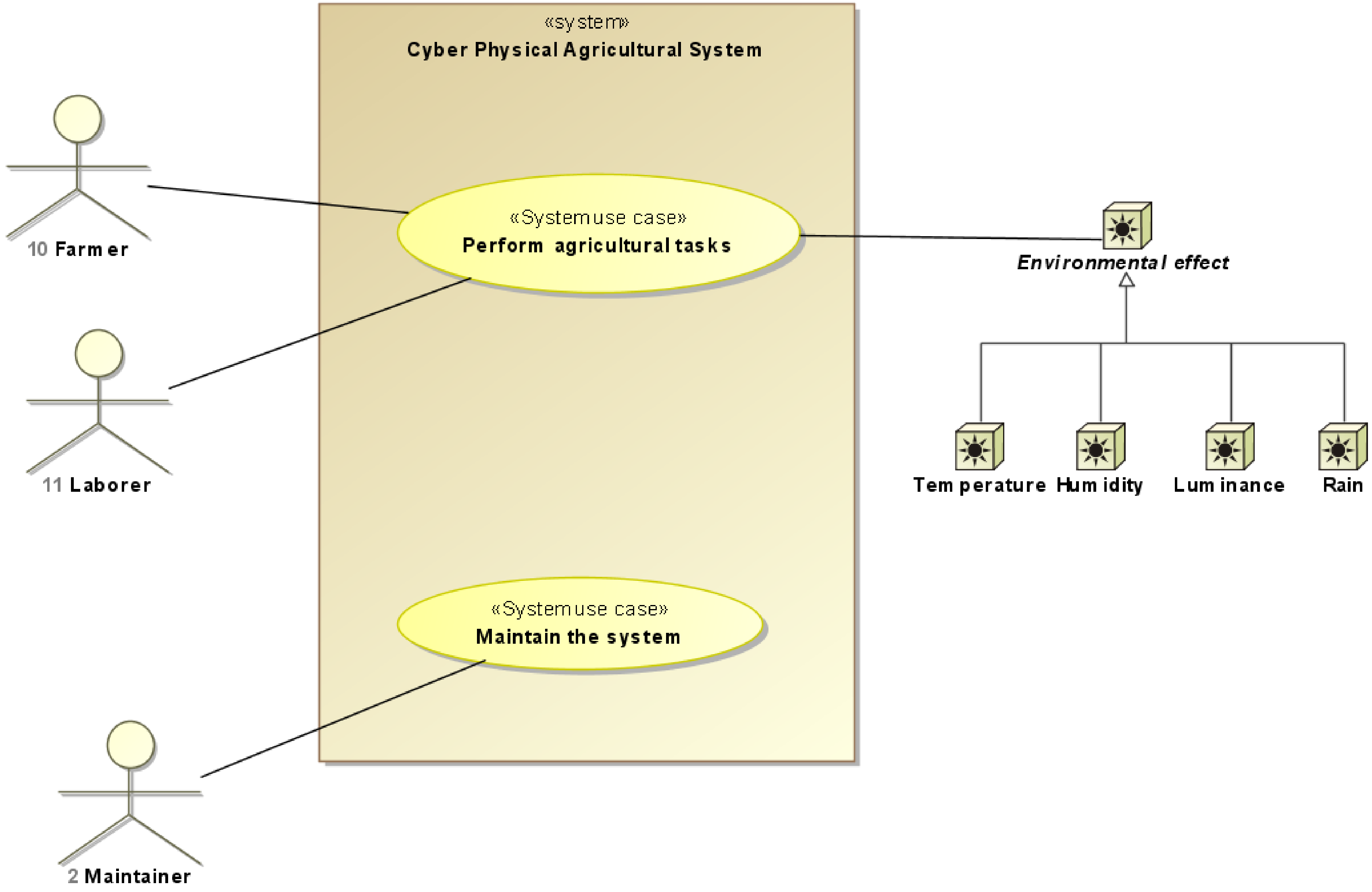
Generalization of Actors (abstraction)



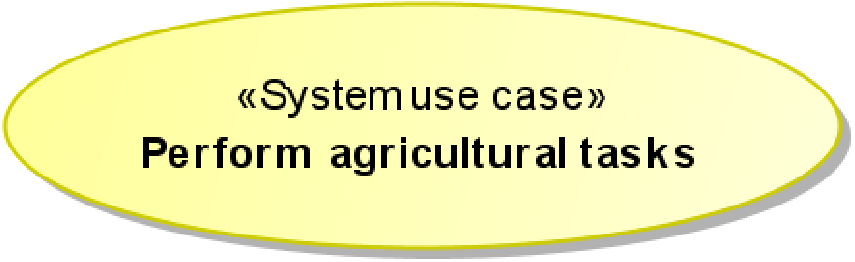
Actor Generalization (Inheritance)

Abstraction /refinement

System-level overview (User)



How to handle complex functionality?

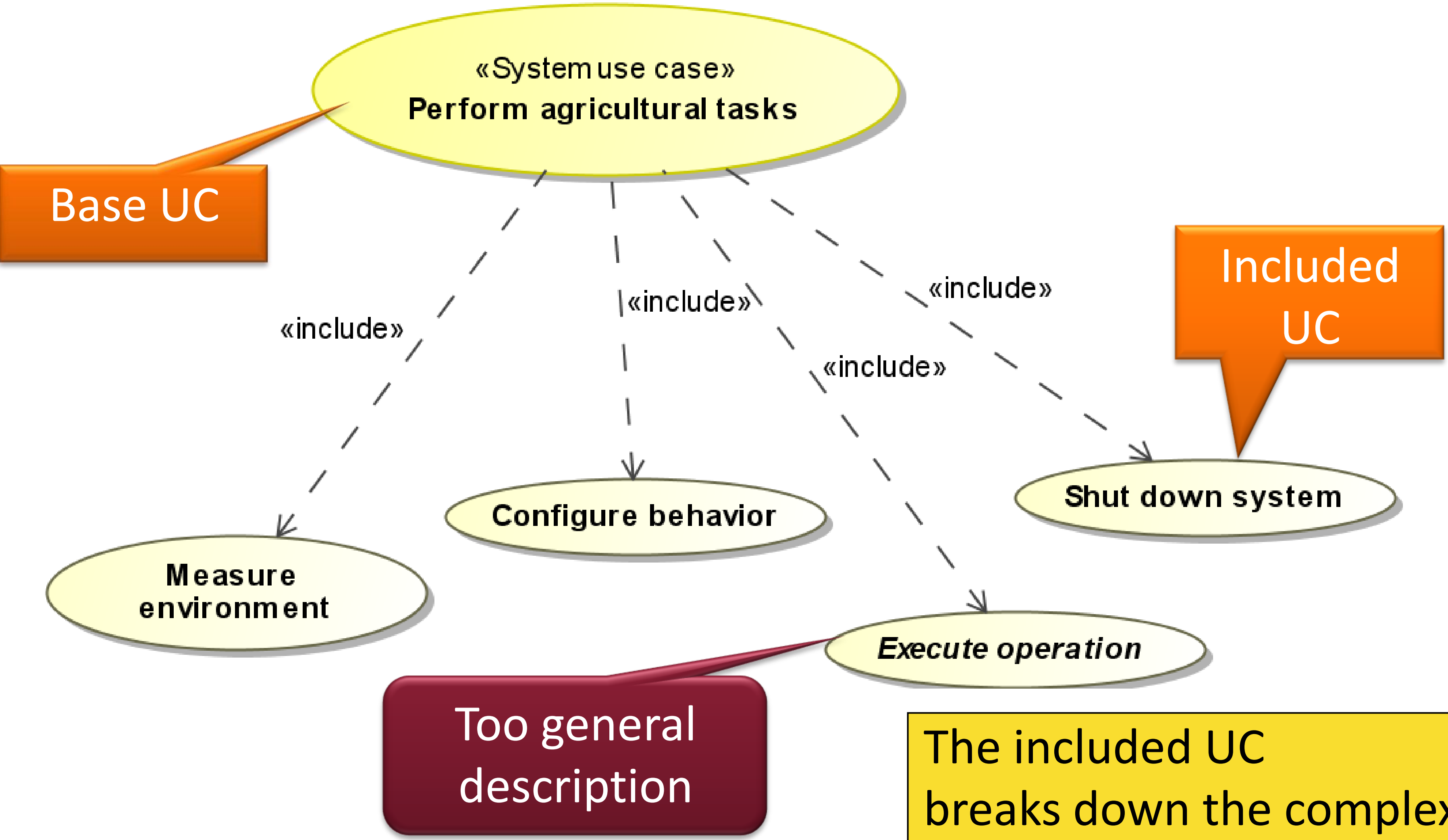


«System use case»
Perform agricultural tasks

Perform agr. tasks =

- Measure the environment
- Configure the system
- Execute agr. operations
- Shut down the system

Refinement with include relation



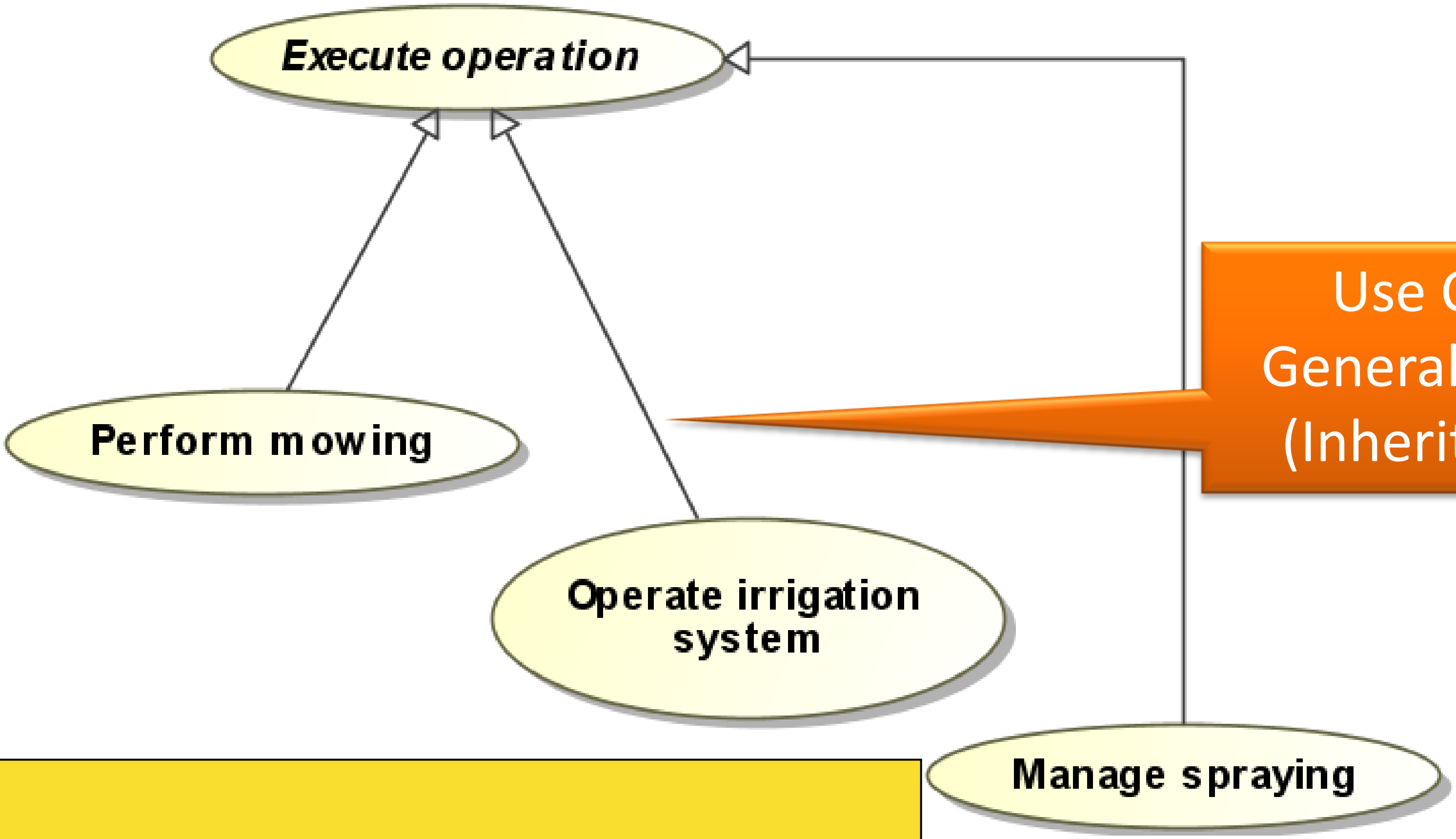
Base UC

Included UC

Too general description

The included UC breaks down the complex core functionality into more elementary steps

Generalization of UCs

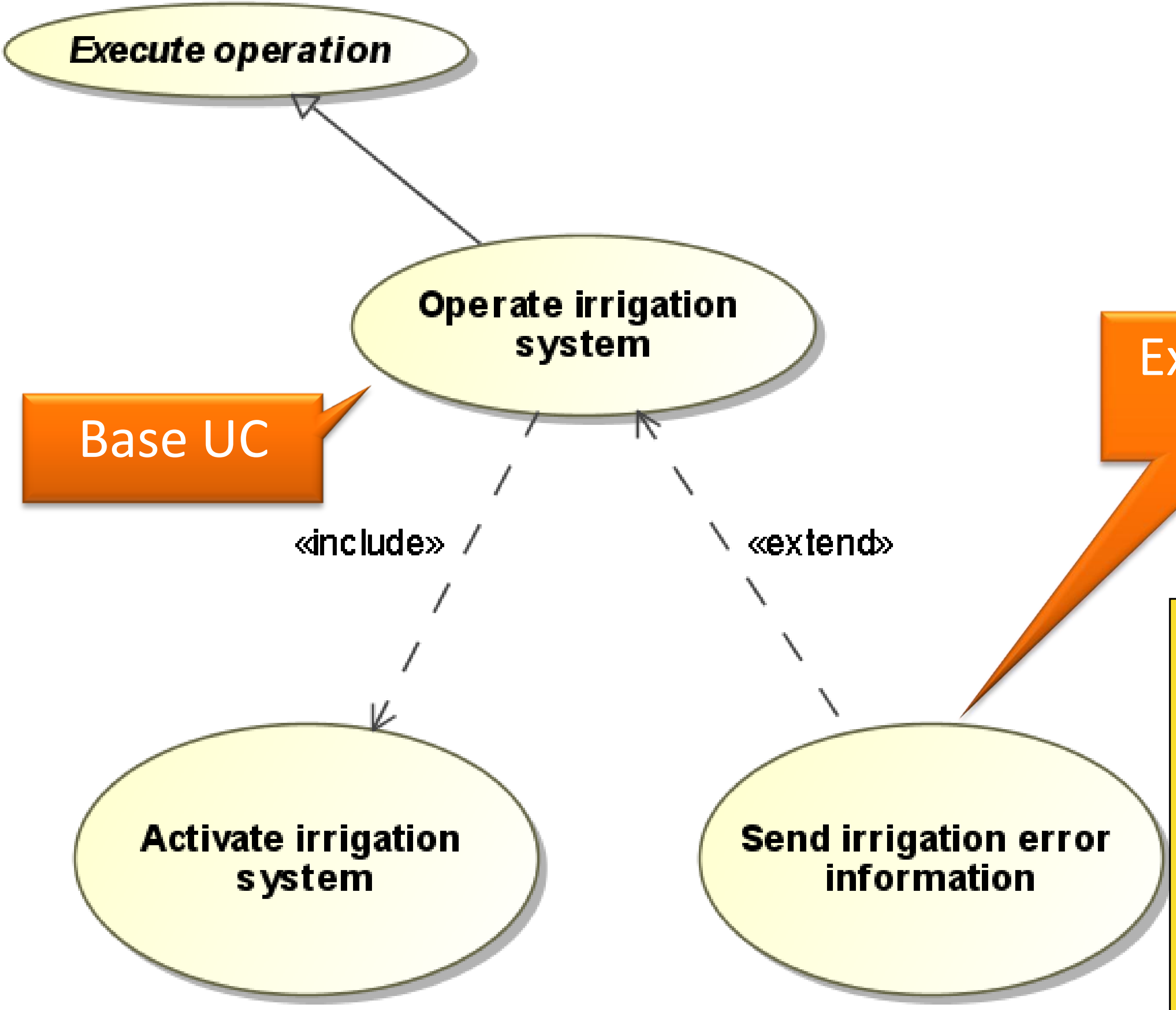


Use Case Generalization (Inheritance)

What happens if

- the irrigation operation fails?

Extend relationship


Execute operation
Operate irrigation system
«include»
«extend»
Activate irrigation system
Send irrigation error information

Base UC

Extension UC

The extension UC Extends core functionality by handling unusual (*exceptional*) situation

Summary: UC Relations

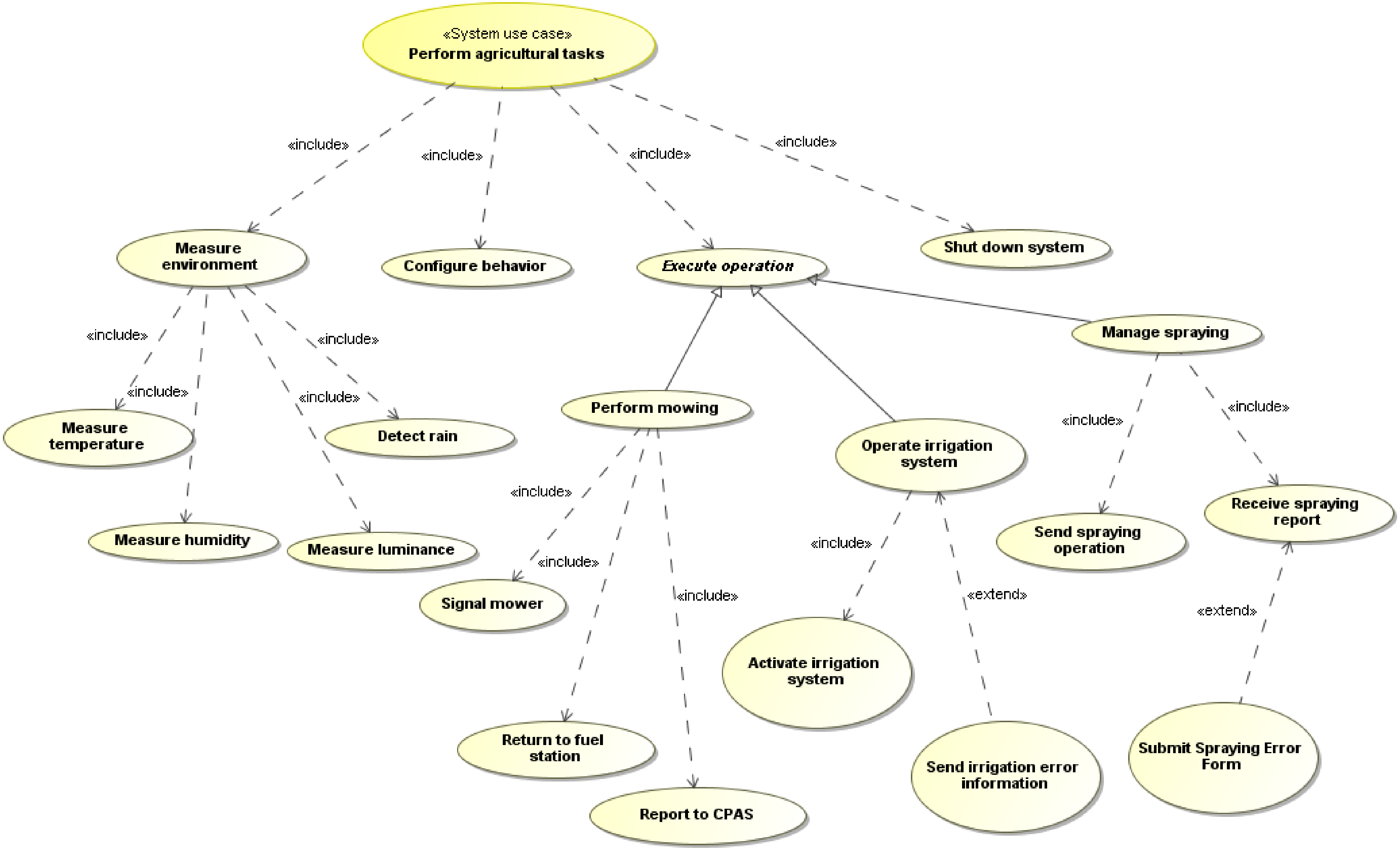
- **Association** (Asszociáció)
  - actor – use case
  - the actor initiates (or participates in) the use of the system

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

Summary: UC Relations

- **Generalization** (Általánosítás)
  - actor – actor
  - use case – use case
  - a UC or actor is more general / specific than another UC or actor
- **Include** (Beszúrás)
  - 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



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

<u>Context of the Modeling Aspect</u>

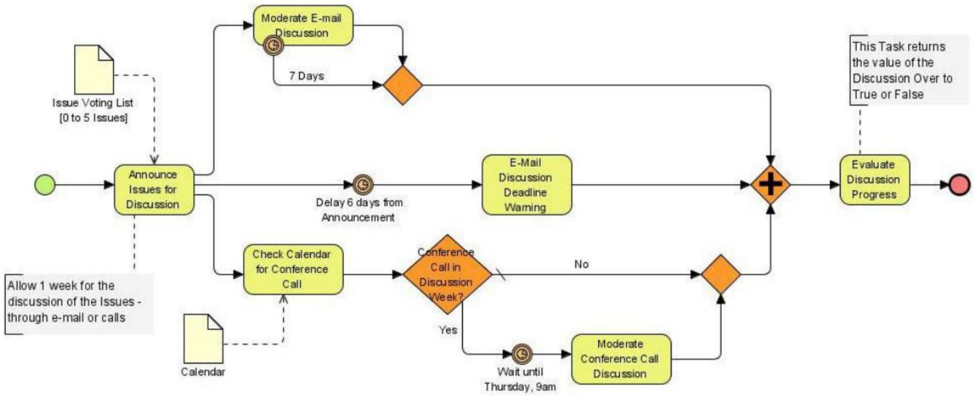
Modeling Elements & Notation

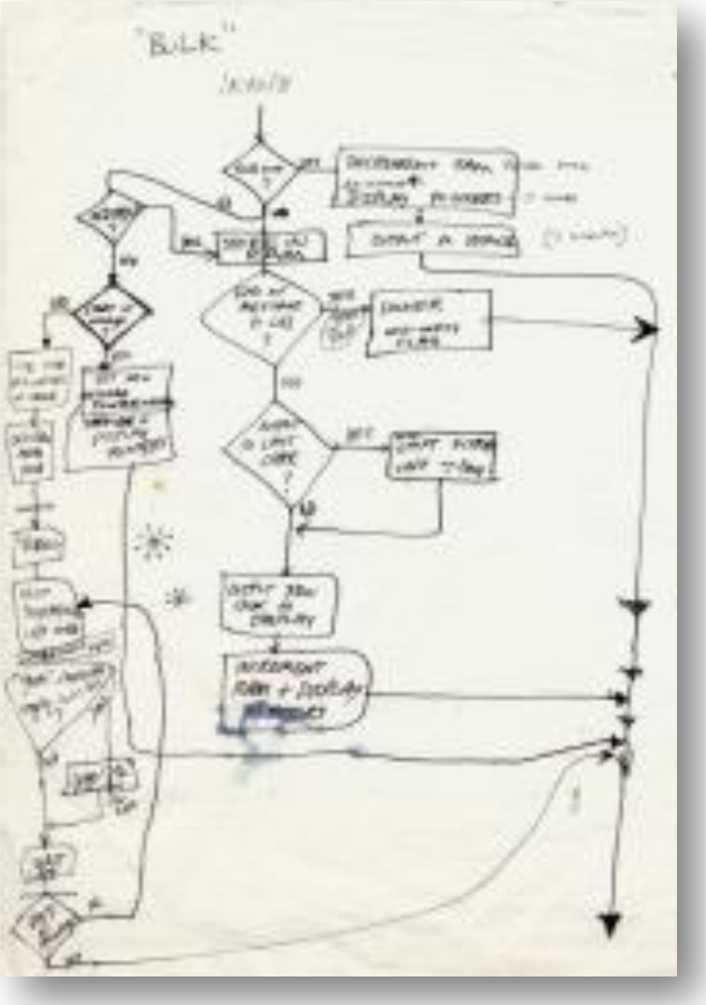
Semantics of the Model

Summary

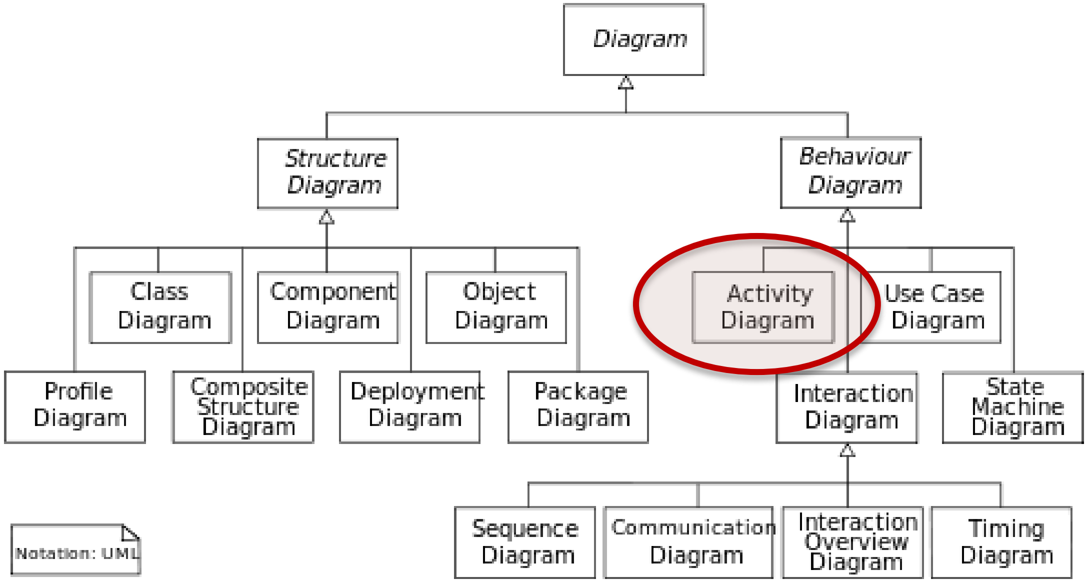
Roots & Relations

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

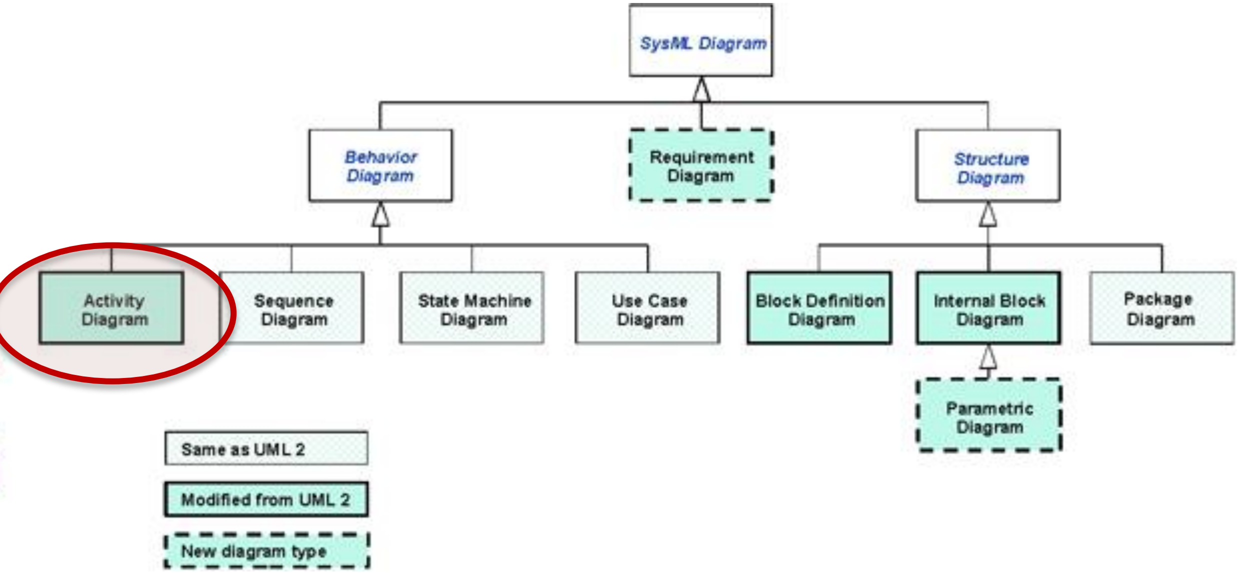

Moderate E-mail Discussion
7 Days
Issue Voting List [0 to 5 Issues]
Announce Issues for Discussion
Delay 6 days from Announcement
E-Mail Discussion Deadline Warning
This Task returns the value of the Discussion Over to True or False
Evaluate Discussion Progress
Allow 1 week for the discussion of the Issues - through e-mail or calls
Check Calendar for Conference Call
Conference Call in Discussion Week?
No
Yes
Calendar
Wait until Thursday, 9am
Moderate Conference Call Discussion



Activity Diagram

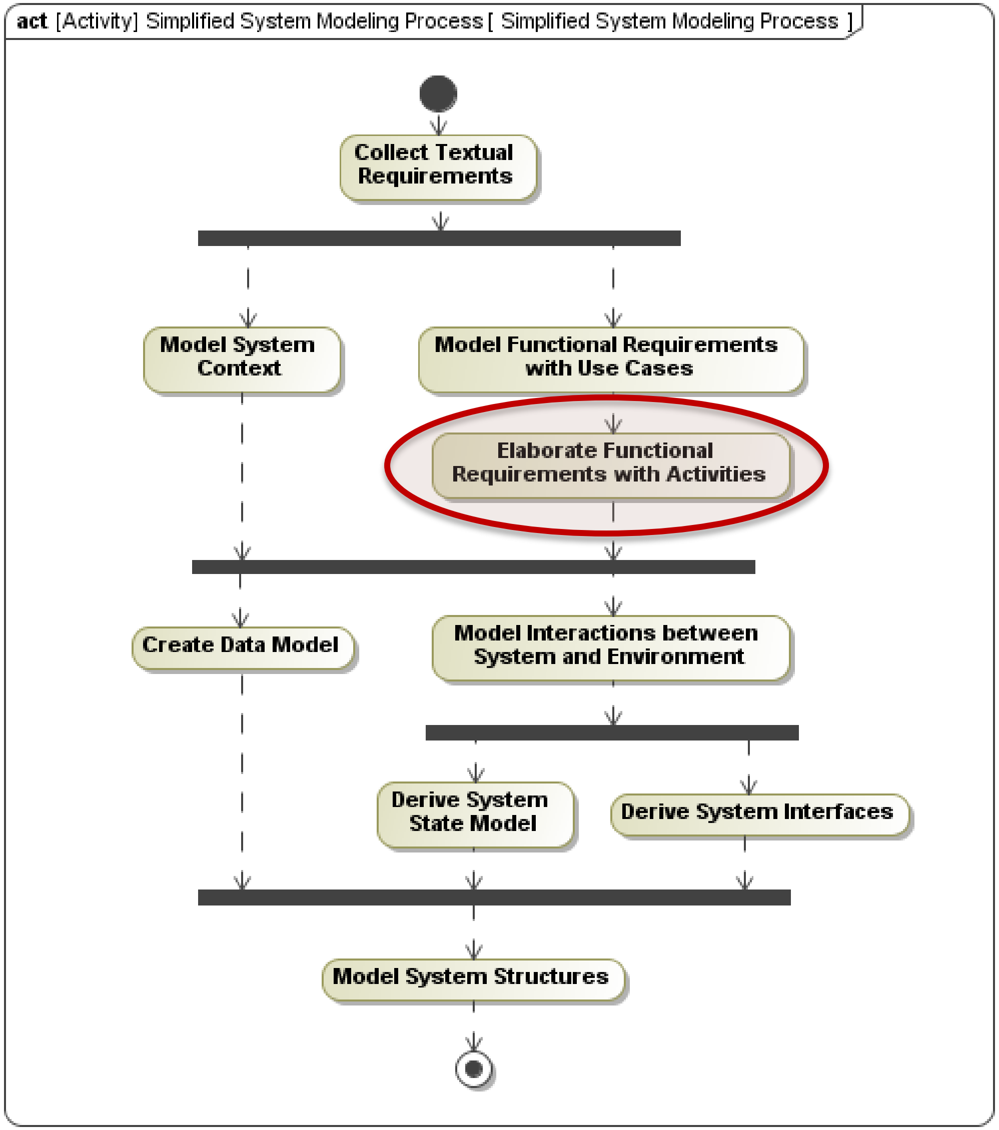


UNIFIED MODELING LANGUAGE™



OMG SYSTEMS MODELING LANGUAGE™

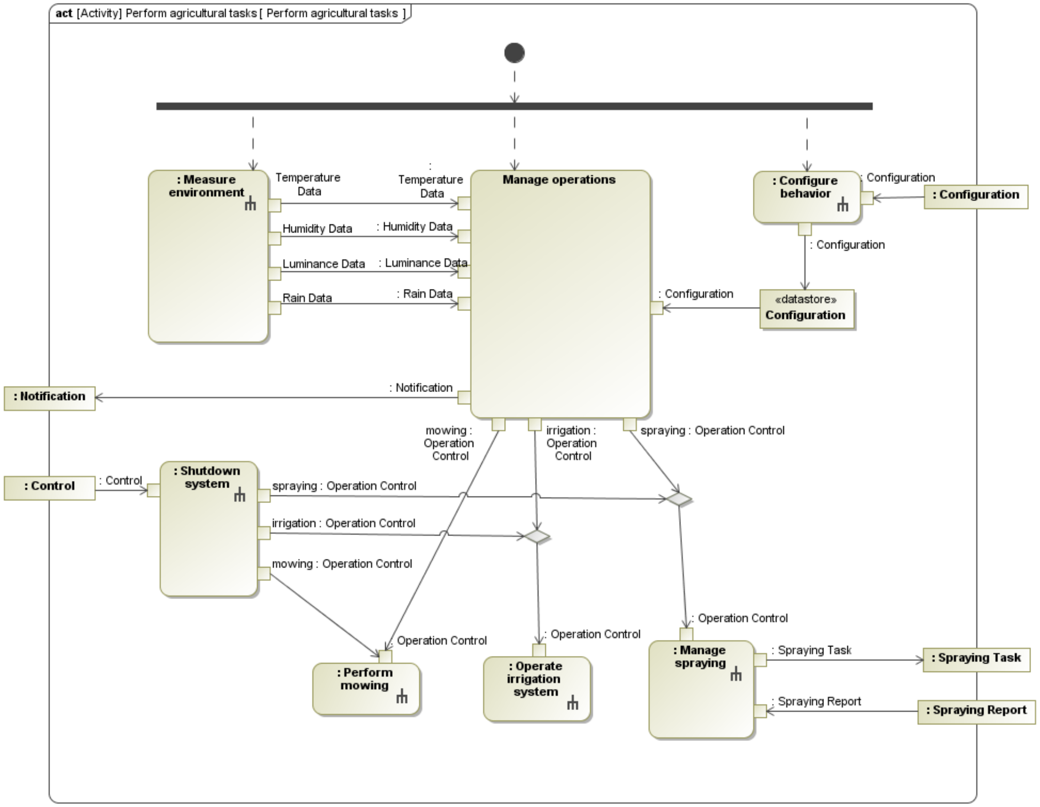
System Modeling Process


act [Activity] Simplified System Modeling Process [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

Modeling Aspect

What are the steps in a process?

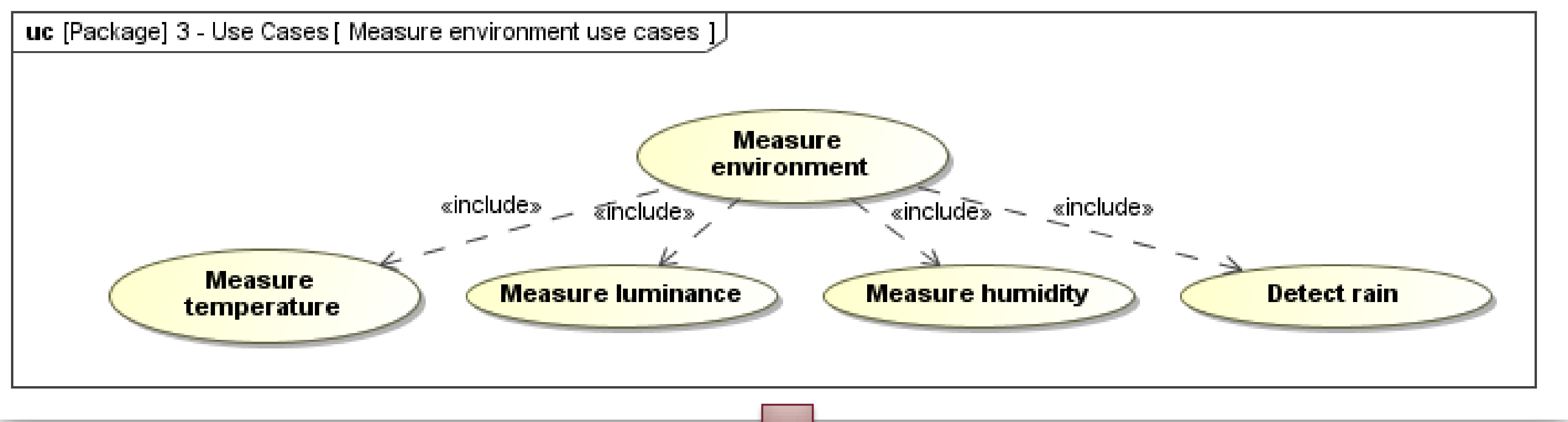
What data flows in the process?

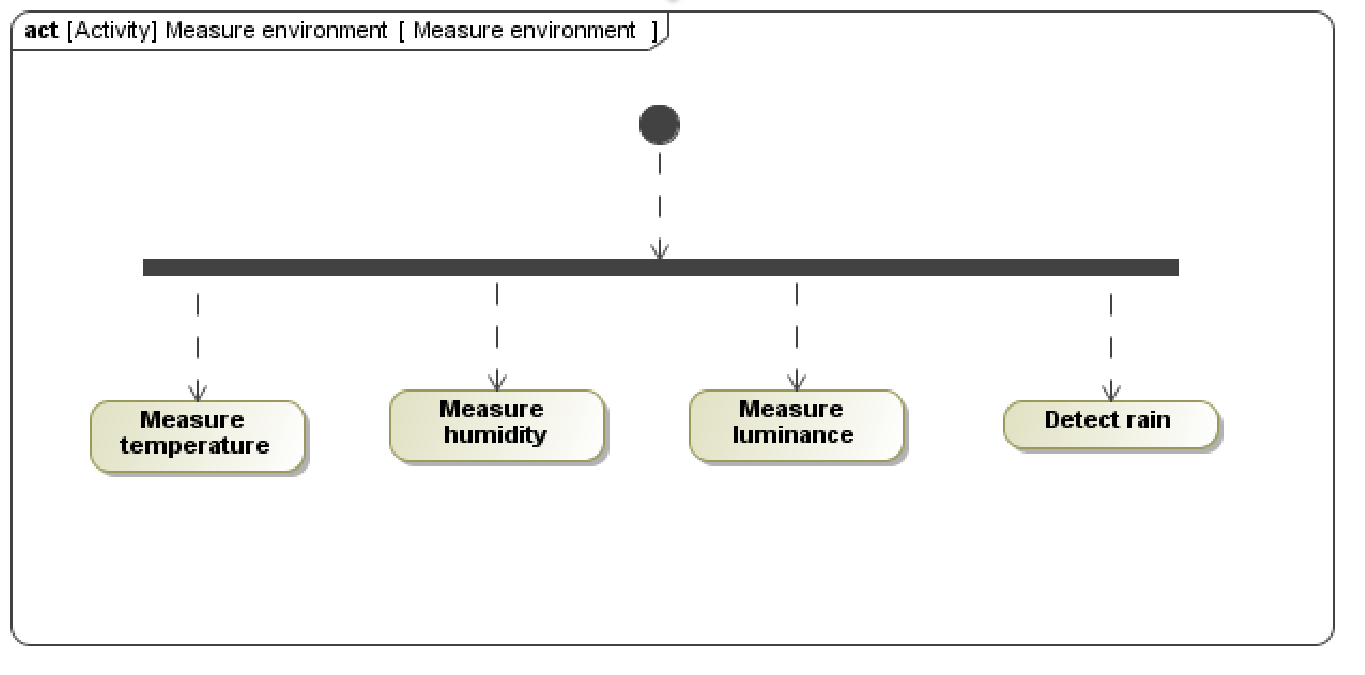


Objectives

- 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

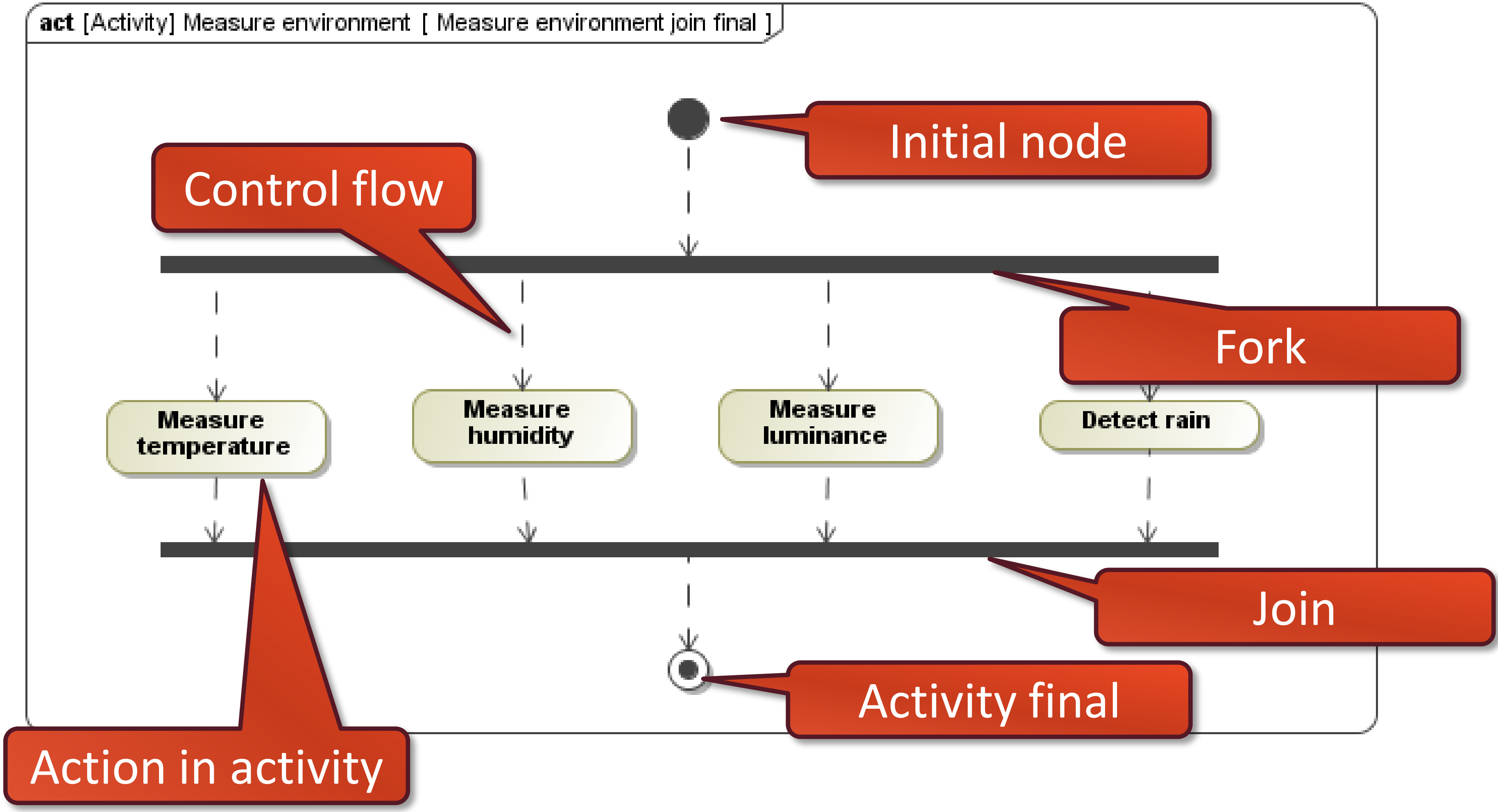
Context of the Modeling Aspect

Modeling Elements & Notation

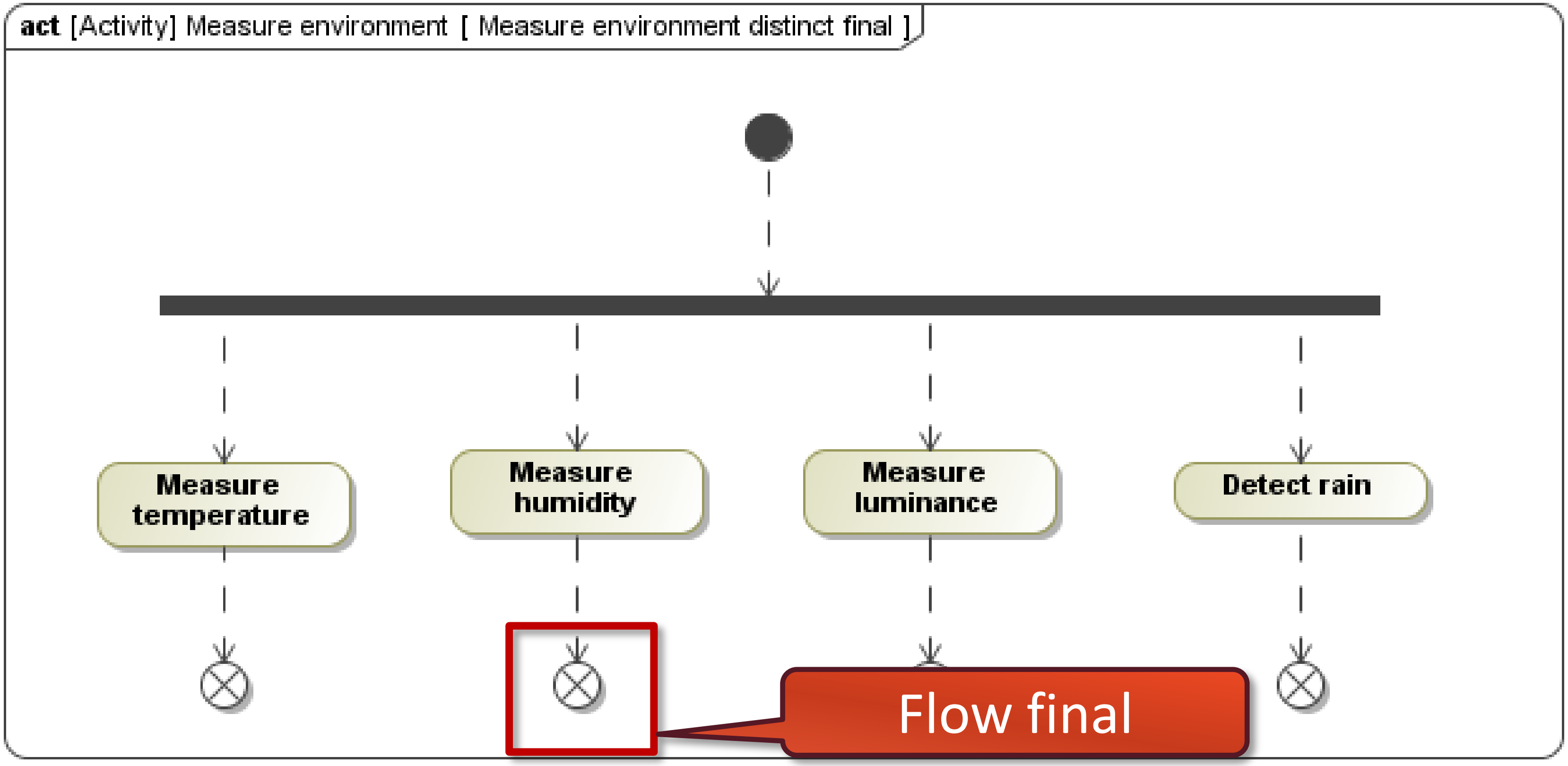
Semantics of the Model

Summary

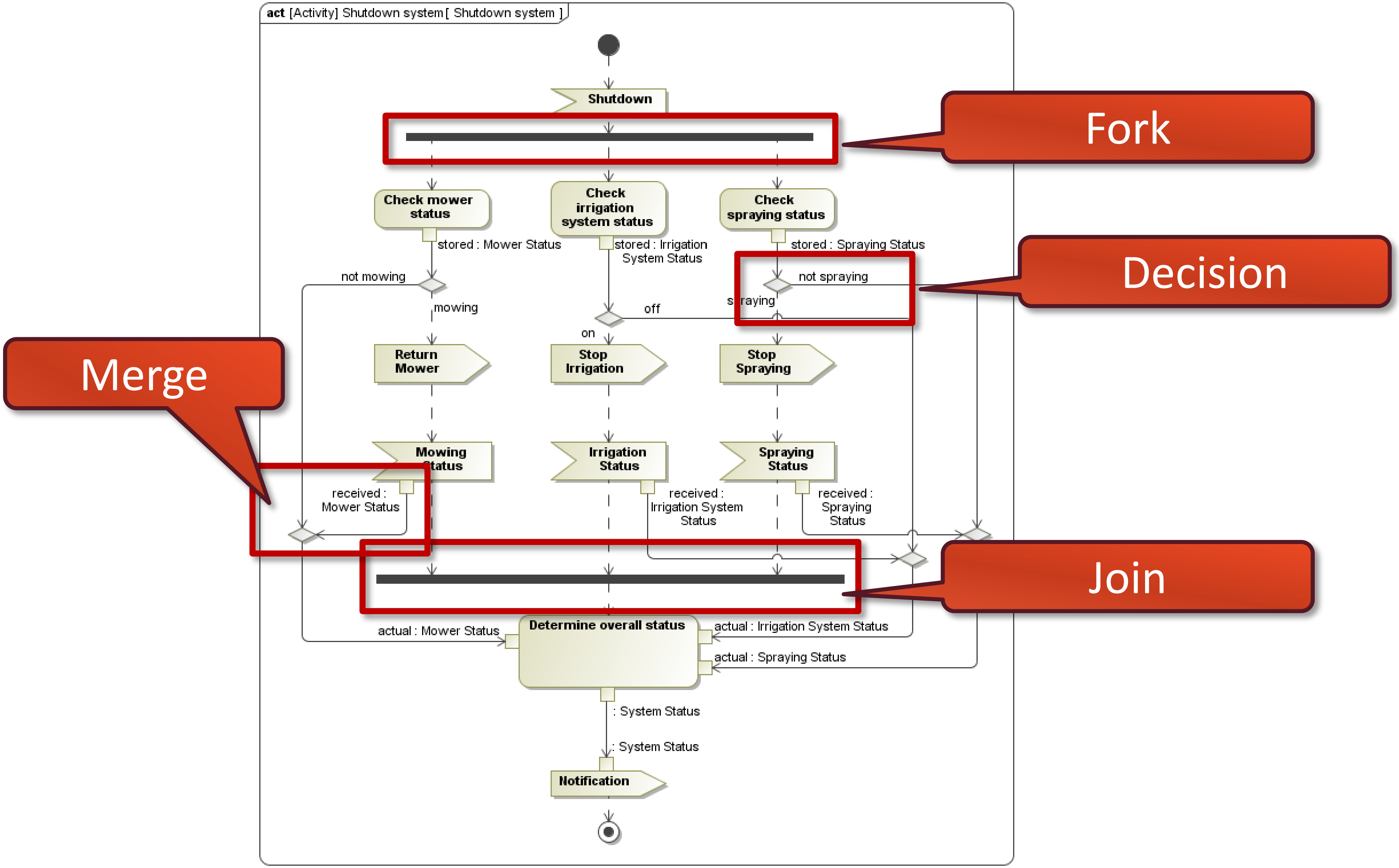
Control flow



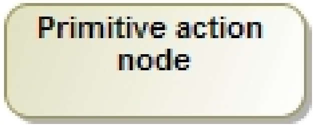
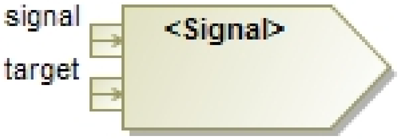
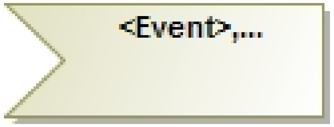
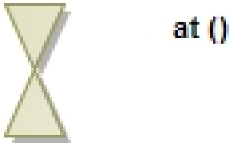
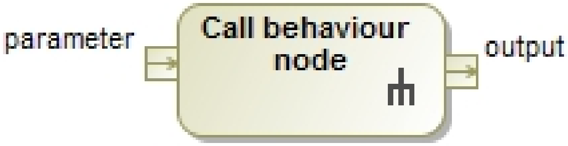
Control flow with flow final


act [Activity] Measure environment [Measure environment distinct final]
Measure temperature
Measure humidity
Measure luminance
Detect rain
Flow final

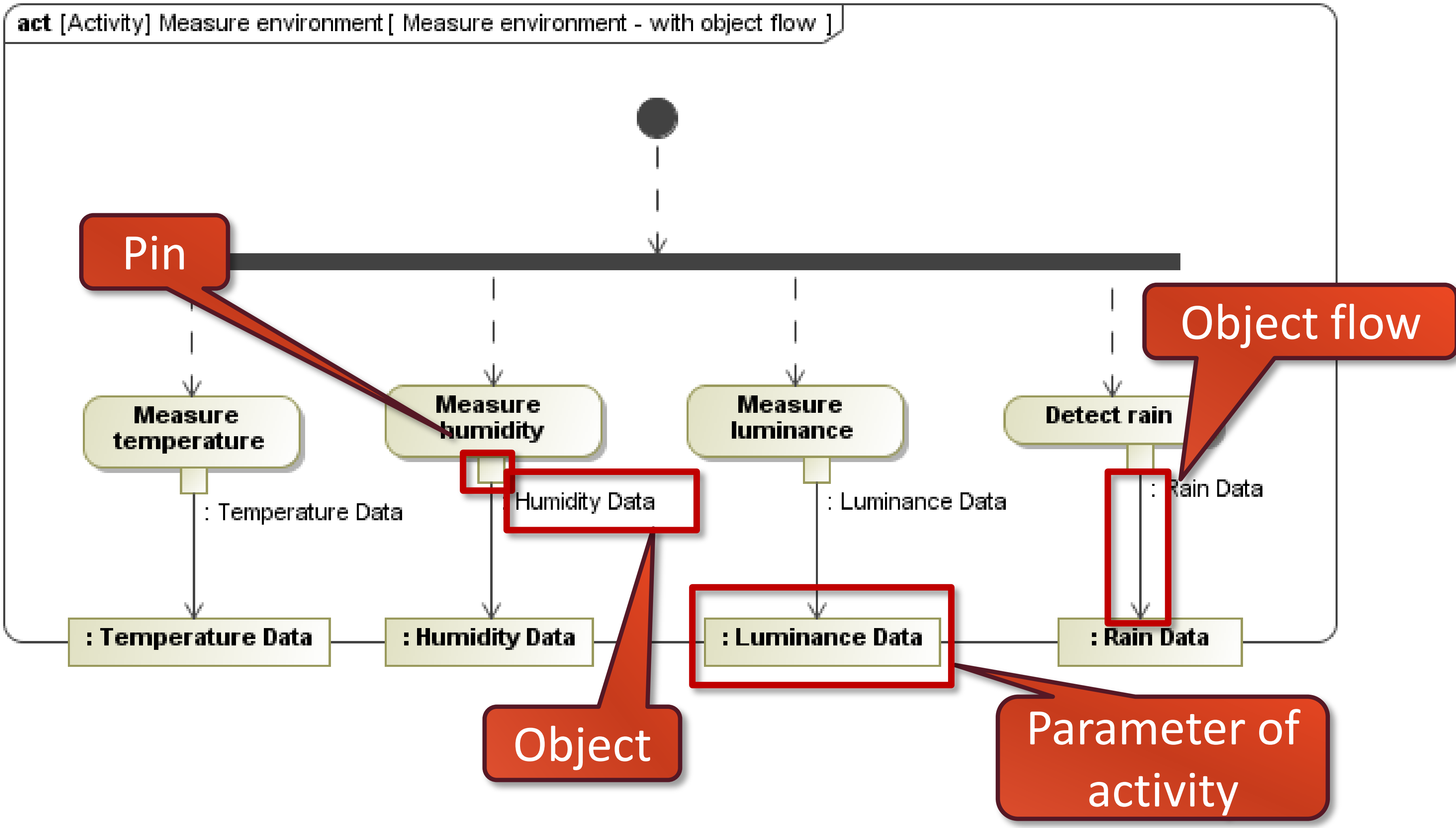
Fork, join, decision, merge



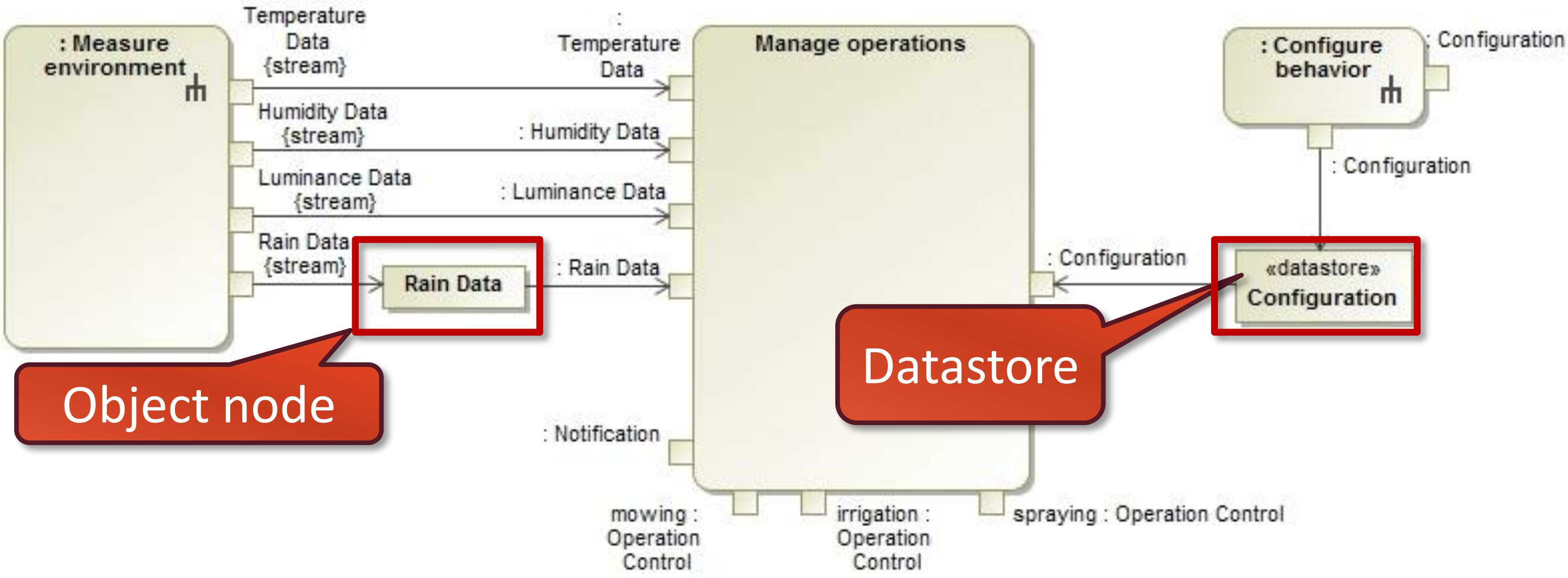
Action types

Primitive action node		Primitive actions include object access, update and manipulation actions.
Send signal node		Send signal data to target.
Accept event node		Accept events, typically has output pins for received data.
Accept time event node		Time event corresponds to an expiration of an (implicit) timer.
Call behavior node		Call other behavior (e.g. another activity).

Combined control and data flow

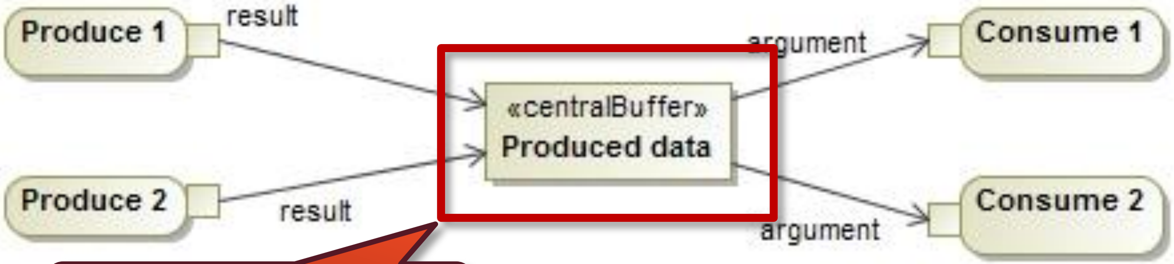


Object nodes



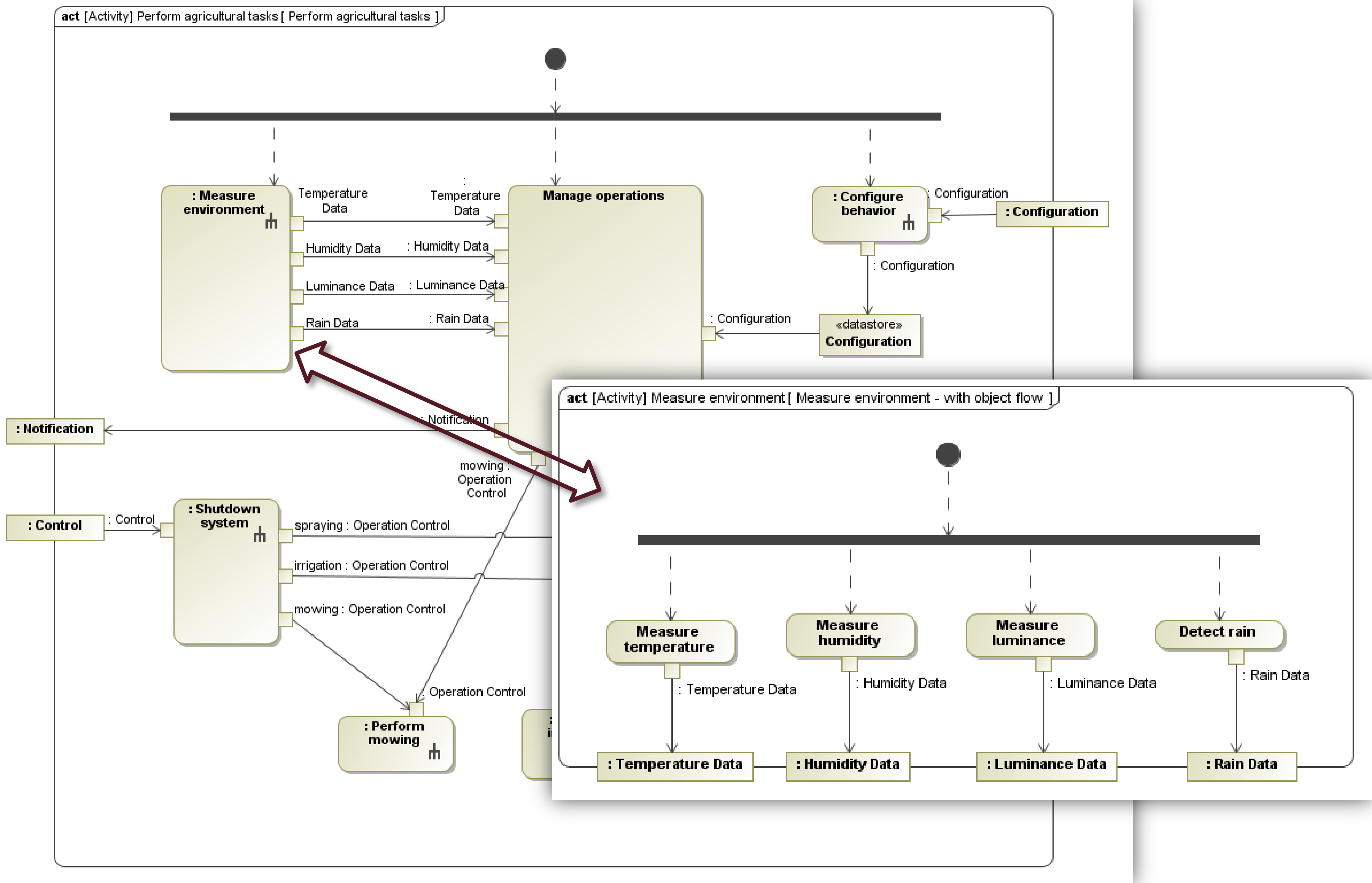
Object node

Datastore

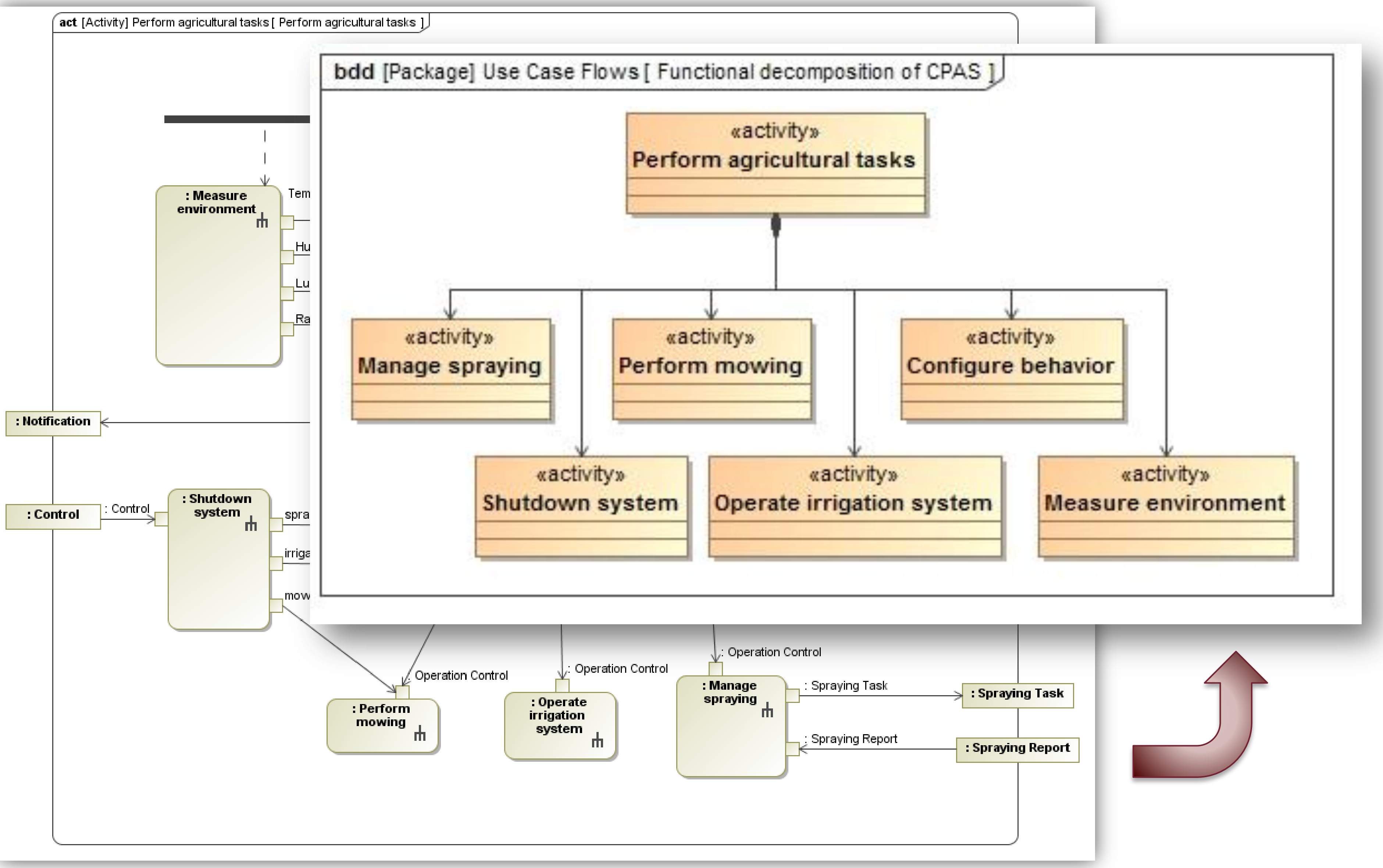


Central buffer

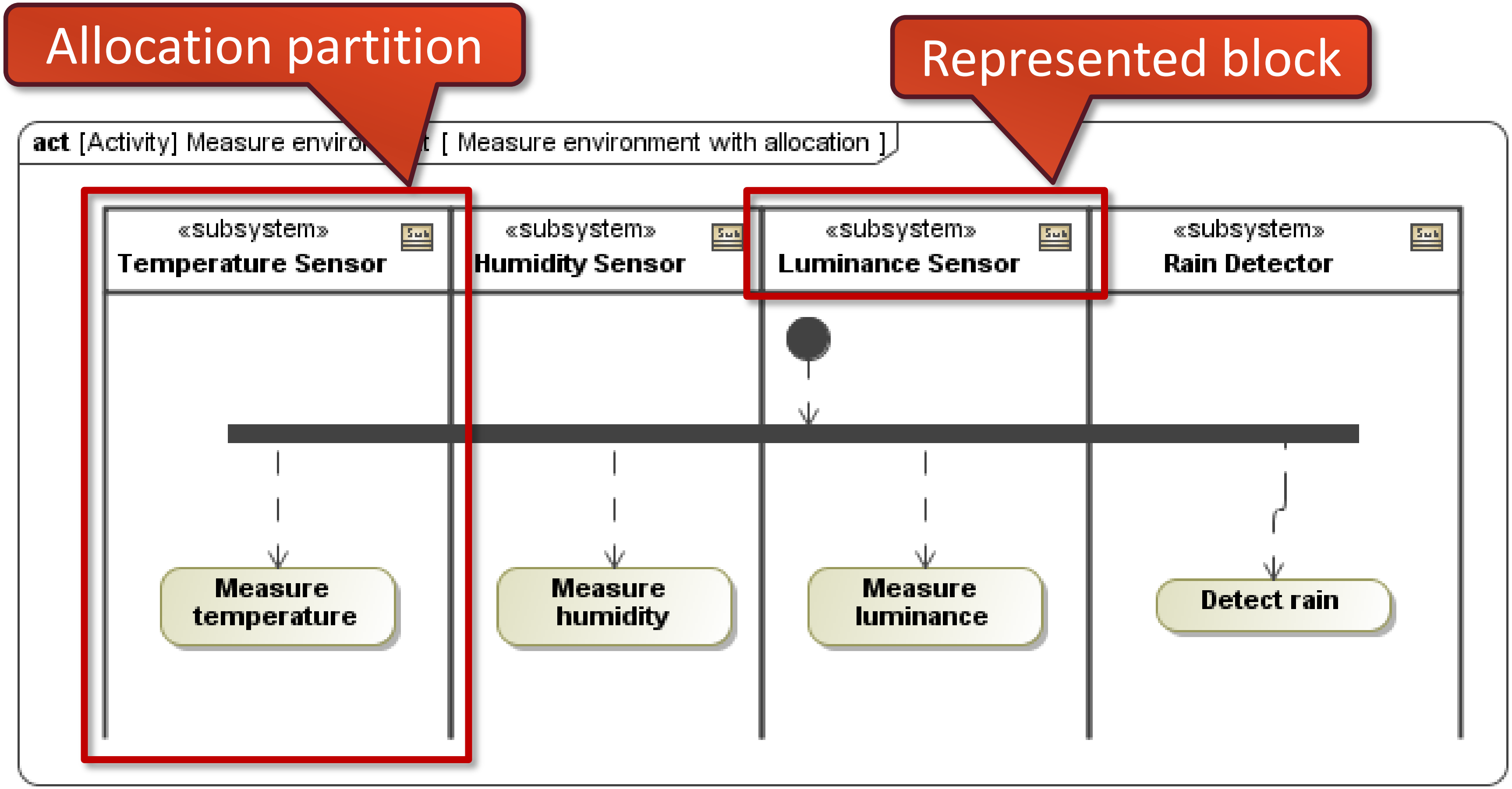
Activity decomposition



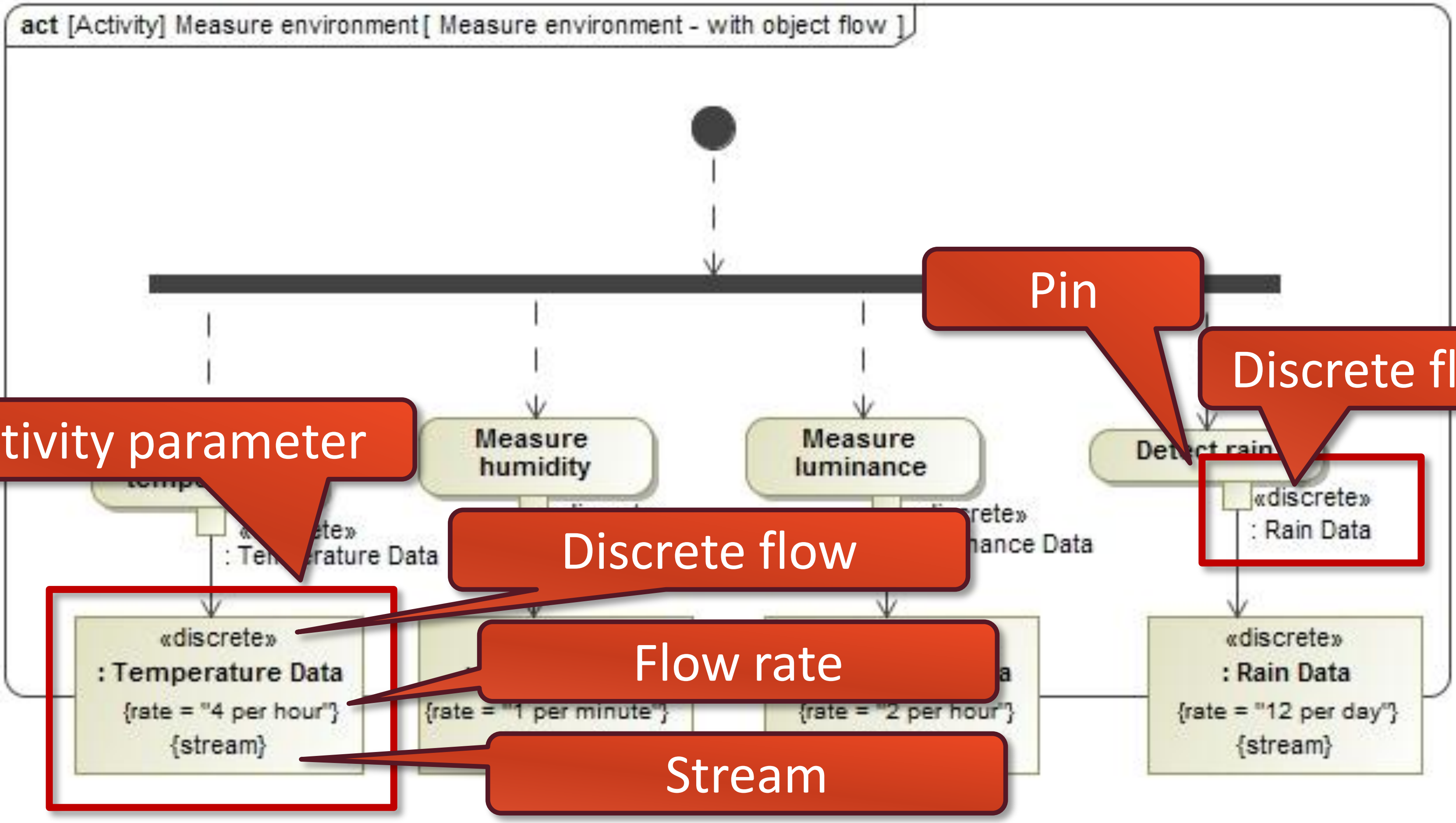
Activity Hierarchy (Functional Decomposition)



Allocating actions



Modelling Streams (SysML)



Pin

Discrete flow

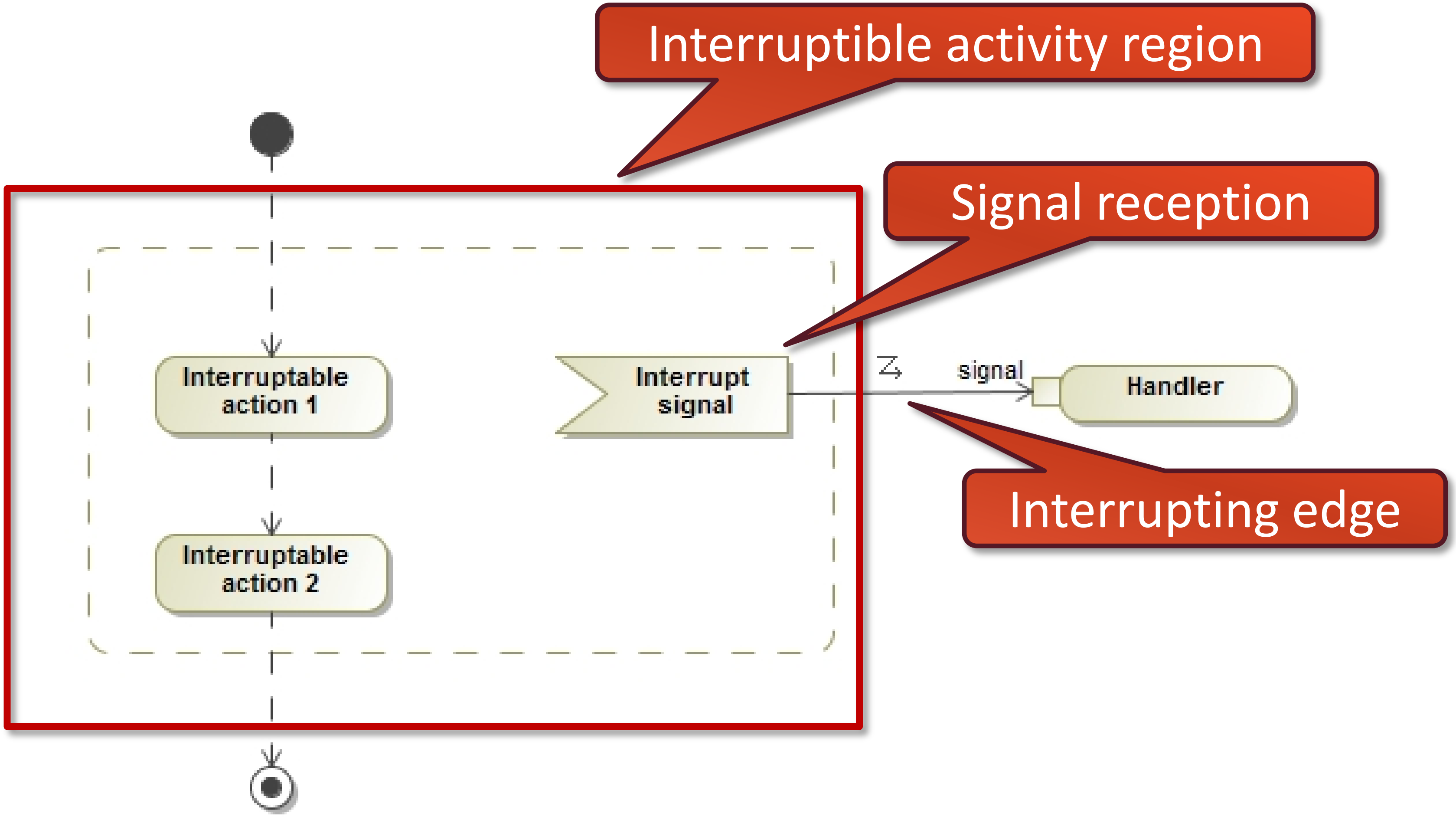
Activity parameter

Discrete flow

Flow rate

Stream

Interruptible Activity Region



Modeling Flow Based Behavior with Activities

Context of the Modeling Aspect

Modeling Elements & Notation

Semantics of the Model

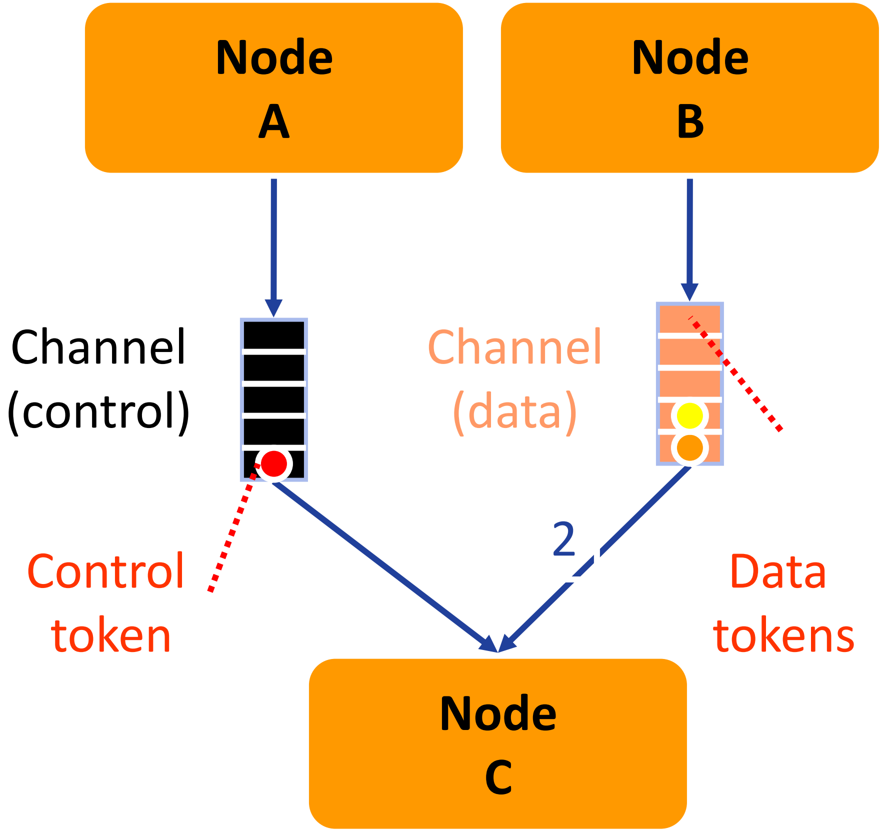
Summary

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

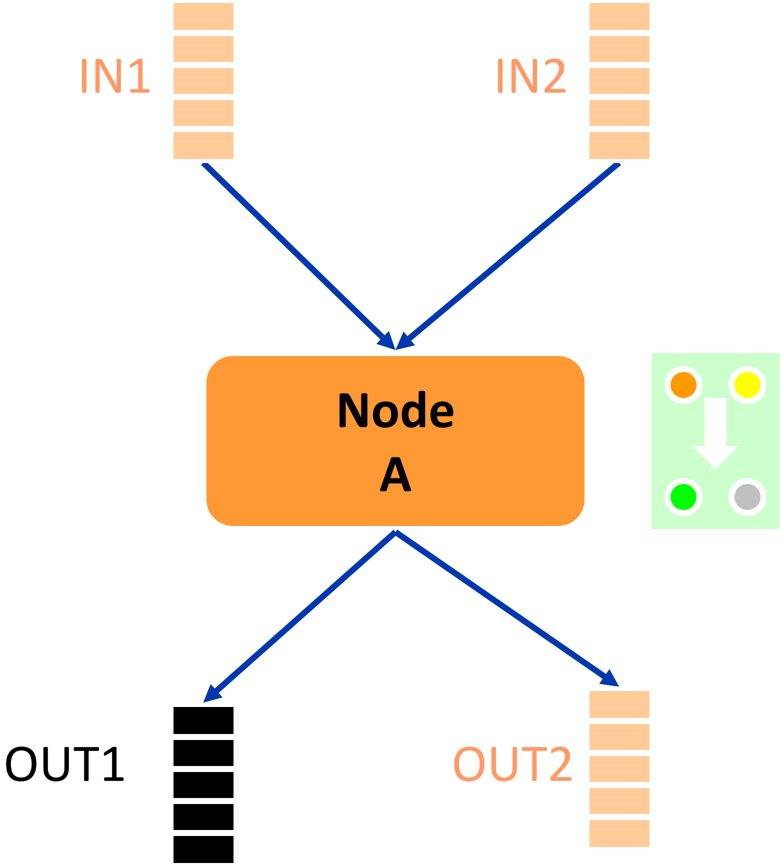
Semantics: Dataflow Networks

- 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



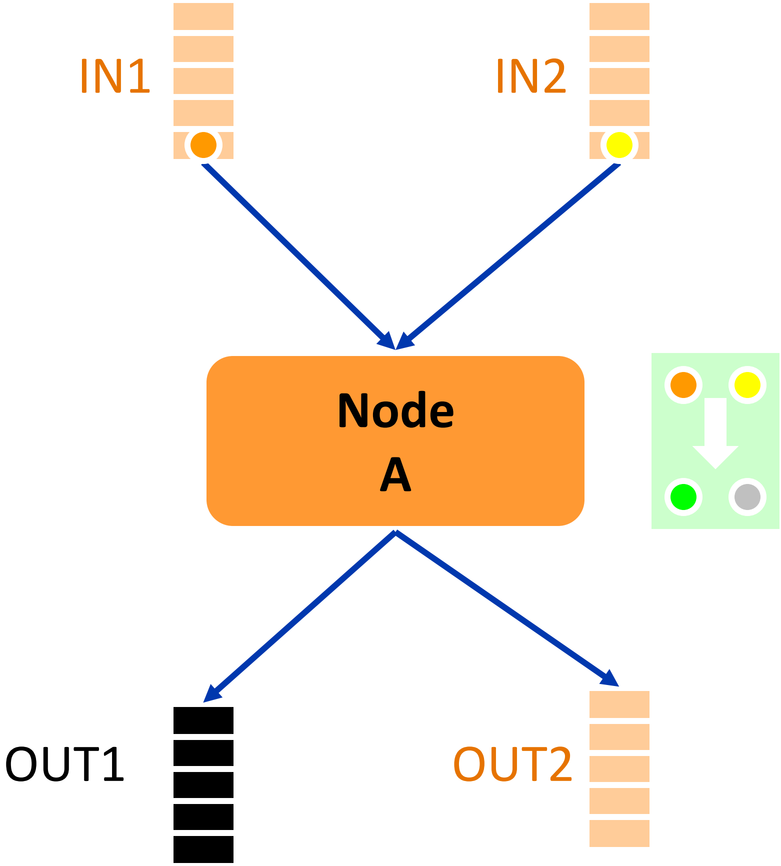
Semantics: Dataflow Networks

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



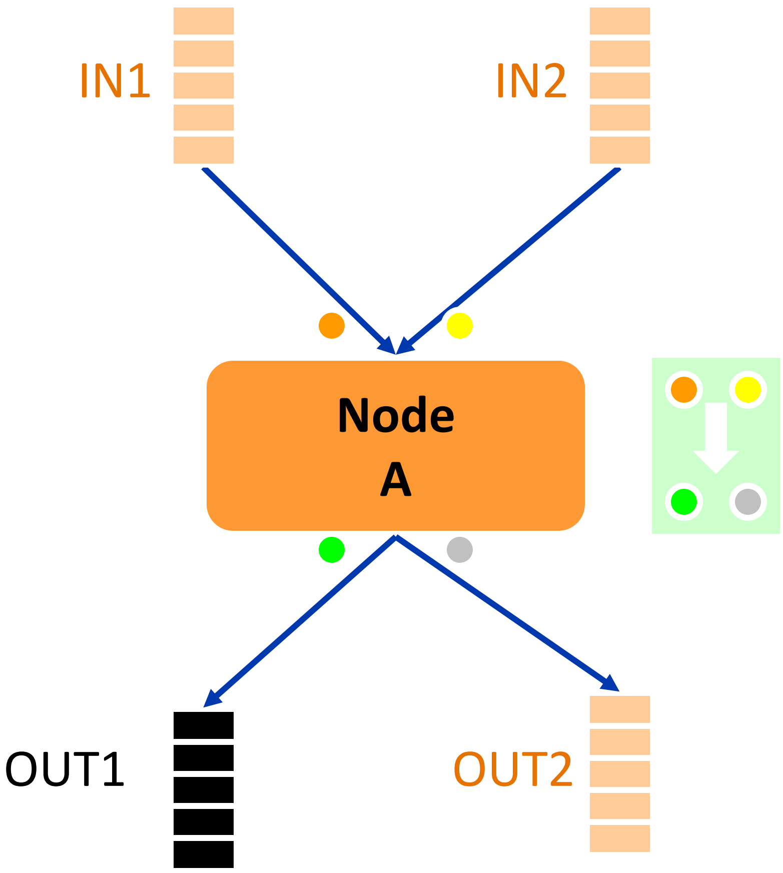
Semantics: Dataflow Networks

- 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?



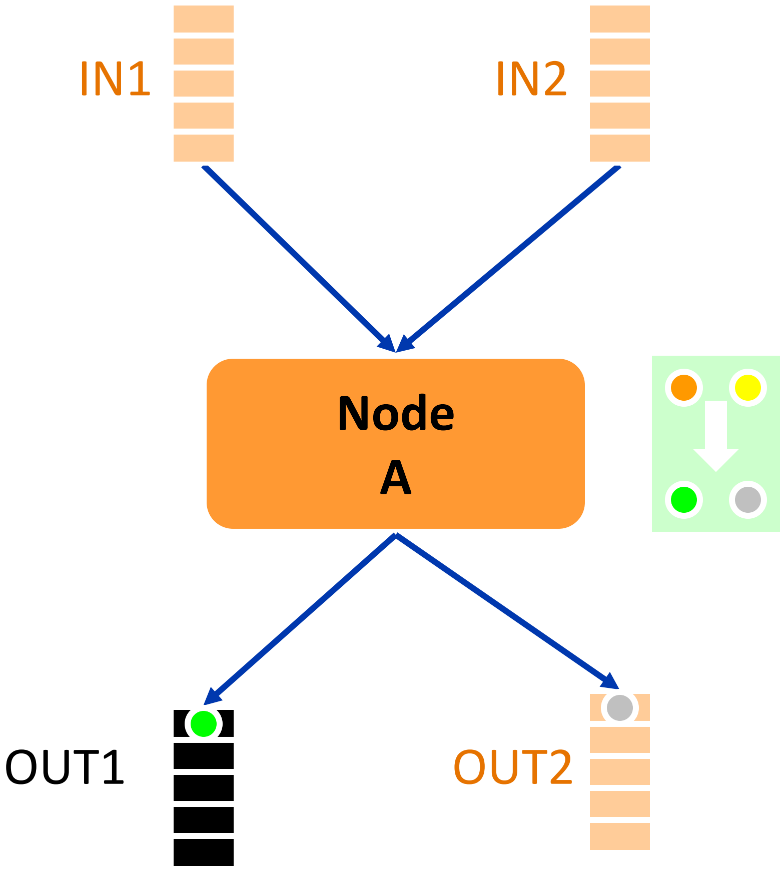
Semantics: Dataflow Networks

- 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?
  - Execution of action

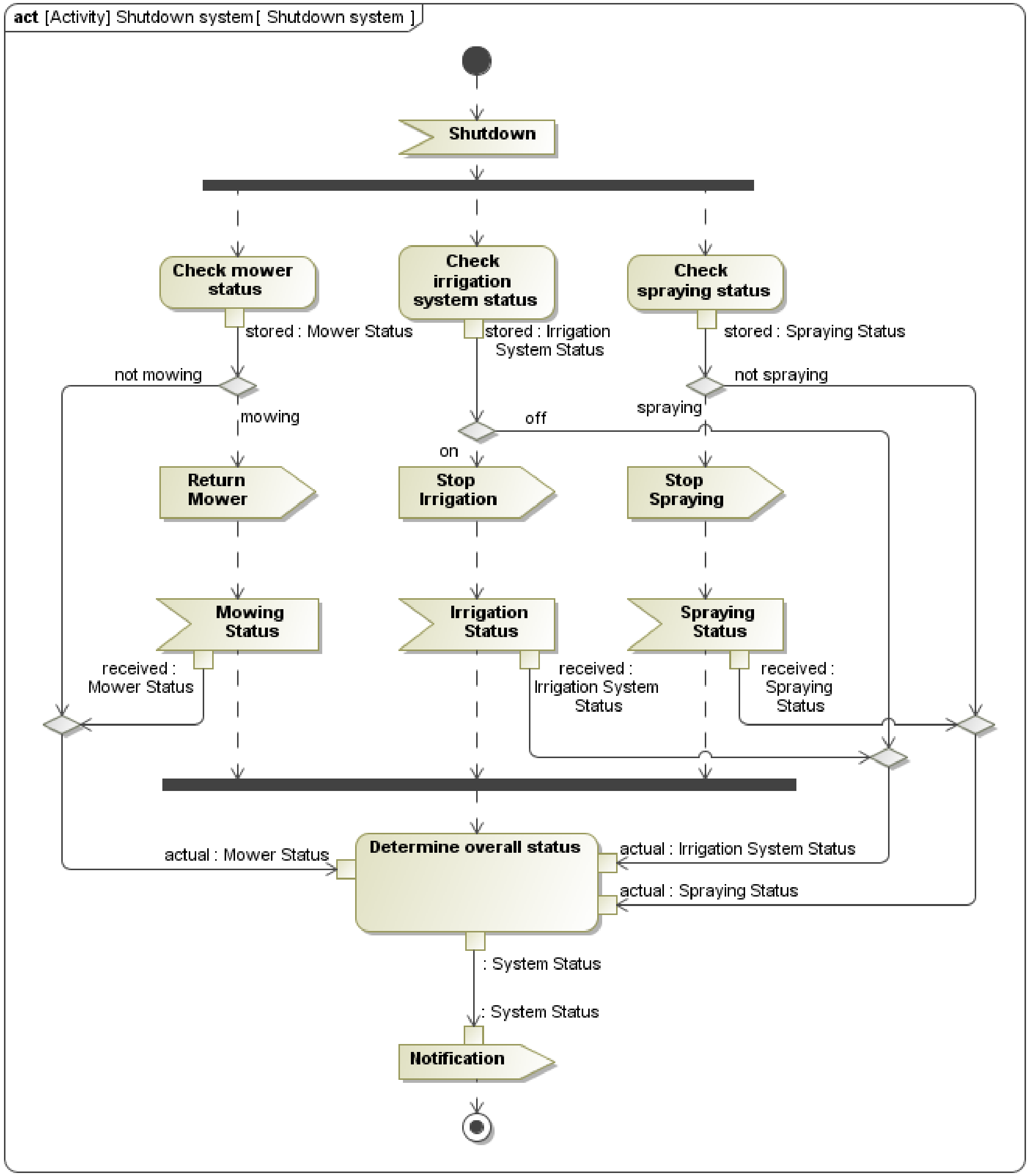


Semantics: Dataflow Networks

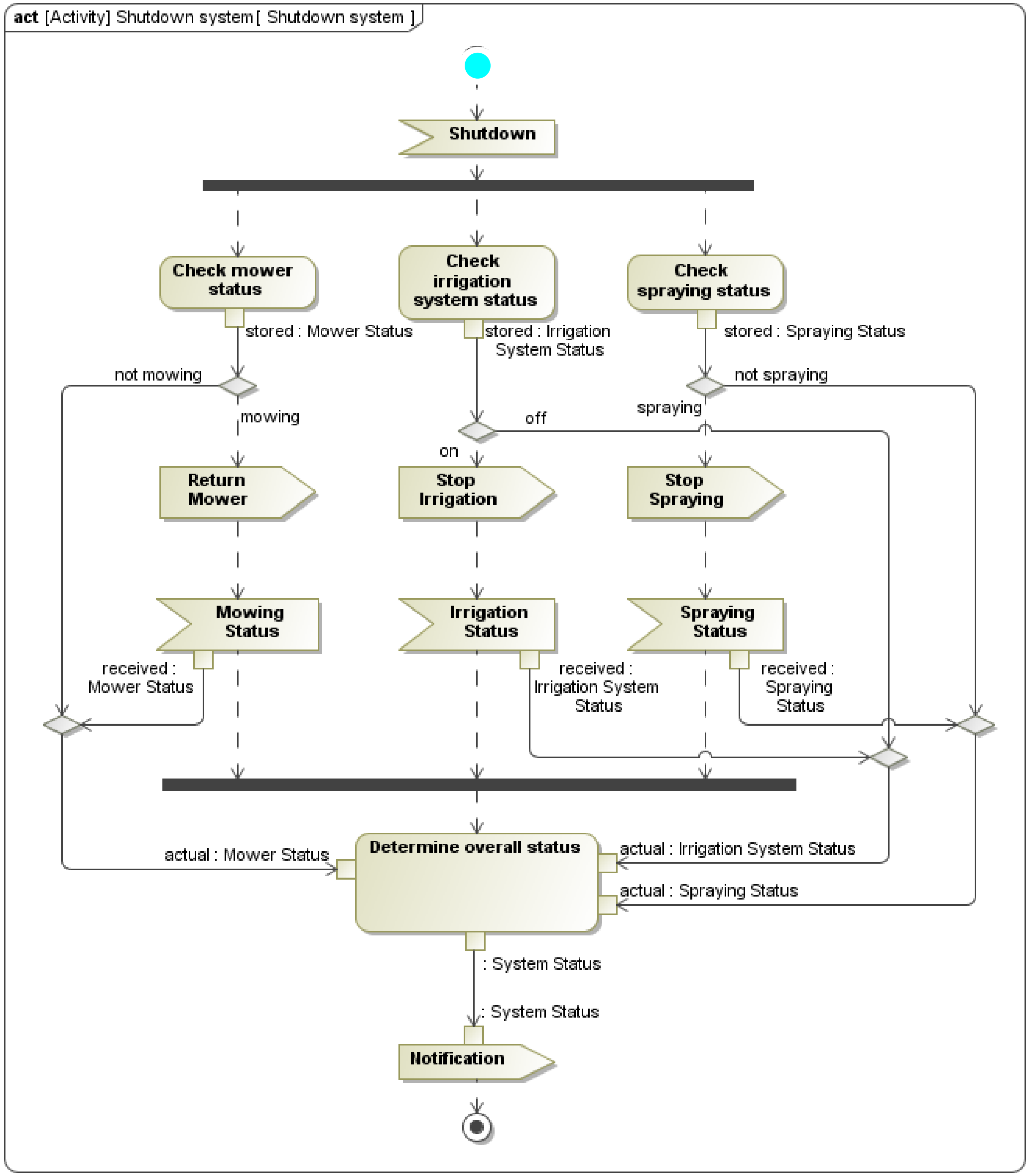
- Firing rule (cont.):
  - precondition:
    - input tokens + current state
  - postcondition
    - output tokens + new state
- Execution of a firing:
  - Is there token on all inputs with
    - Right amount?
    - Right type?
  - Execution of action
  - Sending the output tokens



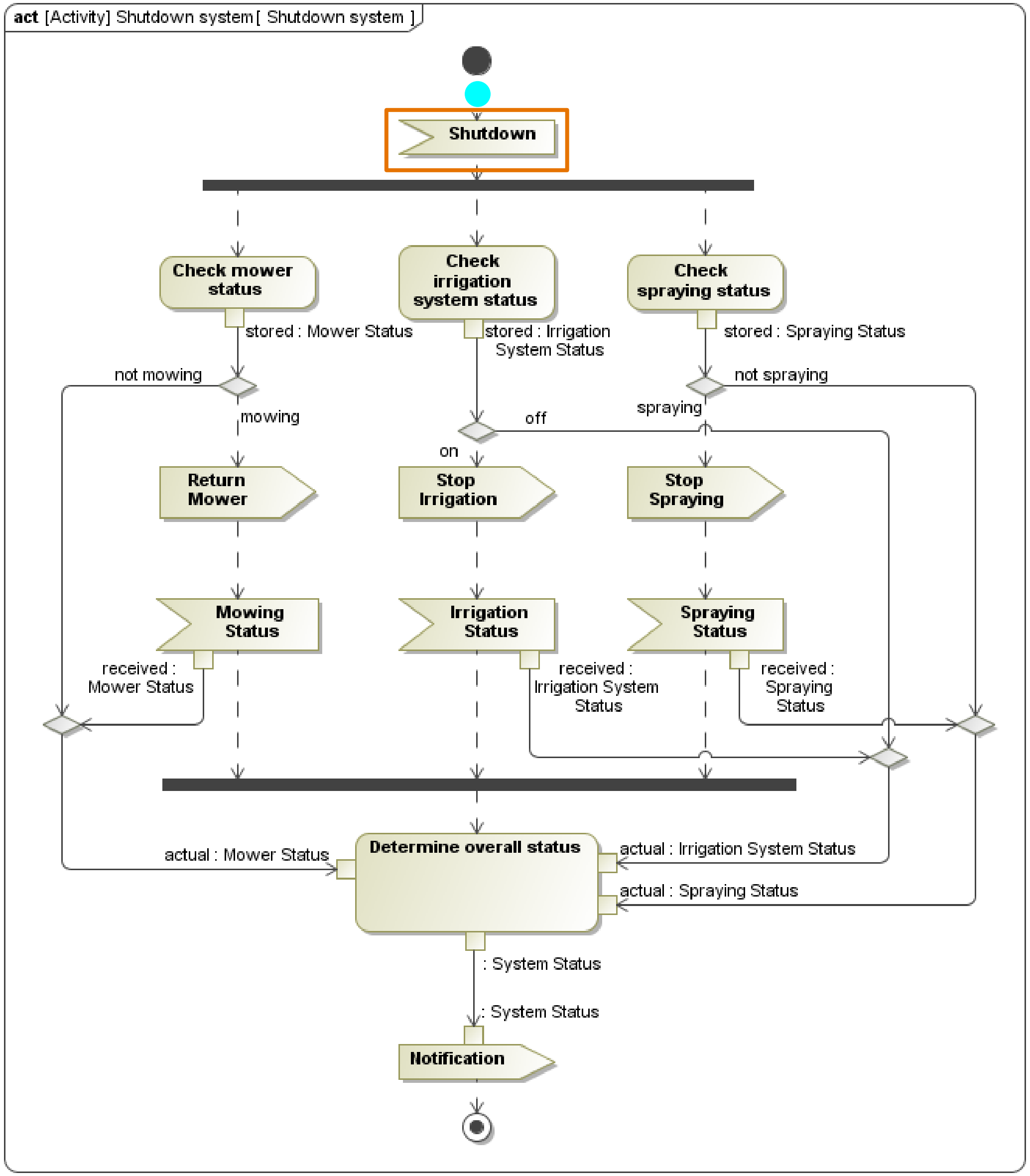
Example: Shutdown system



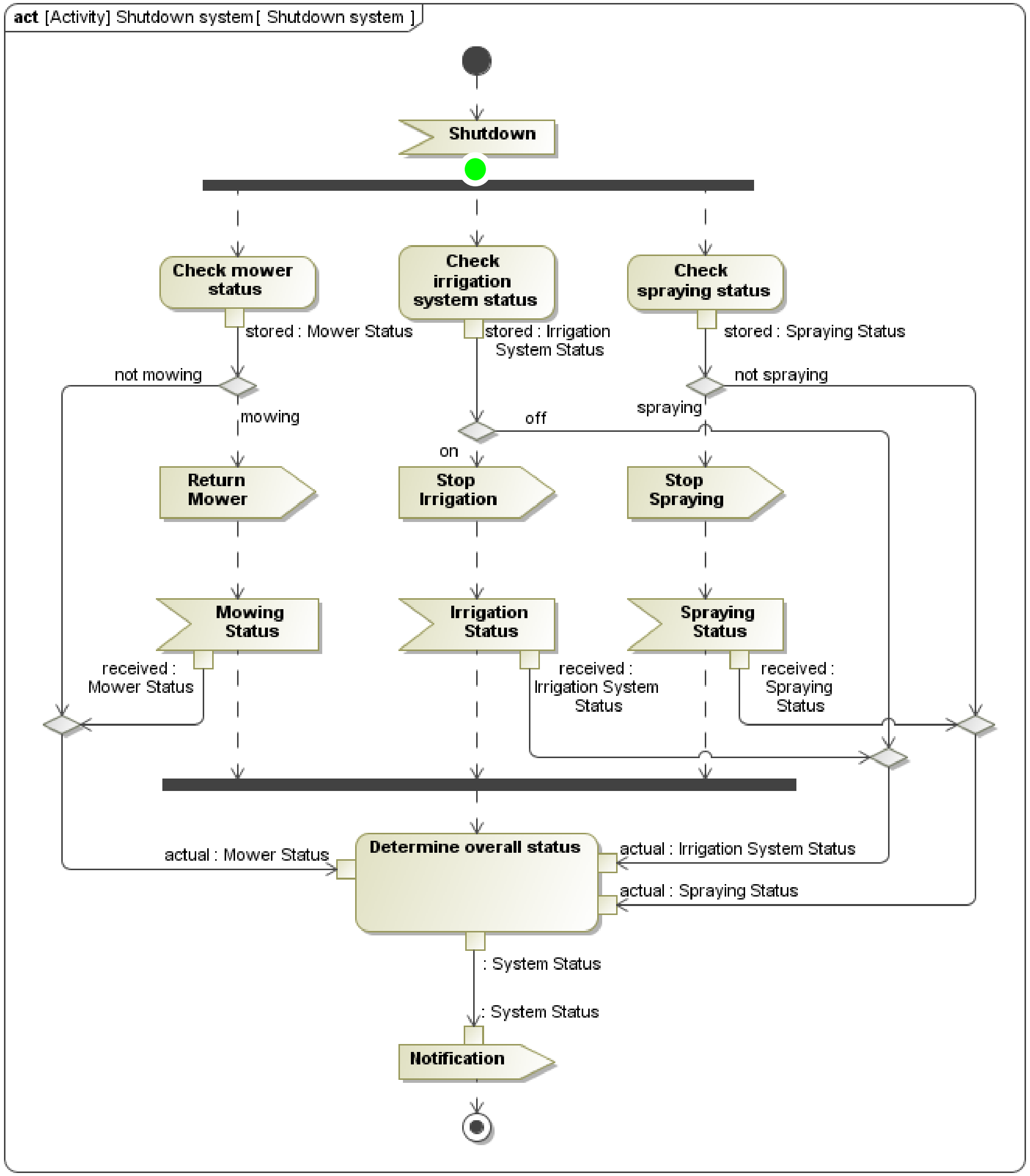
Example: Shutdown system



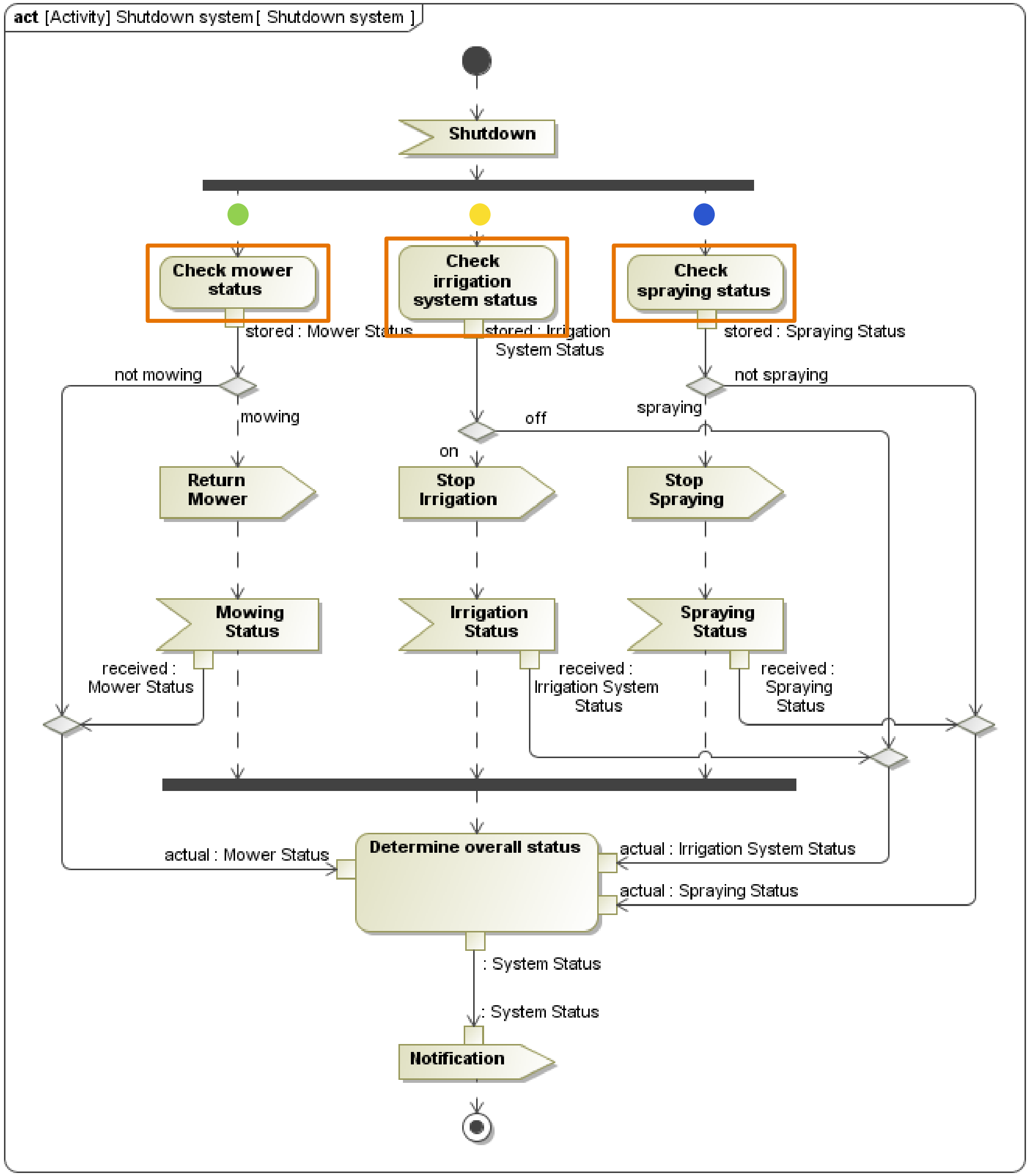
Example: Shutdown system



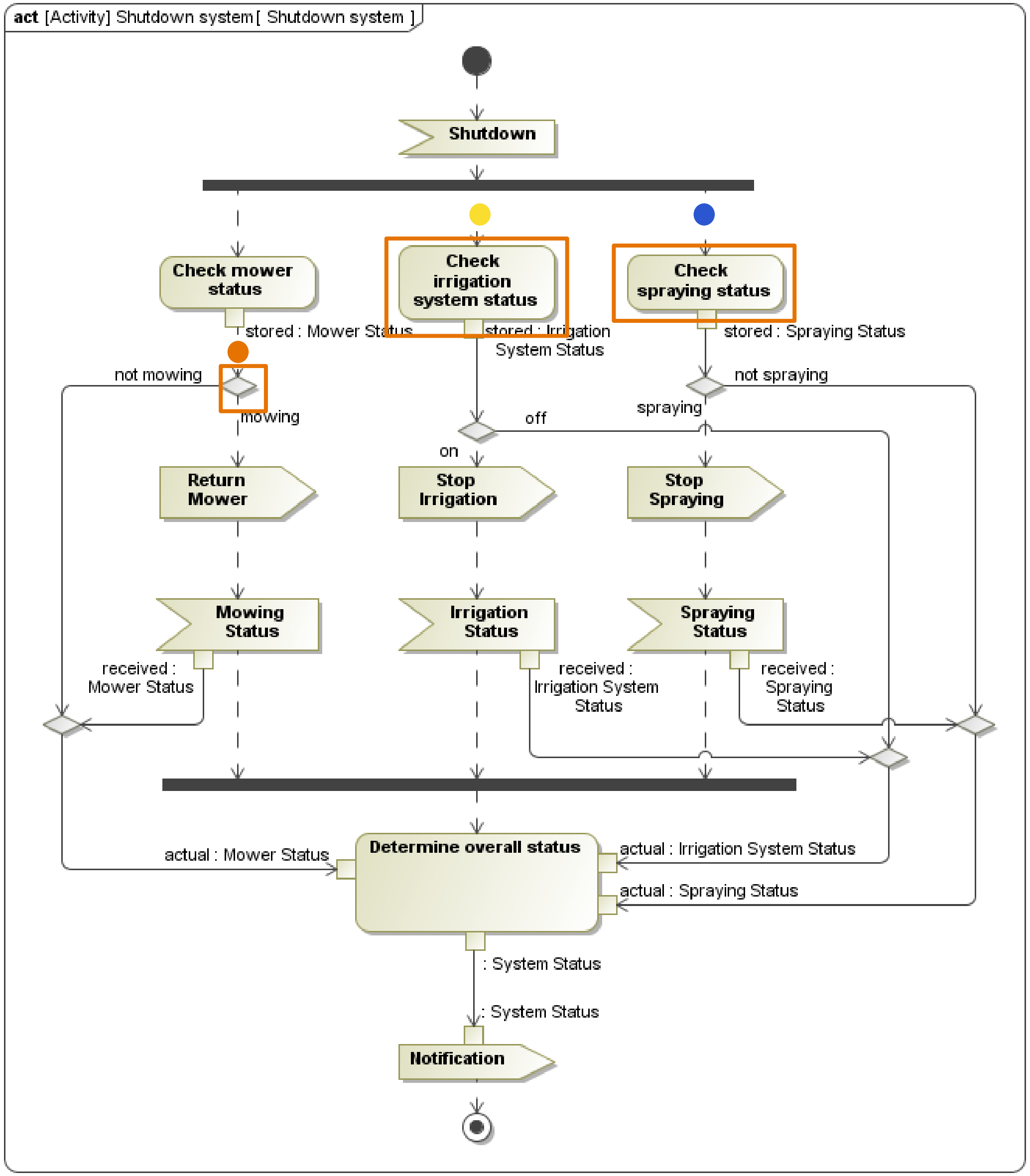
Example: Shutdown system



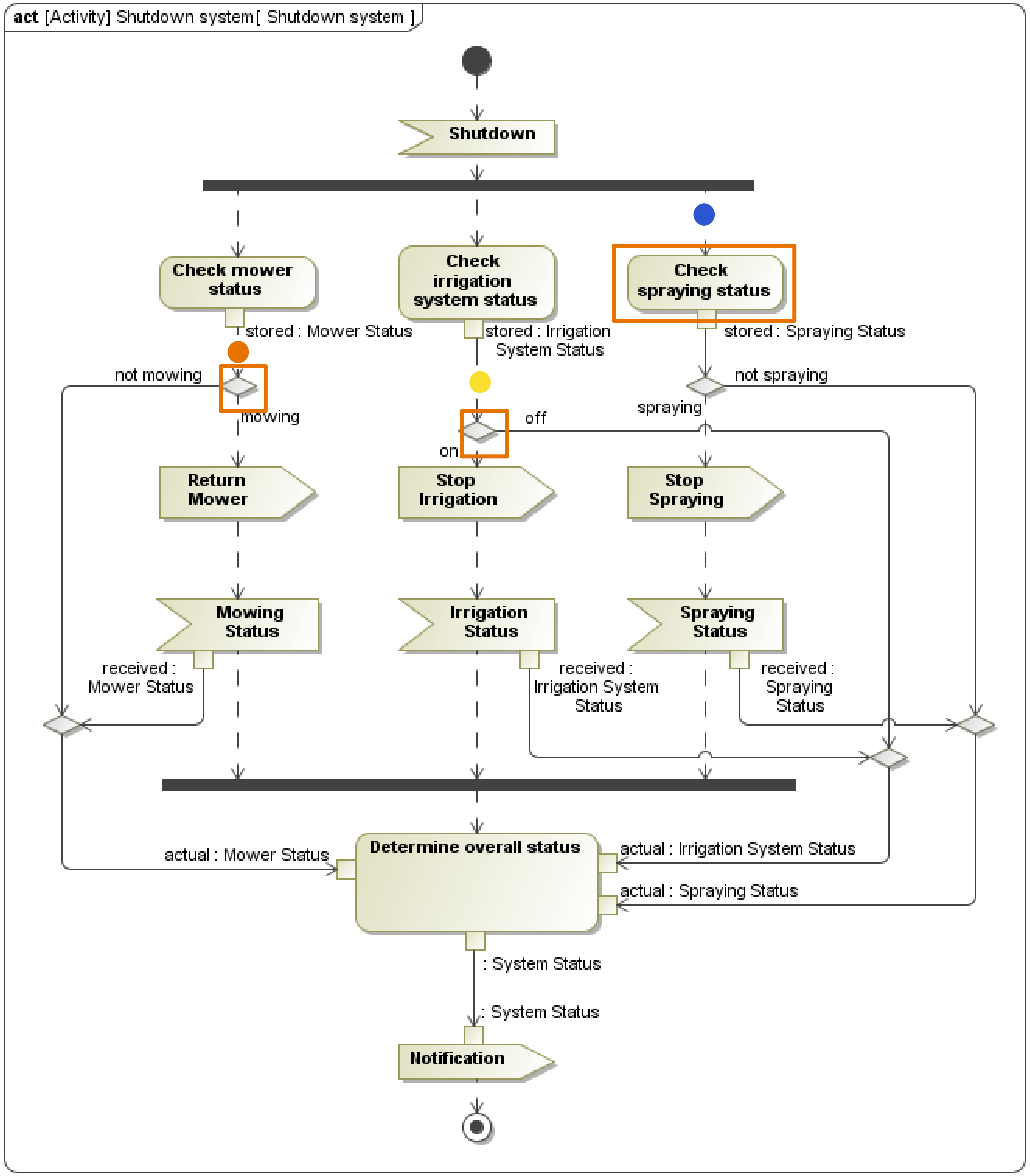
Example: Shutdown system



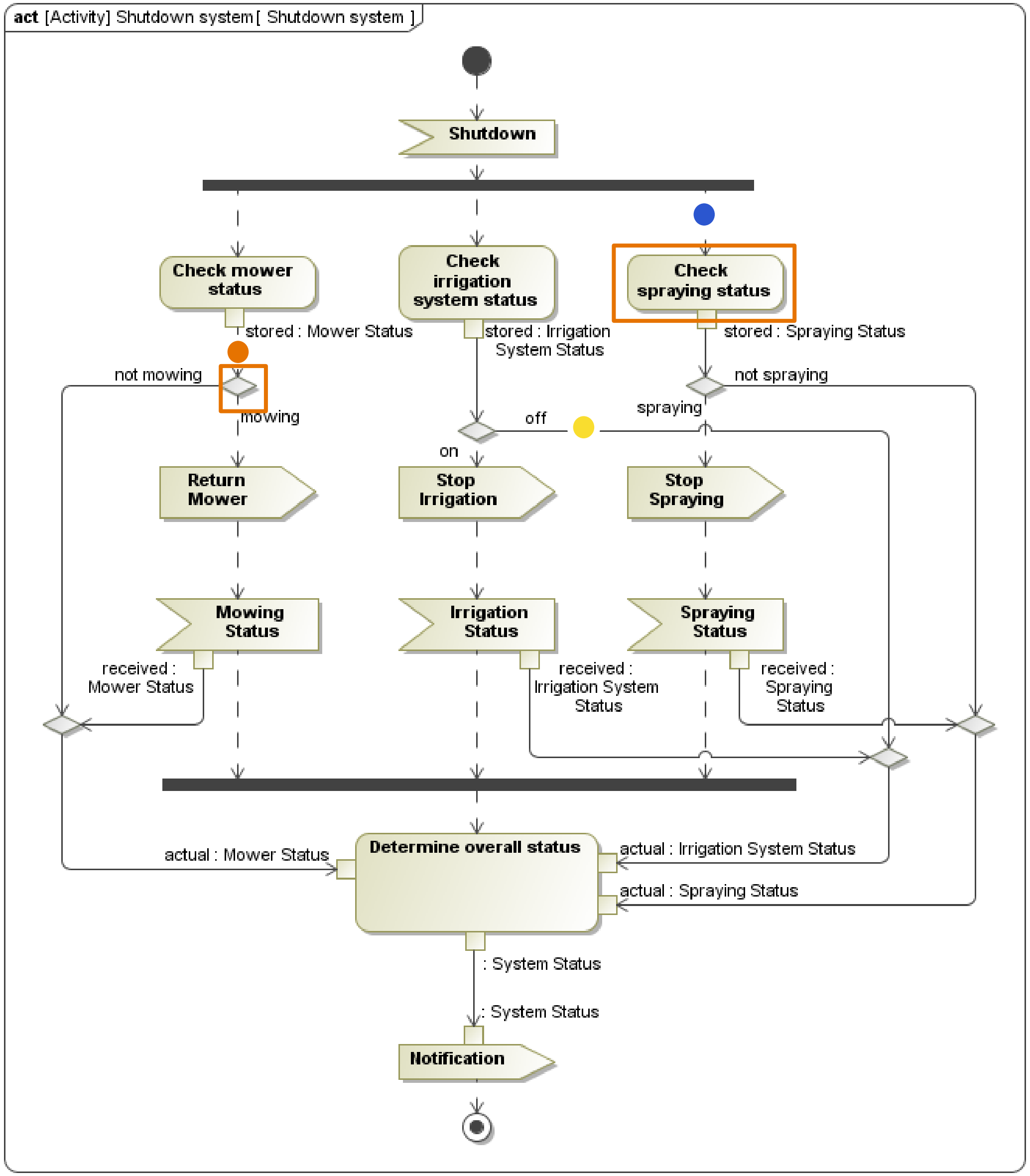
Example: Shutdown system



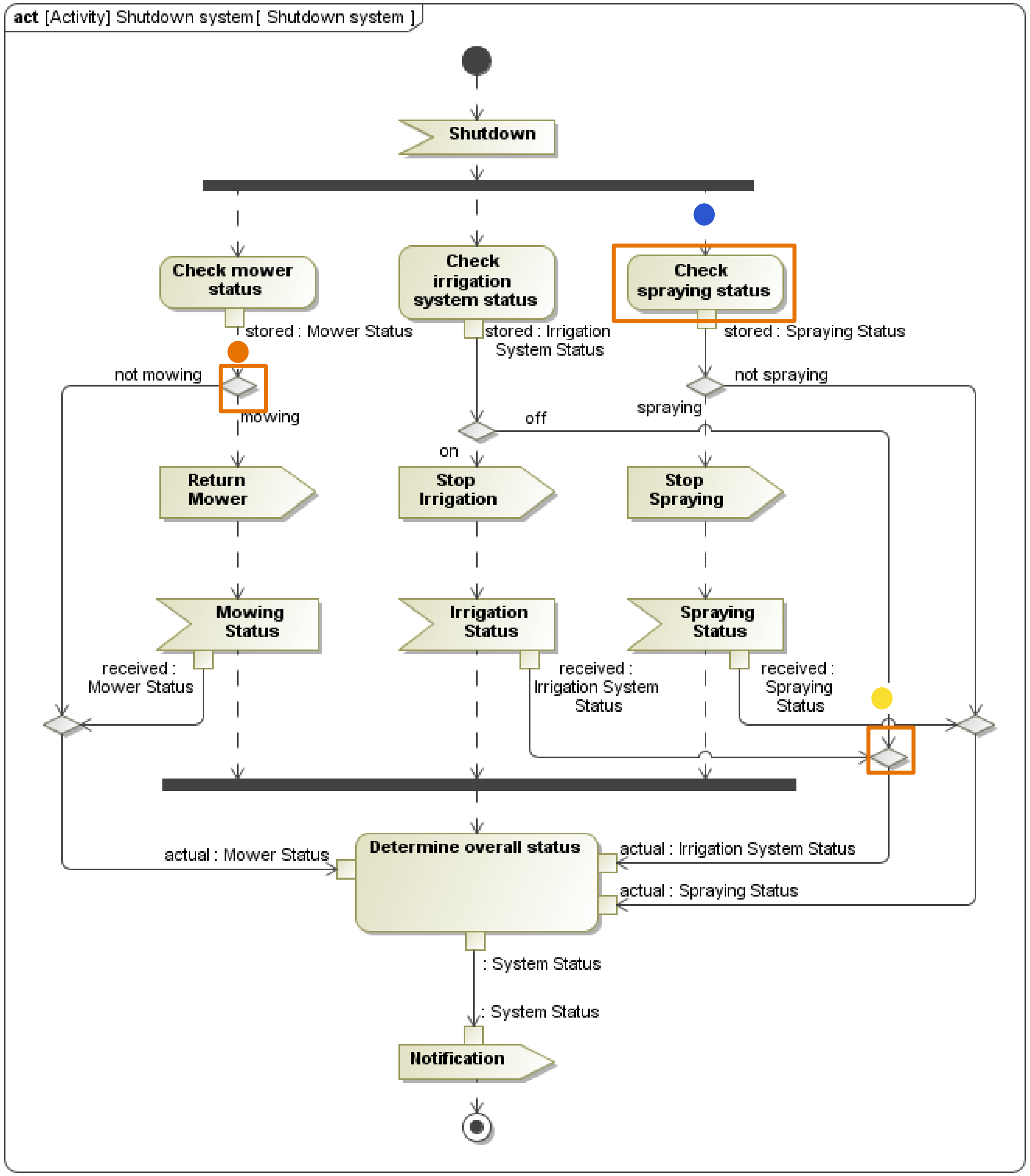
Example: Shutdown system



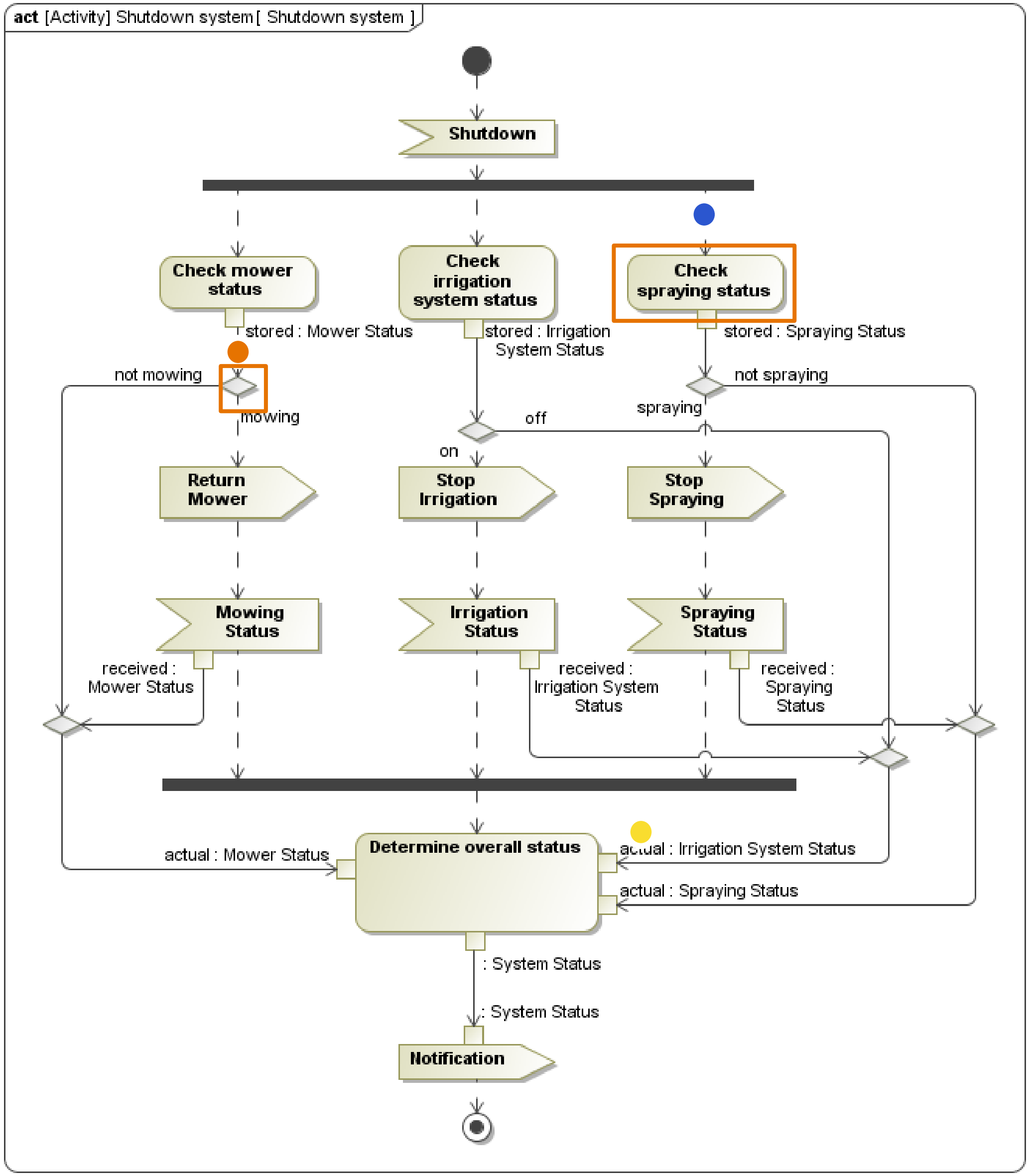
Example: Shutdown system



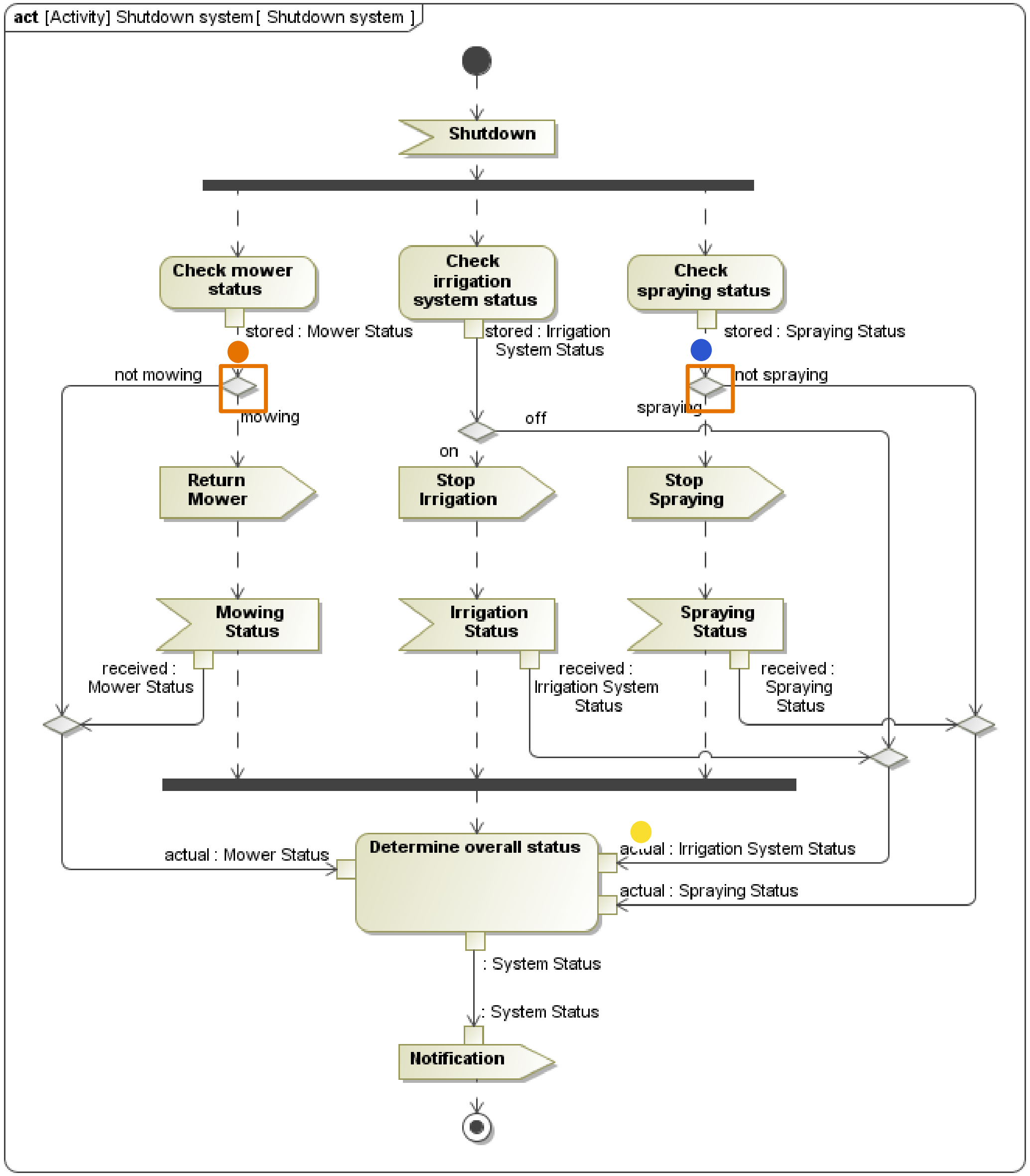
Example: Shutdown system



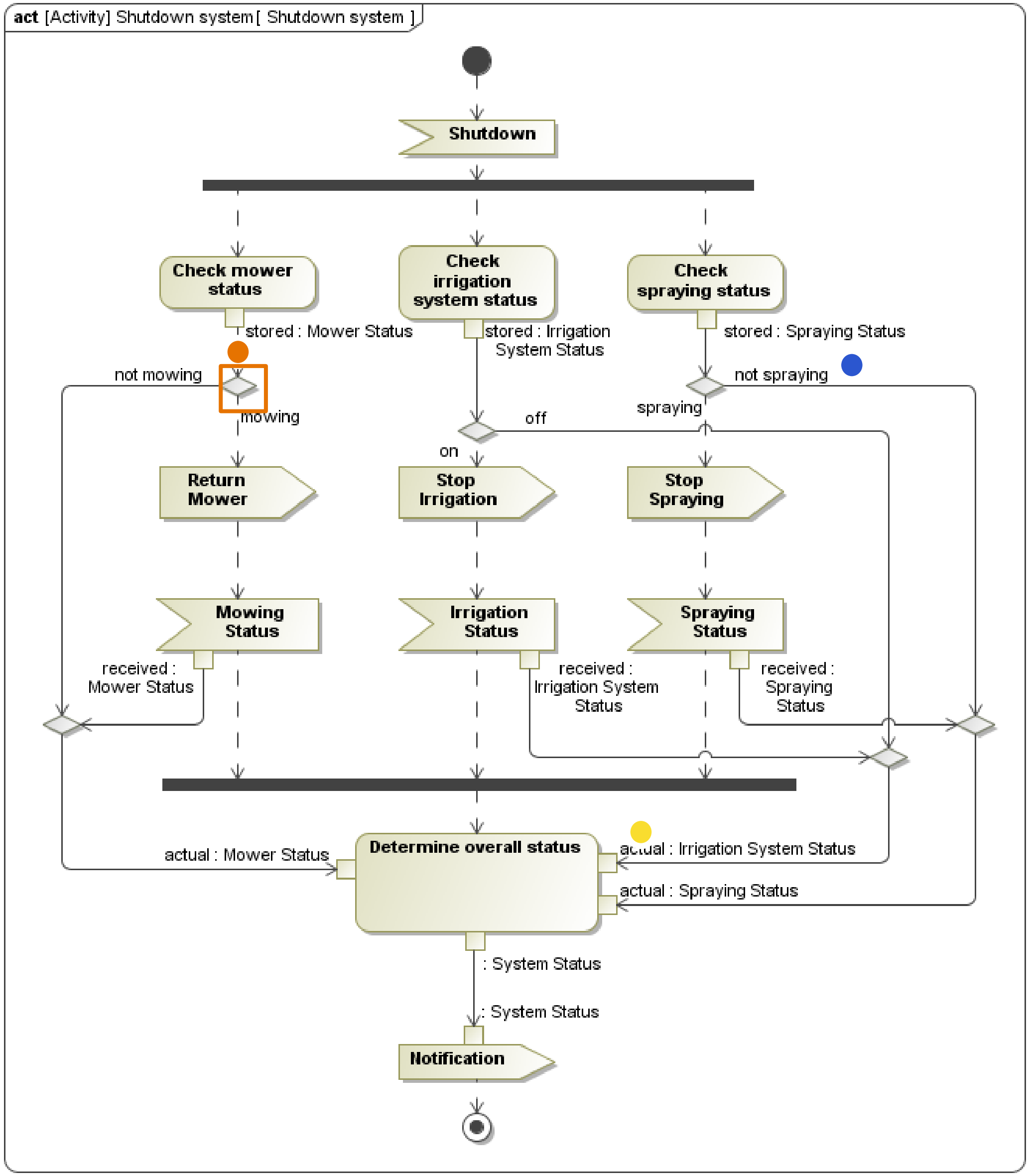
Example: Shutdown system



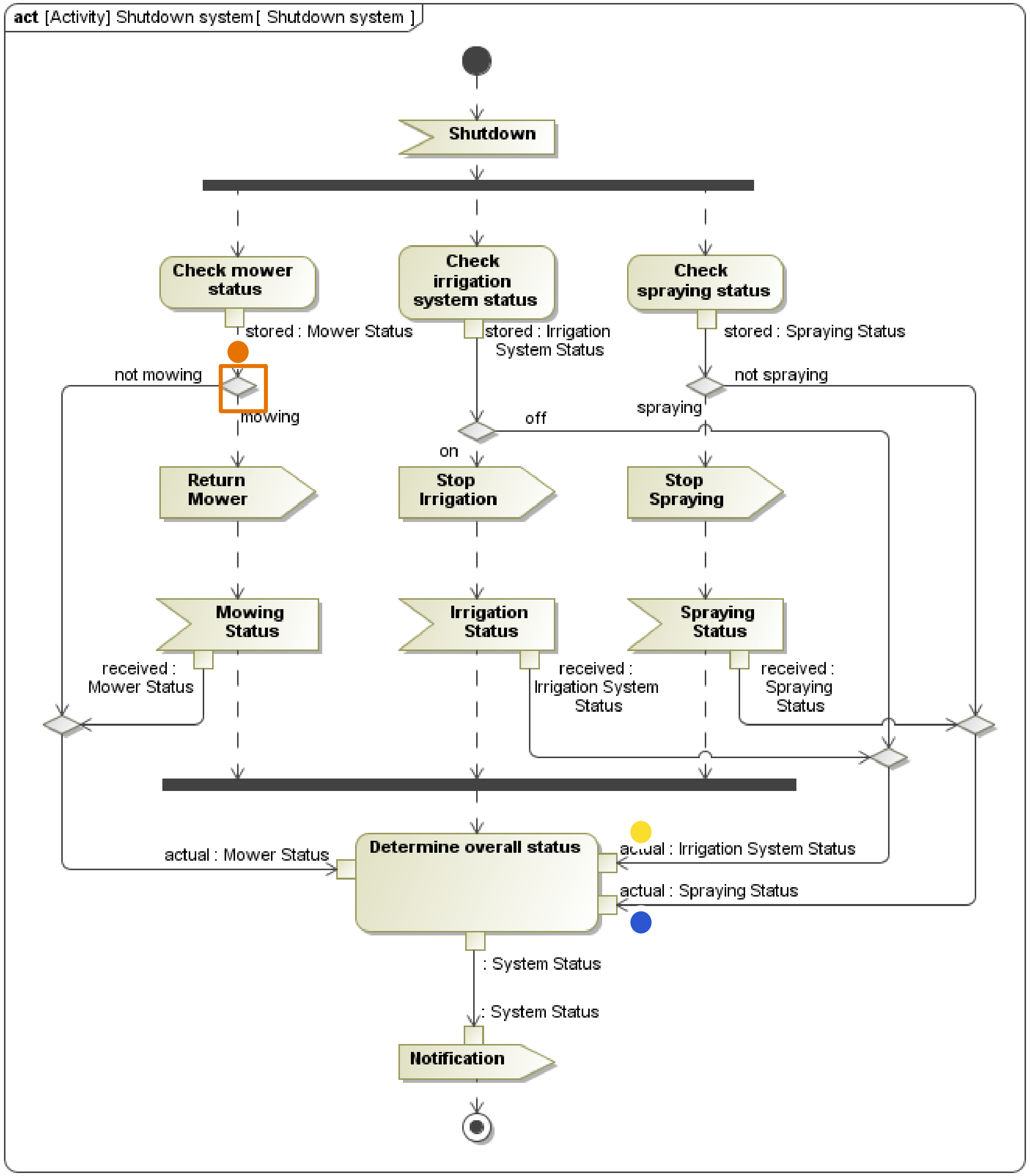
Example: Shutdown system



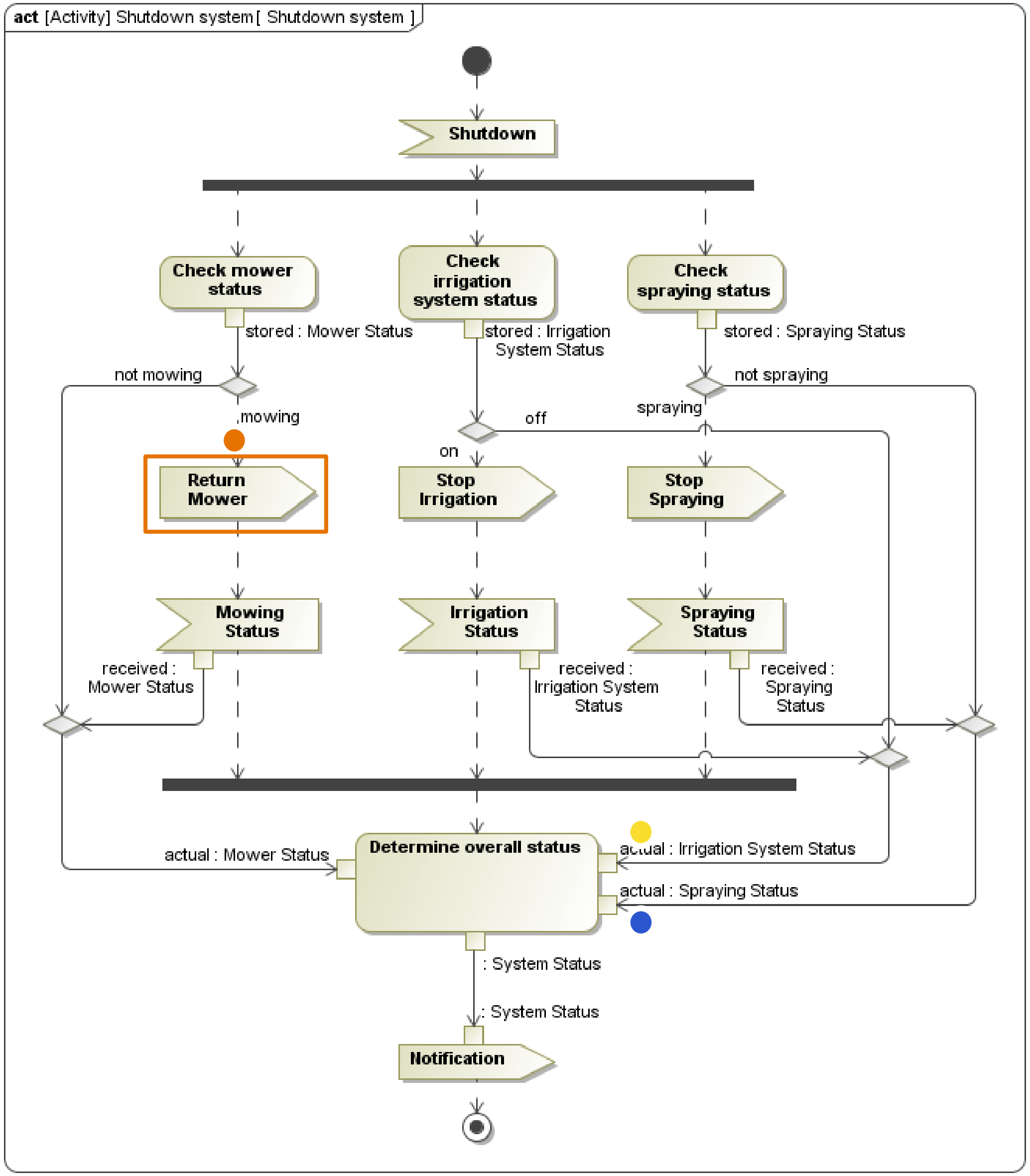
Example: Shutdown system



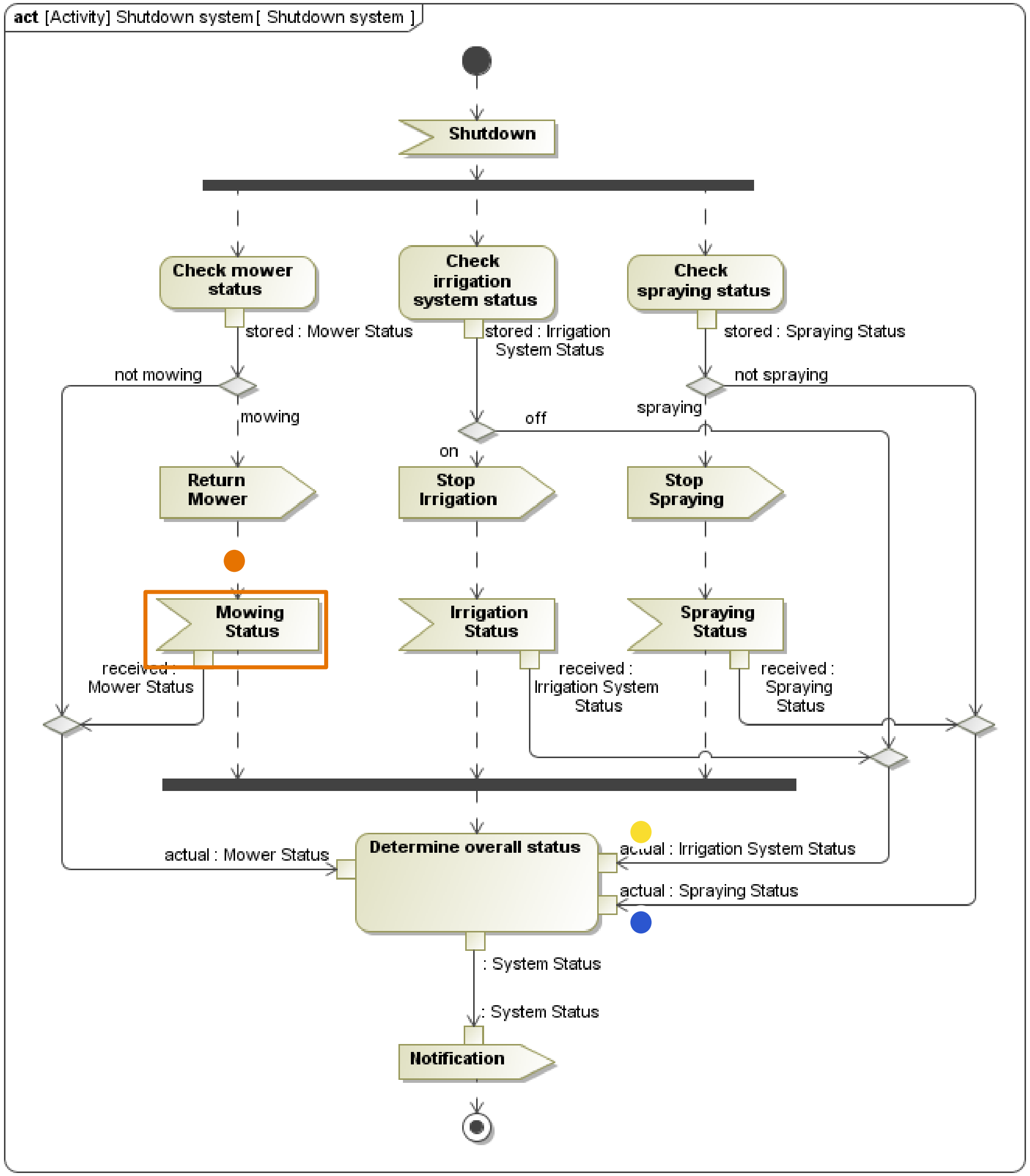
Example: Shutdown system



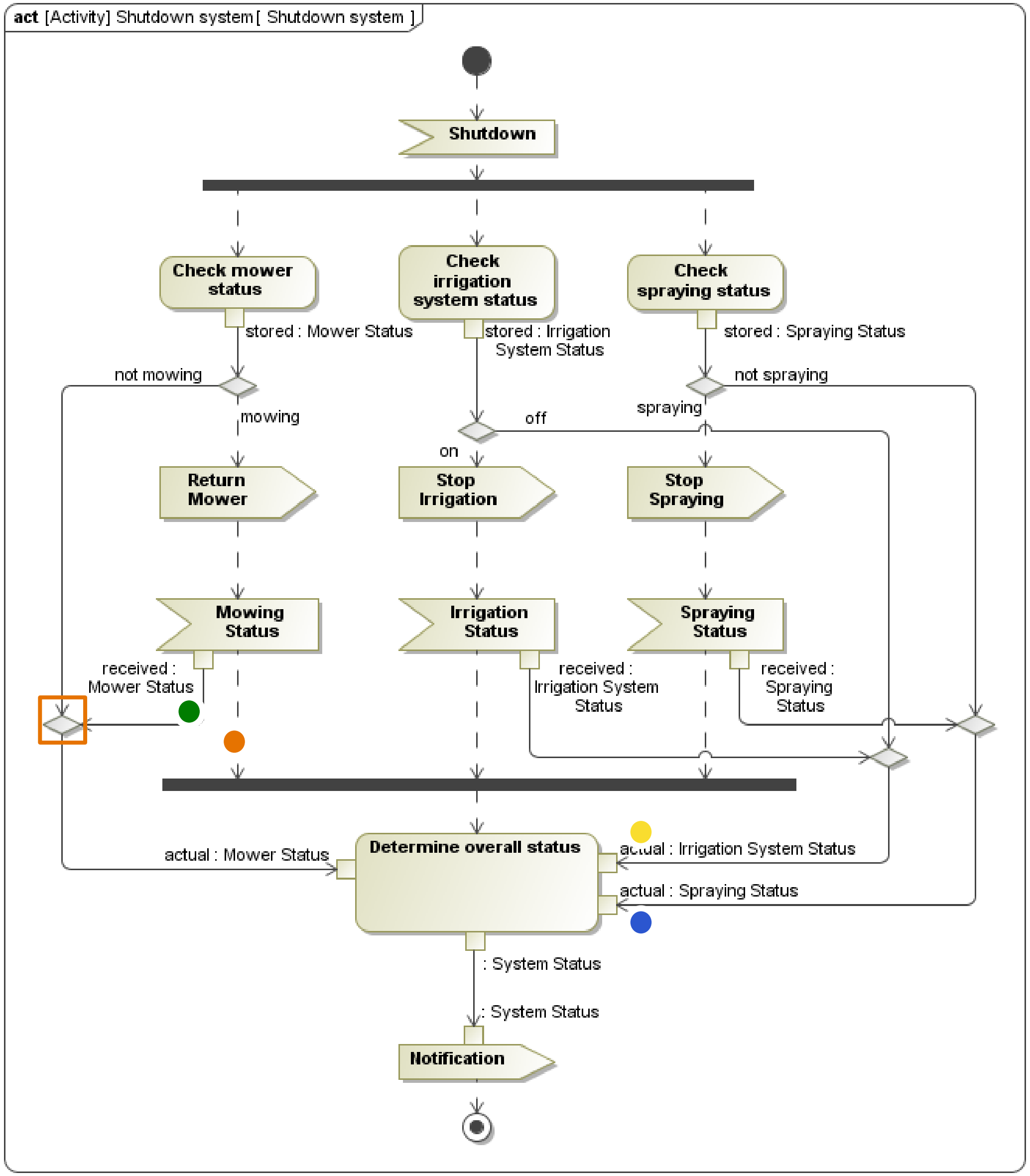
Example: Shutdown system



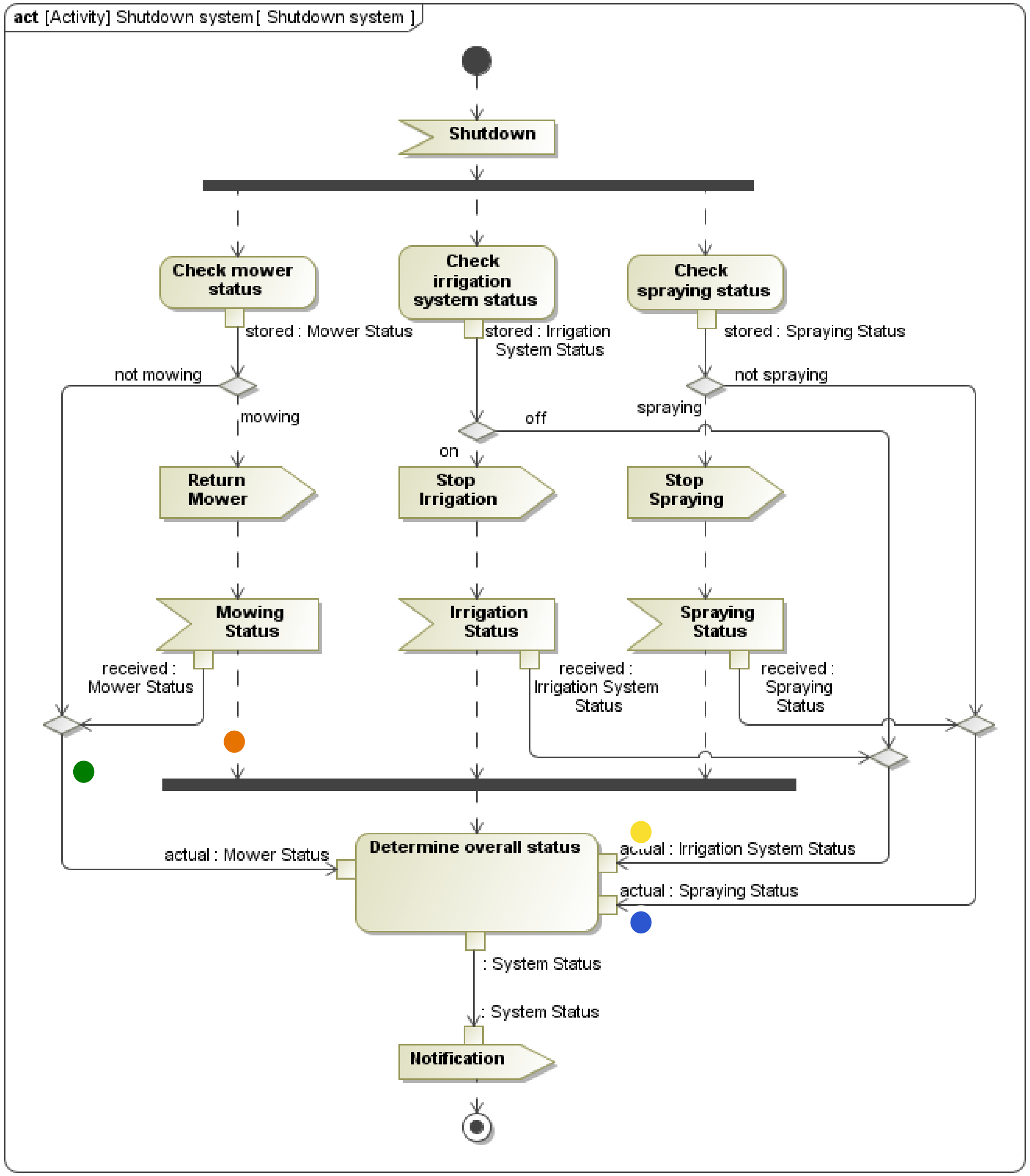
Example: Shutdown system



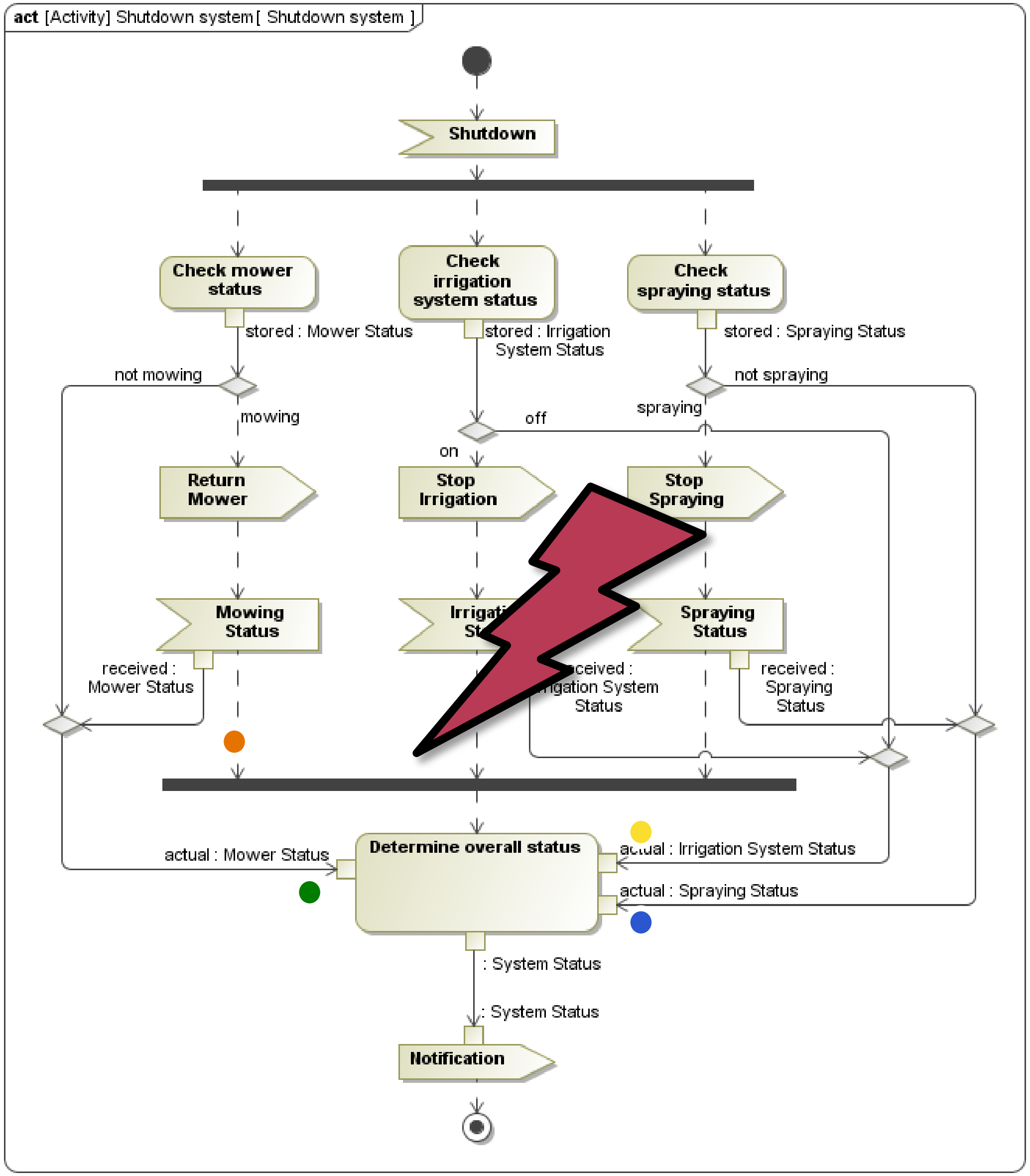
Example: Shutdown system



Example: Shutdown system



Example: Shutdown system



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