CESAR - Cost-**e**fficient methods and processes for **sa**fety **r**elevant embedded systems

CESAR – Papyrus MDT Training
Feb 29th, 2012

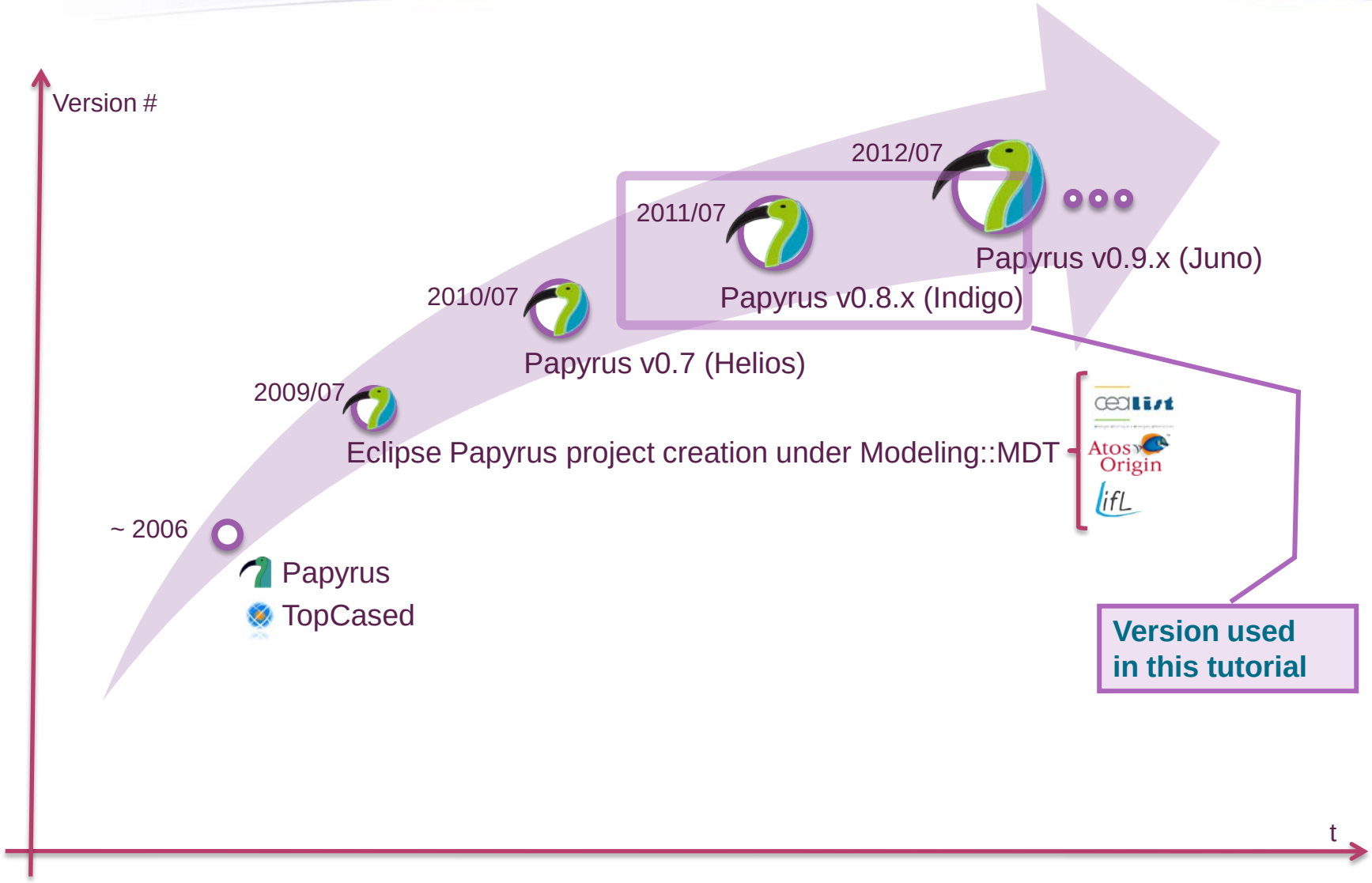
Getting started with Papyrus MDT

Hubert Dubois, CEA LIST/LISE, Hubert.Dubois@cea.fr

Agenda

1. MDT Papyrus versus Papyrus 1.x
2. MDT Papyrus for System Design in SysML
3. Other MDT Papyrus facilities
4. Q&A

Papyrus History



Papyrus 1.x versus new Papyrus

- **Papyrus v1.x**
  - CEA initiative started in
    - In collaboration with Cédric Dumoulin from LIFL
  - Scope: UML modeler and DSML based on UML
    - Includes also both SysML and MARTE standards.
  - Technology
    - Handmade Java-programming based on both GEF and EMF frameworks
- **Papyrus new generation**
  - New Papyrus is an official Eclipse project for Modelling::MDT
    - www.eclipse.org/papyrus
  - Why version number = 0.x ?
    - Due to Eclipse rules: 0.7.x = usual first version # for incubation project
  - Scope: UML2, SysML and DSML
    - Details of initial proposal here: http://wiki.eclipse.org/MDT/Papyrus-Proposal
  - Technology
    - Model transformation / code generation and handmade Java programming based on GMF framework

Papyrus Committers & Supporters

- **CEA LIST**
  - Arnaud Cuccuru, Sébastien Gérard, Camille Letavernier, Vincent Lorenzo, Ansgar Radermacher, Rémi Schneckenburger, David Servat, Yann Tanguy and Patrick Tessier.
- **ATOS**
  - Raphaël Faudou, Tristan Faure, Vincent Hemery, Thibault Landre, Emilien Perico and Mathieu Velten.
- **LIFL**
  - Cédric Dumoulin.

- **Main Current Industrial Supporters (in alphabatical order):**
  - AIRBUS, ATOS, CEA, Ericsson and Esterel Technologies (http://www.listerel.org/)

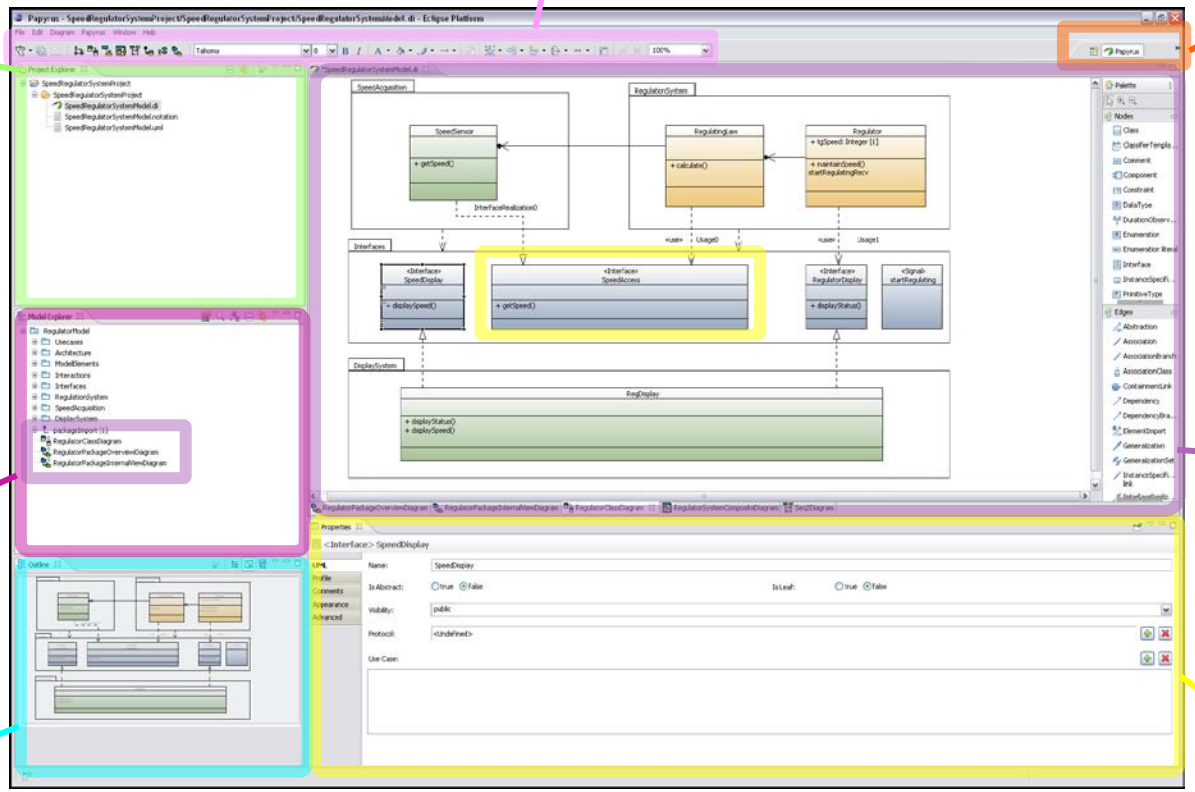
Papyrus facilities in a nutshell

- **Support for Major OMG Standard Modeling Languages**
  - UML2, SysML and MARTE.
- **Support for DSMLs**
  - Based on the UML2 extended by specific profiles
  - Supporting any specific domain notations, either graphical or textual
  - Providing powerful and easy-to-use tool customization facilities
- **Enabler for a full model-based engineering**
  - Model compare and merge
  - Team working
  - Documentation description and generation
  - Model validation

Outlines of the Papyrus perspective

Project explorer: used to manage Papyrus projects at file system level.

Main toolbar: diagram creation, graphical editing (align, distribute…), show /hyde, …



Perspective: switch the modeling context, define windows (eclipse views) arrangement, define the list of available diagrams, define the available menus and toolbars.

Model editors: model editor enabling to edit models through a given modeling language.

Property view: form-based model editor enabling to view & edit model element properties.

Outline view: provide overview of the model (read only).

Model explorer: tree-based model editor covering the whole model.

Authoring models in graphics, but also in texts and tables!

- **Graphical-based editors**
  - UML2
    - Support for Class, Composite Structure, Deployment, Component, Use Case, Sequence, Statemachine, Activity and Profile diagrams.
  - SysML
    - Support for Requirements, BDD, IBD and Parametric diagrams.
- **Textual-based editors**
  - Extension point to embed textual editors within Papyrus to edit partially a diagram.
    - E.g.: Attributes or Operation of Classes, Port of Composite Structure, State of Statemachine.
  - Framework for using the Xtext technology.
  - Miscellaneous:
    - Textual editor for ALF (“Action Language for fUML”) .
    - Textual editor for VSL in the MARTE plug-in of Papyrus.
- **Table-based Editors**
  - Based on NatTable and customizable via EMF Facet.
  - Available generic table editor allowing to show any kinds of element including its stereotypes and related properties.
  - Two customizations for SysML: Requirement and Allocation table editors.

… but modeling is also:

- **Validation**
  - Based on EMF Validation project of Eclipse
  - Integrated within Papyrus GUI
- **Compare and merge**
  - Based on EMF Compare project of Eclipse
    - Specialized for UML models (e.g., specific case of the Stereotypes applying)
  - Integrated within Papyrus GUI
- **Team working**
  - Control mode: enables to split one model into several files to be able to use versioning systems on sub-parts of one model.
- **Documentation generation**
  - Integration of the Gendoc2 component of TopCASED
  - Enable generation of ODT and DOCX documents
  - Available via the market place of Eclipse: http://marketplace.eclipse.org/
- **Code generation and model transformation**
  - All code generators and model transformation engines can be used and connected to Papyrus

Agenda

1. MDT Papyrus versus Papyrus 1.x
2. MDT Papyrus for System Design in SysML
3. Other MDT Papyrus facilities
4. Q&A

About this tutorial…

- **A design inspired from the Accord|$_{UML}$ methodology**
  - Accord|$_{UML}$ [Gérard-Terrier, 2004]: a UML-based approach for Embedded Real-Time Systems modeling
    - ➔ Needs to be adapted to SysML design
  - Activity 1: Defining a Preliminary Design
    - First analysis of a specification
    - Objectives:
      - To structure the modeling process
      - To help ensuring system requirements from early steps of modeling
      - To have a “black-box” view of the system
  - Activity 2: Defining a Detailed Design
    - Detailed analysis: structural and behavioral aspects
    - Objectives:
      - To see the system as a “white-bow”
      - To define behaviors in dedicated diagrams

About this tutorial…

- **What it will be:**
  - A first introduction in using Papyrus MDT tool
  - How starting to use Papyrus modeling environment
  - A short and quick usage for different structural & behavioral diagrams:
    - Requirement Diagram
    - Use-Case Diagram
    - Block Definition Diagram
    - Internal Block Definition Diagram
    - Sequence Diagram
- **What it will not be:**
  - A tutorial an SysML modeling
  - A large usage of each construction of the SysML language

Case study: Elevator Controller

- **Starting Point:**
  - A Requirements Specification document
- **Objectives:**
  - To define a structure for an elevator controller
  - To define few behavioral elements
- **Method:**
  - To follow the DIAPASON = Accord$|_{SysML}$ methodology
  - To use dedicated SysML diagrams for:
    - Preliminary Analysis
    - Detailed Analysis

Activity 1: Defining a Preliminary Analysis

- **Structure of this Preliminary Analysis**
  - SysML Requirements Diagram
    - Create the diagram
    - Create packages and structure for the requirements

Elevator Controller Requirements

2.2 Product Functions

The primary function of the EC is to control the up and down movement of the elevator cars (by way of controlling the elevator sheave motors). It does so to position the elevator cars at floors where passengers have selected (either for pick-up or drop-off). It also controls the opening and closing of elevator car doors and floor entrance doors to allow the *safe* entry and exit of passengers into and out of the elevator cars.

In addition, the EC controls illumination and delumination of floor indicators and buttons.

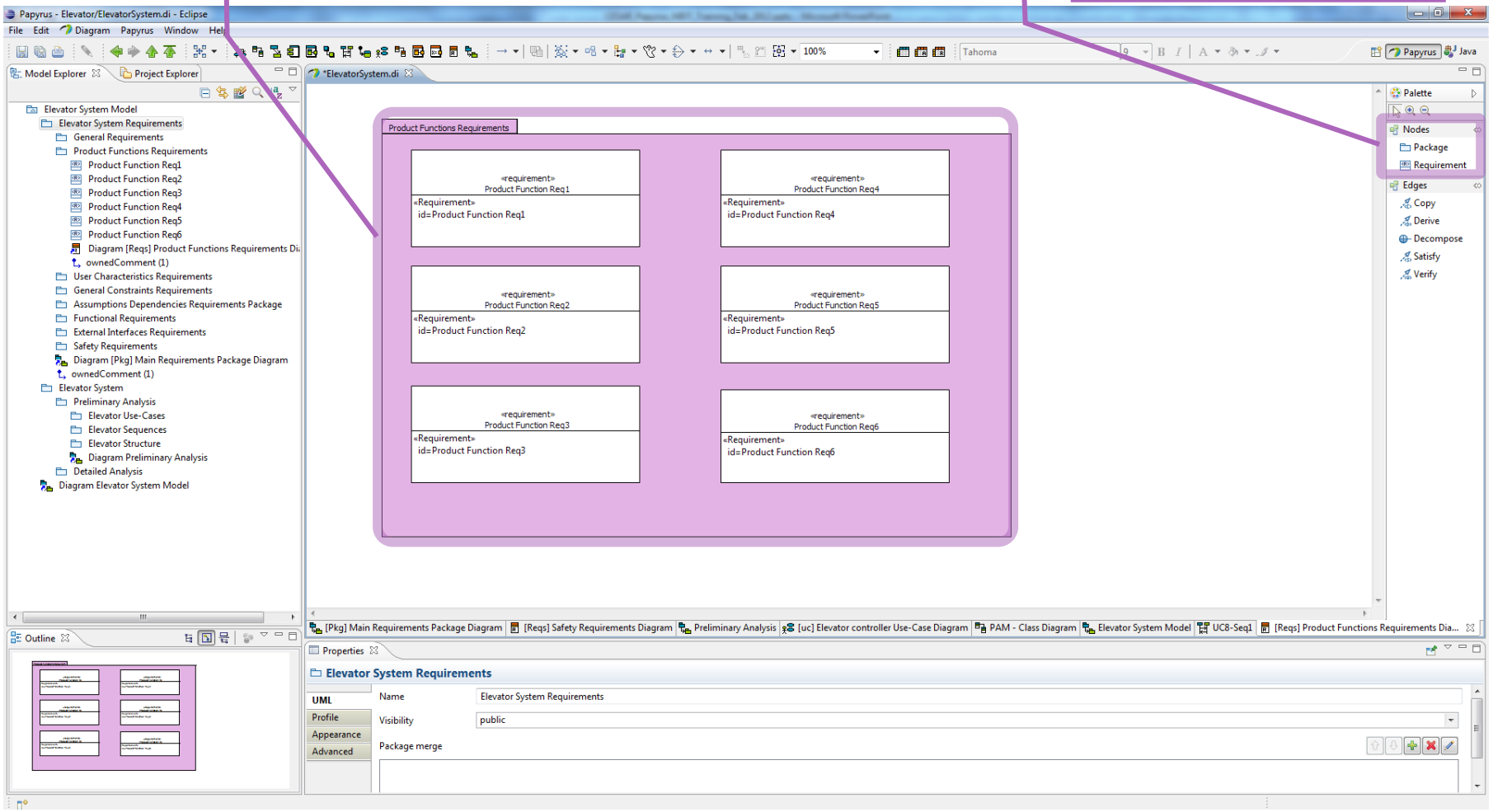
The EC supports **two operational modes** and **one recall mode**: AUTO, HOLD, and SERVICE. The specific mode in which the EC is operating determines the way in which it controls the up-down movement of the cars and open-close actions of the doors. The mode is selected per elevator car using the mode key-switch on in-car button panels. Behaviour of each mode is as follows:

- In **AUTO** mode (operational), the elevator behaves as a typical elevator would. Elevator cars are sent to floors where pick-up or drop-off requests have been made (via up/down button panels at each elevator entrance or in-car button panels, respecticely)
- In **HOLD** mode (operational), the elevator behaves as a service elevator, ignoring any passenger pick-up requests, but instead changing floors only when floor selection is made by a passenger *inside* the elevator car (using the in-car button panel). While in HOLD mode, the elevator doors stay open indefinitely at each destination floor (doors are closed while the elevator is in motion, of course). Only one destination floor can be selected at a time while in this mode. HOLD mode is generally used to facilitate move-ins (i.e. someone moving into a building) or planned transportation of large items.
- In **SERVICE** mode (recall), the elevator is returned to a (pre-configured) default/recall floor and remains on that floor with the doors open. Reanimation of the stopped elevator car then requires operator action (operator must use the in-car mode key switch to change to one of the two operational modes).

SysML Requirement Diagram

Product Functions Requirements Package

Modeling Elements

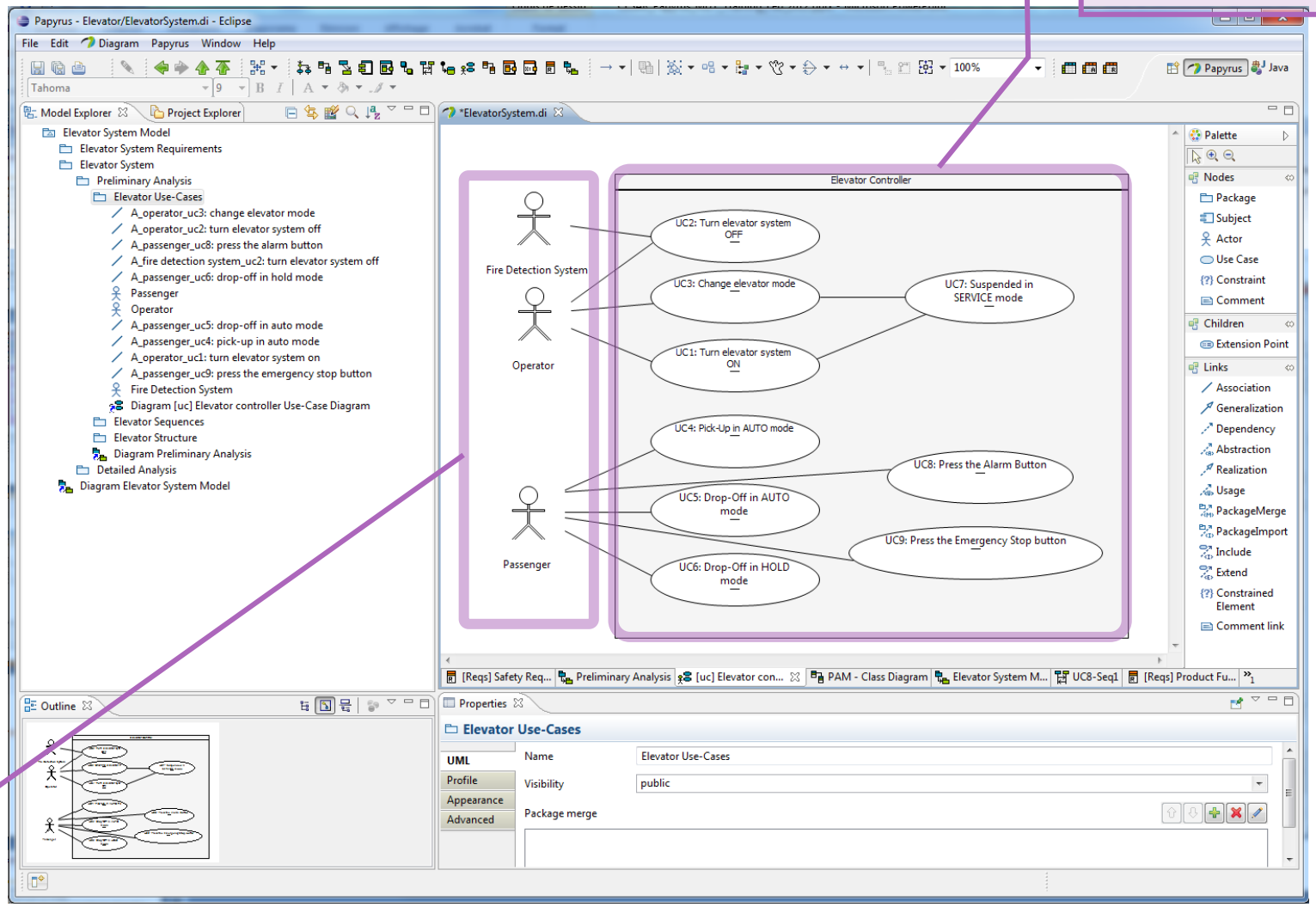


Activity 1: Defining a Preliminary Analysis

- **Structure of this Preliminary Analysis**
  - SysML Requirements Diagram
    - Create the diagram
    - Create packages and structure for the requirements
  - SysML Use-Cases Diagram
    - Creation of the diagram
    - Definition of the system and the context
    - Definition of use-cases

SysML Use-Cases Diagram

Elevator Controller Use-Cases

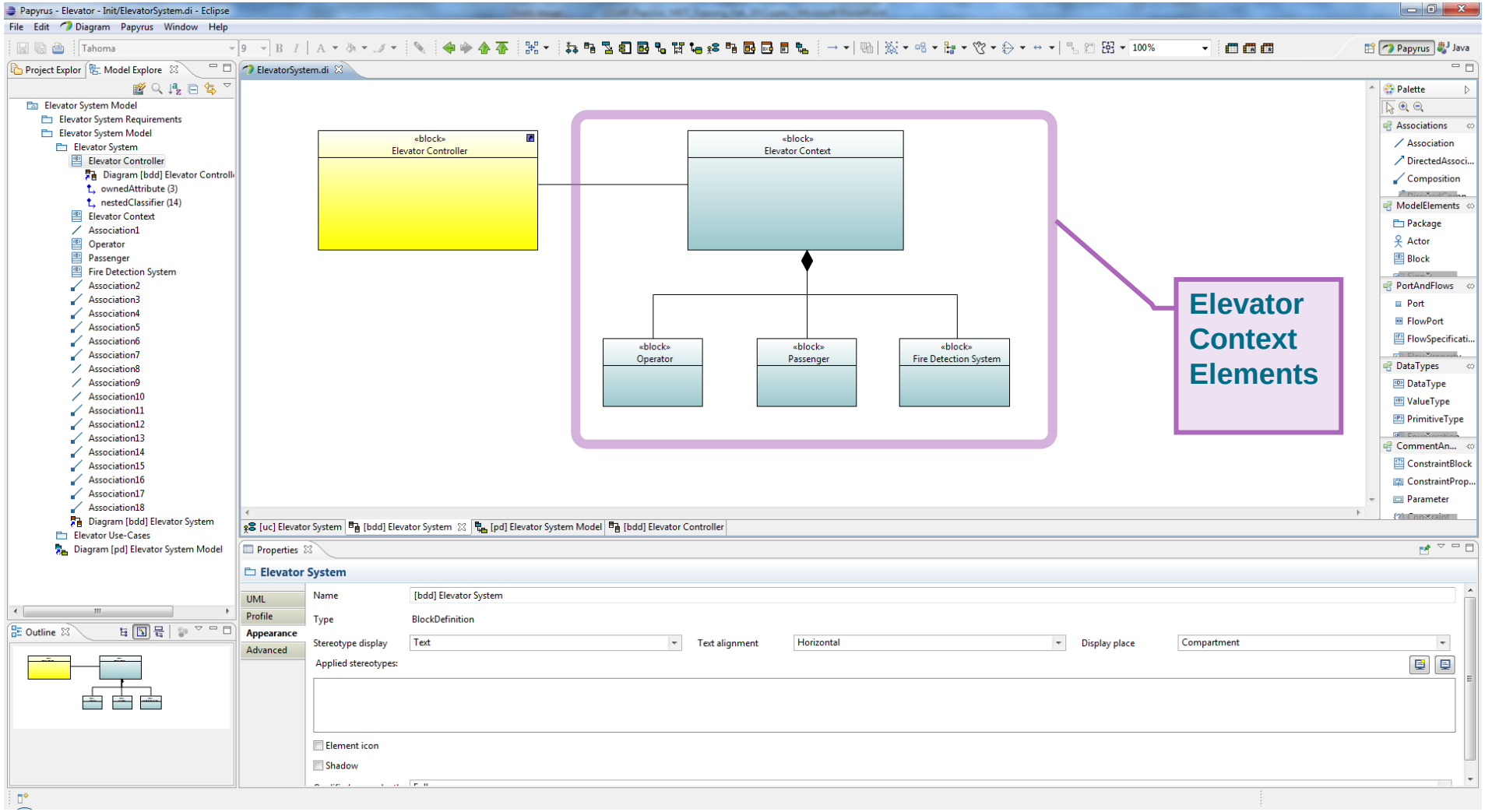


Elevator System Actors

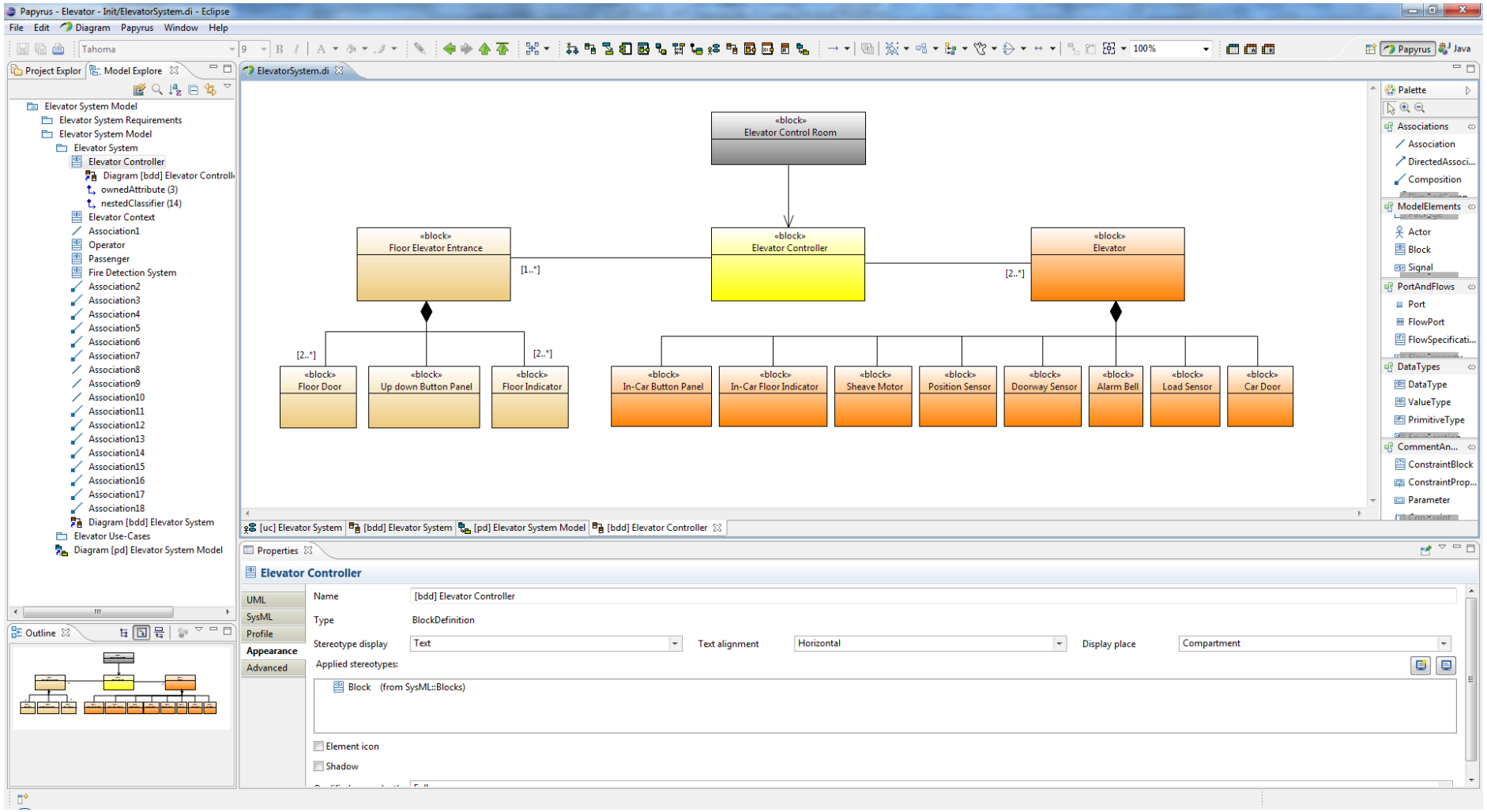
Activity 1: Defining a Preliminary Analysis

- **Structure of this Preliminary Analysis**
  - SysML Requirements Diagram
    - Create the diagram
    - Create packages and structure for the requirements
  - SysML Use-Cases Diagram
    - Creation of the diagram
    - Definition of the system and the context
    - Definition of use-cases
  - SysML Block Definition Diagram
    - Creation of the diagram
    - Definition of the system blocks
    - Definition of relations between these blocks

SysML Block Definition Diagram



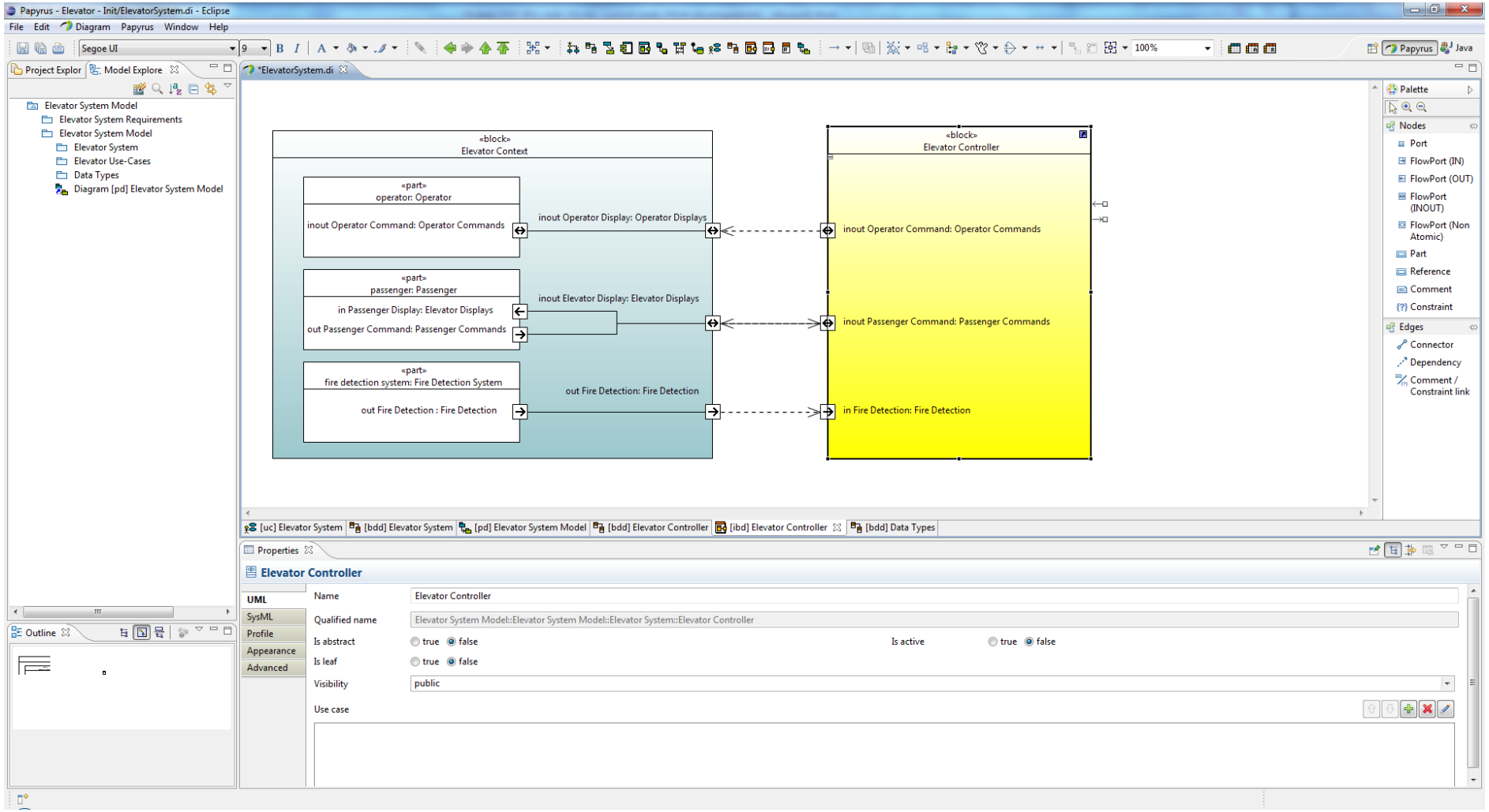
SysML Block Definition Diagram



Activity 1: Defining a Preliminary Analysis

- **Structure of this Preliminary Analysis**
  - SysML Requirements Diagram
    - Create the diagram
    - Create packages and structure for the requirements
  - SysML Use-Cases Diagram
    - Creation of the diagram
    - Definition of the system and the context
    - Definition of use-cases
  - SysML Block Definition Diagram
    - Creation of the diagram
    - Definition of the system blocks
    - Definition of relations between these blocks
  - SysML Internal Block Definition Diagram
    - Creation of the diagram
    - Definition of the system parts for high-level sequences

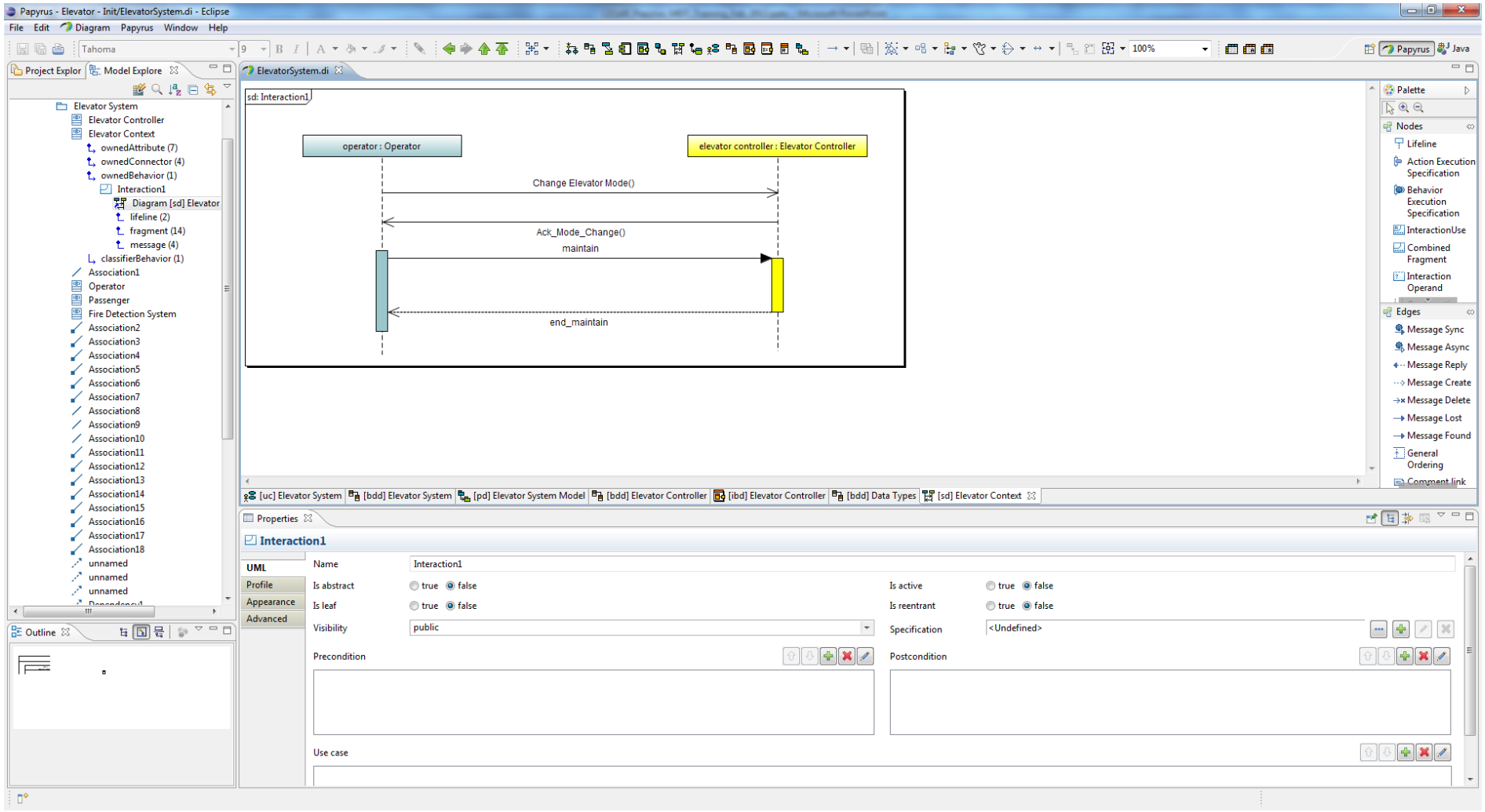
SysML Internal Block Definition diagram



Activity 1: Defining a Preliminary Analysis

- **Structure of this Preliminary Analysis**
  - SysML Requirements Diagram
    - Create the diagram
    - Create packages and structure for the requirements
  - SysML Use-Cases Diagram
    - Creation of the diagram
    - Definition of the system and the context
    - Definition of use-cases
  - SysML Block Definition Diagram
    - Creation of the diagram
    - Definition of the system blocks
    - Definition of relations between these blocks
  - SysML Internal Block Definition Diagram
    - Creation of the diagram
    - Definition of the system parts for high-level sequences
  - SysML Sequence Diagram
    - Creation of the diagram
    - Definition of high-level sequences between subsystems and actors

SysML Sequence Diagram



SysML Sequence Diagram: Basics

Sequence Diagram of interaction

LifeLine

Execution Specification

asyncCal

syncCall

reply

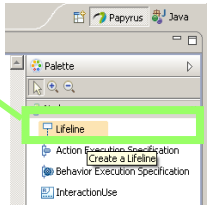
Model elements of interactions

- **Lifelines**
- **Fragments**
  - Execution Specifications
  - Events
  - Combined Fragments
- **Messages**
  - asyncCall
  - sync
  - reply
  - create
  - delete

SysML Sequence Diagrams: Basics
Lifeline creation

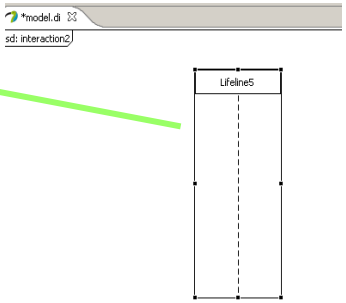
1

Select Lifeline tool in the palette



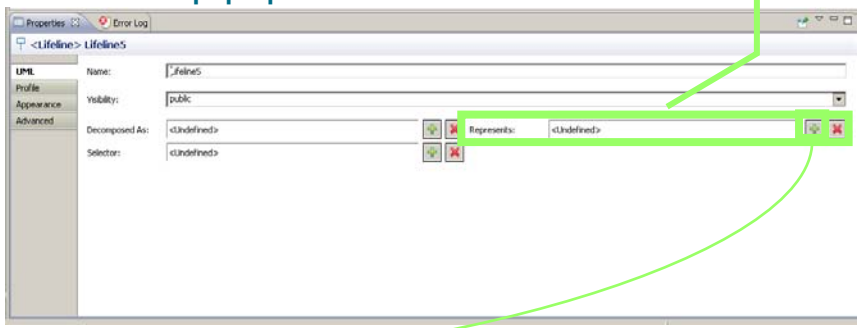
2

Click in the diagram to drop the lifeline



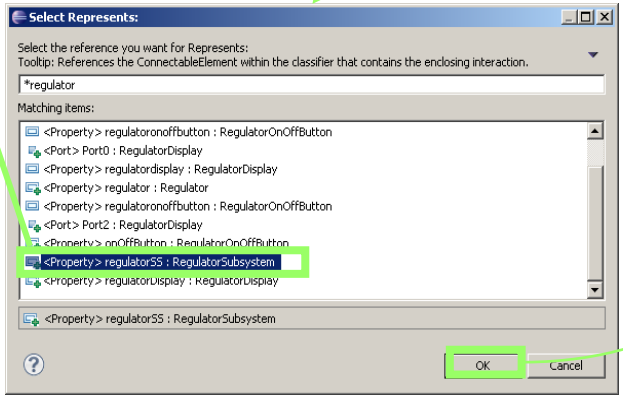
3

Set the represents propety : click on the red cross then select a part in the pop-up



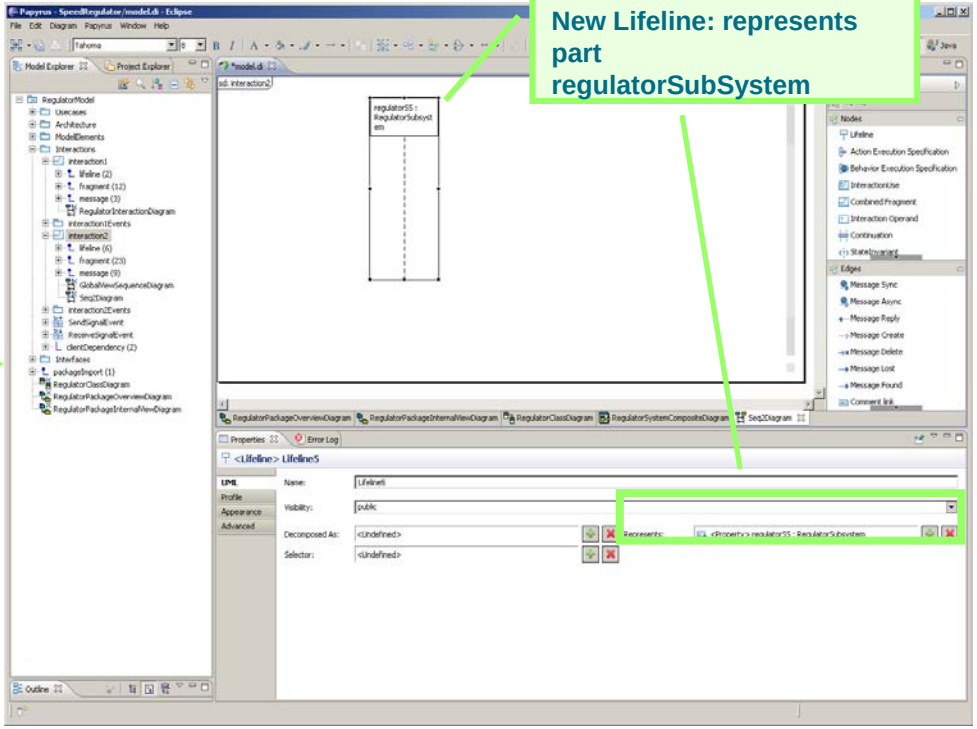
3

Select a part in the pop-up and press OK



4

New Lifeline: represents part regulatorSubSystem

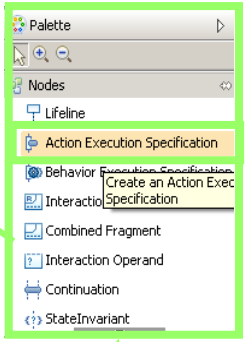


SysML Sequence diagrams: Basics
Execution Specification creation

- **Two kinds of Execution Specification can be created**
  - Action ES
  - Behavior ES

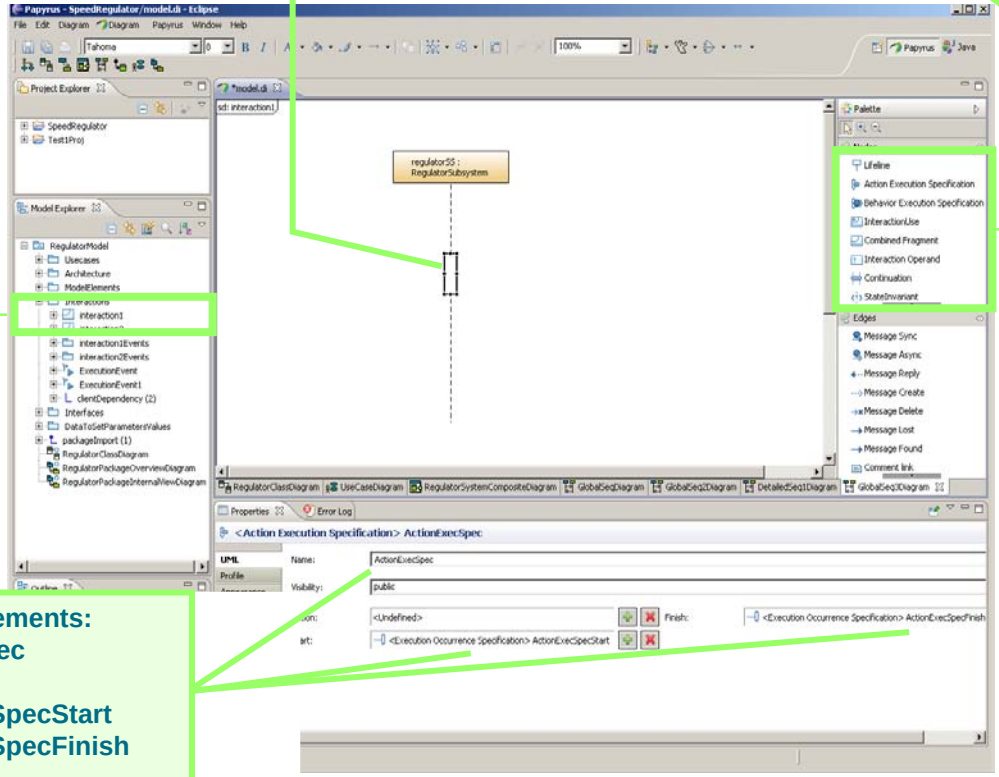
1

Select ES tool in the palette



2

Click and drop the ES on the lifeline.



New created UML Elements:
- **ActionExecutionSpec**
- **2 events**
  - **ActionExecutionSpecStart**
  - **ActionExecutionSpecFinish**

SysML Sequence diagrams: Basics
Message creation

- **Papyrus MDT provides dynamic support for message creation**
  - Selection of operation or signal attached to message
  - Dynamic creation of operation/signal
  - Dynamic creation of ES for synchronous message

1

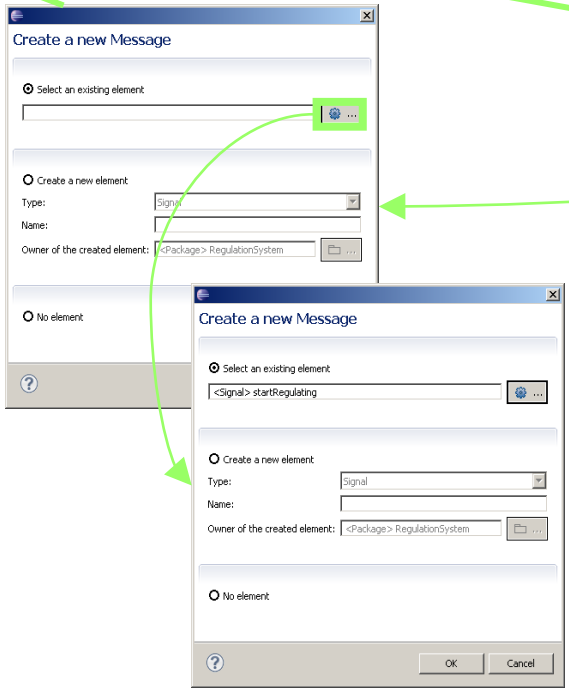
Select Message Creation tool from palette.

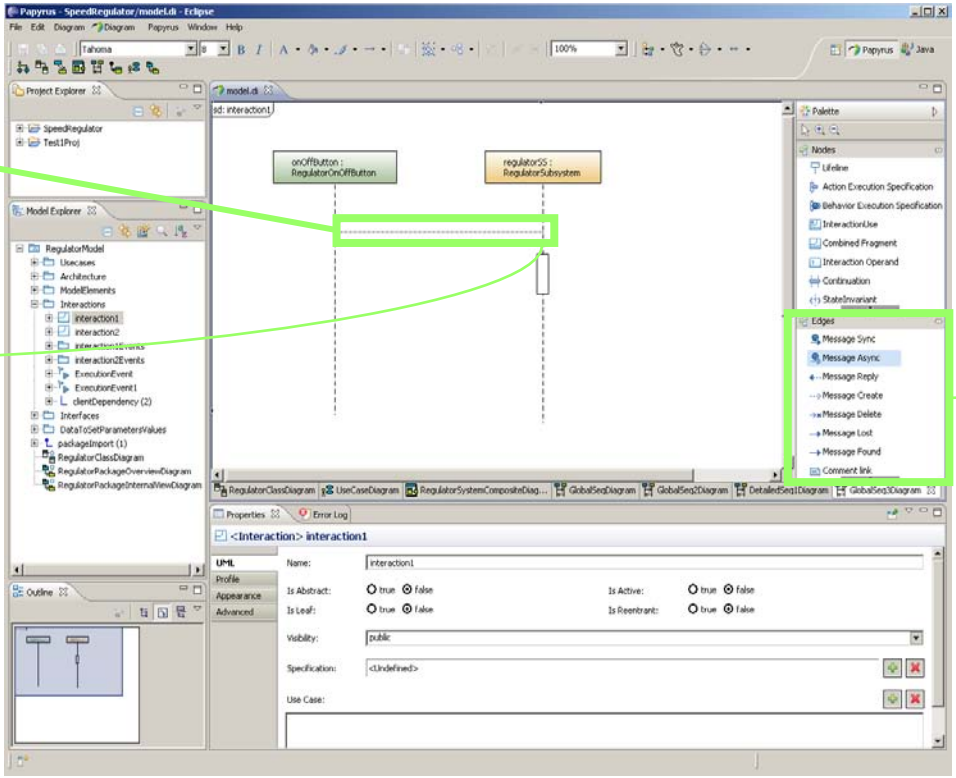
2

Place the message on the diagram click to set source, maintain mouse button down an drag towards target then release

3

Select an operation or signal or create a new one

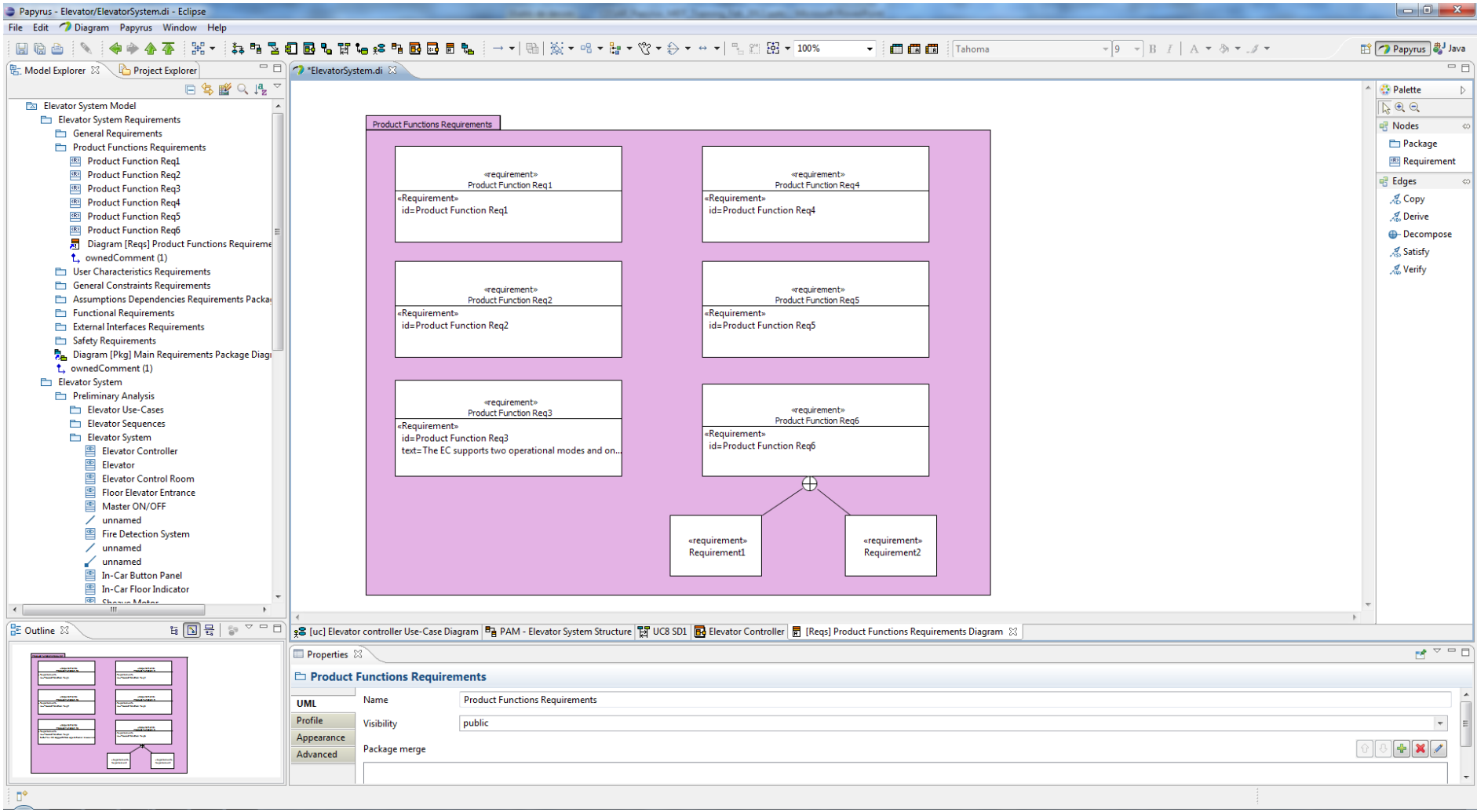




Activity 1: A Complete Preliminary Analysis

- **Structure of this Preliminary Analysis**
  - SysML Requirements Diagram
    - Create the diagram
    - Create packages and structure for the requirements
  - SysML Use-Cases Diagram
    - Creation of the diagram
    - Definition of the system and the context
    - Definition of use-cases
  - SysML Block Definition Diagram
    - Creation of the diagram
    - Definition of the system blocks
    - Definition of relations between these blocks
  - SysML Internal Block Definition Diagram
    - Creation of the diagram
    - Definition of the system parts for high-level sequences
  - SysML Sequence Diagram
    - Creation of the diagram
    - Definition of high-level sequences between subsystems and actors
  - SysML Requirement Diagram
    - To ensure requirements satisfaction

SysML Requirement Diagram



Activity 2: Defining a Detailed Analysis

- **Structure of the Detailed Analysis**
  - SysML Requirements Diagram
    - Derivation of new requirements from preliminary ones
  - SysML Internal Block Definition Diagram
    - To define the interconnection and interfaces between the parts of a block
    - Creation of the diagram
    - Few elements from the Block Definition Diagram defined in the Preliminary Analysis

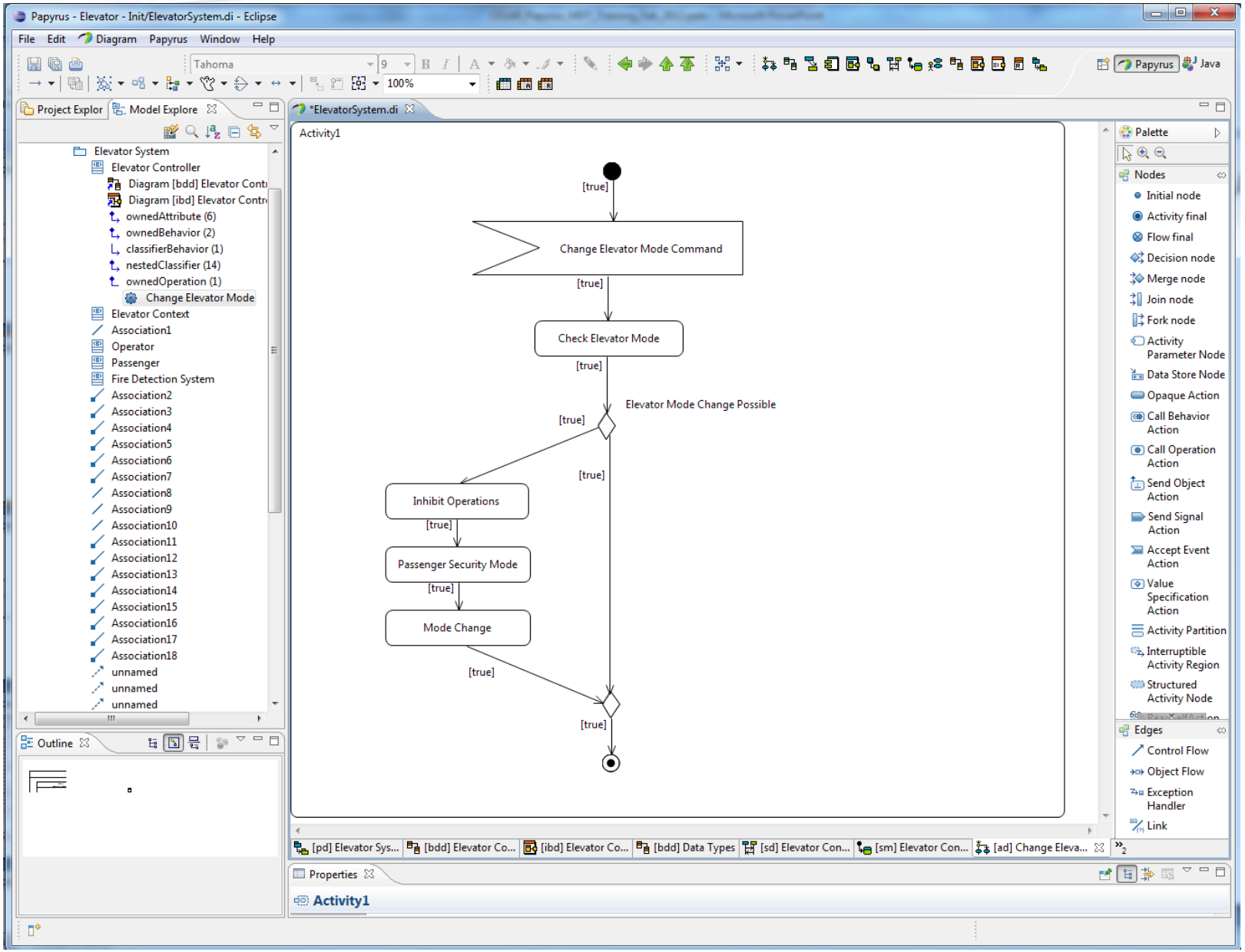
Activity 2: Defining a Detailed Analysis

- **Structure of the Detailed Analysis**
  - SysML Requirements Diagram
    - Derivation of new requirements from preliminary ones
  - SysML Internal Block Definition Diagram
    - To define the interconnection and interfaces between the parts of a block
    - Creation of the diagram
    - Few elements from the Block Definition Diagram defined in the Preliminary Analysis
  - SysML Sequence Diagram
    - To represent the behavior in terms of sequence of messages between parts
    - We detail the Sequence Diagram of the Preliminary Analysis

Activity 2: Defining a Detailed Analysis

- **Structure of the Detailed Analysis**
  - SysML Requirements Diagram
    - Derivation of new requirements from preliminary ones
  - SysML Internal Block Definition Diagram
    - To define the interconnection and interfaces between the parts of a block
    - Creation of the diagram
    - Few elements from the Block Definition Diagram defined in the Preliminary Analysis
  - SysML Sequence Diagram
    - To represent the behavior in terms of sequence of messages between parts
    - We detail the Sequence Diagram of the Preliminary Analysis
  - SysML Activity Diagram
    - To represent behavior in term of ordering of actions based on inputs, outputs, control

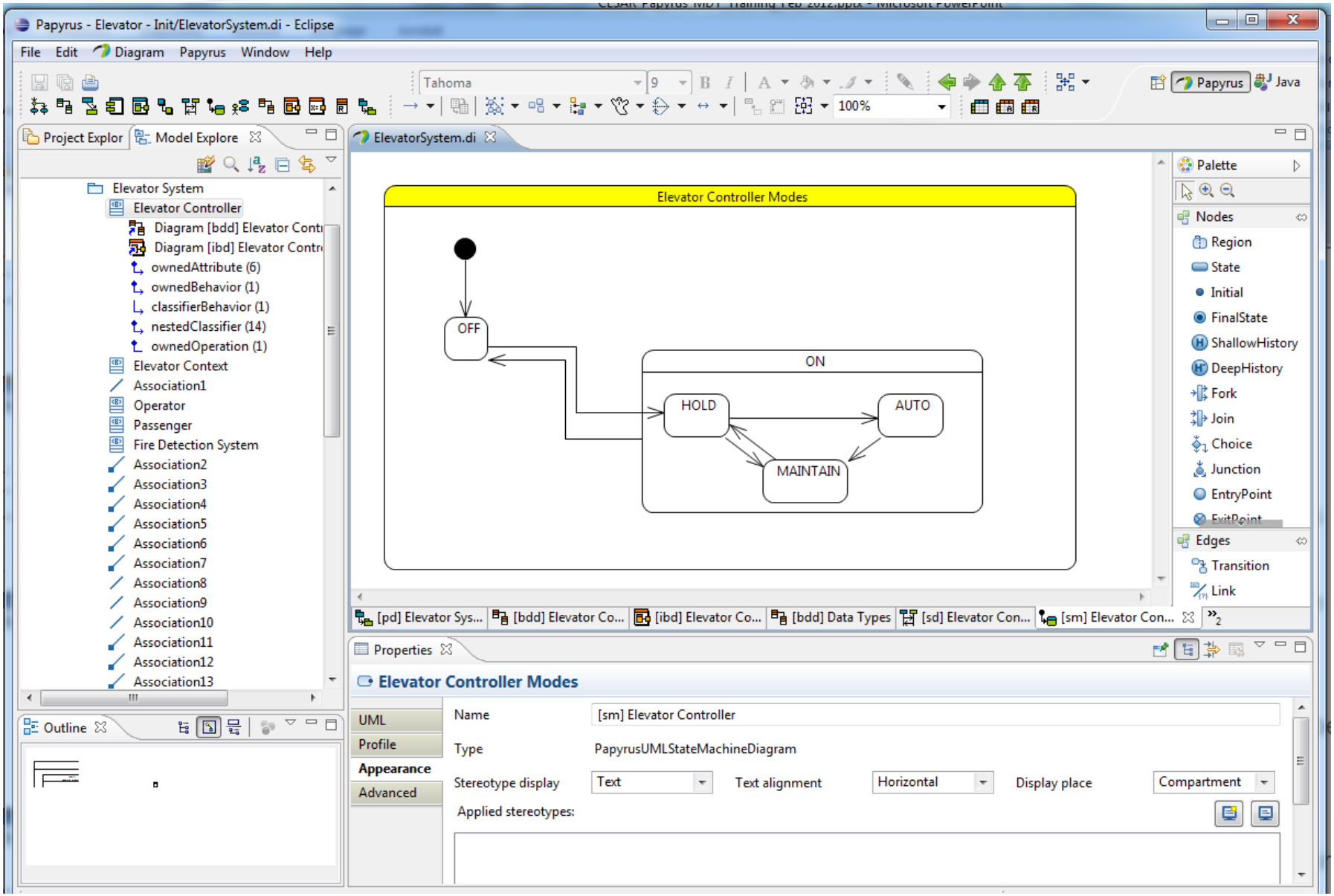
SysML Sequence Diagram



Activity 2: Defining a Detailed Analysis

- **Structure of the Detailed Analysis**
  - SysML Requirements Diagram
    - Derivation of new requirements from preliminary ones
  - SysML Internal Block Definition Diagram
    - To define the interconnection and interfaces between the parts of a block
    - Creation of the diagram
    - Few elements from the Block Definition Diagram defined in the Preliminary Analysis
  - SysML Sequence Diagram
    - To represent the behavior in terms of sequence of messages between parts
    - We detail the Sequence Diagram of the Preliminary Analysis
  - SysML Activity Diagram
    - To represent behavior in term of ordering of actions based on inputs, outputs, control
  - SysML State-Machine Diagram
    - To represent behavior in terms of transition between states trigerred by events

SysML State Machine Diagram



Agenda

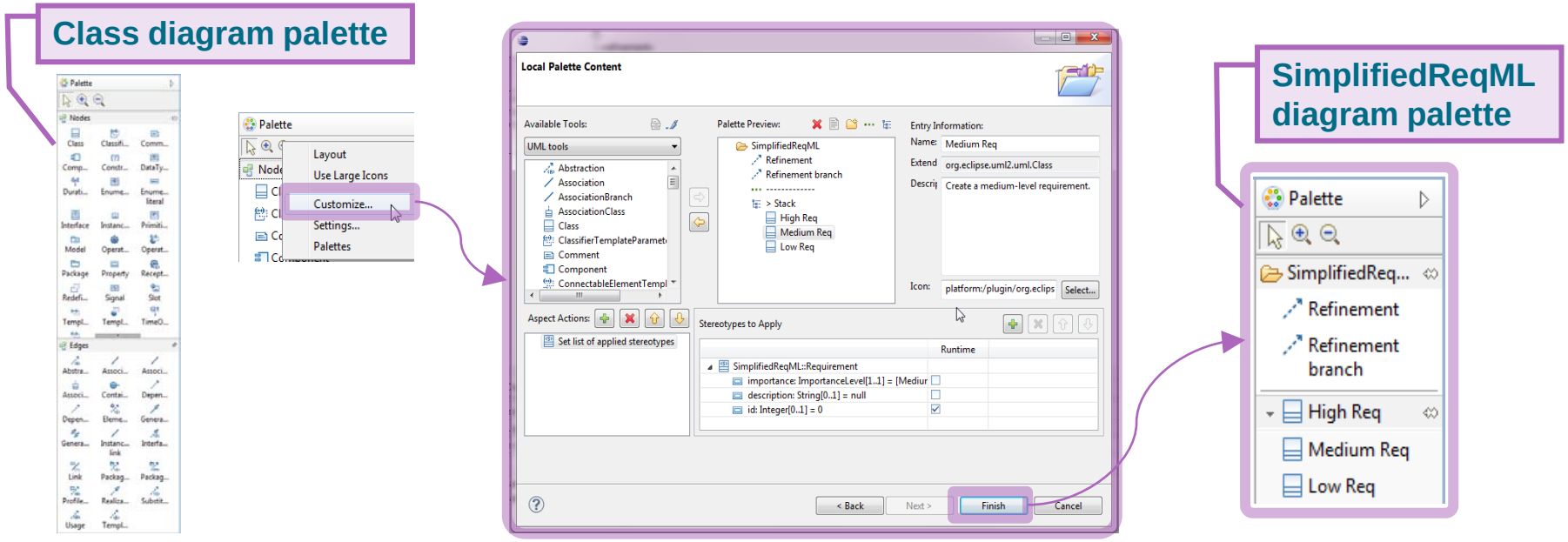
1. MDT Papyrus versus Papyrus 1.x
2. MDT Papyrus for System Design in SysML
3. **Other MDT Papyrus facilities**
4. Q&A

UML profile-based DSML within Papyrus

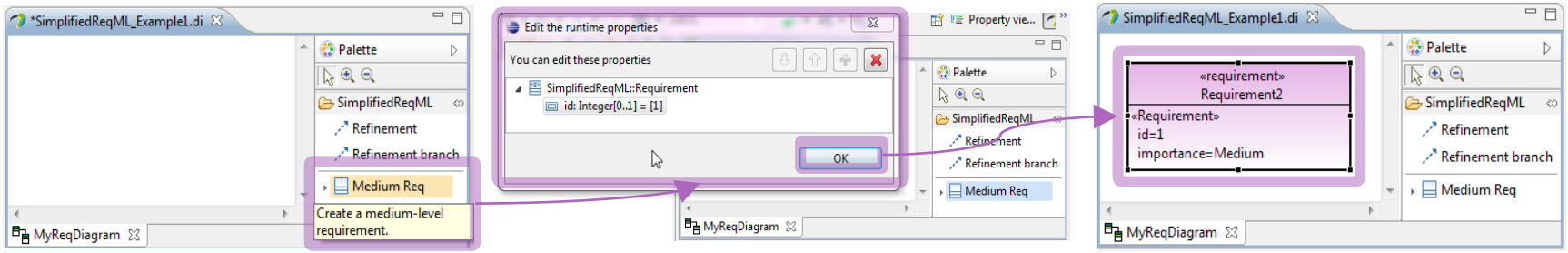
- **By-default definition of UML profile-based DSML**
  - UML profile are used to design the DSML in terms of UML extensions
    - Define the abstract syntax of the Language
    - Propose a default notation used to annotate UML model elements: «StereotypeName »
      - Include possible specific icons and shapes!
- **Advanced design of UML profile-based DSML**
  - Specific palettes
  - Specific properties editor
  - Specific model browser
  - Specific model validation rules
  - Specific editors
    - Custom Tabular Editor
    - Specific textual editor (based on Xtext)
    - Inherited Diagram Editors
    - Brand new editors (based on either EMF or GMF technologies)
  - Specific model wizzards

Specific Palette in Graphical Editors

- **Papyrus enables to customize the palette of its graphical editors**
  - Provide a very simple to use editors to adapt the palette content to specific needs

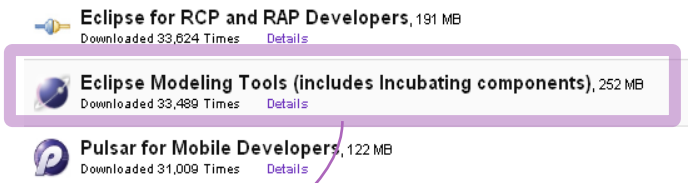


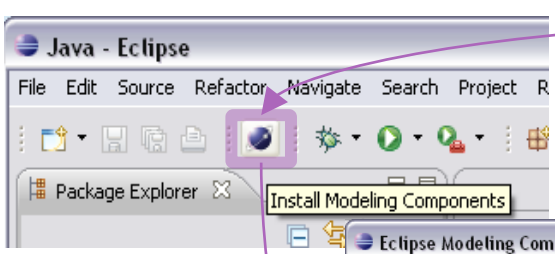
  - Usage illustration on a DSML for Requirements

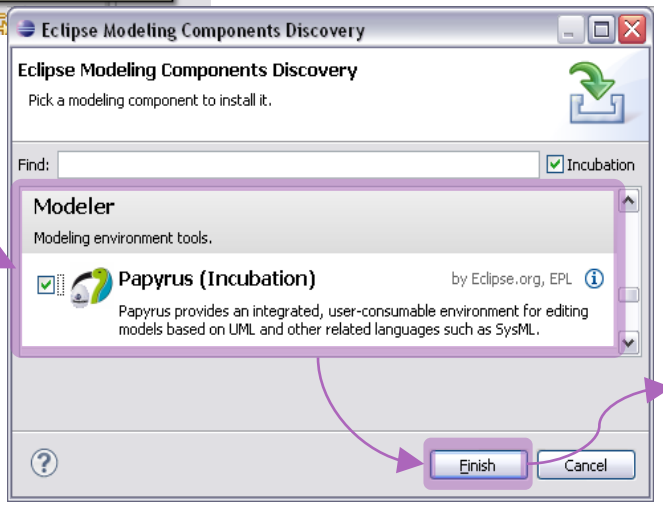


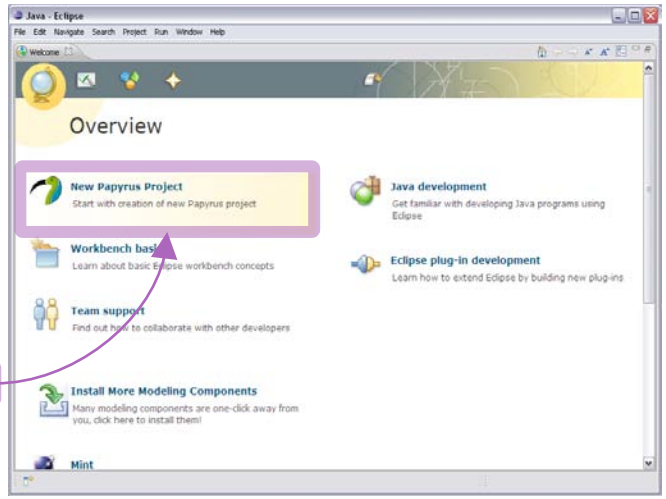
Papyrus standard download

- **Via the standard Eclipse Modeling Platform**
  - Download the Eclipse Modeling Platform
    - Helios: www.eclipse.org/downloads
    - Indigo: Indigo: http://download.eclipse.org/releases/indigo/
  - Unzip the downloaded file and start Eclipse.exe,
  - Launch the Modeling discovery site update,
  - Check Papyrus and start installation.









Papyrus in life…

Agenda

1. MDT Papyrus versus Papyrus 1.x
2. MDT Papyrus for System Design in SysML
3. Other MDT Papyrus facilities
4. Q&A

More Information

- **For developers...**
  - http://wiki.eclipse.org/Papyrus_Developer_Guide
  - http://dev.eclipse.org/mailman/listinfo/mdt-papyrus.dev

- **For users...**
  - http://www.eclipse.org/papyrus
  - news://news.eclipse.org/eclipse.papyrus

- **Papyrus project lead contact:**
  - sebastien.gerard@cea.fr