What is Model-Driven Development (MDD) and Further understanding of BridgePoint

Kenji Hisazumi

Kyushu University

Overview of this lecture

- Goal
  - Why should we model?
  - What is Model-Driven Development, and xtUML
  - xtUML Modeling in Detail

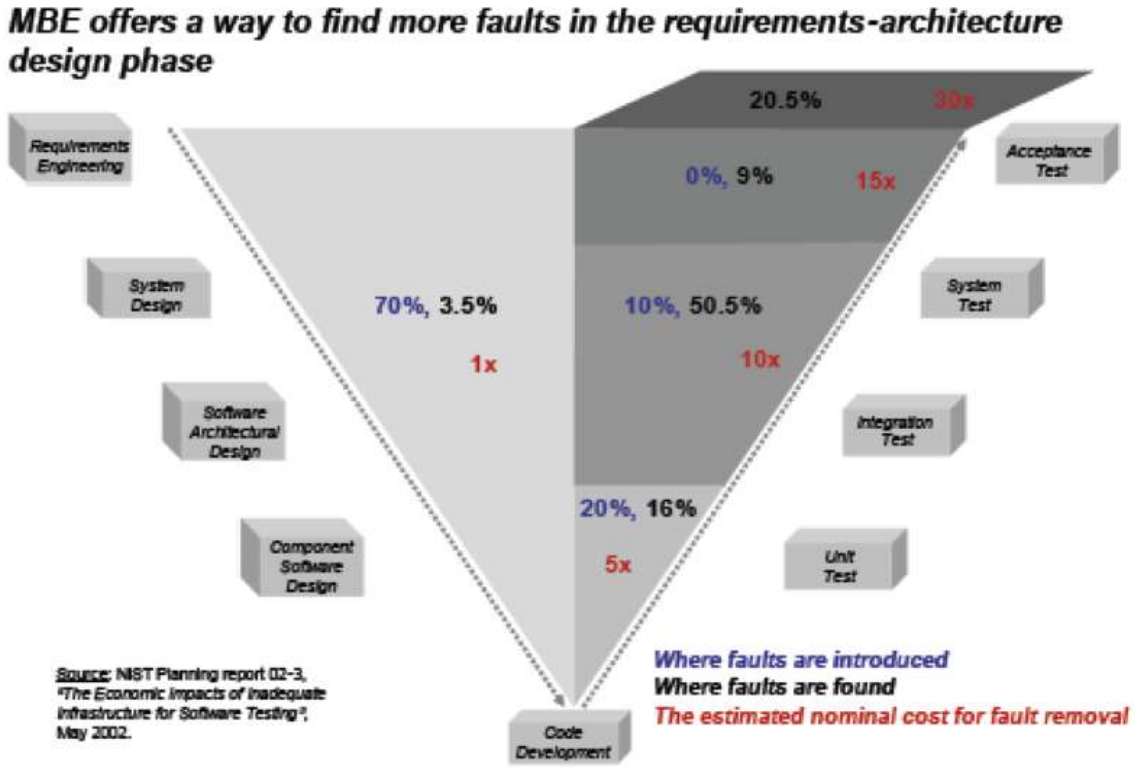
What is Model?

- Why should we bother modeling?
  - We already have Programming Language!
    - C, C++, Java, Python…
- Image software development
  - If you misunderstand requirements…
  - If you do not design properly…
  - And you find the defect after coding…

参考：不具合の発生・発見確率と訂正コスト

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

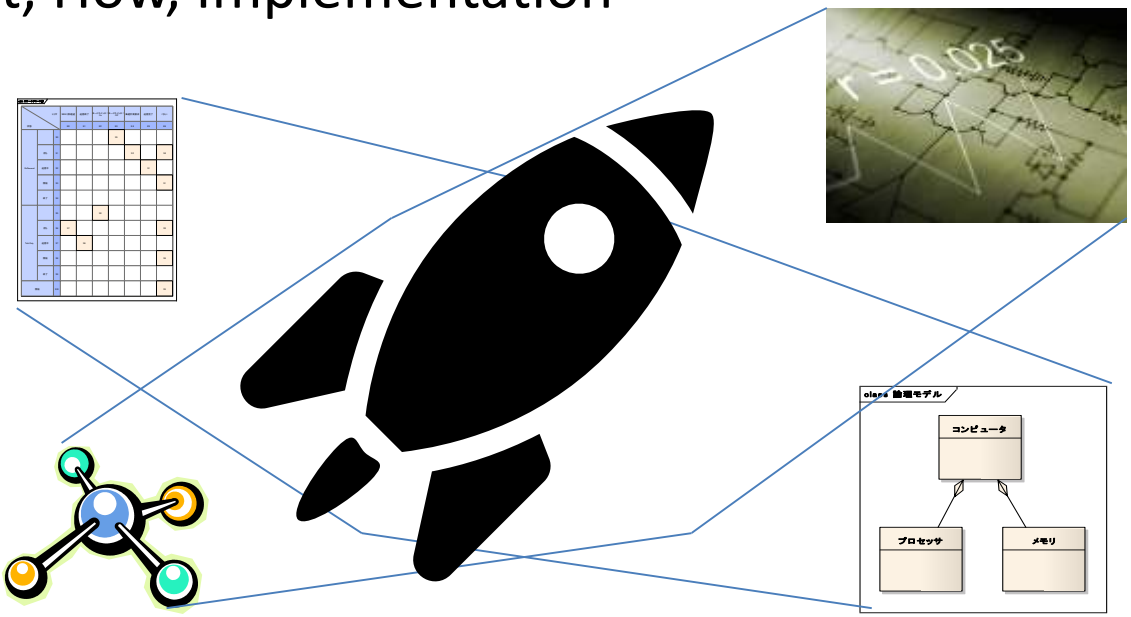
- 不具合が上流工程(設計)で混入する割合70%に対して、発見できたものは3.5%
- これをテスト段階で訂正すると、訂正コストは設計時の30倍



NIST Planning report 02-3, "The Economic Impacts of Inadequate Infrastructure for Software Testing", May 2002

Software System Model

- Visualize complex system, business process, *etc*.
  - From some different point of views
    - Function, Structure, and Behaviors
  - From some different abstraction level
    - What, How, Implementation



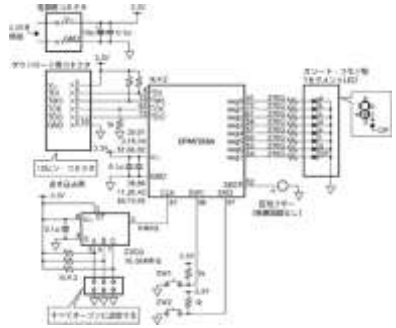
Software System Model: Stepwise Refinement

I want to measure time when I am cooking…..

Abstract

I want to measure any time

System should notify me when the time has elapsed



Concreate

Stepwise refinement

What (Requirement)

How (design)

Implementation

Model

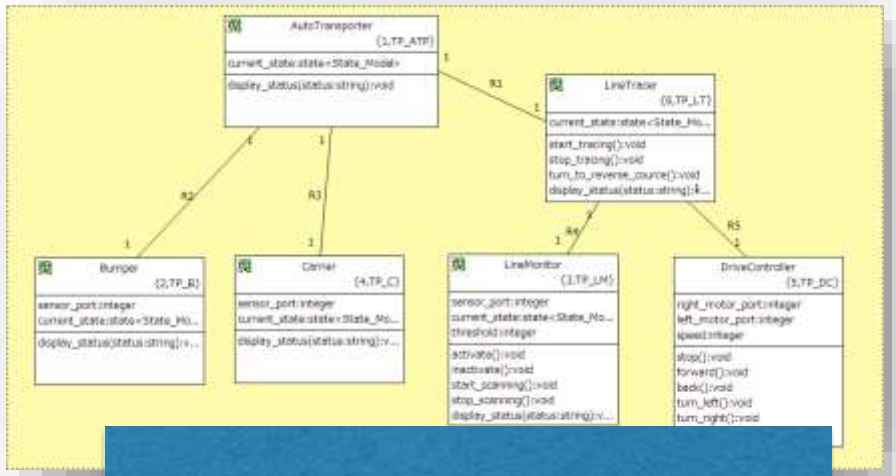
- UML: Unified Modeling Language
  - Industry standard *notation*
  - Family of diagrams

- A lot of type of diagrams
  - Requirement
  - Structure
  - Behavior
- You can choose diagrams what you want to model

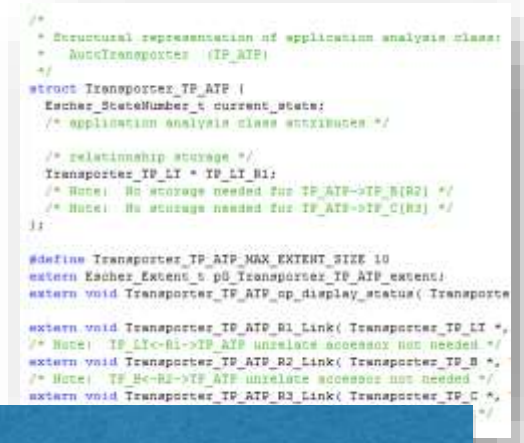
What is Model Driven Development (MDD)

Model Driven Development; MDD

Design Model



Source code

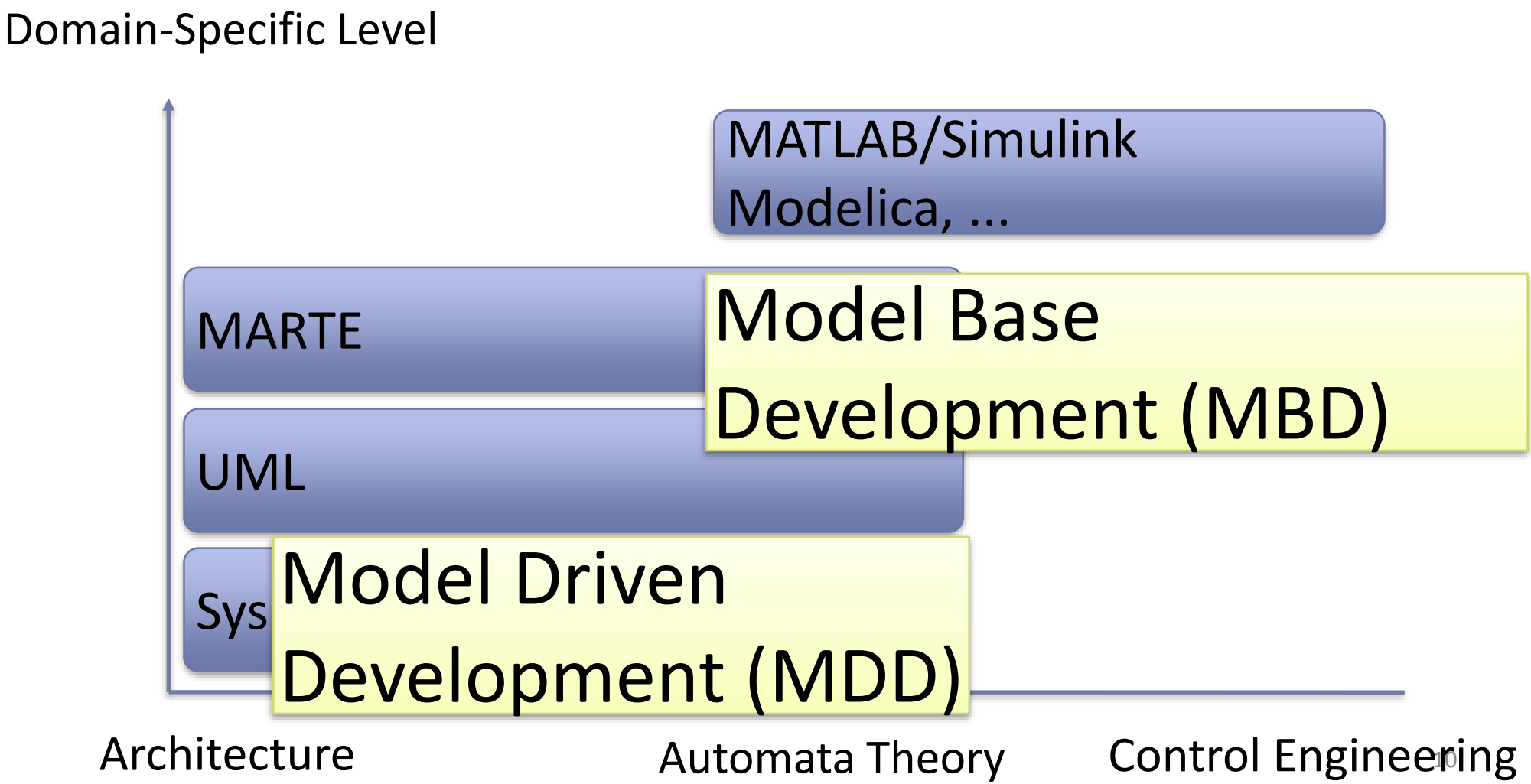


Verifiable in design phase to run high-level model

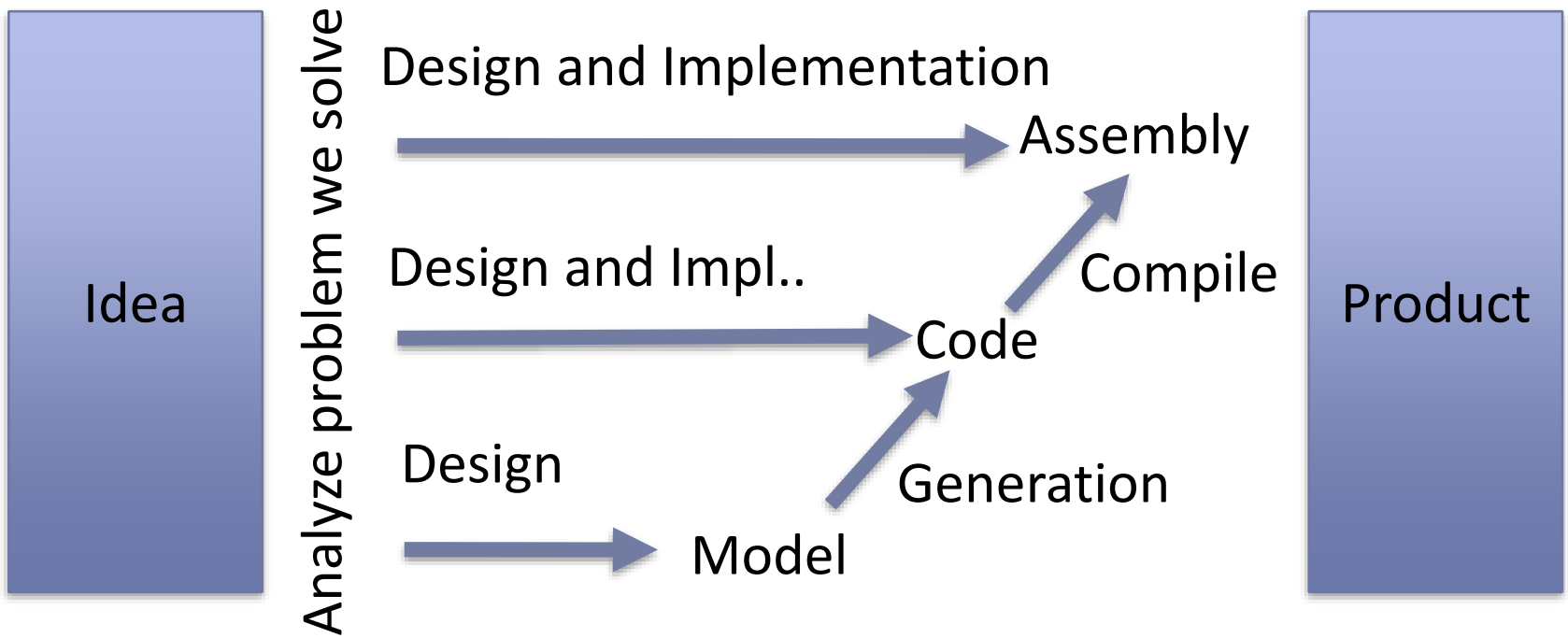
Automatic generation of source codes

MDD improves productivity and quality

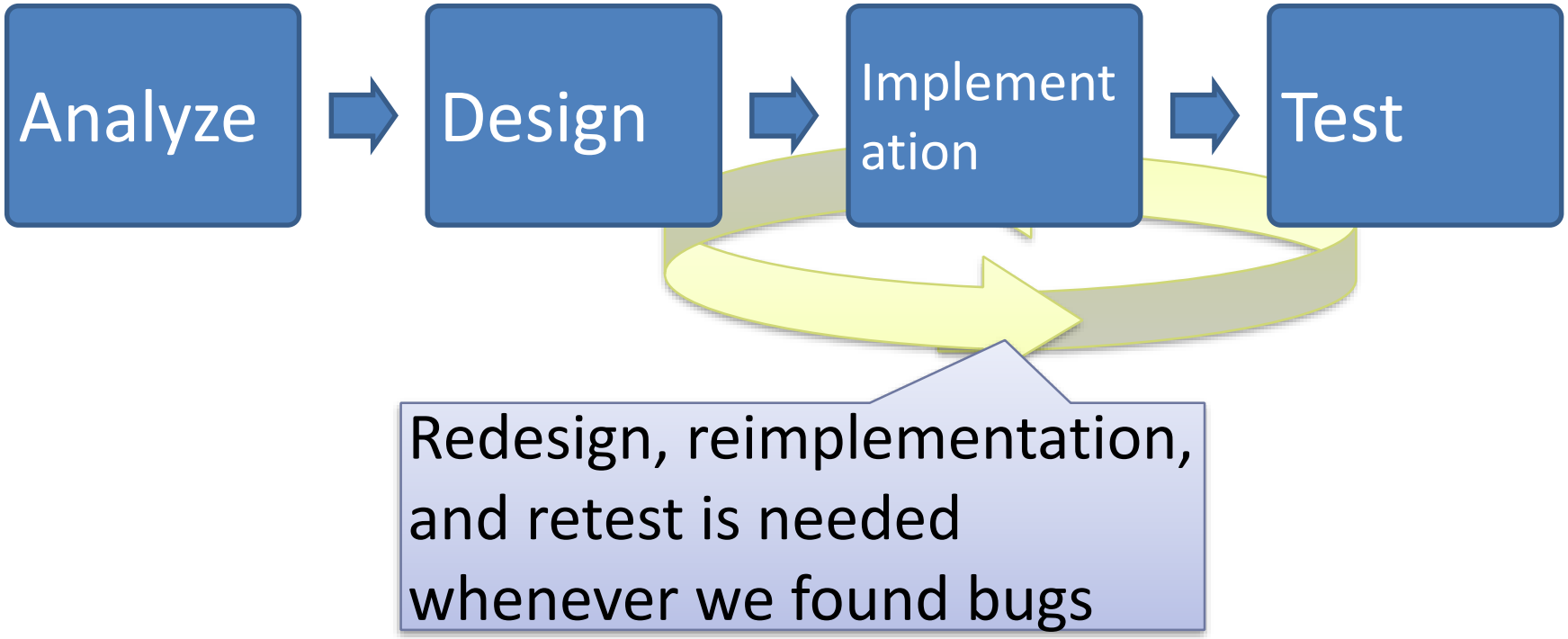
Variations Type of Models



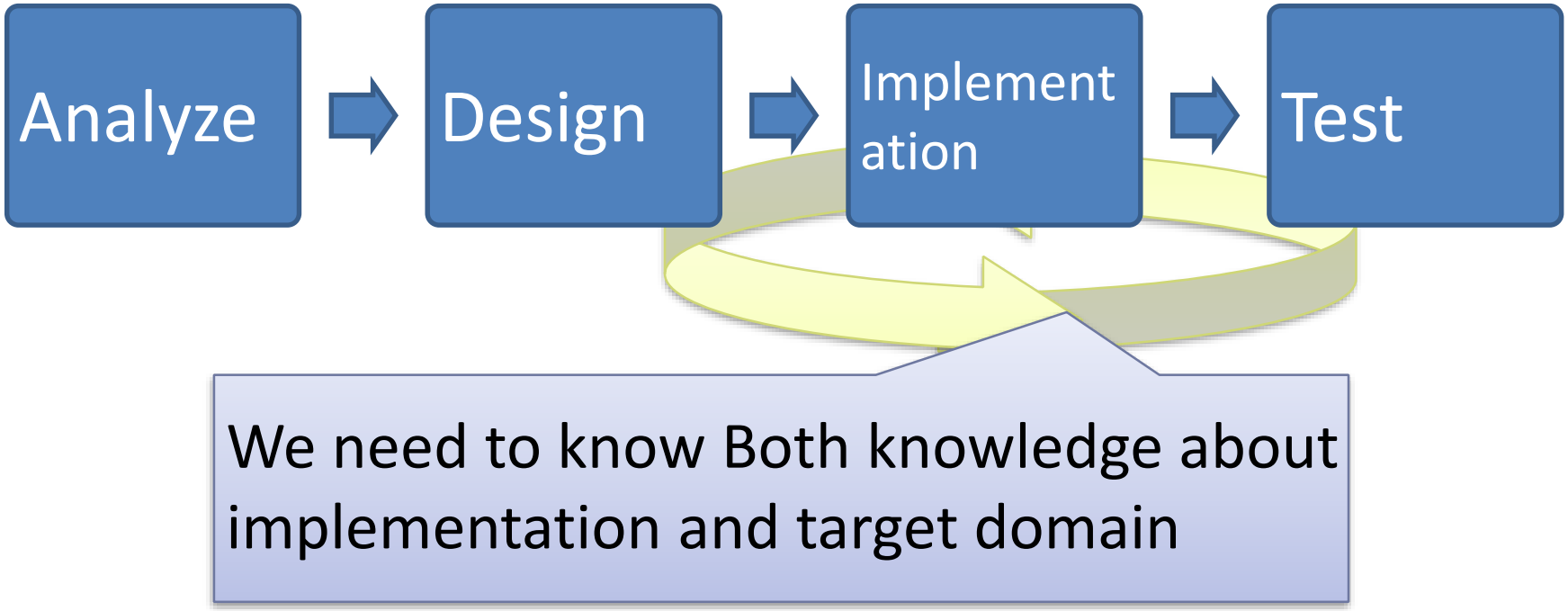
Productivity and Raise of Abstraction



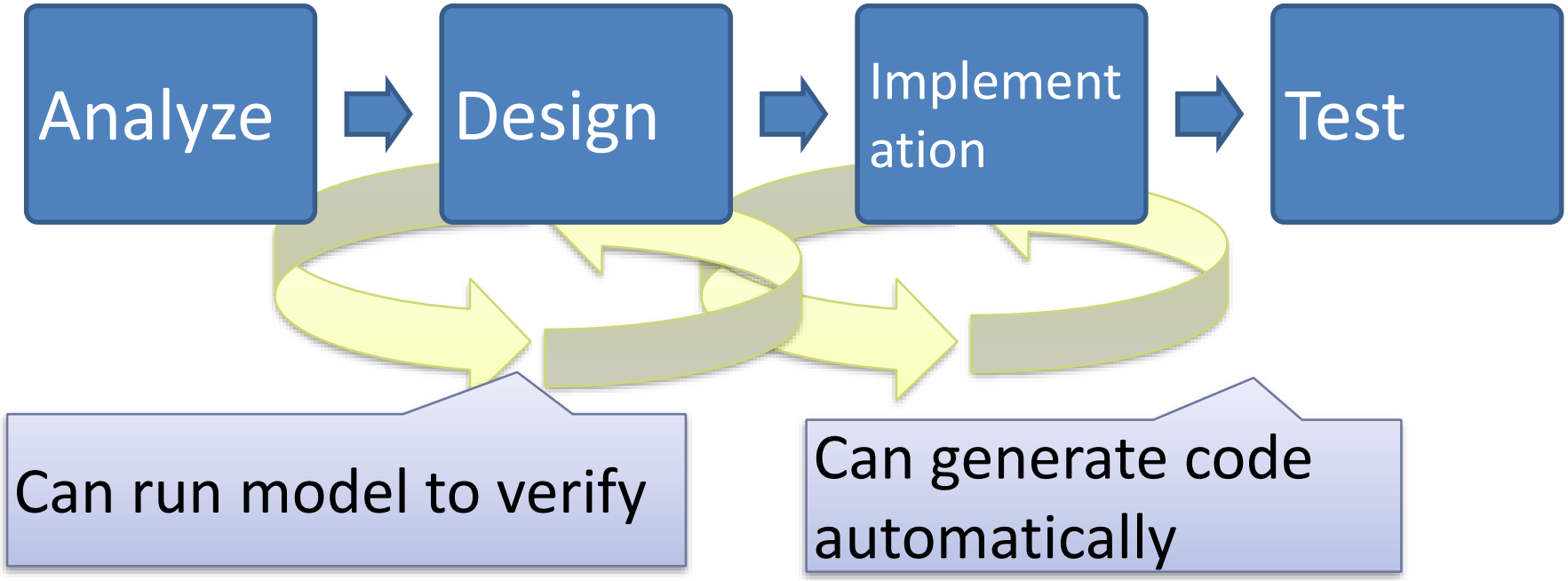
Traditional Way to Develop Software (1)



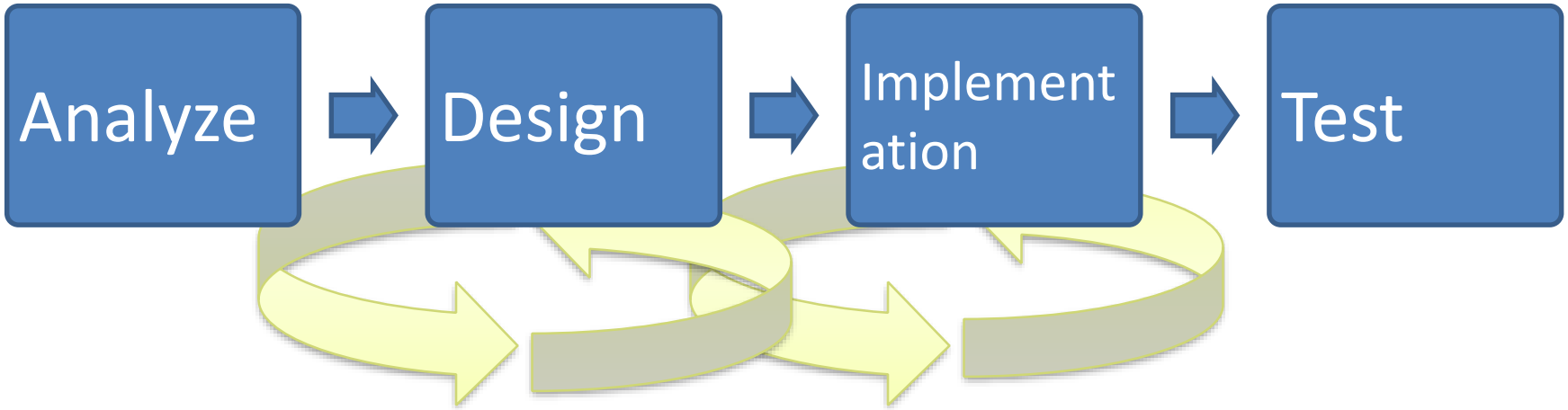
Traditional Way to Develop Software (2)



After MDD (1)


Analyze
Design
Implement ation
Test
Can run model to verify
Can generate code automatically

After MDD (2)



We can split a task to

- Domain expert in design phase (draw models)
- Implementation expert in implementation phase (model compiler)

Further Modeling using BridgePoint and xtUML

xtUML – eXecutable and Translatable UML

- Defines a method, including:
  - Semantics of diagrams
  - Relationship between diagrams
  - Action language
  - Execution rules
  - Order of construction
  - Path to implementation

xtUML Model Hierarchy

High level

Component Diagram
- **Decompose the application**
- **Define Interfaces**

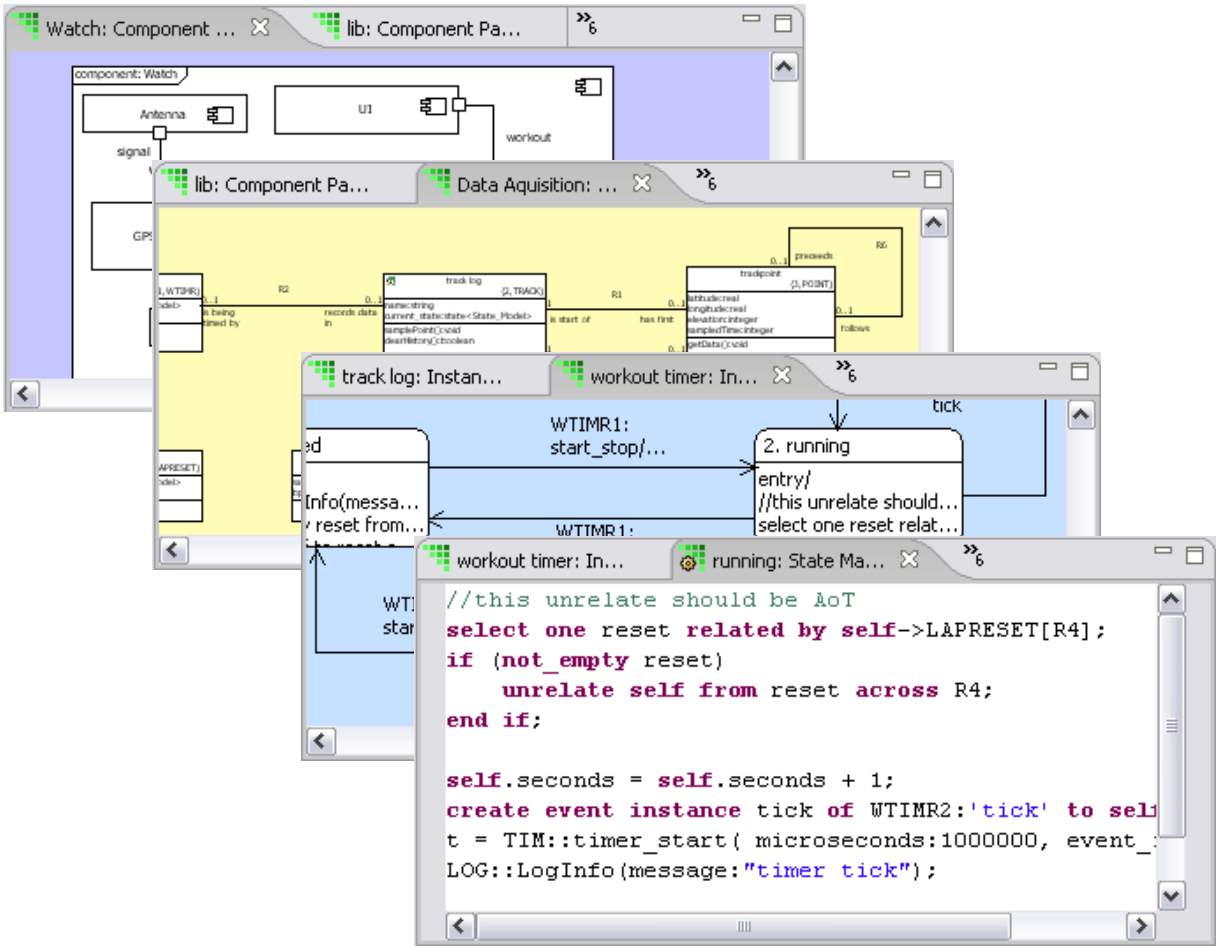
Class Diagram
- **Abstractions, associations**
- **Operations**

State Diagram
- **Functional lifecycle**
- **Event handling**

Action Specification
- **Processing**

Low level

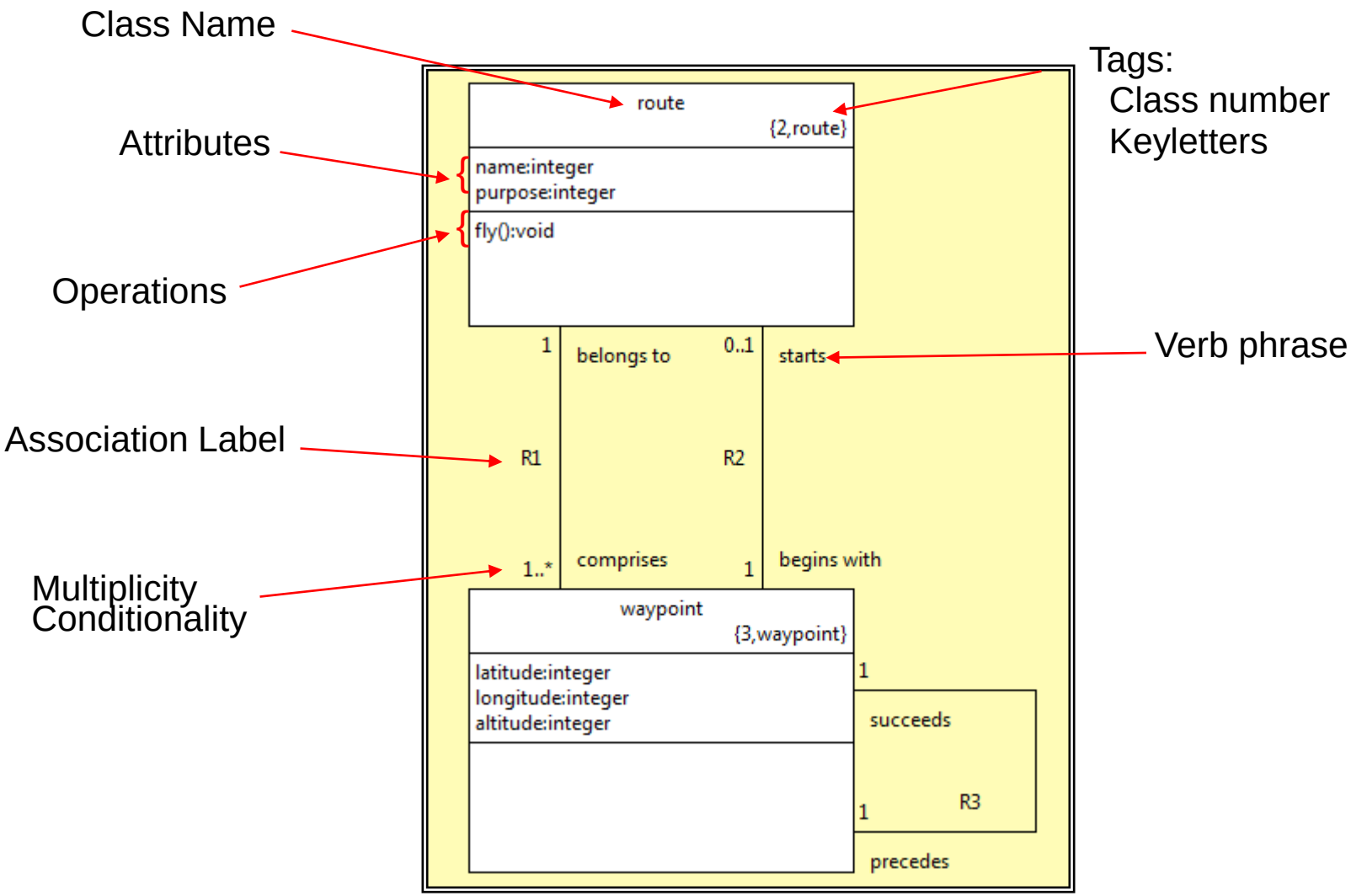
Stepwise refinements



Class Diagrams

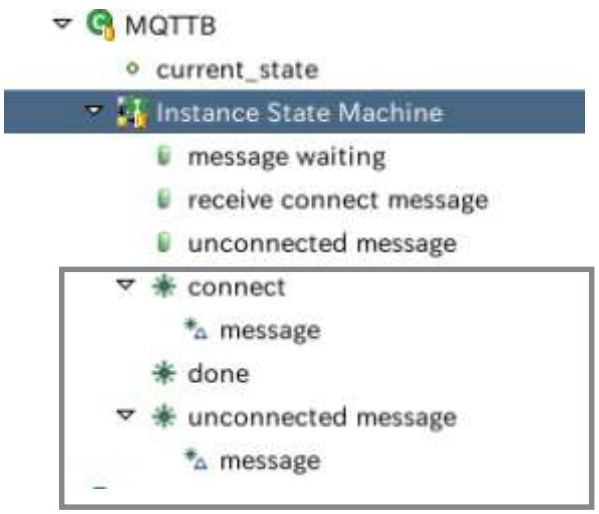
- Identify the types of object the component is concerned with and draw them as classes.
- Abstract the characteristics that define the classes; these are the class attributes.
- The choice of classes and attributes depends on the purpose of the component.
- Draw associations to represent real world relationships that exist between objects.
- During execution, instances of these classes and associations will be created as necessary to represent the real world.

Class Diagram Elements



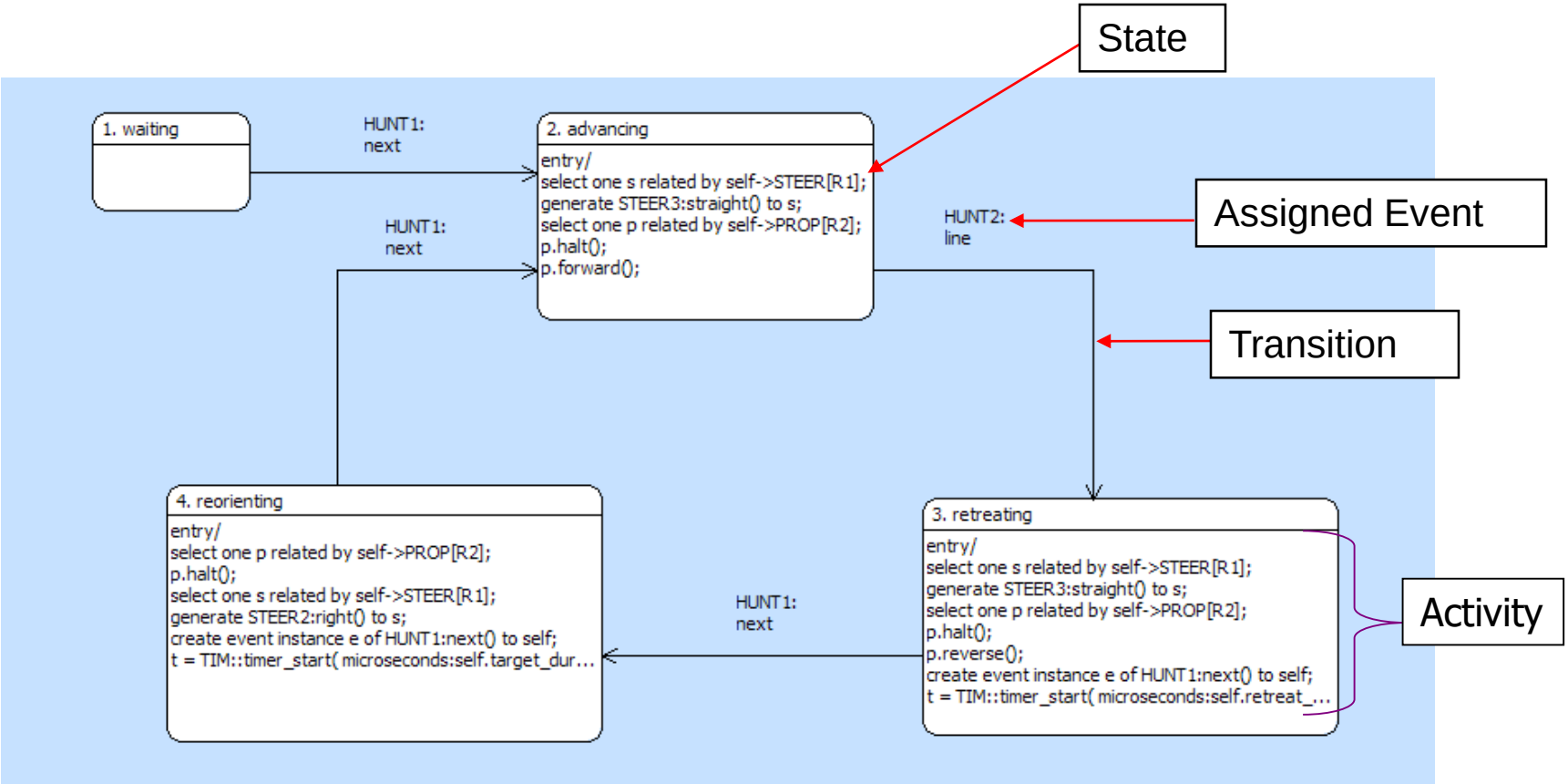
State Machine Diagram

- Define behaviors of a class using a state machine diagram
  - States, Transitions, and Events
  - Events
    - Should be predefined
    - Has parameters
- Multiple state machines are executed parallelly (theoretically).

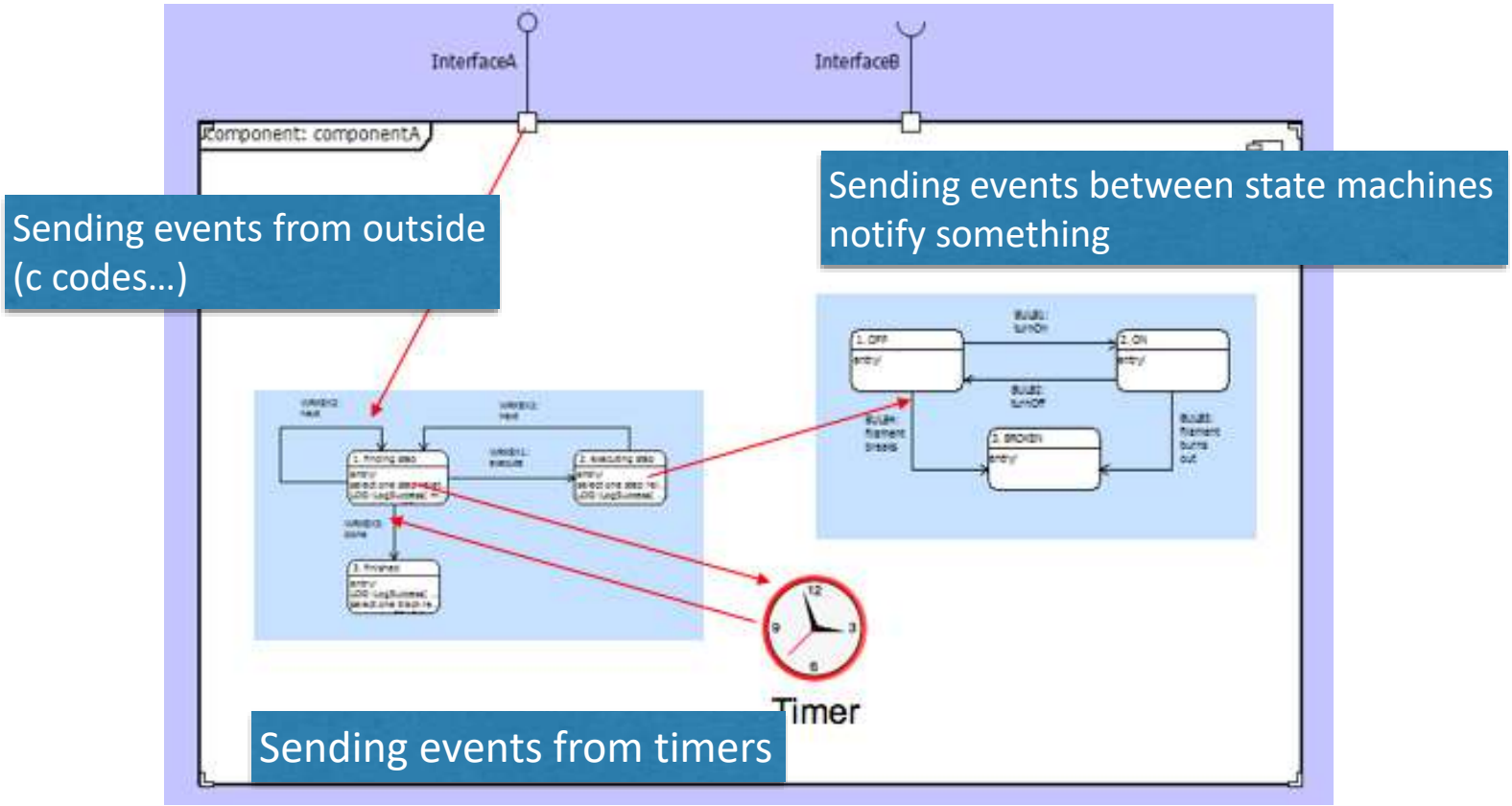


Definitions of events in a state machine

State Machine Diagram Elements



State Machines and Events



Note: In OOP, we write method call to ask something to other object normally. In xtUML, we use *events* for it normally.

Object Action Language; OAL

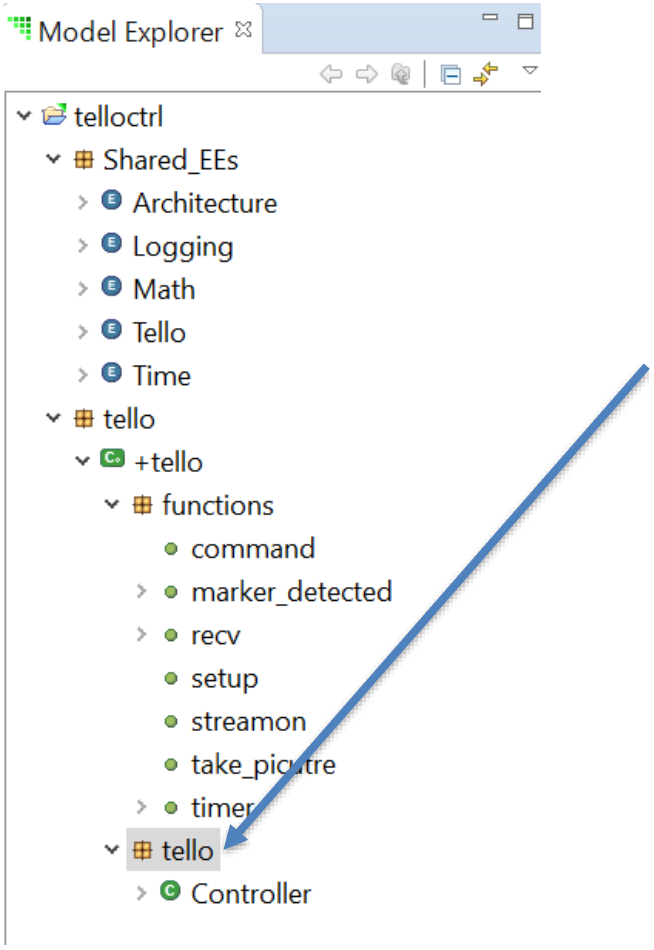
- We describe detail behaviors of functions, operations, and states using OAL
  - Abstract programming language
    - Independent generated codes (C language, etc)
  - SQL like
    - We can query to instances

Exercise 1: Save Maker Id

- Receive marker events and save the id
  - Save to an attribute of a *Controller* class

Add an Attribute (1)

1) Open the class diagram to click *tello* package



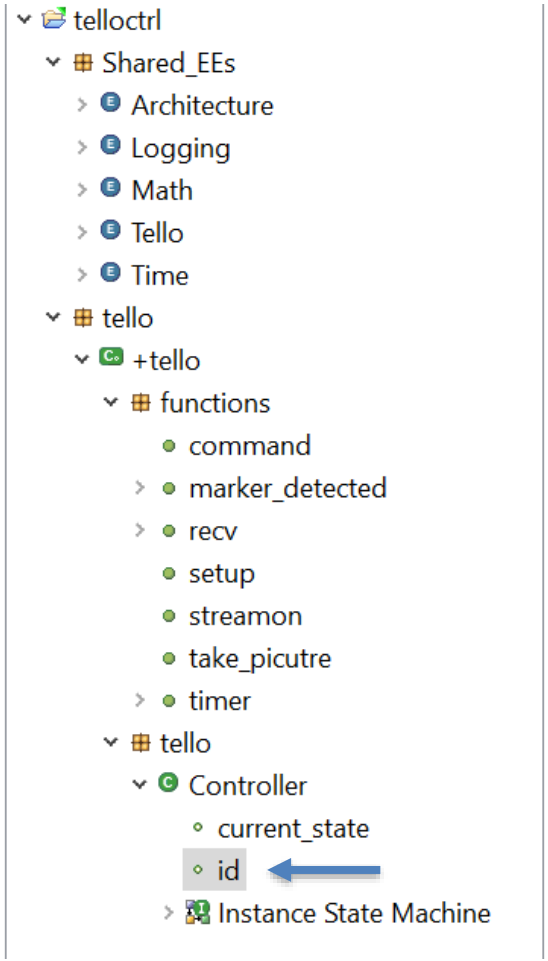
2) Right click the class you want to add an attributes

3) New -> Attribute

4) Give a name, here *id*

Add an Attribute (2)

- Set a type of the attributes in *Properties* view



Properties | Problems | Outline | Console | Git Staging

Property	Value
Basic	
Array Dimensions	
Attribute Name	id
Attribute Name Prefix	
Attribute Prefix Mode	No Prefix
Attribute Root Name	id
Default Value	
Description	
Type	integer
Non Derived Attribute	
New Base Attribute	as New Base Attribute

Set or Get a Value to Attribute

- In states or functions…
  - Set

Controller | tello | Controller::waiting

```
self.id = 10;
```

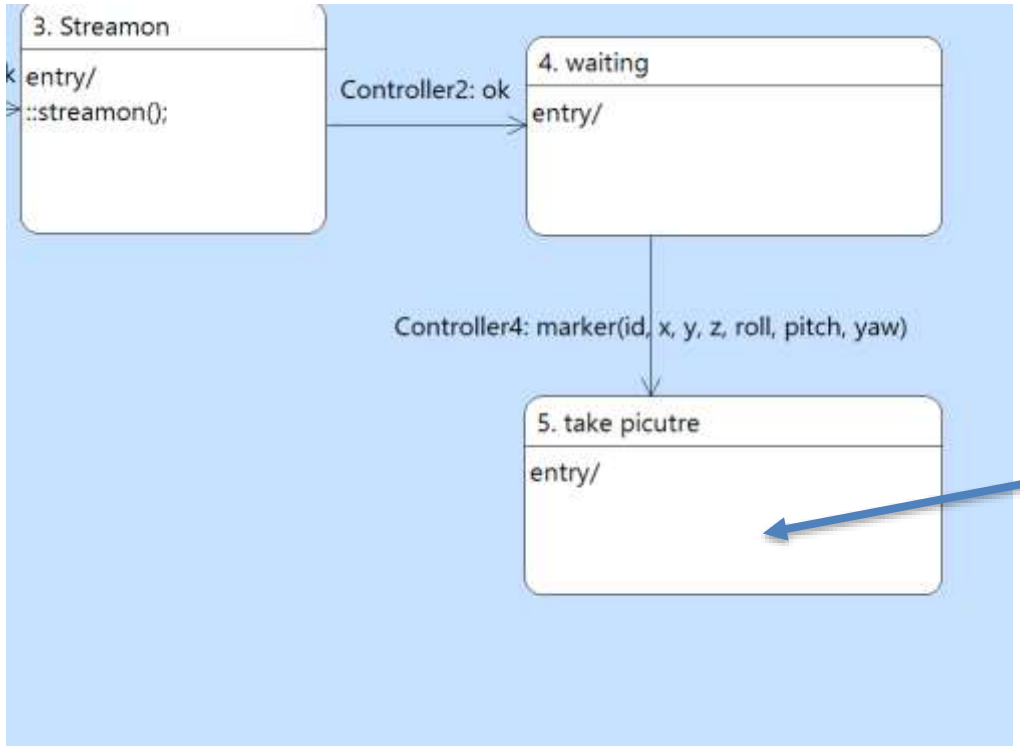
  - Get

Controller | tello | Controller::waiting

```
a = self.id;
```

Getting Parameters in an Event

- *rcvd_evt* is a special variable representing received events parameters




```
self.id = rcvd_evt.id;
```

State Event Matrix

tello | Controller

Events

	Controller1: init	Controller2: ok	Controller3: timeout	Controller4: marker
init	Send com...	Can't Happen	Can't Happen	Can't Happen
Send command	Can't Happen	Streamon	Can't Happen	Can't Happen
Streamon	Can't Happen	waiting	Can't Happen	Can't Happen
waiting	Can't Happen	Can't Happen	Can't Happen	marker
marker	Can't Happen	Can't Happen	Can't Happen	Can't Happen

State Transitions

States

What happened when there is no transitions?

- Can't Happen: Stop and handle errors
- Event Ignored: Just ignore the events

marker event can always happen
Marker event should be ignored.

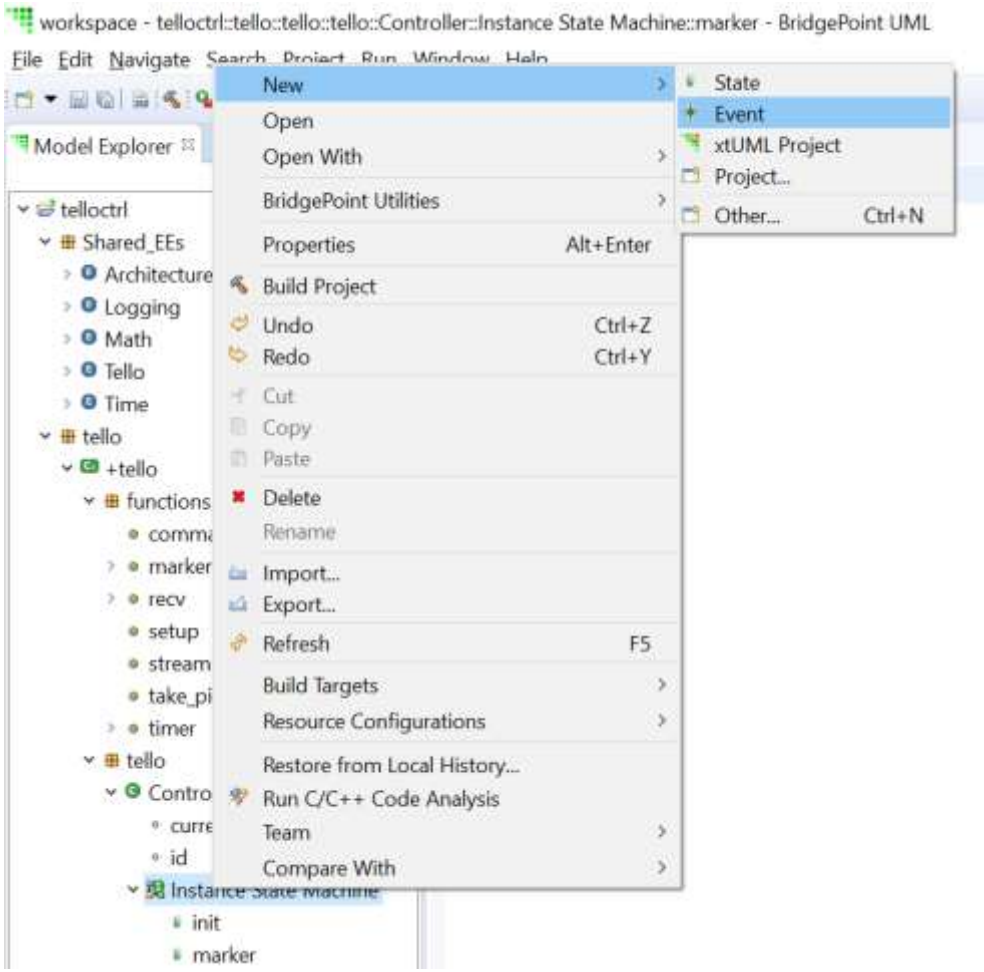
Graphical Editor | State Event Matrix

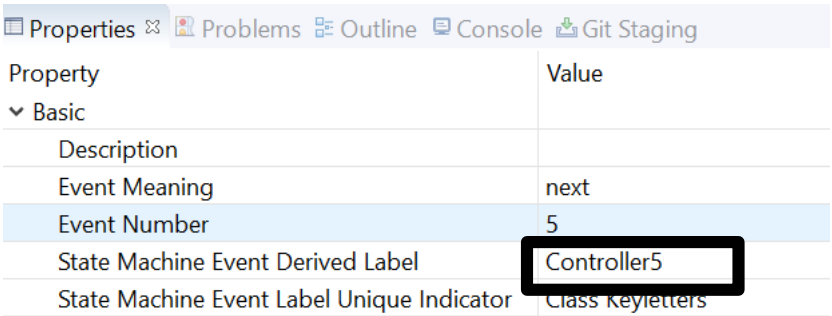
Exercise 2: Send an Event to Myself

- Steps
  - Add an event definition
  - Add program to send event
  - Add a state to transit

Define an Event

- Right Click *Instance State machine*
- New -> Event
- Give an appropriate name
- Check event name from property



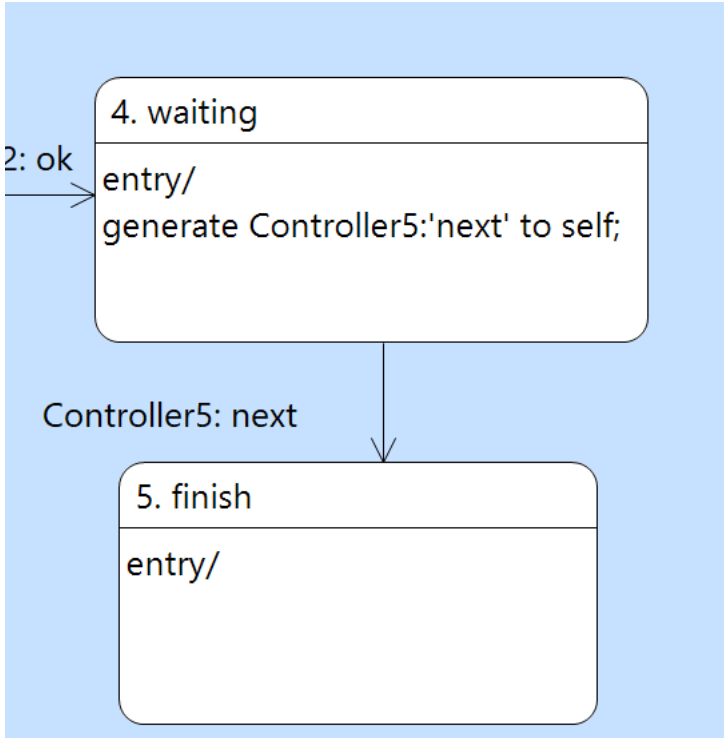


Generate Event in a State

- generate [event name]:[comment] to [receiver]

```
generate Controller5:'next' to self;
```

- Check reference manual in detail

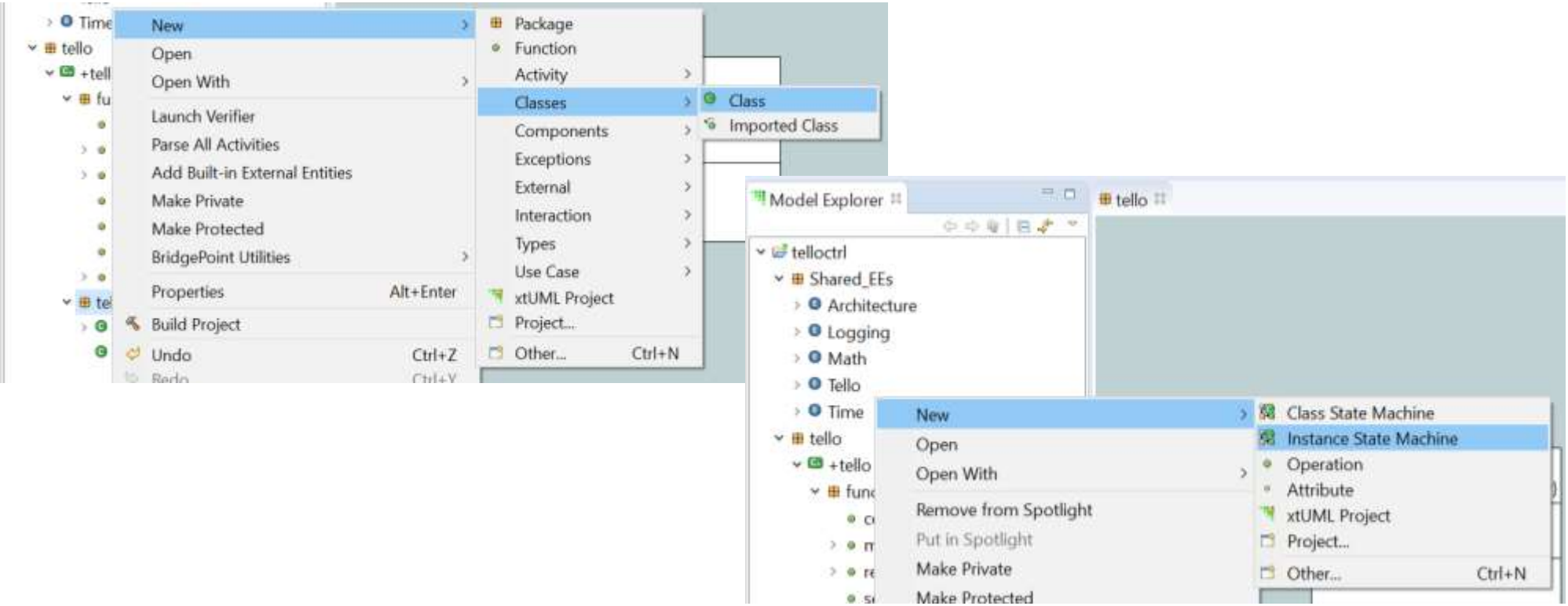


Exercise 3:
Ask something to an instance

- Send (generate) event to ask something to an instance
  - We can send event to both of myself and another instances.
- Steps
  - Make new class

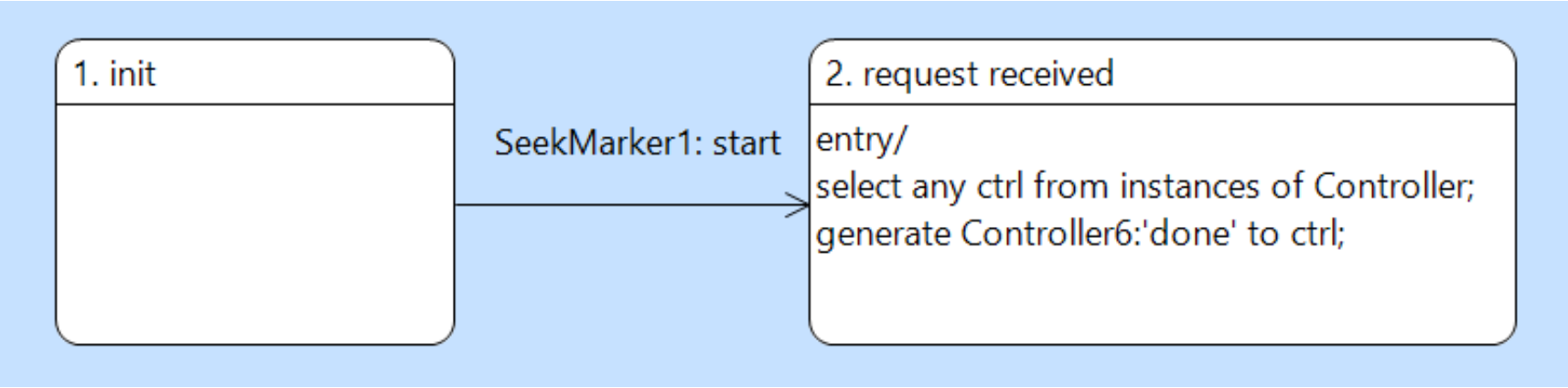
Add a Class and a State Machine

- Add class *Seek Marker* in the *tello* class diagram
- Add an instance state machine to the class

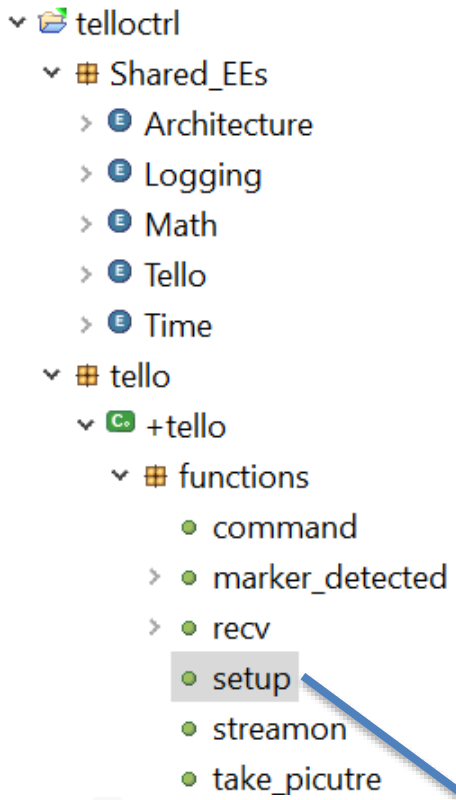


Define Events to receive and send request

- Define *start* event in the Seek Marker class
- Define *done* event in the Controller class
- Draw class diagram to receive the event
- Send *done* event from Seek Marker to Controller
  - Get an instance of Controller using *select any*



Create an Instance of the Class



1) Open function *setup*
2) Add a line to create an instance

```
create object instance ctrl of Controller;
create object instance seekmarker of SeekMarker;
generate Controller1:'init' to ctrl;
```