LECTURE 6

SPECIFICATION AND DESCRIPTION LANGUAGE

SUBJECTS

SDL and its Goals

SDL Structure

- System
- Block

SDL Behavior

- Process

SDL Signals

- Process Implicit and Explicit Addressing

Example of SDL System Definition

WHAT IS SDL

SDL is a formal language developed and standardized by The ITU-T

Intended for the specification of *complex*, *distributed*, *event-driven*, *real-time*, and *interactive* applications

These applications typically involve many concurrent activities that communicate using discrete signals

WHAT IS SDL

SDL describes the *architecture*, *behavior* and *data* of *distributed* systems in *real-time* environments

Useful for specifying the specification of telecommunication system behavior

SDL covers different levels of abstraction, from a broad overview down to detailed design level

APPLICATION AREA

Type of systems:

- Real time
- Interactive
- Distributed
- Heterogeneous

Type of information:

- Behavior, structure

Level of abstraction:

- Overview to details

USE OF SDL

SDL can be used for various stages of the development:

- Formal specification modeling
- Design
- *Implementation?*
  - *SDL was not originally intended for implementation, but translation to code is possible*

GOALS

Use of computer based tools to create, maintain and analyze specifications

- Formal high quality descriptions produced better, quicker and cheaper
- Analyzing specifications for completeness and correctness
- Support human communication and understanding of requirements

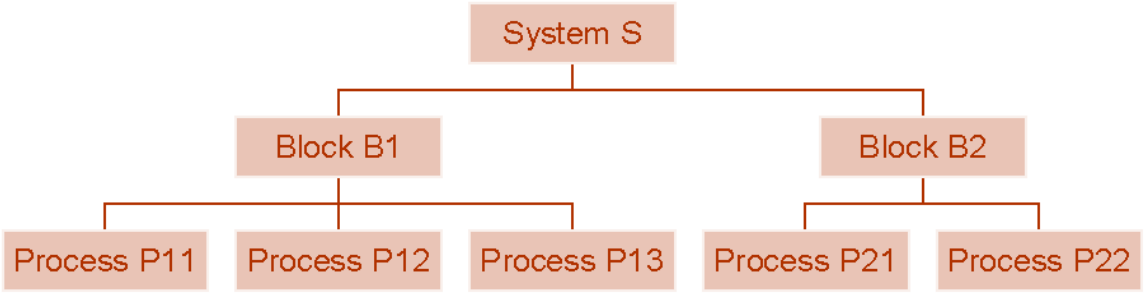
Can be used for design optimization

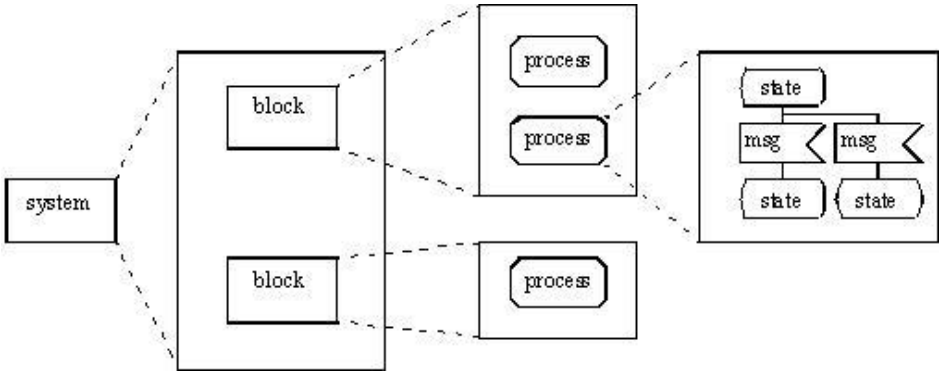
- Provides programmers with an easy way to do verification, validation of the design

SDL STRUCTURE

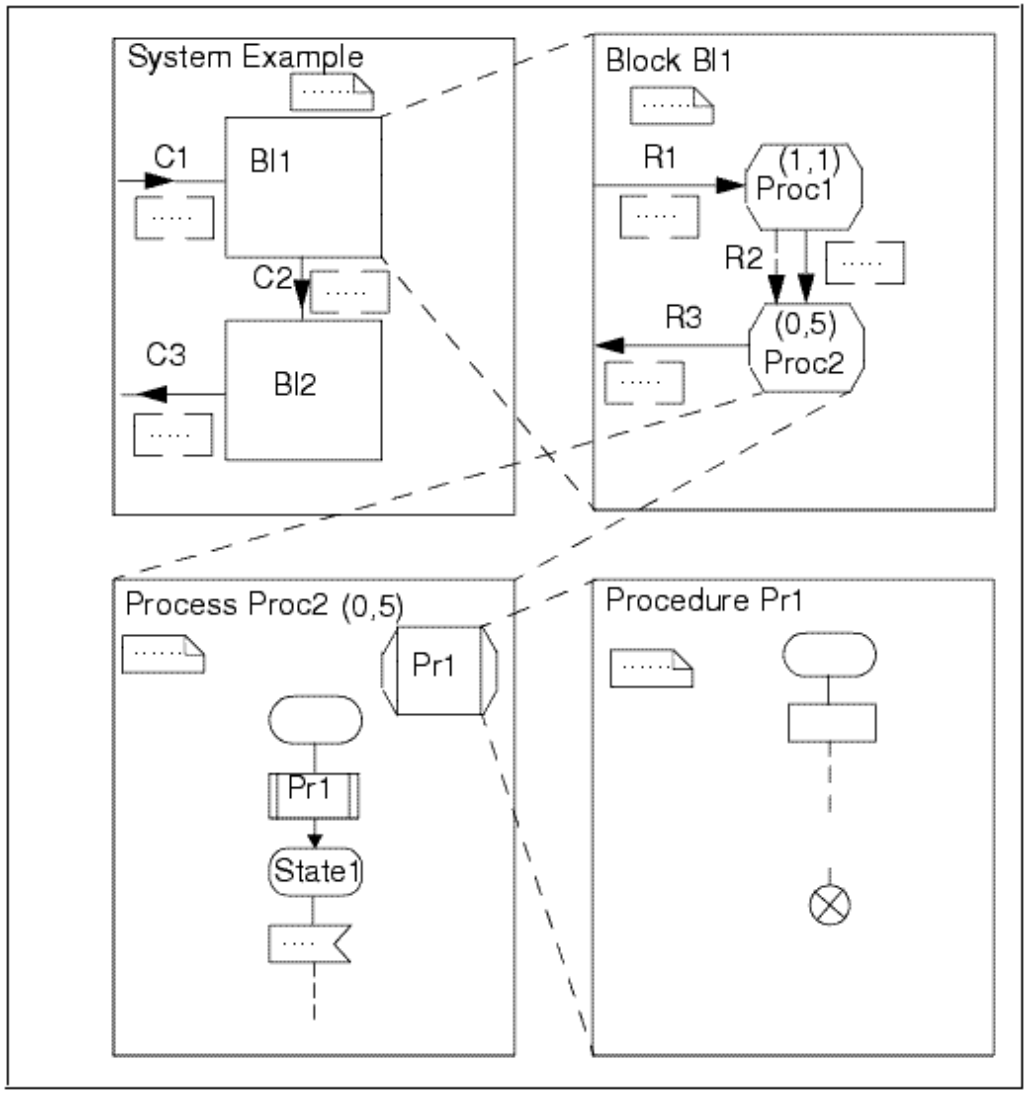
Comprises three main hierarchical levels:

- System
- Block
- Process





SDL STRUCTURE



SDL STRUCTURE

A system contains one or more blocks, interconnected with each other and with the boundary of the system by channels

- Processes communicate with each other using signals sent over the channels

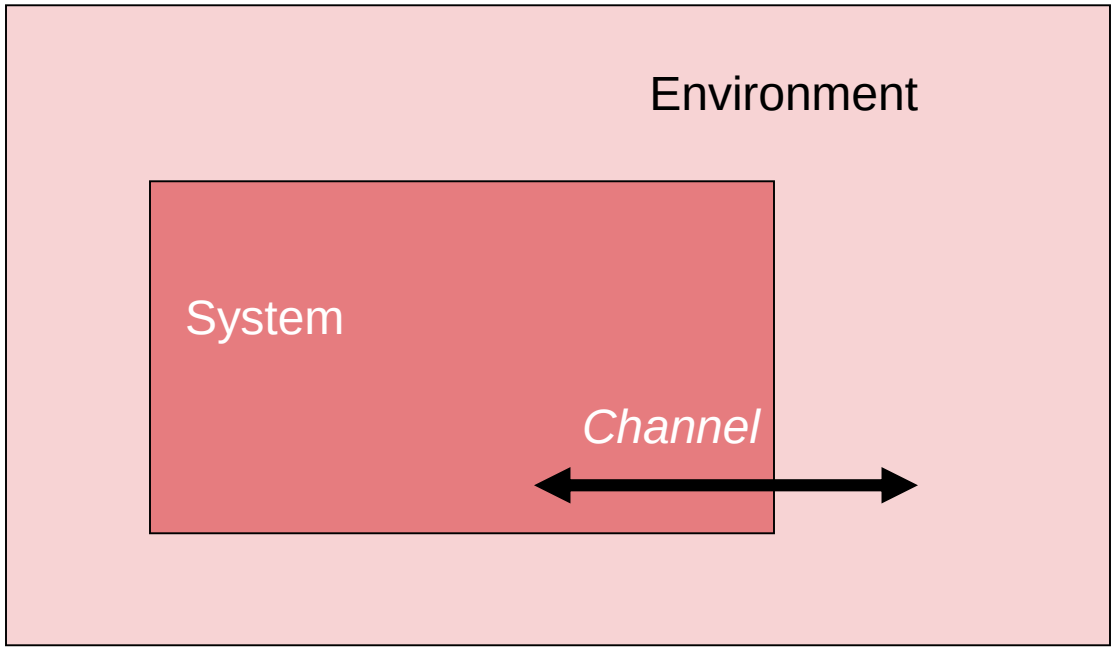
The block is the main structural concept

A block can be partitioned into sub-blocks and channels

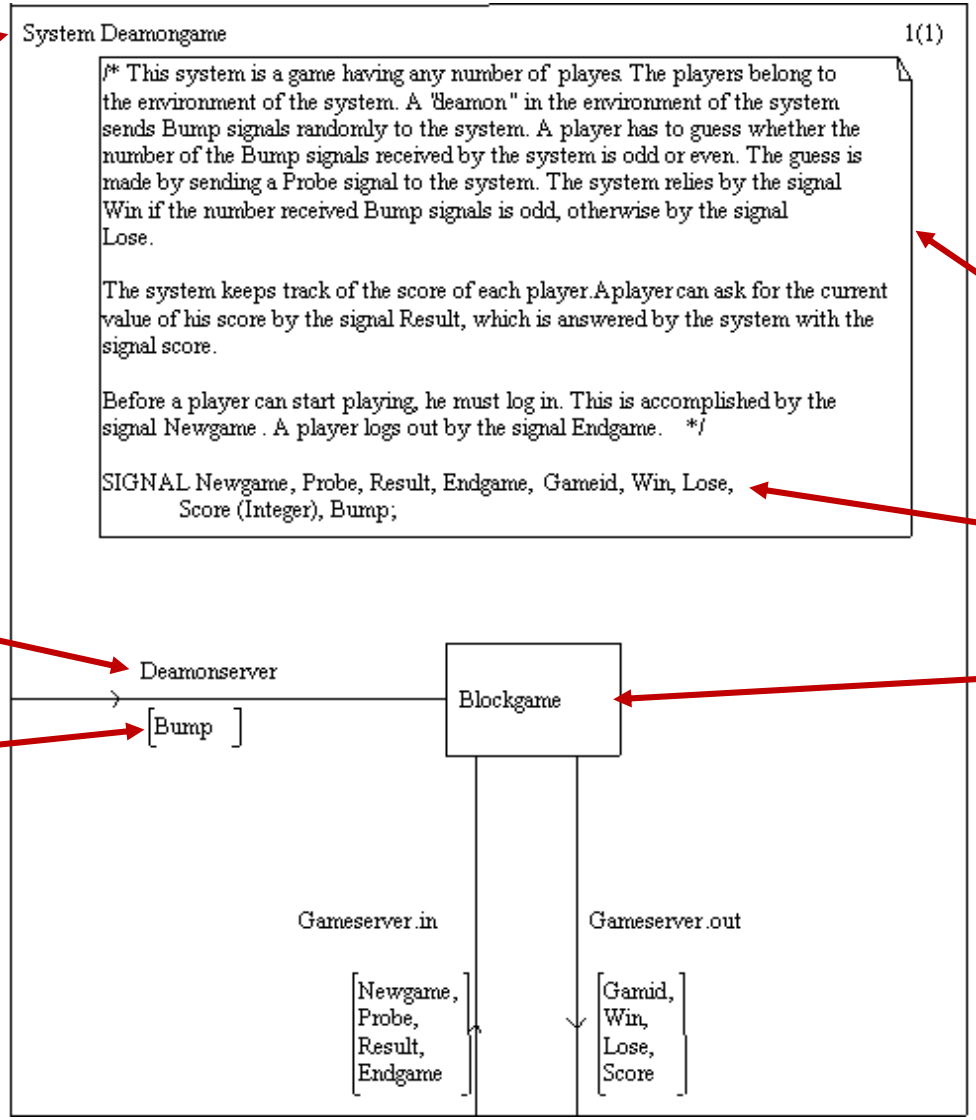
A channel is a means of conveying signals

SYSTEM

The system represents an an abstract machine communicating with its environment



SYSTEM



System name

Text symbol

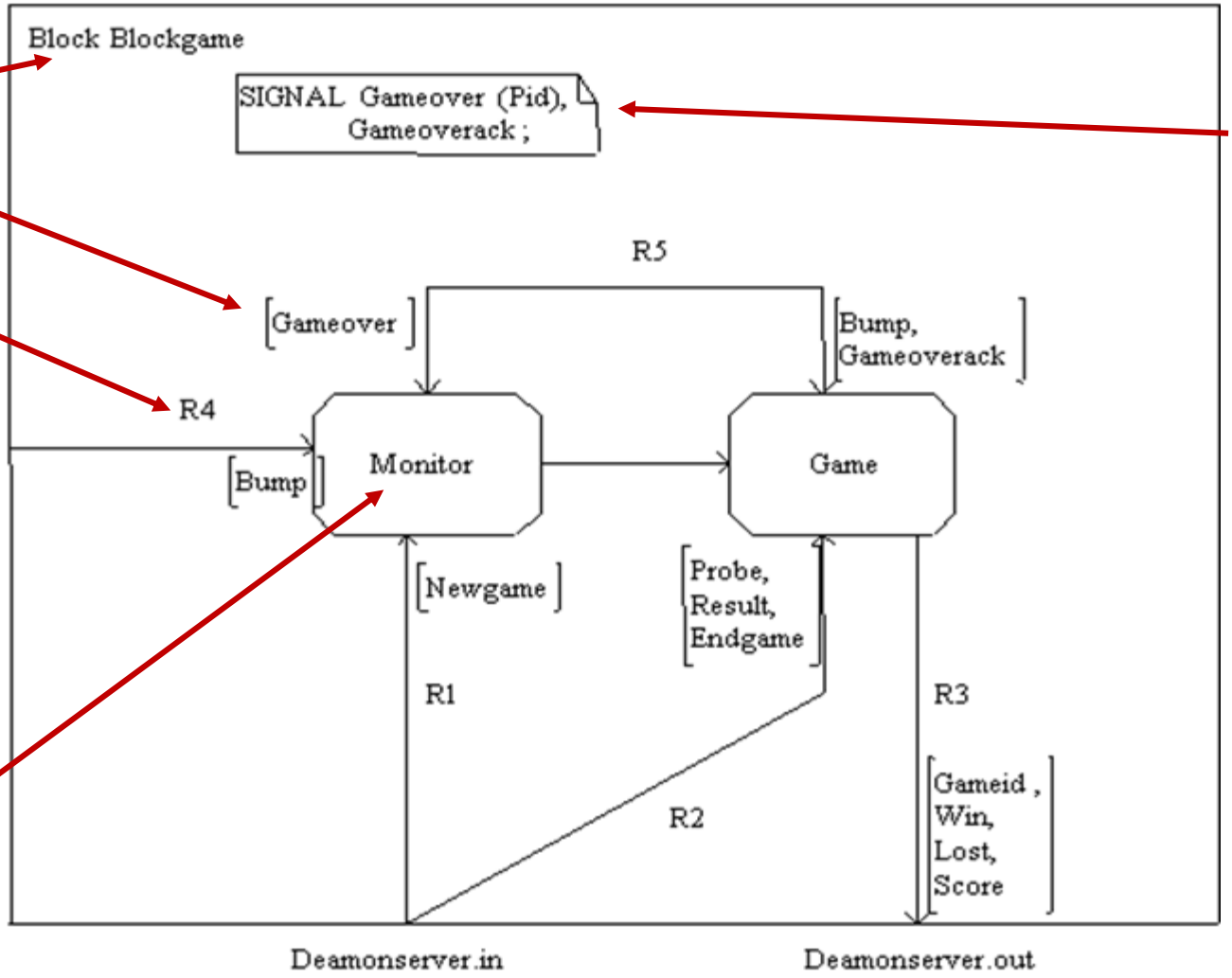
Signal declaration

Channel

Block

Signal

BLOCK



Block name

Signal

Signal route

Channel to route connection

Process

Signal declaration

PROCESS

A process in SDL is an extended finite state machine

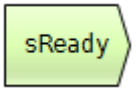
The behavior of a finite state machine is described by states and transitions

A process description is given through a process diagram

The following are the basic constructs used for the description of a process:

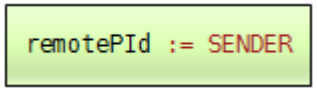
Process start

State

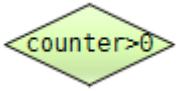


Send Signal

Receive Signal



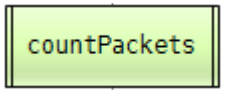
Task



Decision

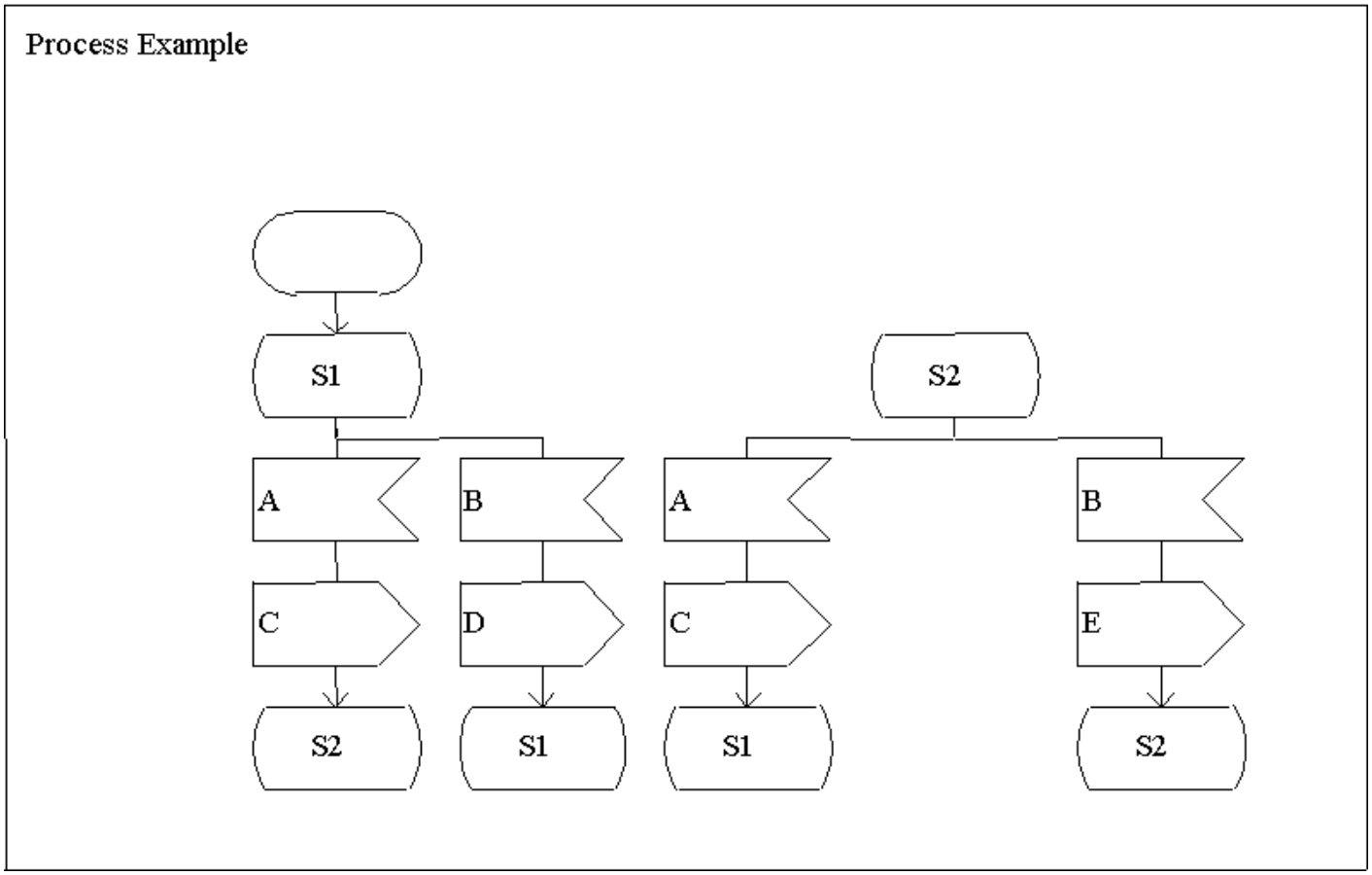
Process or procedure termination

Dynamic process creation



Procedure call

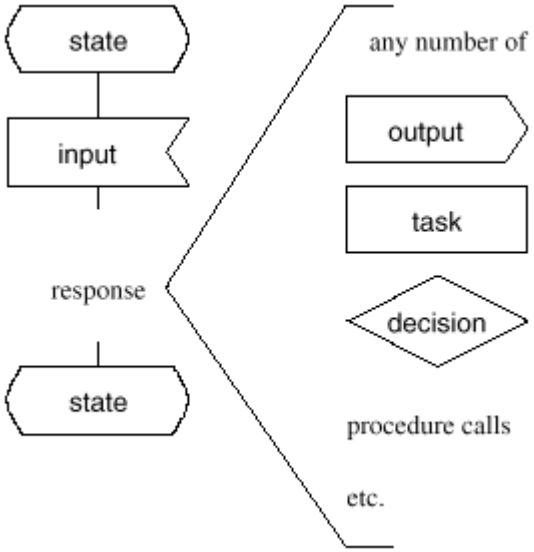
PROCESS



FORM OF PROCESS DIAGRAM

uOttawa
L'Université canadienne
Canada's university

In general, each state transition has the following form



In a state transition a process sits in its current state until an expected input event is received.

PROCEDURE

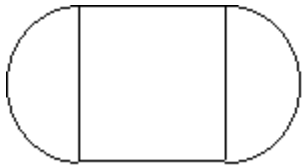
The procedure construct in SDL is similar to the one known from programming languages

A procedure is a finite state machine within a process

- It is created when a procedure call is interpreted, and it dies when it terminates

A procedure description is similar to a process description, with some exceptions.

The start symbol is replaced by the procedure start symbol



DESCRIBING BEHAVIOR WITH SDL

The behaviour of a system is constituted by the combined behaviour of the processes in the system

A process is defined as an extended finite state machine, that works autonomously and concurrently with other processes

A process reacts to external stimuli in accordance with its description

A process is either in a state waiting, or performs a transition between two states

DESCRIBING BEHAVIOR WITH SDL

The co-operation between the processes is performed asynchronously by discrete messages called signals

Every process has an infinite input queue, which acts like a FIFO queue

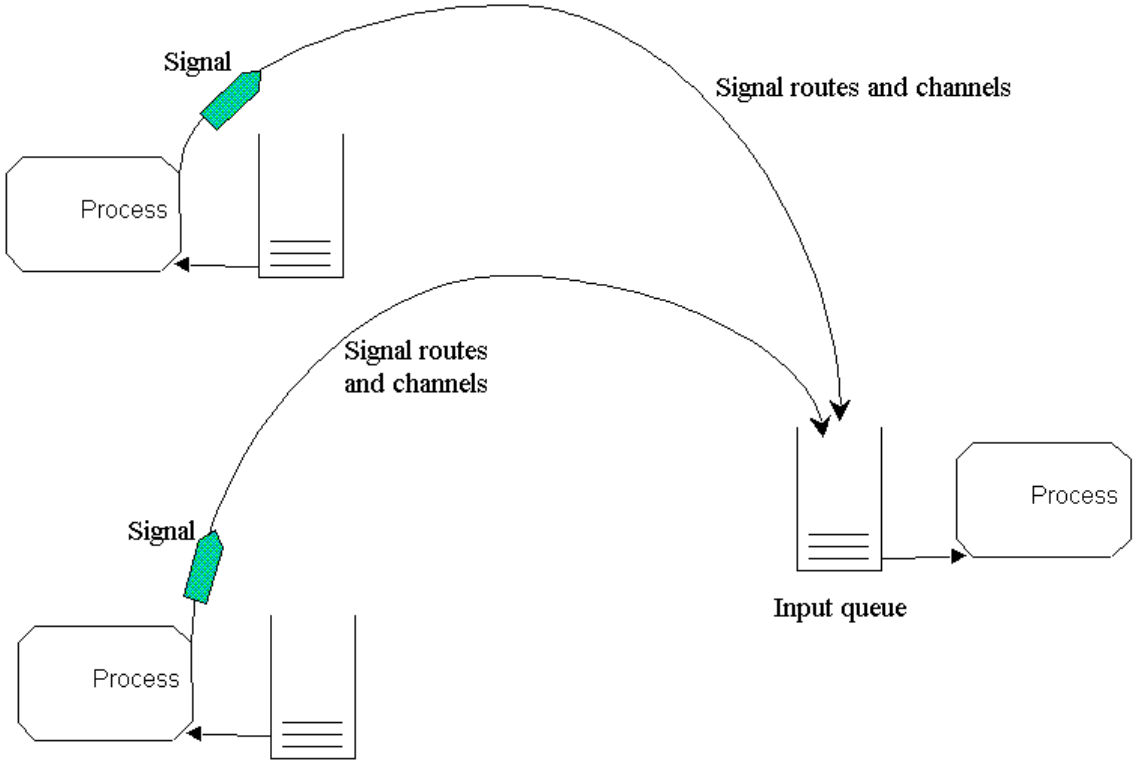
Any signal arriving at the process is put into its input queue

- The predefined *valid input signal set* defines those signals that the process is prepared to accept

DESCRIBING BEHAVIOR WITH SDL

uOttawa
L'Université canadienne
Canada's university

When a signal has initiated a transition, it is removed from the input queue



PROCESS VARIABLES

A process may locally use and manipulate data stored in variables

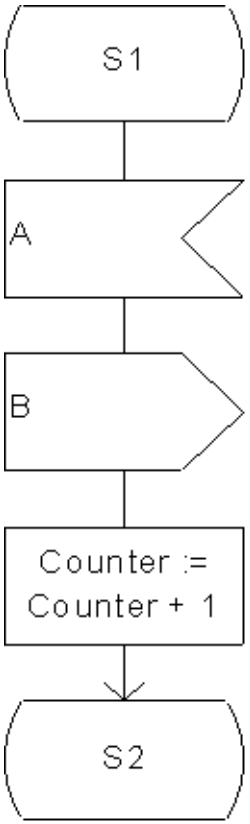
- These are defined, in a text symbol added to the process definition
- The keyword DCL introduces definition of one or more variables in a text box
- DCL stands for Digital Command Language

When there is more than one variable defined after DCL, they are separated by commas

- The definition list is ended by the semicolon

PROCESS VARIABLES

During a transition the process can use and manipulate its own local variables, using the task construct



PROCESS IDENTIFIER

Every process instance has a unique ID

The ID is not determined by the modeler

- It is created by the SDL machine (i.e. program used for simulation) during process creation
- The SDL machine guarantees that every process gets a unique ID

Process instance ID is not identical to the process name

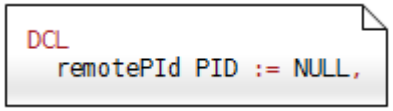
- There maybe many instances of a single process

PROCESS IDENTIFIER

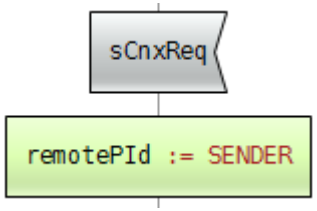
To refer to a process instance, you can use the following keywords

- SELF: returns the ID of the process instance itself
- SENDER: returns the ID of the process instance from which the last consumed signal has been sent
- OFFSPRING: returns the ID of the process instance that has been most recently created by the process
- PARENT: returns the address of the creating process

Process instance ID can be assigned to variables of PID



Declare variable of type PID



Assign SENDER PID to the declared variable (save it for future reference)

SIGNAL ADDRESSING

For any signal sent by a process there must be one and only one destination

Destination can be specified:

- Implicitly
- Explicitly

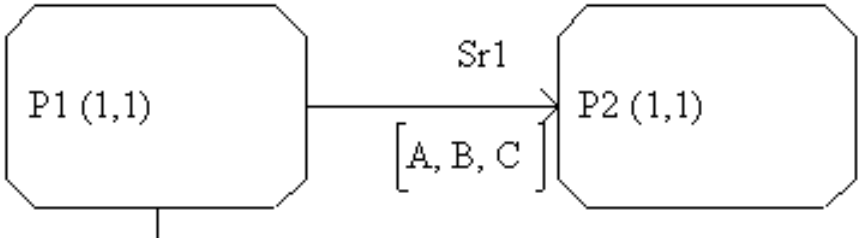
EXPLICIT ADDRESSING

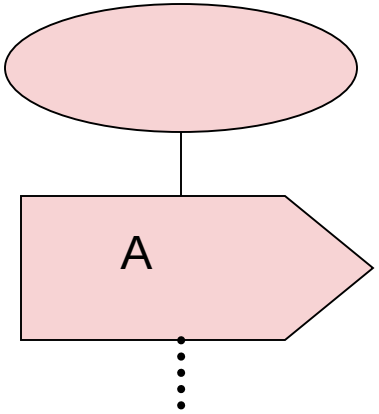
SDL has the TO construct for the explicit addressing of processes

The keyword TO is used in an output, and it is followed by an expression containing the ID of the destination process

IMPLICIT ADDRESSING

The explicit specification of a destination address is not necessary if the destination is uniquely defined by the system structure



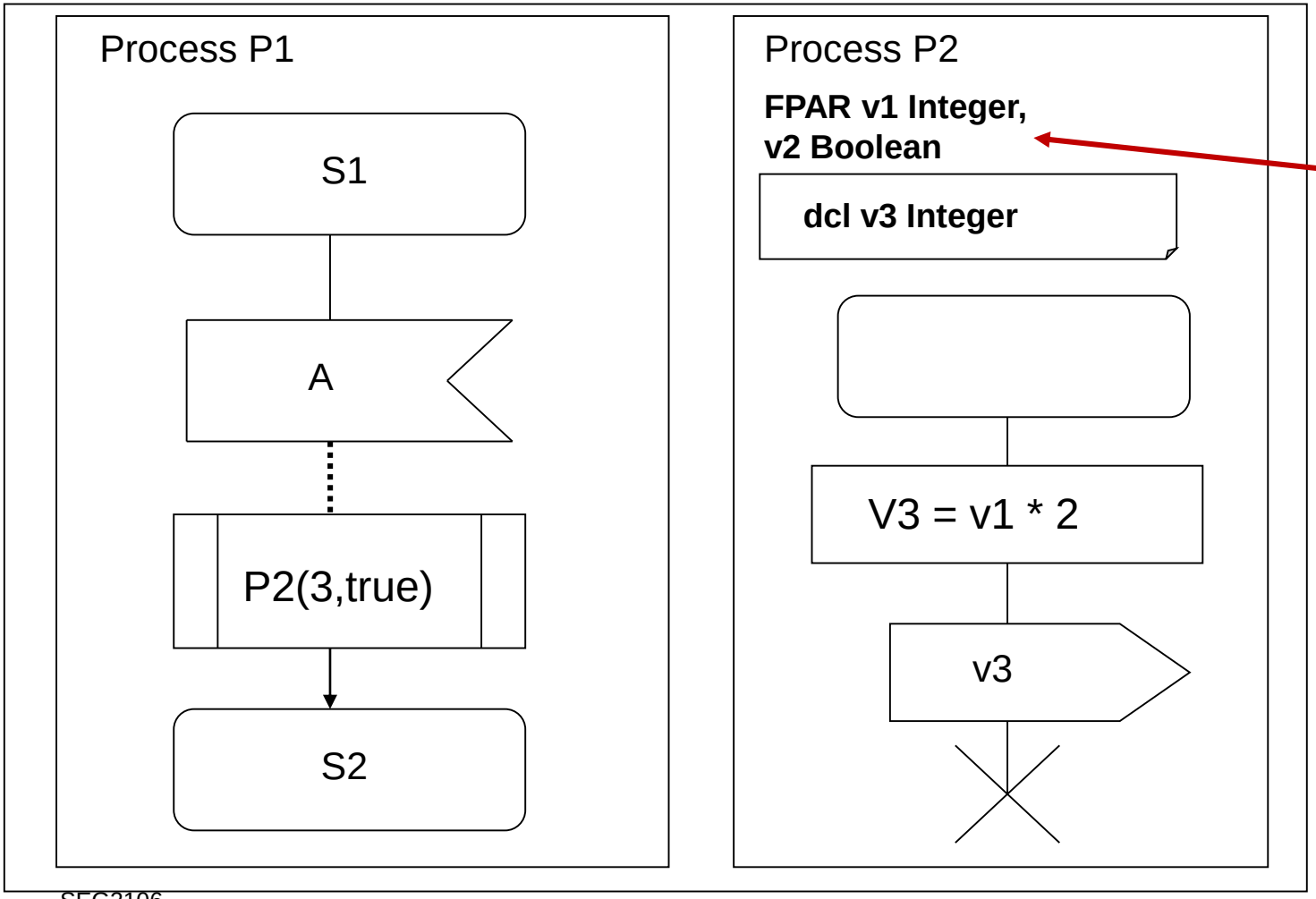


PROCESS CREATION/TERMINATION

Processes can be created by other processes dynamically at interpretation time

The creating and created process must belong to the same block

PROCESS CREATION/TERMINATION



SMART HOME EXAMPLE

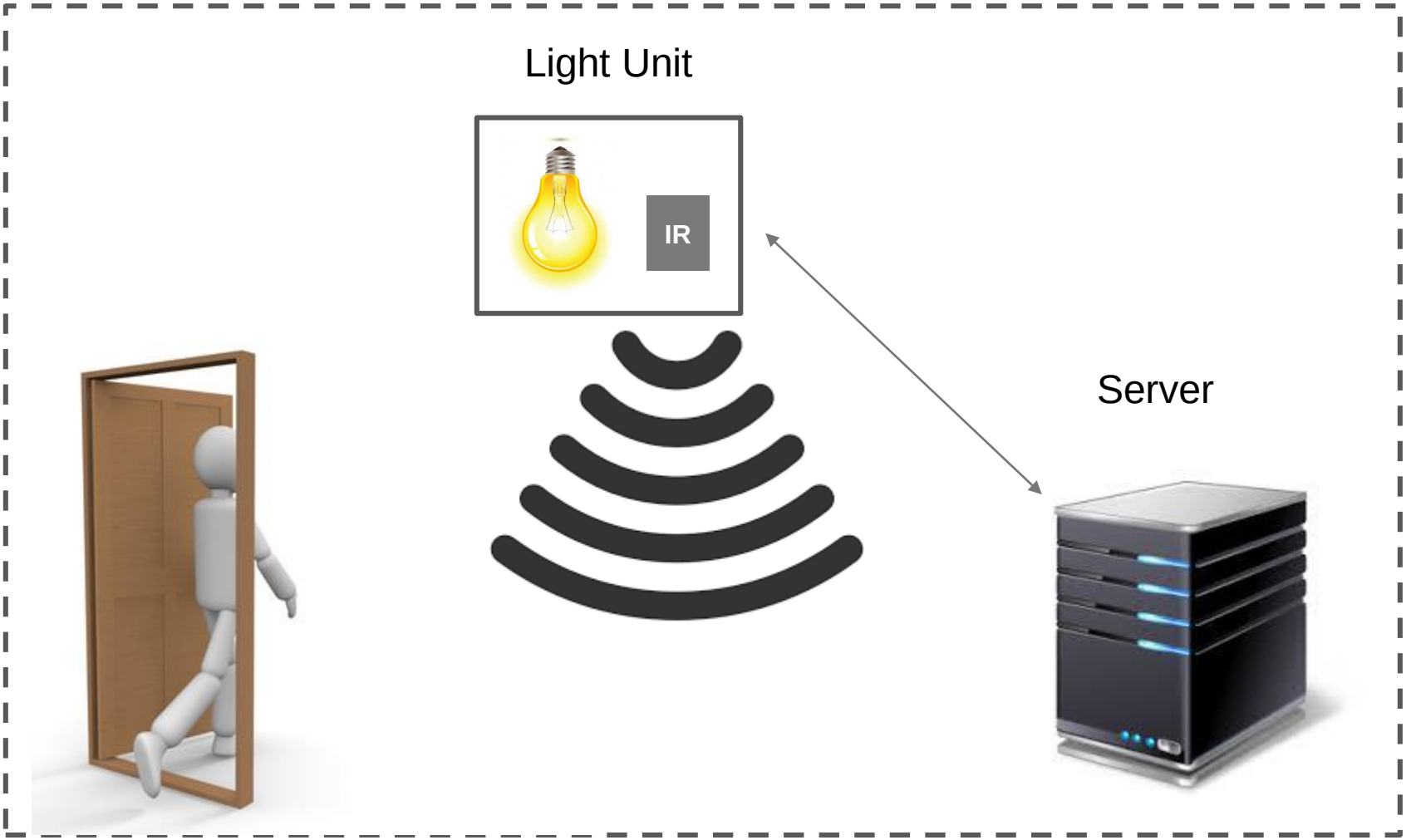
We will model a small module of a smart home system

In a smart home, a central process running on a server controls various smart objects within the home

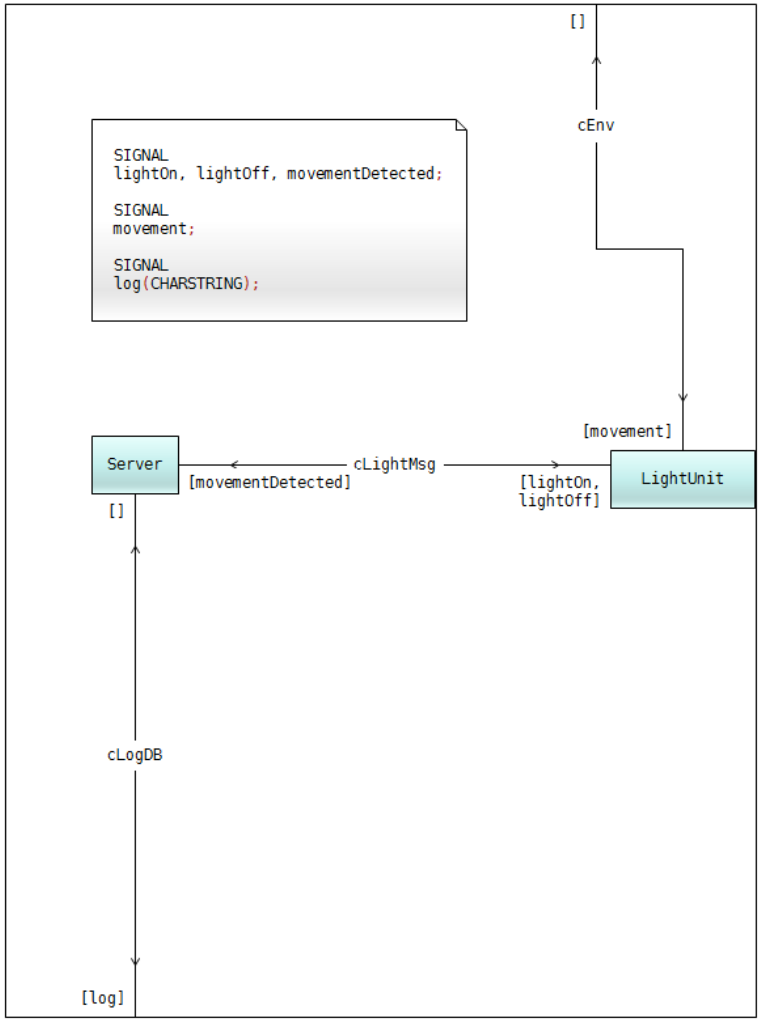
We will model the interaction between the server process and a light unit

- The light unit automatically turns on when someone enters the room
- The light unit has an infra-red sensor that detects movement
- The light turns off if no movement is detected for 5 seconds
- The server is saving log messages related to its operation to a DB outside the boundaries of the system

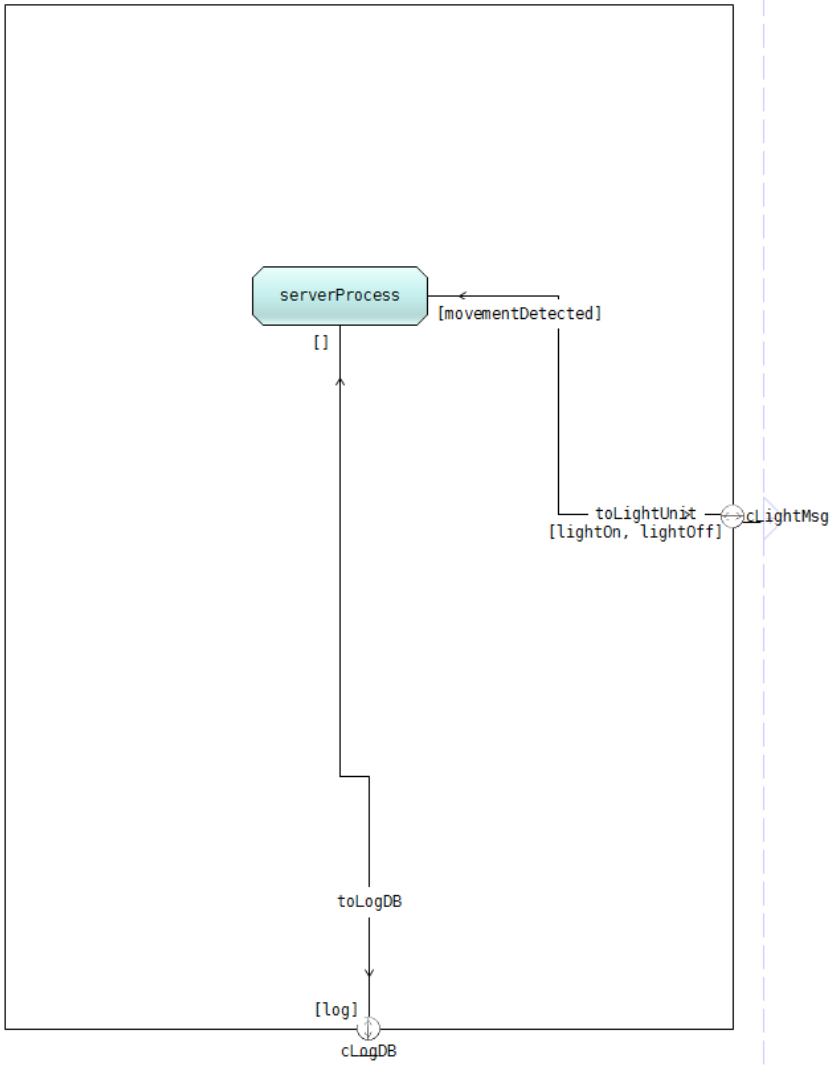
INTERACTION BETWEEN SERVER AND LIGHT UNIT



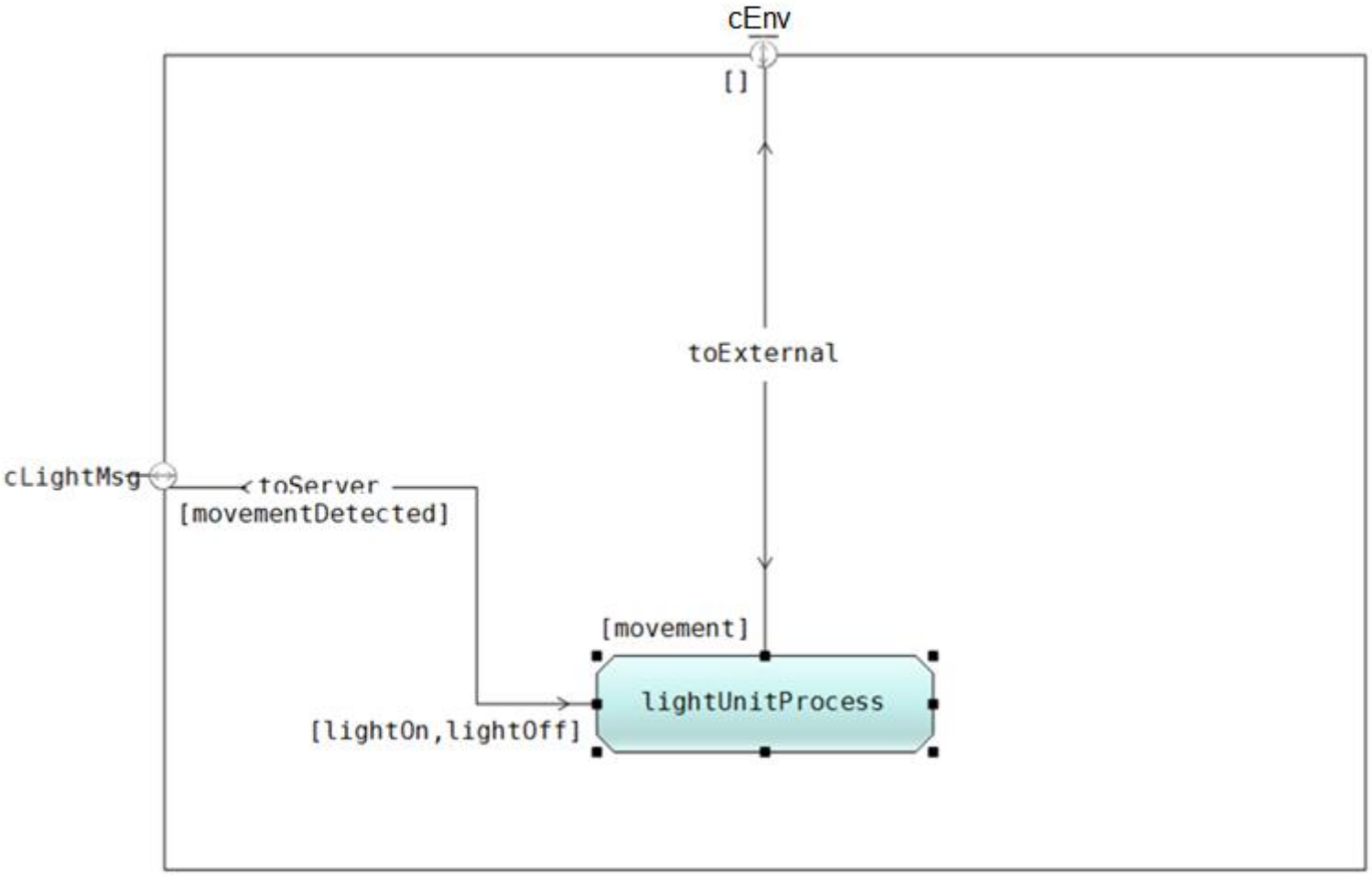
SYSTEM DEFINITION



SERVER BLOCK DEFINITION

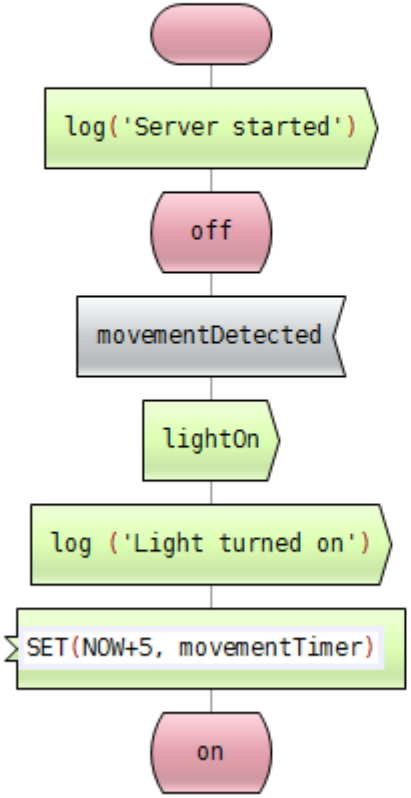


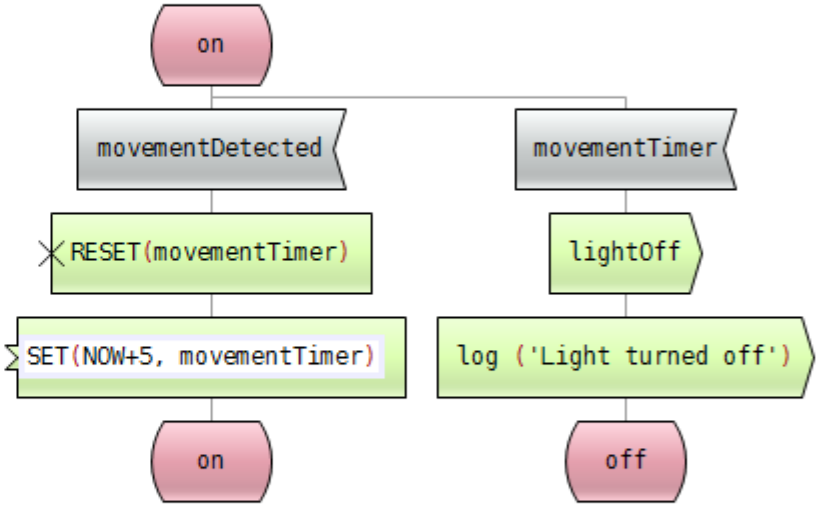
LIGHT UNIT BLOCK



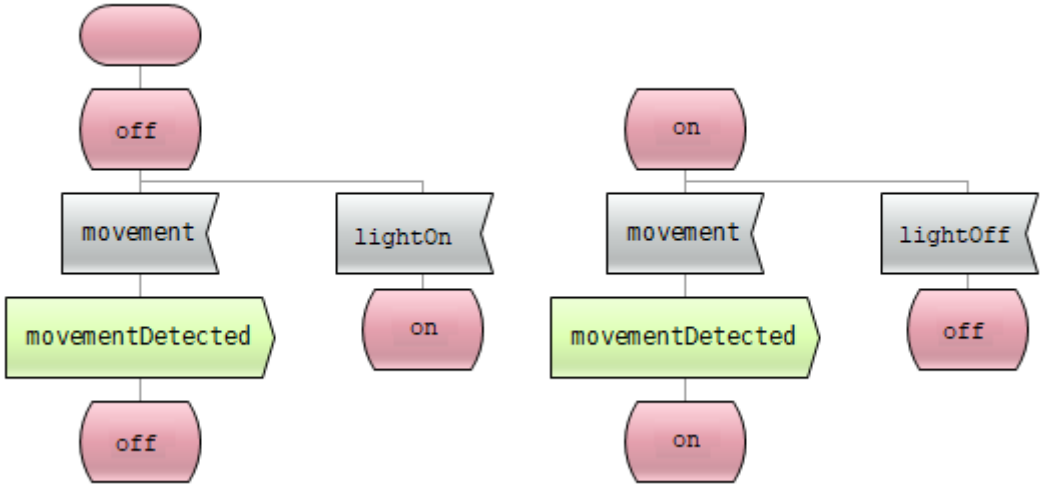
SERVER PROCESS DEFINITION

```
TIMER
movementTimer;
```





LIGHT UNIT PROCESS



THANK YOU!

QUESTIONS?