

LABORATORIO DI INGEGNERIA DEI SISTEMI SOFTWARE

Introduction

Software System project using distributed **QActorK**. Development of the theme **ES0**.

Requirements

Design and build a software system (named **workshift**) that allows to manage a machine according to three turns:

1. in the **first turn** (in the morning) the machine is able to handle messages (dispatches) of type **m1:m1(V)**
2. in the **second turn** (in the afternoon) the machine is able to handle messages (dispatches) of type **m2:m2(V)**
3. in the **third turn** (in the night) the machine 'sleeps'

Messages of types m1 and m2 can be sent by external entities at every time.

Requirement analysis

After interviewing client, meaning he associates with nouns have been clarified:

- **Turn**: there are three turn that represent different moment of the day. Every turn define the behaviour that machine in which is installed the software system has to assume.
- **Handle**: in the requirements are not defined specific actions that the machine assume during message handling. In this case, we have assumed that in case a message has to be handled, it will be printed.
- **Message**: a simple message with the form specified in requirements **m1:m1(V)** or **m2:m2(V)**. Messages can be sent at every time.
- **Sleeps**: during this period, the machine do nothing.
- **Morning**: period of the day that start around **06:00** and ends around **13:00**.
- **Afternoon**: period of the day that start around **13:00** and ends around **19:00**.
- **Night**: period of the day that start around **19:00** and ends around **06:00**.
- **Externarl Entities**: a machine (or a group of machines) that is separate from the internal machine (on which is installed the software system). This machine have the task of send message, of m1 or m2 type, to internal computer.
- **Send**: we only know that the external entity have to send message to the internal computer, but is not specified specified the way with which the message will have to be sent.

The hours that define moments of the day were decided by my personal interpretation.

A first user story.

When the system starts up, according to the time of day the machine is in, a specific turn is activated. Let's take it as an example, that turn at the start of machine is that of **morning**. The machine in the **morning** is able to handle message of **m1** type. Messages are sent by External entities at any moment of day, and those could be of **m1** or **m2** type. If at certain moment a message of **m1** type arrive, machine that is in the **morning** state, can immediately handle this according to the decided behaviour. If instead there is a message of type **m2**, this is saved in a queue and it will be handled after in the specific turn.

At the end of turn (if we are in morning turn, the end will be at 13:00) there will be a transaction of turn. Then machine will pass from the morning turn to that of afternoon. In the same way the machine will have a transaction at the end of afternoon, when it will pass to night turn.

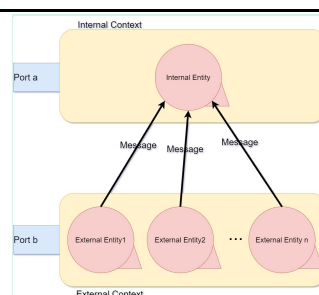
Problem analysis

Aspect noted

- There are three different period of the day, **Morning**, **Afternoon** and **Night**. Any of this periods can be intended as different states in which our machine can be during the day.
- It's not specified the specific moment in which the machine will be started. For this reason, at startup the machine will take the state according to the time of the day in which it is.
- Messages are sent from external entities without any kind of request by the internal entity, then those will be sent with **Dispatch** mode (Fire and Forget).
- Considering that messages are sent at any moment and without regulation on which type by the external entities, those will be placed in a queue and handled in their specific turn.
- For this project is requested the use of the QActorK.

Logical Architecture

- This software system is distributed. The **internal entity** is situated in a context that we assume as **"Internal"**. In this context is situated the machine that have to change on the three



described turn, and have the goal to handle the received messages.

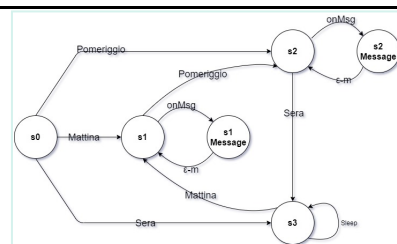
- The **external entities** instead are situated in another context that we have called "**External**". In the picture on the side is represented an architecture with only one external context. But this is not specified explicitly, then could exist more than one external context. Every context can contain more entities. Those entities have the goal of sending messages to the internal entity at every time with no specified regulation.
- Messages that are sent have this kind of syntax: **m1:m1(arg)** or **m2:m2(arg)**. **arg** indicates the content of the message. When those arrive to internal machine, it's not for sure that will be handled immediately. For this reason those are placed in a queue and handled in their specific turn.

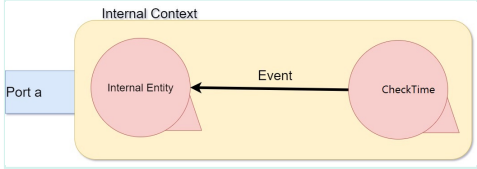
State Model

At the application startup the machine is in state **s0**. In this state it's checked the actual hour, on the basis of which will be decided the correct state.

- Morning turn (**s1**)
 - Morning turn message (**s1 Message**)
- Afternoon turn(**s2**)
 - Afternoon turn message (**s2 Message**)
- Night turn (**s3**)

The Internal Entity actor waits for a message, if none have arrived yet, or check the queue. Then passes in



the state in which messages are handled. At the end it returns to control state.	
<p>The change of turn (s1→s2, s2→s3, s3→s1) is decided by the arrive of an event. For this reason it's necessary to introduce an another entity that help the machine.</p> <p>This new entity is internal (placed in the Internal Context), and cyclically check the current hour. When the actual time corresponds to the start time of a state, this entity send and event to the machine which switchs to the next state.</p>	 <pre> graph LR subgraph Internal_Context [Internal Context] Internal_Entity((Internal Entity)) CheckTime((CheckTime)) CheckTime -- Event --> Internal_Entity end Port_a[Port a] --- Internal_Entity </pre>

Test plans

Project

Testing

Deployment

Maintenance

My git repo: https://github.com/LeoManto/Mantovani_Leonardo

By Leonardo Mantovani email: leonardo.mantovani2@studio.unibo.it

