University of L'Aquila

Department of Engineering and
Information Science and Mathematics

Report Homework #2

Water Distribution, Leakage and Quality Control System

Professor

Vittorio Cortellessa

Students

Gaetano Fichera & Giovanni Lezzi

Github Project Repository:

https://github.com/GaetanoFichera/Water-Quality-Control-System

Academic Year 2017-2018

Index

Chapter 1

Our Homework

The task in this second homework is to create a Performance Model based on the system that we have modeled in the first homework. Using the Sequence Diagrams, we built the Execution Graph, and, through the Deployment Diagram we have built a Queueing Network. From the analysis of the Execution Graph we obtained the Demand Vectors that we used to parameterize the Queueing network. After that, the network has been resolved through a tool in order to obtain the output parameters that would allow to evaluate the system performance. In the final phase through the Architectural Description Language Aemilia we had the possibility to have a further evaluation of the system's performance.

Chapter 2

Work Planning

The workflow was:

- Light rework on our UML Model;
- Study of theoretical concepts of Queuing Networks and how to create the Execution Graph from Sequence Diagram;
- Connection between Deployment and Component Diagram;
- Drafting of the Executiong Graph;
- Calculation of the consumption of physical resources in an overhead matrix;
- Drafting of the Queueing Network;
- Performance Analysis and evaluation, and Refactoring of the Queueing Network Model;
- Study of Aemilia;

- Drafting of the State Diagrams for the implementation in Aemilia;
- Drafting of the Flow Graph;
- Performance analysis in Aemilia;

Chapter 3

Light Rework On Our UML Model

After serious consideration of our last homework, we noticed that there were failures. Below is a list of rework:

- We have applied the stereotypes to the Communication Paths within the Deployment Diagram;
- in order to improve system performance and to better stratify the deployment, we decided to add 3 new nodes:
  - "SeaweedPickingInlandControlUnit";
  - "Seaweed Picking Outgoing Control Unit";
  - "MagikarpControlUnit".

  and , for this purpose we have also inserted a new profile called "Control Unit" from the server stereotype, these 3 new nodes are connected to the server with a wired connection and to sensors with a wireless connection;
- For the connections between the Control Center Server and the two Inland and Outgoing Seaweed Picking Control Units we have inserted a new communication path stereotype that does not consume resources because when we started the modeling work for the Queueing Network we considered the two Control Units located in the same physical space as the Control Center Server, then we use a "Internal Connection" stereotype with zero consumption of resources;
- We have added the operations to the components subsequently reused in the Sequence Diagram;
- we realized the lack of an internal server to the Control Center Pomezia that managed the requests coming from the App inside it;
- we realized that in the sequence diagram "check quality" the call from the component "check quality parameters inland / outgoing" was missing to the component "parameters Quality archieve" to retrieve the desired water parameters;
- we have established the types of connections between one node and the other of the deployment diagram, obtaining:
  - between the Control Center Server and the two Apps a Wired Connection;
  - between Control Center Server and Seaweed Picking a Wireless Connection;
  - between the Control Center Server and the Water Company Server and the Purification System Center an Internet Connection.

- we have agreed that there is a single database that is connected to the water company server or water archives;
- we realized that in the water sampling phase, the sensors will send the data of the water samples to the Control Center Server which, in turn, using the sample archieve component will send the data to the water company server, the problem is that in the SD after the component sample archive is not invoked any component that refers to the water company server. Solution:
  - We have decided to add a component called "Sample Archive" to the water company server which is responsible for saving data on the DB. In going to add this correction we realized that in fact we have failed to use sample sender. Then we have made a small change:
    - sample sender is inside the control center server;
    - sample archive is located inside the water company server and manages the data on the db.
- We modified the sequence diagram of "SturtUpSamplingWater" as we realized that the component sample data on the SeadweedPIckingInland / Outgoing node communicated with the "SampleSender" component on the ControlCenterServer node but from our component + deployment diagram it was not.

Below you can find the description of the new profiles popping out.

3.1 Wired Connection Profile

Wired Connection	
Metamodel Class	Communication Path
Description	It is a representation of the physical meaning of Wired Connection
Tagged Values	
Constraints	

3.2 Internet Connection Profile

Internet Connection	
Metamodel Class	Communication Path
Description	It is a representation of the physical meaning of Internet Connection
Tagged Values	
Constraints	

3.3 Wireless Connection Profile

Wireless Connection	
Metamodel Class	Communication Path
Description	It is a representation of the physical meaning of Wireless Connection
Tagged Values	
Constraints	

3.4 Internal Connection Profile

Internal Connection	
Metamodel Class	Communication Path
Description	It is a representation of the physical meaning of Internal Connection
Tagged Values	
Constraints	

3.5 Control Unit Profile

Internal Connection	
Metamodel Class	Node
Description	It is a representation of the physical meaning of Control Unit
Tagged Values	
Constraints	

Chapter 4

Use Cases Decision

In building the Performance Model we considered only two use cases. To satisfy the Homework request the Use cases choosen are:

- StartUp Sampling Water activated by Sample Supervisor;
- Check Water Quality activated by Quality Control Supervisor.

Chapter 5

Identification Of Performance Requirements

The following consideration has been made:
one kilometer of the route with respect to the connection point with the system is taken into account for an Entry Water Channel or Exit of 5 meters radius, and every 10 meters must be 10 Seaweed Picking, with a total of 1000 Seaweed Picking.
Non-functional requirements are:

- Each sensor must take 100 ms to carry out a sampling;
- The time between a sampling and an other is 60 s;
- The time that each node must use to send the data to the archive is 200 ms;
- The time of use of each node must be less than 90
- The response time of an Actor Task must not exceed 300 ms.

Chapter 6

Development Of Component Diagram Into Deployment Diagram

In reference to the Use Cases taken into consideration, to better understand our architecture, we have combined the Deployment Diagram with Component Diagram. In the figure below this is represented.

Figure 6.1: Deployment Diagram + Component Diagram

Chapter 7

Model WCS With Execution Graphs

7.1 Demand Vector

Analyzing the sequence diagram, component and deployment diagram we have obtained this demand vector which represents virtual resources related to our architecture:

Wired Connection Request	Estimated consumption of the request to the Wired Connection
Wireless Connection Request	Estimated consumption of the request to the Wireless Connection
Internet Connection Request	Estimated consumption of the request to the Internet Connection
Database Request	Estimated consumption of the request to the Database
Control Center Server CPU	Estimated consumption of the request to the Control Center Server CPU
Water Company Server CPU	Estimated consumption of the request to theWater Company Server CPU
Purification System Pomezia CPU	Estimated consumption of the request to the Purification System Pomezia CPU
Seaweed Picking Inland Control Unit CPU	Estimated consumption of the request to the Seaweed Picking Inland Control Unit CPU
Seaweed Picking Outgoing Control Unit CPU	Estimated consumption of the request to the Seaweed Picking Outgoing Control Unit CPU
Seaweed Picking Outgoings Sample Request	Estimated consumption of the request to the Seaweed Picking Outgoing Sample
Seaweed Picking Inlands Sample Request	Estimated consumption of the request to the Seaweed Picking Inlands Sample

The Execution Graphs obtained are:

- Sampling Water activated by Sample Supervisor:

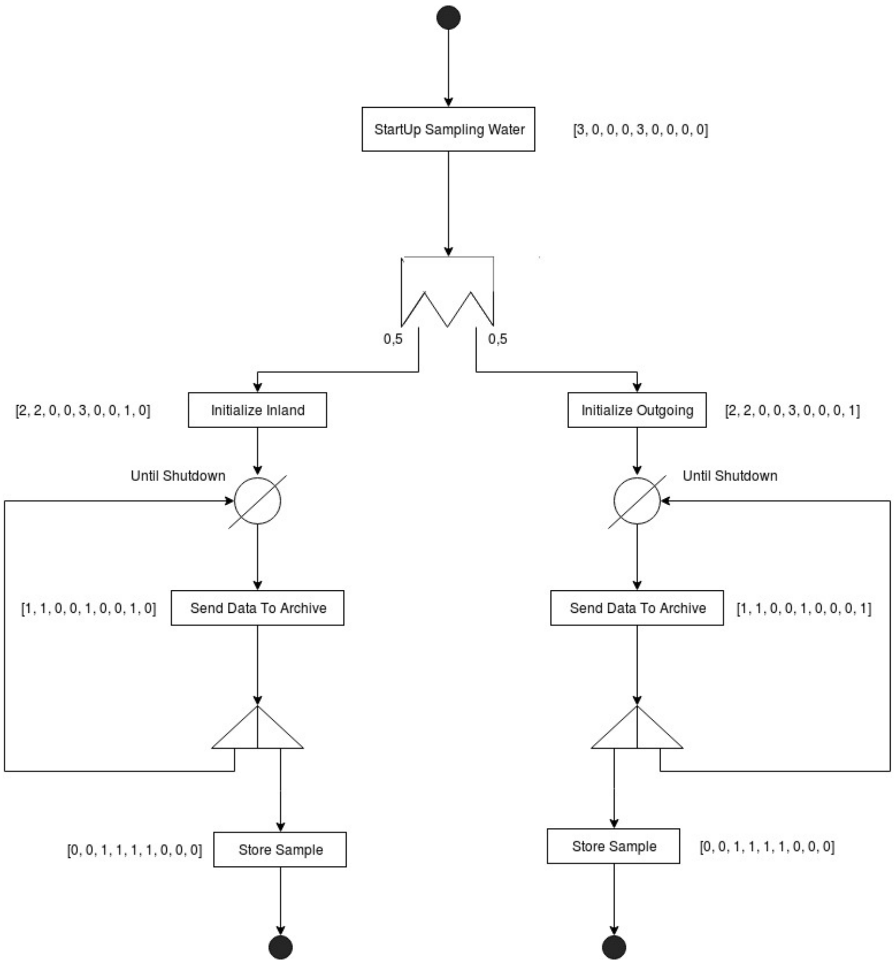


Figure 7.1: EG UC1 StartUp Sampling Water

- Check Water Quality activated by Quality Control Supervisor:

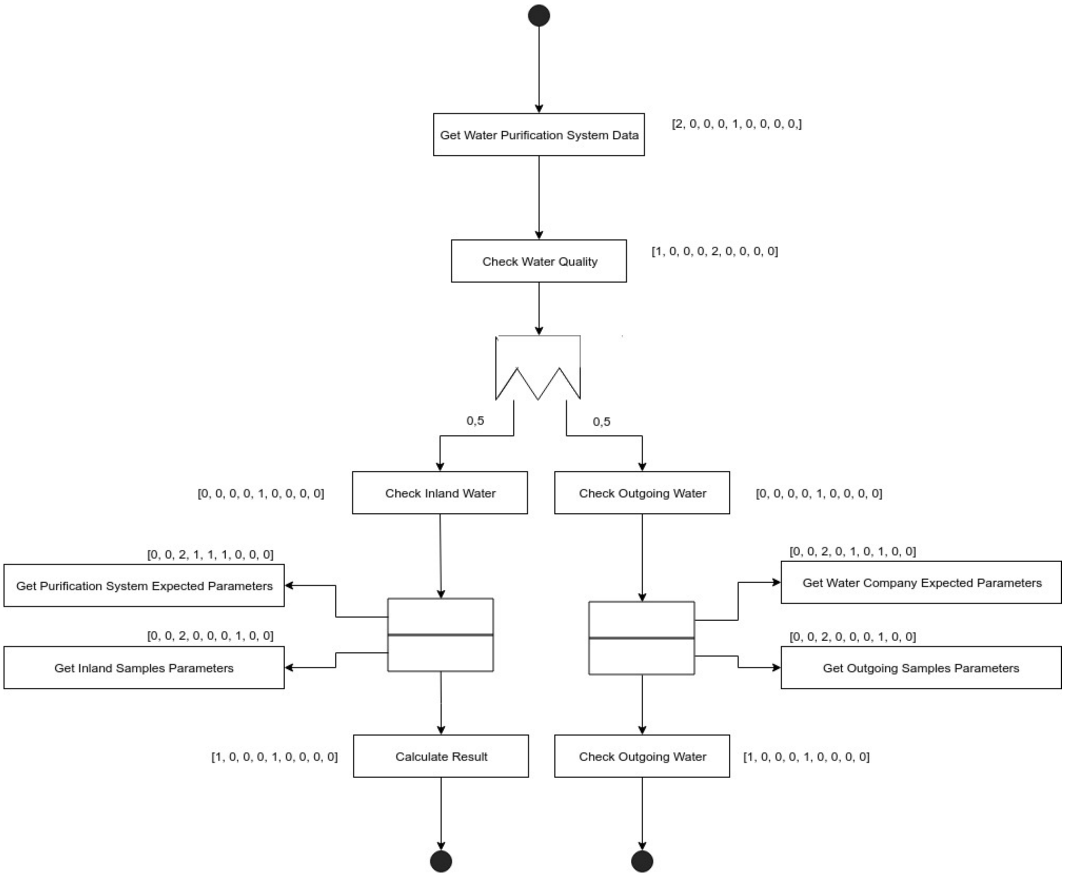


Figure 7.2: EG UC3 Check Water Quality

Chapter 8

Queueing Network Model

The jobs identified in our Queueing Network Model are:

- StartUp Sampling Water Inland;
- StartUp Sampling Water Outgoing;
- Check Water Quality.

The first two jobs refer to the same Execution Graph. In the diagram below you can see how the jobs interface with our system and how the nodes are interconnected with each other:

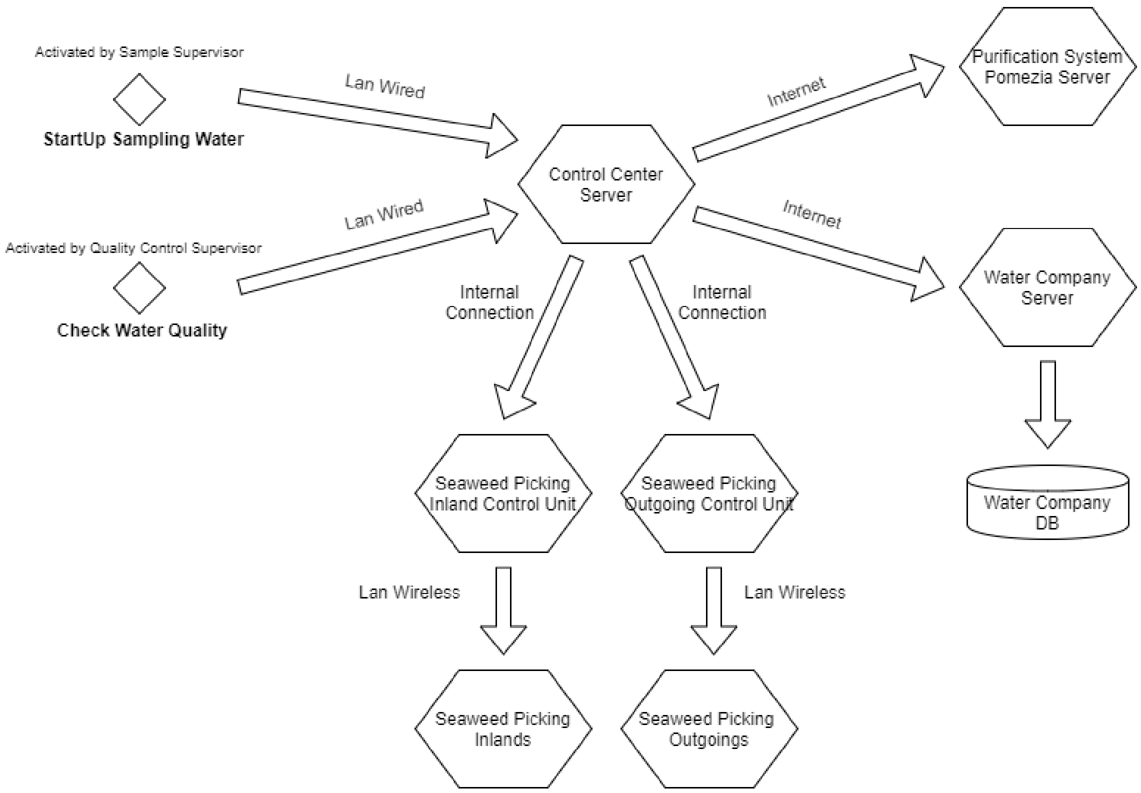


Figure 8.1: Physical Nodes

This is the Queueing Network so obtained:

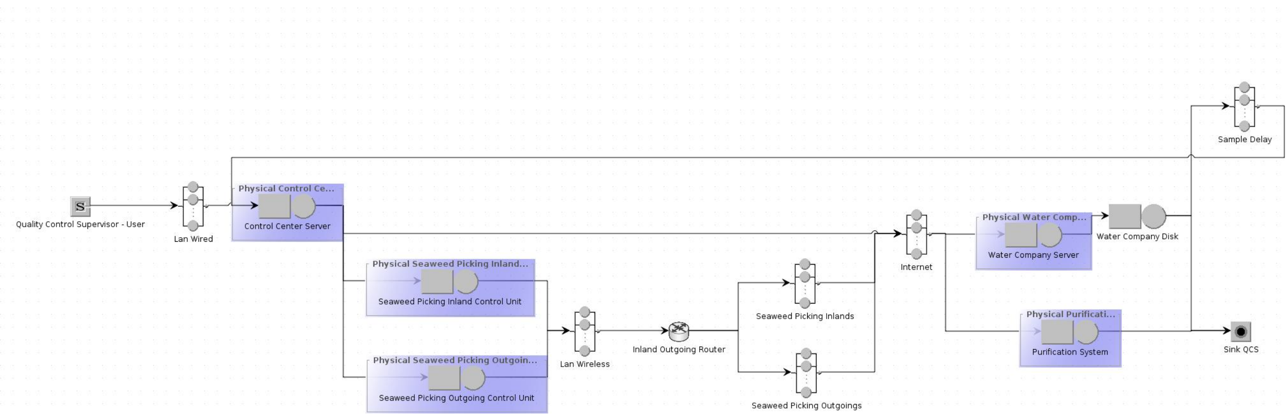


Figure 8.2: Queueing Network

Below physical nodes will be listed:

- Computational Nodes;
  - Control Center Server;
  - SeaweedPicking Inland Control Unit;
  - SeaweedPicking Outgoing Control Unit;
  - Water Company Server;

    - Purification System.

- Node that store data:
    - Water Company Disk.
- Delay station that simulates Network delays:
    - Lan Wired;
    - Lan Wireless;
    - Internet.
- Delay station that simulates the sampling:
    - SeaweedPicking Inlands;
    - SeaweedPicking Outgoing.
- Delay station that simulates the deterministic interval time between sampling of the same Seaweed Picking:
    - Sample Delay.

After several tests, we agreed to reduce the sampling time to 500 ms to refine the performance analysis. On JMT for the same reason we have only one SeaweedPicking Control Unit with doubled performance and no two for In / Out. We have also decided to eliminate the finite capacity regions as useless for the purposes of our project.
After other tests we have splitted each StartUp Sampling Water job into:

- StartUp Sampling Water;
- Sampling.

This choice was driven by the fact that we did not know how many times the samples were made, as you can see in EG UC1.

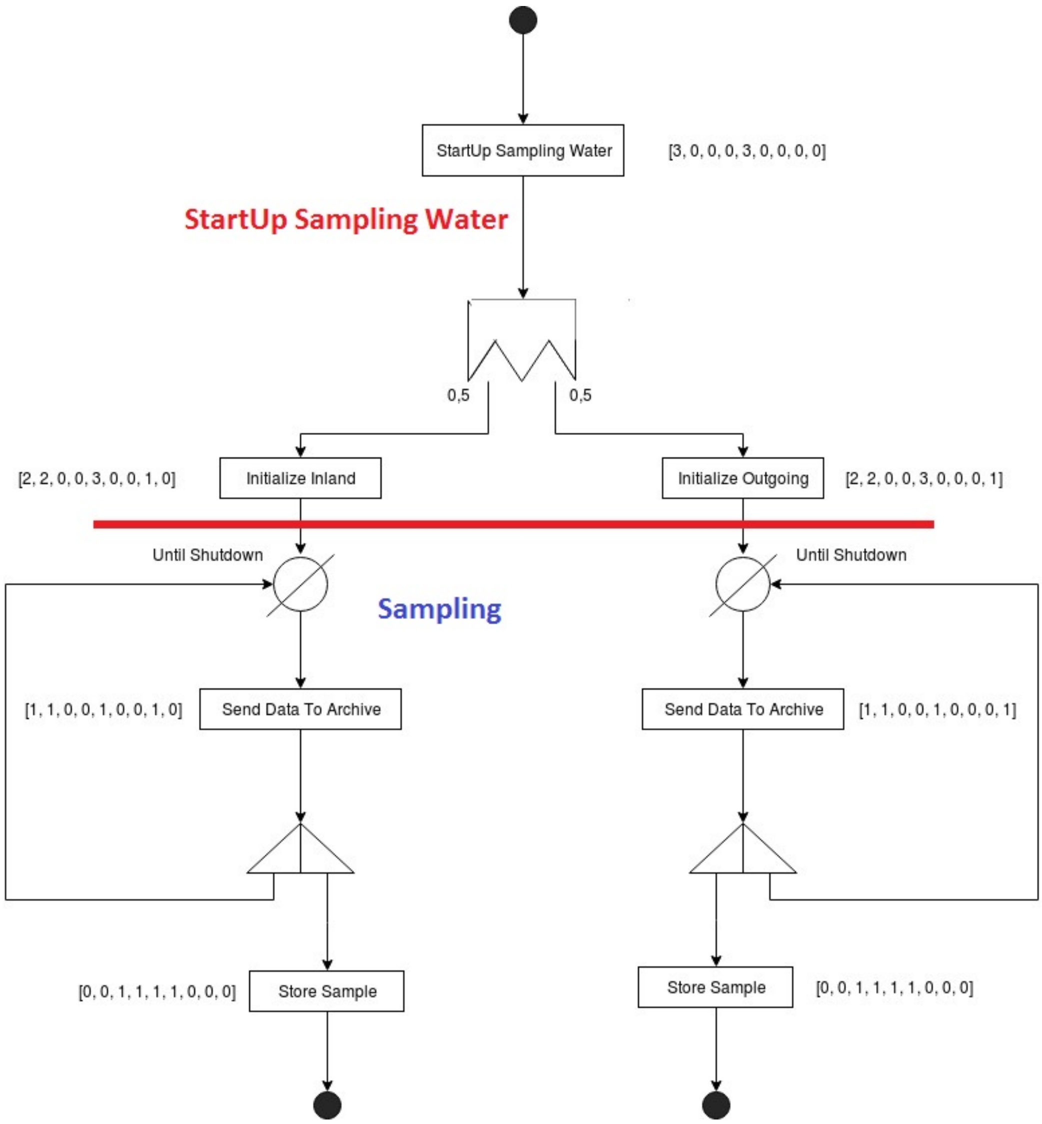


Figure 8.3: EG UC3 Check Water Quality

For this purpose a Class Switch Node is introduced where each StartUp Sampling Water becomes a Sampling job. So our final Queueing Network has become this:

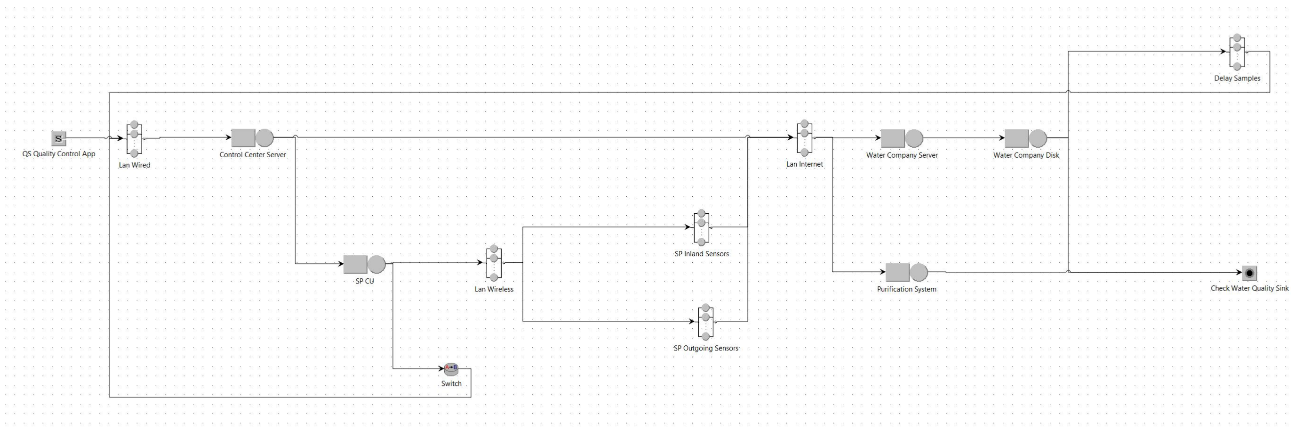


Figure 8.4: Final Queueing Network

Chapter 9

Parameterization

In order to have a mapping beetwen the consumption of our Physical Resources and Virtual Resources we identified a Overhead Matrix:

		Physical Resources										
		CC Server CPU	SP In Control Unit	SP Out Control Unit	SP In Sensor	SP Out Sensor	Water Company Server	Water Company Disk	Purification System	Lan Wired	Lan Wireless	Internet
Virtual Resources	Wired Connection Request	5	5	5	0	0	5	0	5	10	0	0
	Wireless Connection Request	6	6	6	3	3	6	0	6	0	10	0
	Internet Connection Request	10	10	10	0	0	10	0	10	0	0	5
	Database Request	10	10	10	0	0	20	50	0	0	0	0
	CC Server CPU	10	10	10	0	0	10	0	10	0	0	0
	Water Company Server CPU	10	10	10	0	0	10	0	10	0	0	0
	Purification System Pomezia CPU	10	10	10	0	0	10	0	10	0	0	0
	SP Ins Sample Request	15	15	15	13	13	0	0	0	0	0	0
	SP Outs Sample Request	15	15	15	13	13	0	0	0	0	0	0
	Per Operation (ns)	10	20	20	1000	1000	5	500	20	20000	28000	35000
	Per Operation (us) *NON MODIFICARE*	0,01	0,02	0,02	1	1	0,005	0,5	0,02	20	28	35
	Per Operation (ms) *NON MODIFICARE*	0,00001	0,00002	0,00002	0,001	0,001	0,000005	0,0005	0,00002	0,02	0,028	0,035

Figure 9.1: Overhead Matrix

In the last three rows of the table are specified the service time for Physical Resources to execute a basic operation. The next step is obtain the whole consumption of our Virtual Resources for each Execution Graph considering the meaning of each individual block:

	VIRTUAL RESOURCES EG 1									
		Virtual Resources								
		Wired Connection Re	less Connection Re	rnet Connection Requ	tabase Reque	CC Server CPU	ater Company Server CP	cation System Pomezi	P Ins Sample Reque	ts Sample Request
Blocks	StartUp Sampling Water	3	0	0	0	3	0	0	0	0
	Initialize Inland	2	2	0	0	3	0	0	1	0
	Send Data To Archive *IN*	1	1	0	0	1	0	0	1	0
	Store Sample *IN*	0	0	1	1	1	1	0	0	0
	Initialize Outgoing	2	2	0	0	3	0	0	0	1
	Send Data To Archive *OUT*	1	1	0	0	1	0	0	0	1
	Store Sample *OUT*	0	0	1	1	1	1	0	0	0
	SUM without Probability	9	6	2	2	13	2	0	2	2
	SUM with Probability	6	3	1	1	8	1	0	1	1

Figure 9.2: Virtual Resource Usage By Each Block

	PROBABILITY BLOCKS EG1	
Blocks	StartUp Sampling Water	1
	Initialize Inland	0,5
	Send Data To Archive *IN*	0,5
	Store Sample *IN*	0,5
	Initialize Outgoing	0,5
	Send Data To Archive *OUT*	0,5
	Store Sample *OUT*	0,5

Figure 9.3: Block Probability

Considering this last table with the overhead matrix we obtain a matrix containing the consumption of the physical resources related to the considered Execution Graph. This method was applied for both execution graphs and then we reapplied it once we decided to split the jobs.

	FINAL RESPONSE TIME EG1											
		Physical Resources										
		CC Server CPU	SP In Control Unit	SP Out Control Unit	SP In Sensor	SP Out Sensor	Water Company Server	Water Company Disk	Purification System	Lan Wired	Lan Wireless	Internet
Virtual Resources	Wired Connection Request	30	30	30	0	0	30	0	30	60	0	0
	Wireless Connection Request	18	18	18	9	9	18	0	18	0	30	0
	Internet Connection Request	10	10	10	0	0	10	0	10	0	0	5
	Database Request	10	10	10	0	0	20	50	0	0	0	0
	CC Server CPU	80	80	80	0	0	80	0	80	0	0	0
	Water Company Server CPU	10	10	10	0	0	10	0	10	0	0	0
	Purification System Pomezia CPU	0	0	0	0	0	0	0	0	0	0	0
	SP Ins Sample Request	15	15	15	13	13	0	0	0	0	0	0
	SP Outs Sample Request	15	15	15	13	13	0	0	0	0	0	0
	SUM Operations	188	188	188	35	35	168	50	148	60	30	5
	SUM RT (ns)	1880	3760	3760	35000	35000	840	25000	2960	1200000	840000	175000
	SUM RT (us)	1,88	3,76	3,76	35	35	0,84	25	2,96	1200	840	175
	SUM RT (ms)	0,00188	0,00376	0,00376	0,035	0,035	0,00084	0,025	0,00296	1,2	0,84	0,175

Figure 9.4: Final Resources Consumption

Chapter 10

Queueing Network Solving And BottleNeckAnaliyysis

Once parameterized the Queueing Network, the data was inserted into Java Modelling Tool (JMT). It was decided to carry out a simulation with a number of SeaweedPickingInland/Outgiung equal to one thousand for each. Various output Indices were analyzed in first analysis. Performance Indices:

- Utilization:
  - Control Center Server;
  - SeaweedPicking Control Unit;
  - Water Company Server;
  - Purification System;
  - Water Company Disk.
- Response Time:
  - Control Center Server;
  - SeaweedPicking Control Unit;
  - Water Company Server;
  - Purification System;
  - Water Company Disk.
- Response Time Sink (Check Water Quality);
- Throughput for Sink (Check Water Quality).

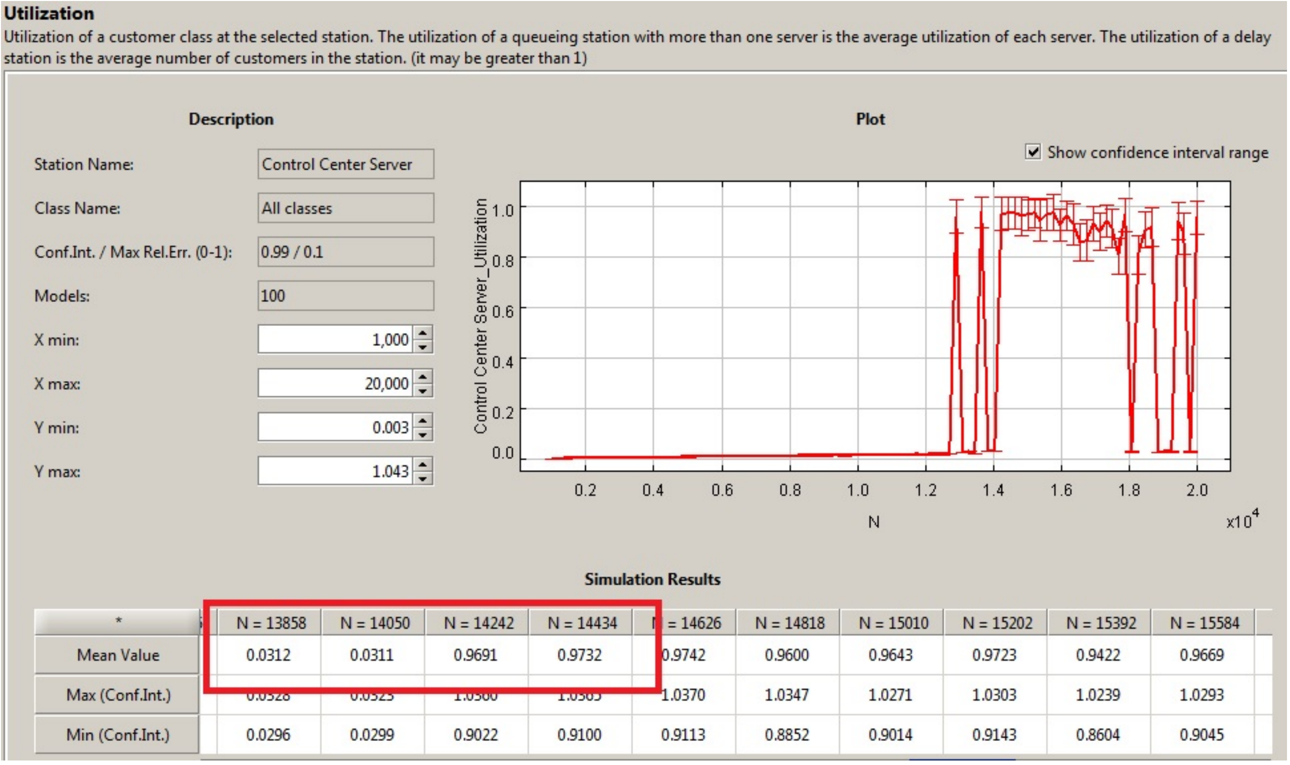


Figure 10.1: Control Center Server Utilization

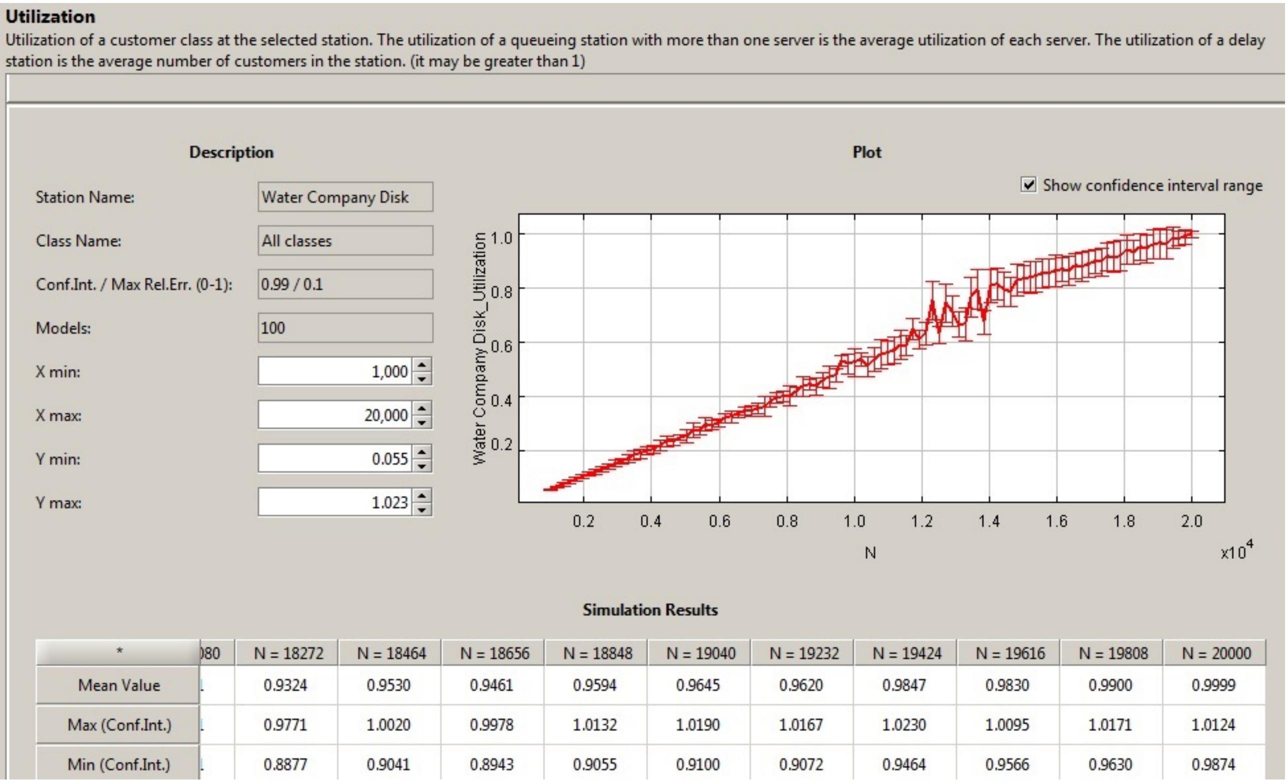


Figure 10.2: Water Company Disk Utilization

It resulted that the simulation was successful for all performance indices. Then we choose to run multiple simulations with increasing number of sensors to understand how the system scales and how it respond to the increasing of inputs. For this purpose we have done a What-If simulation composed of one hundred sub-simulation and it has come out that the Utilization performance of the Control Center Server, Water Company Disk gets worse when the

fourteen thousand sensors are reached.
Other proof of this is the analysis of the Response Time.

10.1 Refactoring

For the refactoring phase we wanted to consider the case in which we want to increase the number of Seaweed Picking to twenty thousand units. in this case, as we saw in the previous analysis, we have a bottleneck linked to the Control Center Server and to the Water Company Disk.

- Software Solution;
- Hardware Solution.

We have chosen a hybrid solution because we have halved the service time for each class of jobs for the Water Company Disk and for the Control Center Server, and decided to have two separate Water Company Disks one for the Sample Inland and one for the Sample Outgoing that translates into a modification of the architecture.
In addition, by resolving the new model we noticed an unexpected increase in the use of SCPU and for this reason we have slightly reduced the service time of this node. The results are shown below:

Figure 10.3: Utilization Refactoring

Chapter 11

AEmilia Model

The model implemented by us in Aemilia is modeled in a phase prior to refactoring.

11.1 Work Planning

The workflow was:

- Study of theoretical concepts of Aemilia;
- Drafting of Flow Graph;
- Drafting of State Diagram;
- Implementation of the Model;
- Test and Analysis of results.

11.2 Aemilia Flow Graph

A Flow Graph represents the topology of an architecture described in AEmilia. It is convenient to start with the flow graph representation of the architectural type and then to specify the behavior of each node.

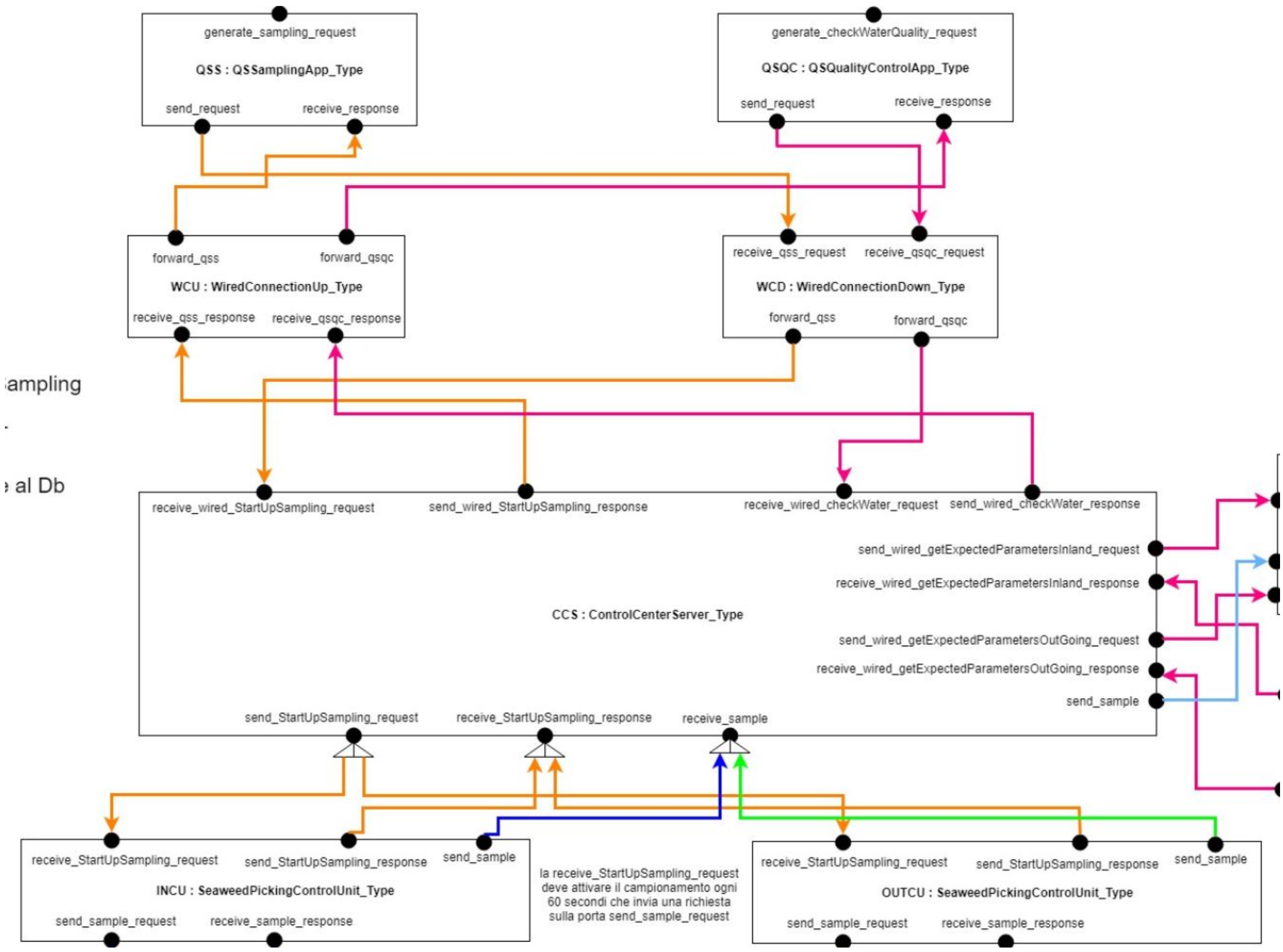
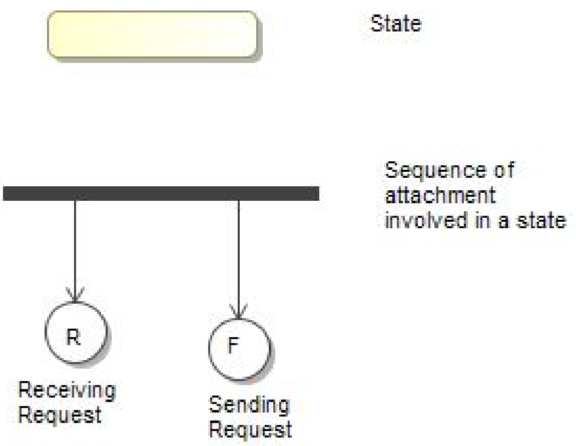


Figure 11.1: Flow Graph Extract

11.3 Aemilia State Diagram

In describing the behavior, we created our Diagrams.



We have a block representing the state, a fork under which there is a sequence of attachments involved within the same state, two circumferences representing these attachments

Figure 11.2: Legenda

Below you can see an extract of it:

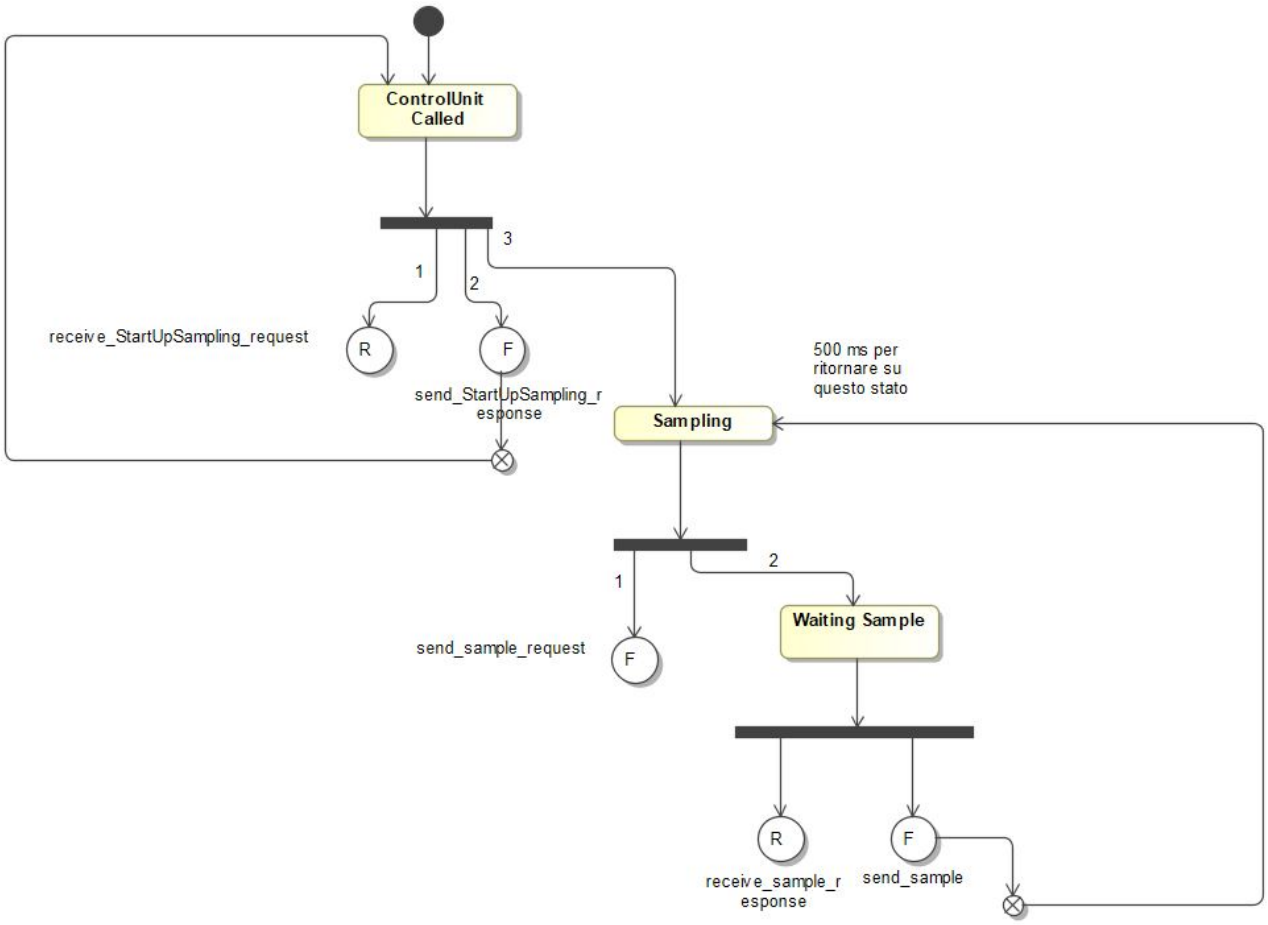


Figure 11.3: State Diagram Extract

In this image we see the behavior of the component "SeaweedPicking Control Unit."

11.4 Implementation Of The Model

Once the behavior of the various nodes, the architecture has been written following the syntax of Aemilia. Below, we will be proposed snippets, coming from previous pictures.

```
ELEM_TYPE SeaweedPickingControlUnit_Type(const rate SPCU_send_StartUpSampling_response_rate,
    const rate SPCU_send_sample_request_rate,
  const rate SPCU_send_sample_rate, const rate SPCU_delay_rate, const integer sensors_num)

BEHAVIOR

  ControlUnitCalled(void; void) =
    <receive_StartUpSampling_request, _> . <send_StartUpSampling_response, exp(
    SPCU_send_StartUpSampling_response_rate)> . Sampling();

  Sampling(void; void) =
    <send_sample_request, exp(SPCU_send_sample_request_rate)> . WaitingSample();

  WaitingSample(void; void) =
    <receive_sample_response, _> . <send_sample, exp(SPCU_send_sample_rate)> . Delay();

  Delay(void; void) =
    <delay, exp(SPCU_delay_rate * sensors_num)> . Sampling()
```

```
INPUT_INTERACTIONS

  UNI receive_StartUpSampling_request;
    receive_sample_response

OUTPUT_INTERACTIONS

  UNI send_StartUpSampling_response;
    send_sample;
    send_sample_request
```

11.5 Test And Analysis Of Results

After building the model, it was written a file describing the performance measures to be analyzed.

```
MEASURE SeaweedPickingINControlUnitUtilization IS
  ENABLED(INCU.send_StartUpSampling_response) -> TRANS_REWARD(1)
  ENABLED(INCU.send_sample) -> TRANS_REWARD(1)
  ENABLED(INCU.send_sample_request) -> TRANS_REWARD(1);
```

After simulating the Model the results are this:

```
- Value of measure "ControlCenterServerUtilization":
  0.003647

- Value of measure "WaterCompanyServerUtilization":
  0.00271904

- Value of measure "WaterCompanyDiskUtilization":
  0.00225506

- Value of measure "SeaweedPickingINControlUnitUtilization":
  0.0017911

- Value of measure "SeaweedPickingOUTControlUnitUtilization":
  0.0017911

- Value of measure "PurificationSystemServerUtilization":
  0.000463978
```

As can be seen from the results the values obtaines from the Aemilia simulation are very close to those of JMT and the non functional requirements specified at the beginning are all respected.

Chapter 12

Our Conclusion

With this second homework we have been able to fully understand the importance of the analysis and validation of UML models on the basis of functional requisites using methods and tools such as queueing Network and Descrption Language For Performance Analsiys like Aemilia. So we leave this experience with an enriched baggage that we hope will be useful for us in a future working environment