

Jiri Michal

11530340

Subject: ITC508

Assignment #: 1

Lecturer: Simi Bajaj

Date: 3-Apr-15

Table of Contents

[Task 1 3](#_Toc416091561)

[Define the term ‘use case’ and explain the various types of actors in a Use Case. 3](#_Toc416091562)

[Explain the difference between the <<include>> and <<extends>> relationships in use case diagrams? Give examples 3](#_Toc416091563)

[Describe with examples ‘encapsulation’, ‘information hiding’, ‘polymorphism’ and ‘data abstraction’. 3](#_Toc416091564)

[What is the difference between USDP and the Waterfall lifecycles in the relationship of between activities and phases? 4](#_Toc416091565)

[Explain Class diagram. Outline the main steps in developing a class diagram for a Use Case (UC). 4](#_Toc416091566)

[Task 2 4](#_Toc416091567)

[Question 1 4](#_Toc416091568)

[Start line run 4](#_Toc416091569)

[Record employee joining the line 5](#_Toc416091570)

[Record employee leaving the line 5](#_Toc416091571)

[Stop line 5](#_Toc416091572)

[Question 2: Communication diagrams 5](#_Toc416091573)

[Start line run 5](#_Toc416091574)

[Record employee joining the line 5](#_Toc416091575)

[Record employee leaving the line 5](#_Toc416091576)

[Stop line 5](#_Toc416091577)

[Question 3: Class diagram 5](#_Toc416091578)

[Bibliography 6](#_Toc416091579)

# Task 1

## Define the term ‘use case’ and explain the various types of actors in a Use Case.

Use case is defined as a description of the system functionality (an activity that the system performs (Satzinger, Jackson, & Burd, 2010)) form users’ view (Bennett, McRobb, & Farmer, 2010).

There are two major types of actors:

* The role of humans who interact with a specific use case in the system.
* The role of other systems which interact with a specific use case in the system.
* The role of devices which interact with a specific use case in the system.

## Explain the difference between the <<include>> and <<extends>> relationships in use case diagrams? Give examples

The <<include>> relationship is used when some functionality is part of many use cases A, B, C ... so we separate this functionality to the separate use case S and then we include the use case S to use cases A, B, C... (Bennett, McRobb, & Farmer, 2010). For example if we have use cases “Pay anyone”, “Transfer”, and “Pay Bills” in each of them we have to check if is enough money on the account. So we form a new use case “Check balance” which will be included in previous use cases and simplifies their description.

The <<extends>> relationship is used when a use case X optionally adds some functionality to a use case Y. For example a use case “check availability of item” can extend a use case “add item to wish list”

## Describe with examples ‘encapsulation’, ‘information hiding’, ‘polymorphism’ and ‘data abstraction’.

The **encapsulation** means that each object contains its description (data) and its functionality (methods) in one unit (Satzinger, Jackson, & Burd, 2010). For example an object “wheel” is described by its centre position and its radius and it can spin (functionality).

The **information hiding** is a design principle in which an object’s data are not visible from outside of the object, and the access to its data can be provided by methods (setters and getters) (Satzinger, Jackson, & Burd, 2010). For example an object “dog” has attribute (data) name, and only way how to access dog’s name from outside is by methods getName(), and setName() .

The **polymorphism** is an ability to take shape in many forms that means in relationship with the object oriented modelling, that there is a possibility to call same message on different classes and the respond will be appropriate (Bennett, McRobb, & Farmer, 2010). For example if we have a array of shapes which contains triangles, squares, and circles. And we call a method getArea() on the array of shapes we get appropriate outcome, even each shape has different formula to calculate its area.

The **data abstraction** is a process of simplifying and pinpointing of important characteristics. The abstraction is widely use in modelling where we have a view of a physical system and with abstraction we reduce the irrelevant data and keep the significant data to model the system (Bennett, McRobb, & Farmer, 2010). What is and is not important depends on a purpose of the model. For example if we have a physical system of pipelines we will keep data about position and length but we omit a colour of the pipes.

## What is the difference between USDP and the Waterfall lifecycles in the relationship of between activities and phases?

The main difference between USDP and Waterfall is that in each phase of project lifecycle (inception, elaboration, construction, and transition) the USDP has many activities (requirements, analysis, design, implementation, and test). For example in the inception phase USDP contains all activities with most stress put on requirements (Bennett, McRobb, & Farmer, 2010). On the other hand the waterfall has an activity in just one phase. For example the requirements activity would be in the inception phase but not in construction phase.

## Explain Class diagram. Outline the main steps in developing a class diagram for a Use Case (UC).

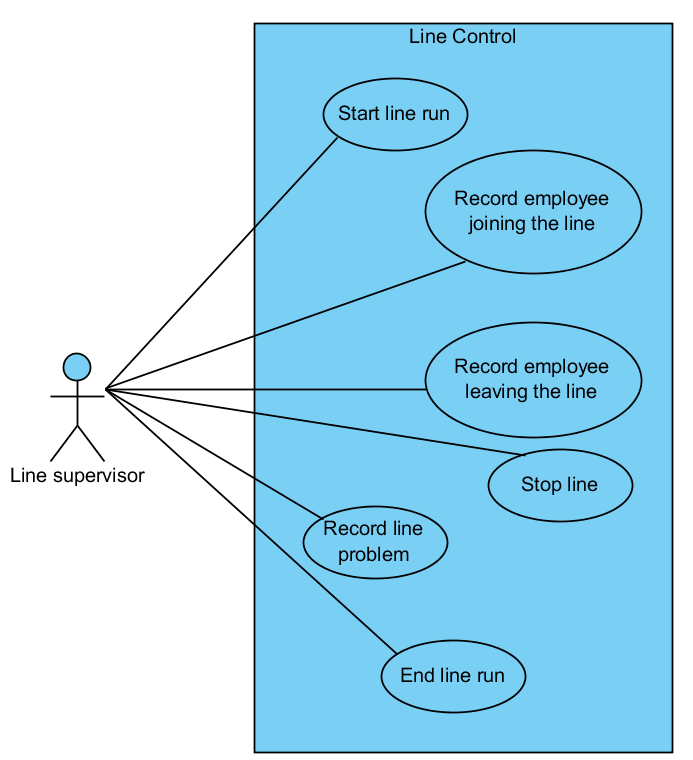
The class diagram (analysis class diagram) consists of entity, boundary, and control classes and their associations. These classes and associations have been identified from several use cases and collaboration diagrams derived and developed from these use cases.

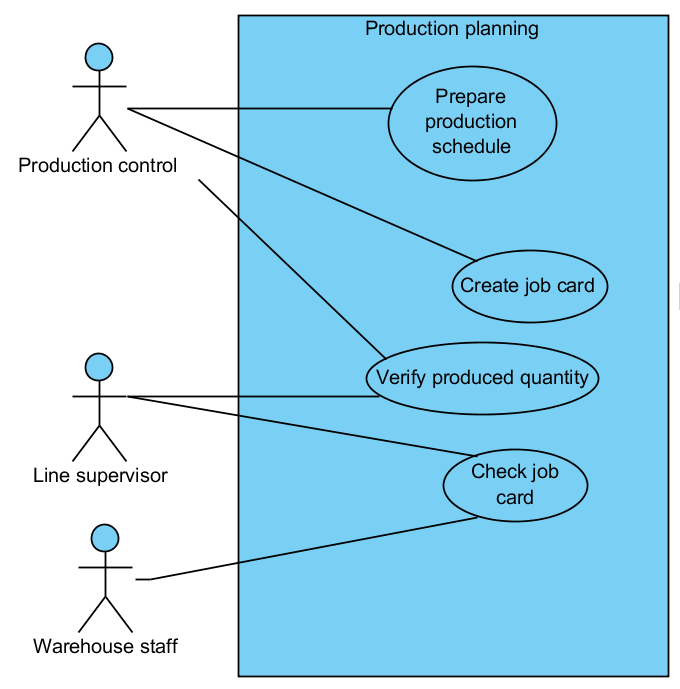
Firstly from a use case we identify entities we need to store information about, and associations among these entities. Next step is to draft a communication diagram for the use case including boundary and control class if needed. With addition of the boundary and control classes we need to redefine association among the classes (possible redirection due to the control class). As the last step we put all classes from the separate communication diagrams together and identify a multiplicity of the associations. Because the control and boundary classes usually belongs to different packages we can omit them in the final class diagram (their presence is optional). Note that an actor form use case (user) does not need to be represented by any class if we do not need to restrain him or her access or keep a record about that access.

# Task 2

## Question 1

### Use cases diagrams





### Start line run

To start the line run the supervisor check if sufficient amount of ingredients needed for the run is available and if required number of staff is present and clocked in. And if there is no visible obstruction, the supervisor records time and date and starts the line.

### Record employee joining the line

Usually an employee is recorded when he/she join the line at the beginning of the shift. Also they can be recorded joining the line during a line run or after an absence. The time, date, location, and job number is recorded.

### Record employee leaving the line

Usually an employee is recorded leaving the line at the end of the shift. Also an employee is recoded leaving the line when he/she is leaving for sickness or is reassigned to different line. Employee is not usually recorded leaving for lunch breaks or downtime due to breakdown. Time, date and circumstances are recorded.

### Stop line

When supervisor stop the line he/she records time and reason of stopping which can be a coffee breaks, a lunch break, and restocking.

## Question 2: Communication diagrams

### Start line run

### Record employee joining the line

### Record employee leaving the line

### Stop line

## Question 3: Class diagram

# Bibliography

Bennett, S., McRobb, S., & Farmer, R. (2010). *Object-Oriented Systems Analysis and Design Using UML (4th edition).* Berkshire: McGraw-Hill Education.

Satzinger, J. W., Jackson, R. J., & Burd, S. D. (2010). *System Analysis and Design in a Changing world (5th edition).* Boston: Cengage Learning.