SOFT20091: Software Design & Implementation

Sean Turner - N0643456

Dan Hughes - N0644583

Melvin Joy - N0640825

Thomas Moore –N0642032

Table of Contents

[Overview 3](#_Toc481506408)

[Requirements Specification 3](#_Toc481506409)

[Table 3](#_Toc481506410)

[Use Case 5](#_Toc481506411)

[Activity/sequence diagram 5](#_Toc481506412)

[Class diagrams 6](#_Toc481506413)

[Software Development Life-Cycle 7](#_Toc481506414)

[Software Functionalities 7](#_Toc481506415)

[MoSCoW 7](#_Toc481506416)

[Analytical Techniques 8](#_Toc481506417)

[Swot 8](#_Toc481506418)

[Design Patterns 9](#_Toc481506419)

[VCM 9](#_Toc481506420)

[Data Structure 10](#_Toc481506421)

[Sort algorithm 10](#_Toc481506422)

[Implementation tings 10](#_Toc481506423)

[Testing 10](#_Toc481506424)

[Approach 10](#_Toc481506425)

[Performance 10](#_Toc481506426)

[Conclusion 10](#_Toc481506427)

[Appendices 10](#_Toc481506428)

# Overview

This report is on the Pokémon Pokedex software, which is created to understand how the design and development of Pokedex Software took place. The Pokémon Pokedex is a software that will be able to store all the details of the Pokémons and when the user initially uses the Pokedex has no Pokémon details stored in the Pokedex, this is so that when they come across new Pokémon they can be added to big database of Pokémon’s with all their details. The Pokedex’s main functions will be user can register, search, update, delete, sort and view evolutionary data of all Pokémon.

# Requirements Specification

## Table

|  |  |  |  |
| --- | --- | --- | --- |
| Number | Type | Requirement Description | Implication |
| 1 | Functional | The ability to add a new Pokémon record to the storage. | User interface to enter Pokémon details |
| 2 | Functional | The ability to hold the data and information of Pokémon. | Use an array to store Pokémon data. Allow it to be dynamic. |
| 3 | Functional | The ability to search through stored data by National Dex Number, ID number or by Pokémon Types (Primary/Secondary). | Implement an appropriate searching algorithm based on a given field. |
| 4 | Functional | The ability to sort records by National Dex Number, ID Number or by Type. | Implement an appropriate sorting algorithm given a field to sort on. |
| 5 | Performance | Chosen sorting algorithm is most efficient for the selected data type(s). | Algorithm chosen must suit a dynamic array of classes. |
| 6 | Performance | Chosen data type(s) store the data in the most efficient way possible. | Limit the amount of unused space in data types by selecting the best data type. |
| 7 | Physical and Operational | A suitable user interface is chosen to allow for the user to easily navigate the Pokedex. | Create an interface that is easy and instinctive. |
| 8 | Functional | Allow Pokémon registered to have multiple typing. | Use of string within class and Boolean to show whether there is or is not a second type with string to show second if it is. |
| 9 | Functional | Allow for all Pokémon to be viewed at once, in order of ID Number. | Implement a sort and display the results in a 10 per page window |
| 10 | Functional | Allow the ability to view Pokémon that evolve and Pokémon that do not in separate lists. | Allow a search of Pokémon and only show ones with an evolution or prevolution |
| 11 | Functional | Allow the viewing of evolutions lines of Pokémon that do evolve. | Allow selection of Pokémon with evolutions and display that evolution tree |
| 12 | Operational | The date that a Pokémon is registered is recorded. | When a Pokémon is entered get system date and store it with the Pokémon |
| 13 | Operational | Keep records of Pokémon registered by the user, to be viewed whenever needed. | Allow entered Pokémon to be stored in data structure. |
| 14 | Operational | Allow Pokémon to be deleted from the system. | Allow the user the choice of deleting a Pokémon record. Then remove this record from our system. |
| 15 | Operational | Allow for the modification or updating of an existing record | Allow the user to make changes to each record then store this changes in the system |
| 16 | Functional | Stop duplication of previously entered Pokémon | If a Pokémon entered is already in the system don’t allow it to be entered twice. Ask if it needs to be edited |

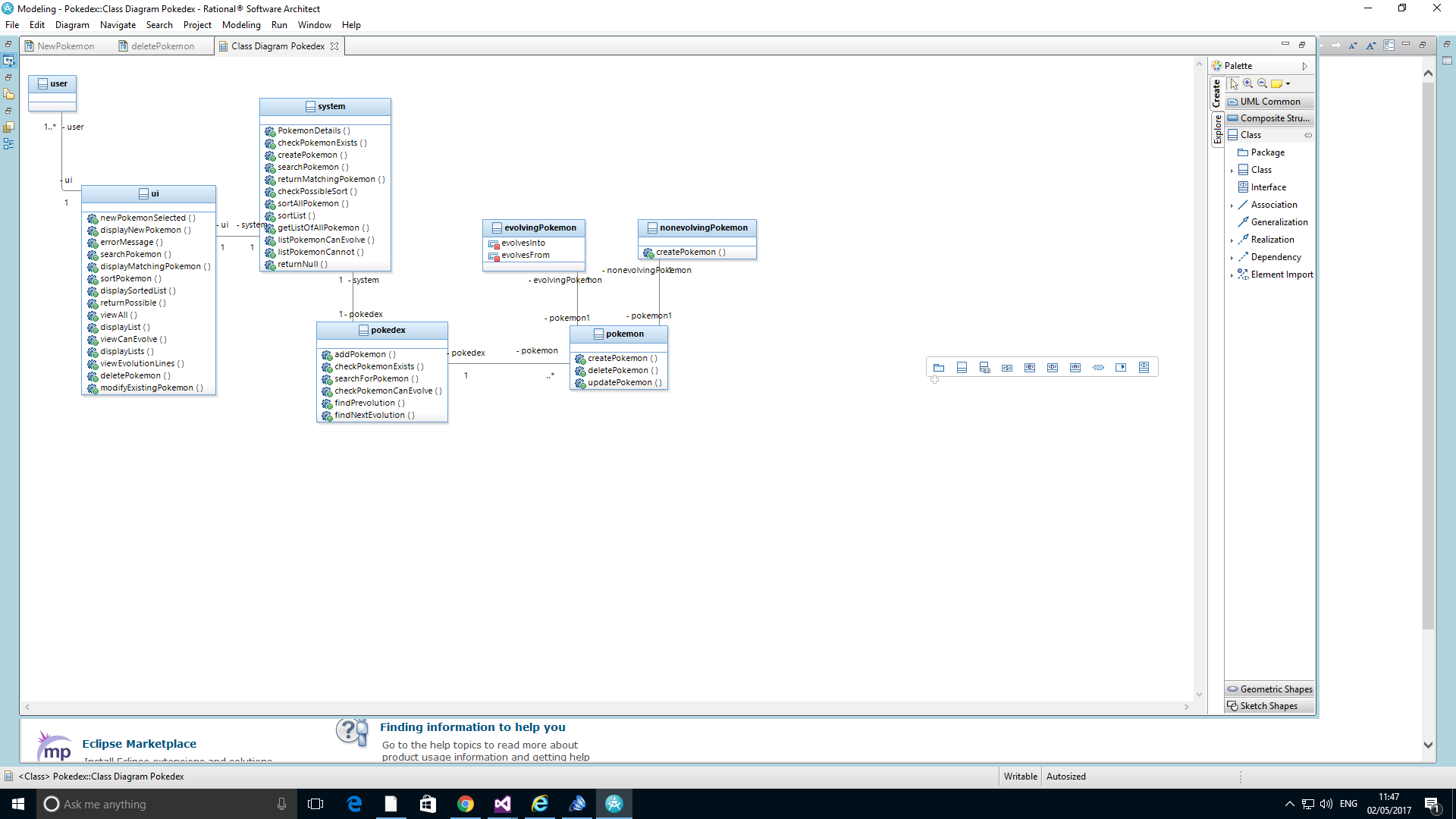
## Use Case



Figure 1

## Activity/sequence diagram

## Class diagrams



# Software Development Life-Cycle

The Software Development Life-Cycle that is used for the development of the project is the agile framework called Scrum. Scrum is used in development for cross functional or self - organising teams of about 4 - 9 people, and uses fixed – length iterations called sprints to produce a potential product every sprint through a series of organised meetings. Scrum has roles of development team, product owner and scrum master. The product owner mainly looks at the requirements for the project, development team builds a potential product every sprint and scrum master acts as the facilitator. Scrum contains artifacts of product backlog and sprint backlog. Product backlog is everything that could be done by priority and sprint backlog is what has been agreed to do for each sprint. (James, 2010)

In the group all of the members will be acting as the development team role, one of the group member will also be given the product owner role and the module tutor will be acting as the scrum master. The advantage of using scrum for the development of the Pokedex is that the group is only size of 4 people and people can easily self – organise through a series of meetings and as there is leader people can assign each other tasks through meetings where issues can be resolved quickly. The sprints and constant feedback means that it will be easy to cope with changes while having a working product throughout, which makes it easier to deliver quality product in the given time.

James, Michael. "An Empirical Framework For Learning". *Scrum Methodology*. N.p., 2010. Web. 1 May 2017.

# Software Functionalities

## MoSCoW

|  |  |
| --- | --- |
| MUST | * Registration of new Pokémon’s. * Sorting of all the Pokémon’s in the database. * Be able search specific Pokémon’s with their national Pokedex number. * Modify details for existing Pokémon’s. * Delete Pokémon’s details. * Be able to see detailed evolutionary line of the Pokémon. |
| SHOULD | * Make sure that there aren’t any duplicate details of Pokémon. * Pokedex Software should be coded efficiently to give the best performance. * User friendly and easy to navigate. |
| COULD | * GUI interface so it makes it is more appealing and user friendly. * Images can be added of each Pokémon and displayed. * XML export and import. * Each Pokémon data can be compared with each other. |
| WONT | * Graphs displayed for each details of the Pokémon. * Abilities of Pokémon can be displayed. * User registration and be able to share Pokémon details. |

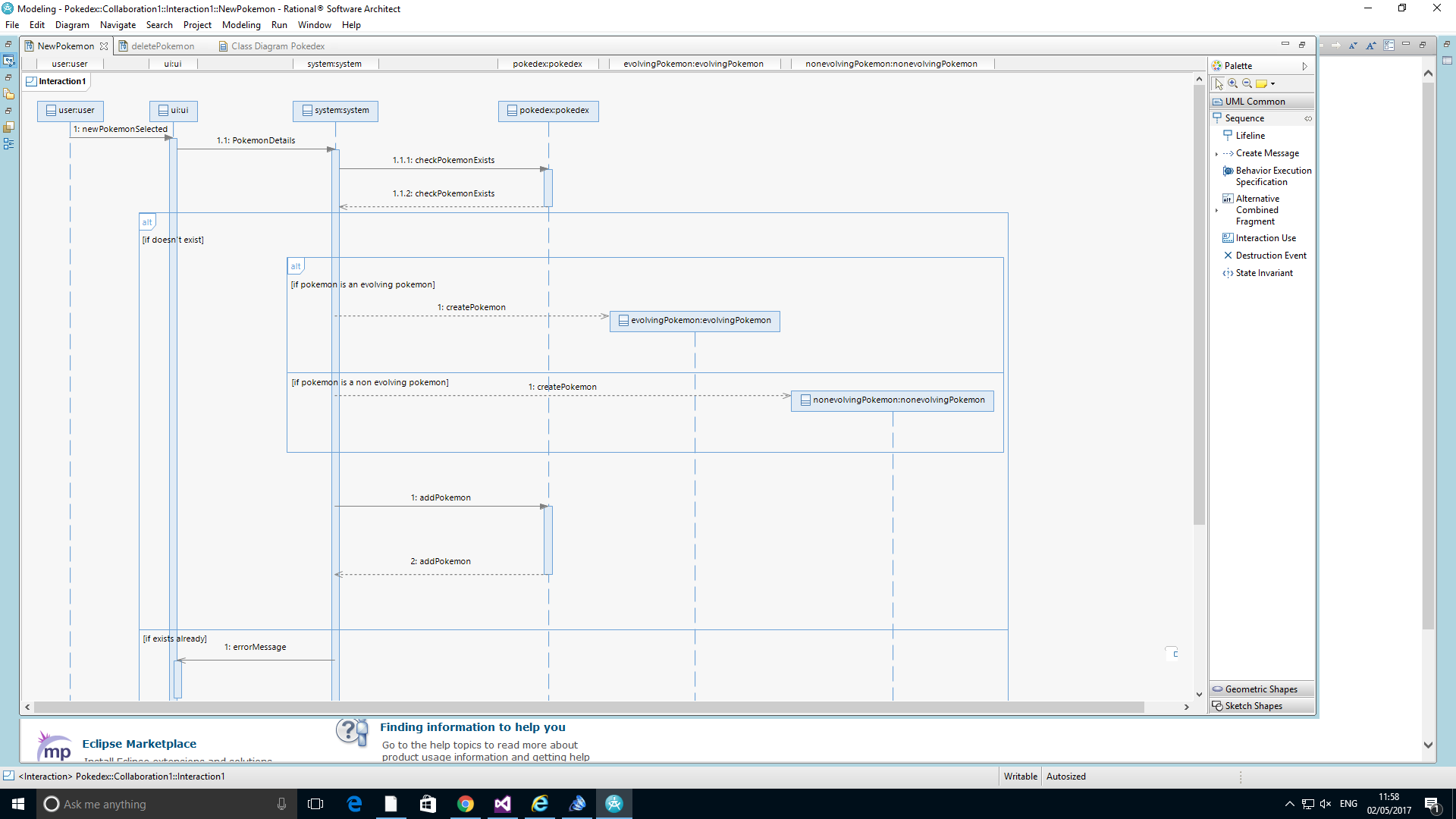
# Analytical Techniques

## Swot

|  |  |
| --- | --- |
| Strength | Weaknesses |
| * Flexible because the user can manipulate the data as they require. * User friendly by easily being able to navigate through the pokedex. * The software performs efficiently by making sure that code uses efficient algorithm and design pattern. | * When user adds Pokémon there is no check system to see if what user added is actually a Pokémon. * Compared to current Pokedex systems in the market, the Pokedex system that is being implemented has no statistical details of the Pokémon’s in the forms of graphs etc. |
| Opportunities | Threats |
| * GUI user interfaces used to enable easier navigation for the user and better appeal to the software. * The details entered in the Pokedex could be compared with the actual big database of Pokemon online to see if entered details are correct. * Be able to register multiple users to the system and enable them to share details of Pokémon’s. | * Storage of data might have a limit in space if a large about of details are added. * Need to make sure that there are no memory leaks in the program. |

# Design Patterns

## VCM



View

Control

Model

For the Pokedex system the design pattern that will be used is the Model View Controller, where the User Interface class is the view, system class is the control and Pokedex is the model. The user selects new Pokémon to be inserted and the UI sends the details to the system and the system then checks if the Pokémon exists. If the Pokémon exists the error is sent to the user from the system to the UI otherwise the Pokémon is added to the Pokedex. Using the Model View Controller makes the design more flexible and extensible and it will make it easier when Pokedex is developed further in the future.

# Data Structure

# Sort algorithm

# Implementation tings

# Testing

## Approach

## Performance

# Conclusion

# Appendices