DEVELOPMENT OF REVISION APPLICATION FOR ANDROID DEVICES

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Bachelor's thesis

Visamäki
Degree Programme in Business Information Technology

Spring 2018

Juho Jauhiainen

Tietojenkäsittely
Visamäki

Tekijä	Juho Jauhiainen	**Vuosi** 2018
Työn nimi	Development of Revision Application for Android Devices	
Työn ohjaaja /t	Lasse Seppänen	

TIIVISTELMÄ

Tämän opinnäytetyön tavoite oli tutkia ja esittää kuinka voitaisiin lähestyä kertausapplikaation kehittämistä Android-laitteille. Työtä on rajoitettu siten että applikaation koodauskieleksi on määritetty C#-koodikieli ja kertausmateriaali on rajattu vain loogiseen ongelmanratkaisuun. Opinnäytetyöllä ei ole toimeksiantajaa ja aihe onkin opiskelijan itsensä valitsema.

Työn teoreettinen osuus käsittelee ja tutkii nykyaikaisen ohjelmistokehityksen vaatimuksia ja käytänteitä ja esittelee kehittämistyökaluja. Tutkimusmateriaalina on käytetty kirjoittajan omia käyttökokemuksia sekä valikoituja kirjallisia ja internetlähteitä.

Opinnäytetyön käytännön osuudessa laadittiin selkeä sovellussuunnitelma, joka pohjautui teoreettisen osuuden havaintoihin ja valintoihin ja noudatti työlle säädettyjä rajoituksia. Käytännöllistä sovellusdemoa ei kehitetty tiukan opinnäytetyön kirjoitusaikataulun vuoksi.

Opinnäytetyön tuloksena syntyi toimiva ja hyödynnettävissä oleva sovellussuunnitelma joka pysyy rajoitusten puitteissa. Tulosta voidaan hyödyntää tulevaisuudessa pohjana opiskelijoiden ICT-projektille tai esimerkkinä yrityksille, jotka olisivat kiinnostuneita kehittämään työssä kuvailtua sovellusta.

Avainsanat Ohjelmistosuunnittelu, mobiilisovellukset, C#, Android, ongelmanratkaisu.

Sivut 22 sivua, joista liitteitä 8 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Business & Information Technology
Visamäki

Author Juho Jauhiainen **Year** 2018

Subject Development of Revision Application for Android Devices

Supervisor Lasse Seppänen

ABSTRACT

The goal of this thesis was to investigate and display how to develop a revision application for Android devices. Work has been restricted so that application must be developed using C#-code language and revision material is limited to logical problem-solving. The thesis has no commissioner and the subject has been chosen by the author.

The theoretical portion deals with and investigates the demands and practices of modern software development and introduces development tools. Background information was collected from writer's own personal experiences as well as from selected literature and internet sources.

The practical portion consists writing an application plan that is based on the discoveries and selections in the theoretical portion and follows the restrictions set within the thesis. The practical application demo was excluded due to strict thesis writing schedule.

The result was successful creation of the application plan that is functional and stays within the limitations. Result can be used as a base for future student ICT-projects or as an example for companies who would be interested in developing described application.

Keywords Software development, mobile application, C#, Android, problem-solving.

Pages 22 pages including appendices 8 pages

CONTENTS

Appendixes

1 INTRODUCTION

When facing an entrance exam situation, just before the doors open to the exam room, many may have experienced the sudden panic and wish to do some quick rehearsal on subjects they have studied, no matter how big or small the subject matter is. Unfortunately, there is currently no tool to help with this stressful situation. And thus, due to interest sparked by own experience and for the sake of versing the writer further into proper software design and development, the goal of the thesis is to explore a potential way for creating a mobile rehearsal application in the form of clearly written and clearly documented application development plan.

Another reason for choosing the thesis subject was its adjustable nature and transformability. If there should ever come a time when entrance exams are abolished, it could be possible for example to then transform the idea into an educational game to use it as a teaching tool.

This thesis is aimed at readers who have previous experience and knowledge on both coding and software development as some of the used terminology may be somewhat alienating to readers who are not familiar with the subject of software development. Due to time restrains, this application is designed with certain limitations these namely being that application must be aimed for Android platforms, contains only logical problem-solving as revision material, must be written with C#-coding language and must support expandability.

As such, the crucial research questions for this thesis are as follows:

– What tools will be used for development?
– How expandable is the application?
– What development methods shall be used?
– How will logical problem solving be approached?
– How would the user testing and effectiveness measure work?

Towards the latter half of the thesis, it was possible to have an interview with a junior software developer from Ambientia who offered valuable, experienced information for the thesis. As it is according to standard procedures and as the interviewed did not request it otherwise, the interview is referred and listed as anonymous.

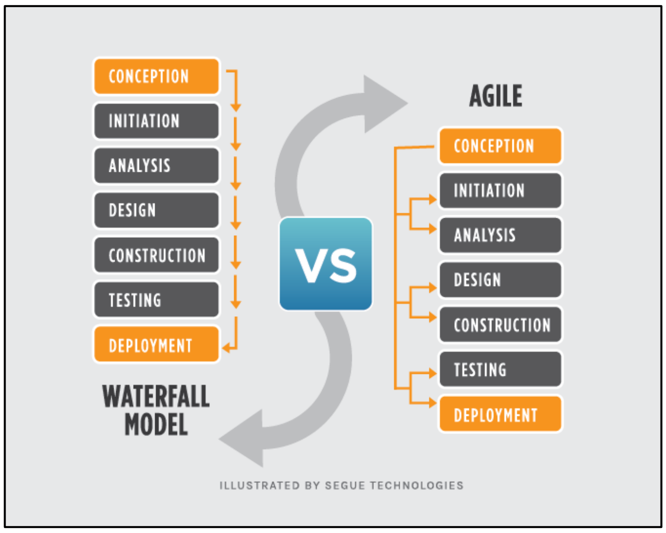
2 MODERN SOFTWARE DEVELOPMENT

Software development hails its roots to the 60s and 70s era when automated information technology became a crucial, integral part of the everyday bureaucracy and ever since then, software is an essential part of most devices used today. As time has passed, however, the modern daily work environment has changed from what it previously had been and as such, modern software development must be prepared for situations and elements that were not present in the past.

2.1 Adaptability to changes

In the past, software development processes were very linear and often relied on software specifications being set in stone through the entire project. This lead in to the utilization of waterfall work models where software specifications were determined and set at the start of the project from which the development would move forward phase by phase often leaving practical testing until the very end of the development process just before deployment. While functional in some cases, waterfall models greatest flaw is that any sudden change in specification, be it either due to a client or unforeseen circumstances, could potentially lead into a complete redo of the project and thus can be too risky and costly development method for a modern development project.

Modern software development utilizes new working methods that allow for better adaptability during development known as agile methods. Agile methods emphasize the importance of constant feedback, iteration and collaboration between developers and business personnel which allows developed software to adapt to changes even late in the development process and supports the continuous evolution of the project. Picture 1 visualizes the differences between waterfall model and agile methods.



Picture 1. Waterfall and Agile model comparison (Lotz 2013).

There is a multitude of agile methods to choose from each with their own unique approach to delivering best results. For the creation of revision application, few selected agile development methods are explored.

2.1.1 SCRUM

SCRUM method embraces close interaction between the developers and the client as with this method, the project starts with defining product backlog into which developers alongside the client discuss and decide what functionalities and features are included and how they are prioritized. After initial product backlog is formed, sprint planning begins where the sprint is a timeframe, usually a month, in which project team agrees what features are to be included into the sprint and what should be completed by the end of it.

After each sprint, the developers and the client or client representative redefine the backlog from which new set of features and functionalities are selected for the next sprint. Most development projects aim to have three to four sprints, but this is adaptable based on the situation. There are also certain team roles that are utilized in SCRUM methodology, these being Scrum Master whose responsibility is the overall project and team management and to arrange interaction with the customer, Scrum Team that handles the execution and development and lastly the client who holds the title and role of Product Owner.

As SCRUM focuses on the close relation of the developers and clients, it has proven to be a very successful and functional way for companies to develop modern software. Because SCRUM methodology is in popular use, many other agile development methods have adapted or otherwise share some similar features and principals with it.

2.1.2 Lean

Like SCRUM, Lean methodology focuses on utilizing the product backlog for its project goal. However, the product backlogs are defined much more compact in Lean as listed features are only limited to the most critical items that the client desires and are then prioritized accordingly in agreement with the client.

Lean differentiates from SCRUM in that it does not utilize or use specific team roles and instead favors offering everyone on the team the authority to make decisions. Lean abolishes team roles as it has deemed based on the use cases of the past that a non-hierarchy-based control is more efficient and quicker for developers. The effective use of team resources and utilization of automated tests are other focus points in Lean methodology. (VersionOne n.d.)

2.1.3 Kanban

Efficiency without overburdening the developer team is the main goal of Kanban methodology and as such, its practices focus on enhancing the workflow. Kanban encourages to use visual presentation of the workflow, limiting the amount of WIP tasks and enhancing the flow by pulling in the next highest item from the backlog after finishing the previous task.

Kanban does not utilize any form of roles and instead of delivering results in a bundle like SCRUM methodology does with its sprints, Kanban focuses on delivering results one item and task at a time. Kanban also encourages to focus on features that hold most value for the product and the client bit similarly to Lean methodology. The benefits of Kanban methodology as such are more rapid delivery rates with software features and the reduction of unnecessary activities in development which saves development time.
(VersionOne n.d.)

2.1.4 Crystal

Crystal methodology may be one of the most adaptive of the agile methods as it considers people as the most important element and regards process as a secondary objective. Because of this, Crystal method advises that a process should be modeled based on the team requirements without using prescribed tools or techniques.

Frequent delivery, discussions and reflections about methodology improvements, close and all-inclusive communication and active feedback are the most important principals in Crystal method and with these principals Crystal avoids rigid processes. The amount and need of management, documentation and reporting are based on the team size and project environment making this method one of the lightest of agile development methods. (Santos 2017.)

2.2 **Mobile application development**

As mobile devices from smartphones to tablets have become more and more of an everyday item for consumers, the modern software development has naturally hopped on the growing market. However, designing software or applications specifically for mobile devices has its own challenges.

2.2.1 Mobile platforms

Mobile application development has been largely aimed towards the biggest names in mobile device markets, which currently are Apple's iOS reliant platforms such as iPhones and other similarly branded devices, Android which runs on multitude of mobile devices available today and Windows with their own smartphone and tablet models. Each of these platforms run on their own designated code-language.

Apple's iPhones and the other devices of the brand utilize Objective-C code language. Objective-C language is derived from C programming language and as such offers elements used in C language, but it also adds its own syntax which is heavily based on object-oriented and dynamic runtime programming.

Android OS that many mobile devices have installed is developed by Google from base in Linux-kernel. The utilized coding language on Android systems is Java language which is also commonly used in Web-development which is why many mobile applications developed in Java share common extensions and add-ons with their browser counterparts.

Windows has been kind of a late-comer to the mobile device market, but their Windows mobile devices have gained some popularity among mobile developers in recent years. The rise in popularity occurred after Windows merged the Application Programming Interface used in their mobile, tablet and desktop into one making design of software and application that can run on multiple Windows platforms much easier.

Mobile Windows devices run on C#-language which has all the features of its predecessor C languages but includes extended functionalities for using Representational State Transfer API and other such services. Despite their increase in popularity among developers, Windows mobile devices have begun to fade away from consumer use and are thus some often-ignored platform devices.

2.2.2 Criteria of mobile application development

Mobile device hardware is often designed to be cost efficient due to the large production amounts, which means the processor, graphic drivers and memory they use are largely inferior when compared to standard PC. Commonly mobile devices for example utilize processors that can reach over 100 MHz at their best whereas the most laptops and desktop PCs have processor reaching two to four GHz depending on the quality of the PC. These hardware limitations present developers some criteria to consider when designing applications for mobile devices.

Because much of the hardware on mobile devices is weaker when compared to PC, the criteria which developer must focus on are light memory usage, low processing power requirement and as a new restriction that is not commonly considered in PC software development, low battery consumption. Similarly, mobile developers must keep in mind the varying screen sizes and screen resolutions between the mobile devices as they are quite different from standard PC and laptop screens and cannot perform heavy visual tasks as easily as traditional computers.

Another criterion for a mobile developer to remember is the higher expectation and critical review rate on mobile applications than they usually are for PC software. On PC software some number of glitches and dysfunctionality is expected but official, chargeable mobile applications are expected to be far more polished. As such, the quality of the application is a high priority in mobile application development. (Mikkonen 2004, 5.)

2.3 Cross-platform development

Often in the modern software development, it is necessary to transfer software from one platform to another, which can be rather difficult due to each platform utilizing its own type of coding language, and sometimes these transfer attempts can lead to software being re-written from scratch to another platform environment or cause the software either partially or completely break down.

To counter the problems of cross-platform development, tools specializing in cross-platform design have emerged in recent decade as either modules to existing Integrated Development Environments or as their own separate IDEs. These tools allow developers to design and code their software on multiple platforms simultaneously while utilizing just one coding language specified by the tool and simplifying some of the framework and feature differences between the platforms. Picture 2 displays the selection of cross-platform tools available.

Cross platform development tools
Adobe PhoneGap
Xamarin
React Native
A framework for building native apps with React.
Sencha
NativeScript
kony
appcelerator
MONOCROSS

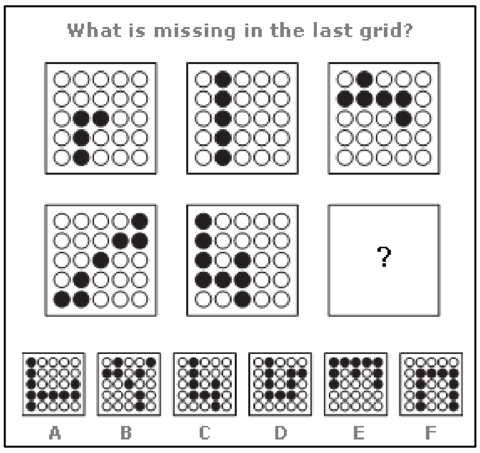
Picture 2. Variety of cross-platform tools (Nylund 2017).

Despite the existence of cross-platform development enabling tools, the large number of mobile platforms still presents problems for mobile developers. While cross-platform development allows for answering the possible client desire to reach a wide audience, there just are some functionalities and features that cannot be performed on all platforms thus forcing the developers to at times make harsh restrictions. (Anonymous 2018.)

3 APPROACH TO LOGICAL PROBLEM SOLVING

In a multitude of professions today, employees from the top to bottom are often expected and required to have some degree of skills in logical problem-solving (McKay 2017). This holds true for software developers as well as they are expected to be able to use logical problem solving when facing down difficulties, be they either something within the software code or something in the project management.

The most usual way to often determine problem solving skills of a candidate during recruit process or of an entrance exam participant is to present them a logical reasoning test which at their simplest are series of images from which last one is to be deduced based on the other ones. Other form of these kinds of test is a truth-or-lie type of puzzle where there are three claims and the last one must be either false or true based on how it compares and contradicts with the previous two. Picture 3 displays one type of logical reasoning puzzle.



Picture 3. Example of an image-based logical reasoning puzzle (IndiaBin n.d.).

When selecting which types of logical problem-solving puzzles should be used in the software, the greatest criteria is to consider the platform the software is developed on. If the software is to be created for PC and tablet devices, the utilization of both text and image-based puzzles is viable as screen resolution and the size of the devices allow for easier image and text viewing. If the software is to be created for smart phones, however, the developer must consider the smaller screen resolution and size which can make reading text quite a chore.

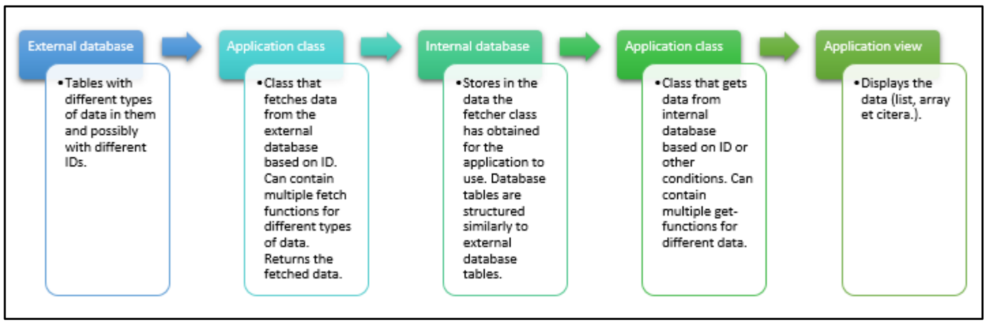
As the revision application focuses on Android devices, the choice between the two implementable puzzle types favors more image based logical reasoning puzzle. Because the smaller screen size and resolution of Android devices, displaying heaps of text would be straining for the eyes and as such image reliant approach is preferable for the sake of a friendlier user experience.

3.1 Approach with databases

One possible implementation of logical puzzles is to hard-code them as a part of the view or page which however is only recommendable if there are no plans for increasing variety of puzzles and only one type of puzzle is planned to be displayed always. If there are plans for increasing the variety and the amount of reasoning or deduction puzzles, utilizing databases is one recommended approach. For database utilization there are options to either implement an internal database or then use an external database both options having their own strengths and weaknesses.

The internal database is coded into the application itself making the stored data easy to access at any part of the application without the need for online connection but can only store small amounts of data before affecting the application runtime as vast internal database increases application byte size and memory usage. External database allows more data storage and an option to add and update the data but requires online connectivity during application runtime and the database connection must be called for each unit that requires it.

A hybrid solution exists where data is first called from external database and then stored into internal database for application to use but this requires that both internal and external databases have similar structure and that the data stored in internal database is small scale in order not to slow down application runtime. Picture 4 offers a crude example of how hybrid solution functions.


External database
•Tables with different types of data in them and possibly with different IDs.
Application class
•Class that fetches data from the external database based on ID. Can contain multiple fetch functions for different types of data. Returns the fetched data.
Internal database
•Stores in the data the fetcher class has obtained for the application to use. Database tables are structured similarly to external database tables.
Application class
•Class that gets data from internal database based on ID or other conditions. Can contain multiple get-functions for different data.
Application view
•Displays the data (list, array et citera.).
Picture 4. Simplified presentation for hybrid database solution.

The choice between database types and implementation ultimately boils down to the amount of data and whether there are plans to add, expand and update the data that application uses from database. At the start of development, the implementation of internal database is recommended for the sake of easy testing and as development advances, database type should be switched to external for easier data management.

Another criterion to consider in terms of database related solutions is how dependent the software is on the network connectivity. Should the software be designed so that its major functionalities or the software in its entirety requires network connection, implementing an external database is a reasonable option. However, if the software is designed to be independent of the network connection or it does not require it to work, a hybrid database solution is recommendable.

3.2 Approach with embedded resources

An alternative way of implementing logical problem-solving into application without the utilization of databases is possible trough embedded resources in some C#-code-based cross-platform development tools. Embedded resources are files that upon creation and addition are inserted into application assembly and as such are accessible to all platform versions of the application.

As embedded resources are one part of the code, they are relatively easy to use in and tie to different types of events like button presses and screen tap related functions. However, as they are one part of the code and the application assembly, having a vast amount of embedded resources could potentially lead to similar problems in runtime like overcharged internal databases. Therefore, the embedded resources should be used in cases where the amount of the resource material is known and set before hand or then it should be used as an initial solution for early builds.

4 APPROACH TO TESTING

Testing developed software and ensuring the proper functionality of features is a crucial part of modern software development and has led to in most cases increased quality of delivered results. Testing was not always kept in such high regard however.

In the past, when software development followed the waterfall style of development work, testing was considered as a phase that was usually performed late in the project process. This, however, could sometimes lead to redoing some of the earlier phases in the project if some major bugs or dysfunctionality were found and as such, testing was often considered to be very time consuming and a costly part of the development process. As a cause for the time and resource consuming nature of testing, many companies and developer teams would sometimes either completely ignore proper testing phase or then keep the tests at the most simple and minimal levels which in return leads to unpolished, glitchy and in worst case broken software on release.

As agile development methods have become more and more common in software development, testing has become an integral part of every stage of the development process and as such the risk of releasing broken and unpolished software has declined excluding the cases of pre-alpha and -beta stage user tests that some software utilize before releasing the final marketable version. Along with the evolution of development methods, the nature of test methods themselves have changed.

4.1 The three levels of software testing

Software testing is divided into three levels – unit, integration and system testing. Methods and ways to perform tests in these categories vary from case to case but the general idea for each category remains the same. Unit testing consists of test cases for individual modules or classes of the software to ensure their functionality. Integration testing sees how the created modules and classes work with each other when combined and lastly system testing checks the overall functionality of the software.

In unit testing a singular unit of code, which can be a function, module or object, is tested immediately after its creation. Unit testing is meant to confirm that the created unit performs its intended tasks correctly. One of the problems of unit testing however is that often some singular units require other parts of the software for its functionality which is why it is often necessary to create accompanying dummy units for testing purposes.

4.2 **Static and dynamic testing**

In software testing it is important to remember the divination of static and dynamic testing methods. Static testing methods can be performed as early as the beginning of the development as it focuses on performing base level testing, often with the help of analytic tools, to remove most glaring problems before more accurate testing is performed. Dynamic testing methods are performed when software has developed into such form that it can go through prototype or partial functionality testing. (Kasurinen 2013, 65.)

4.2.1 Black-box testing

Black-box testing is among the oldest used standard testing methods and it has become somewhat of a synonym for software testing as it is the testing type most people think of when testing is mentioned. In black-box testing software unit is tested by giving it input and seeing if the unit is producing expected output.

Due to the simplicity of the method, black-box testing can be used at any point of the development where there are functionalities involving input and output such as button press events and filling out forms to offer a couple of examples. The same simplicity makes black-box tests structurally ideal for automated tests that must repeat the same check on multiple occasions. As black-box testing does require some level of functionality behind it, it is counted as a dynamic method.

The downside of black-box testing however is that because it focuses only on input and output, it disregards inspecting what happens in the software when it is running and thus the tester often does not have knowledge on the inner workings of the software. Because black-box testing does not inspect in detail what happens during software runtime, testers using this method must produce much documentation on what input was put trough and how it came out and what errors the software notified.

4.2.2 Glass-box testing

Glass-box testing is a much more comprehensive version of black-box testing as it focuses not only on input and output relations but also on how the software forms its result on runtime, something black-box tests ignore. The advantage of this testing method is that in case of errors, the tester can track the problem all the way to the base-code level and will in doing so learn about the software and its functions.

Glass-box testing however, due to its more detailed inspection, requires that the tester already has enough knowledge on the software and practical programming skills and as such it is often too difficult of a test method for beginners. As glass-box testing requires more knowledge and time than black-box testing, it is often performed at the start of the project to ensure that the initial inner logic of the software is correct before adding to it thus classifying it as static test method.

4.2.3 Grey-box testing

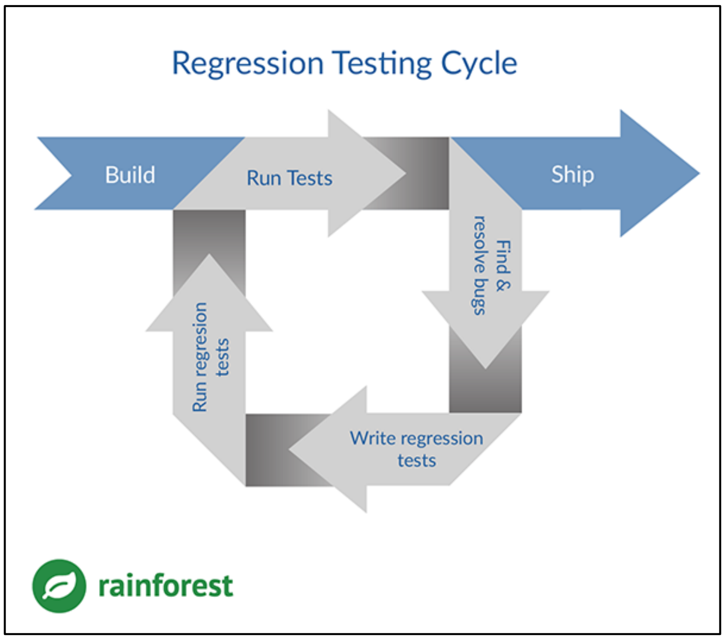
Grey-box testing is the hybrid form of black- and glass-box testing as it combines the best of both worlds by focusing on both the inner functionality of the software as well as to input and output relations. This allows for inspecting the software while also offering the option to inspect and test its individual units.

Grey-box testing is commonly seen in web-stores and similar services where developers and testers are aware of the nature of their service and can inspect it in glass-box level but also can inspect the database and other elements under their service via black-box tests. As this is a hybrid method, it cannot be classified as a static or dynamic test method.

4.3 **Regression testing**

Regression testing is a crucial part of software development that ensures the overall functionality of the software. The term however does not refer to a separate form of testing but is more of a broader term for tests that are performed to ensure that the developed version of the software is functional.

The idea and the need for regression testing lies behind the fact that often most software issues and bugs are related to newly created components or to functions utilizing said components. Thus, the purpose of regression testing is to see that after something new is added or something existing has been changed, it will not create errors that have been already fixed in previous development versions and that the software performs correctly even after the changes. Picture 5 offers a visual explanation on how regression testing is run after each build or version of the software.



Picture 5. Visual presentation of regression testing cycle. (DotterWeich 2016).

As regression tests are performed multiple times during the development for the most likely use cases whenever a new software version is created, they can be and, in some cases, should be performed as automated tests. Automating regression tests however can be infamous double-edged sword as this can cause the developers as well as testers to miss and ignore bugs and issues that may happen outside of written automated test conditions. Because of this, automated tests should have some accompanying manual tests for cases the automated ones could miss.

For the sake of easier test management and reducing rewriting test cases, it is advisable to collect tests that are run every time when a new build or update is completed and inspect the correctness of core functionalities into a regression pack. Utilizing regression packs requires that they are updated on frequent intervals during development so that the pack tests will include cases that consider any new added components.

4.4 Test Driven Development

When it comes to testing during software development, one considerable option that has submerged in recent decades is the Test Driven Development model. TDD is a testing specific development methodology and pattern where developer write a small-scale test, confirm that the error check works, write small bit of a code that passes the test, refactor the code and then repeat the steps throughout the project.

At first glance TDD can seem like a sluggish way to develop software as it gives an image of developers being able to code a miniscule portion of the software in a timeframe. However, TDD has become a viable development method as currently there are many outside tools for every code language that offer automated test solutions thus cutting down time and need to write repeatable test cases repeatedly.

The notable advantages of TDD are the reduced need for debugging as every part of the code is driven through tests before implementation, increased quality of the software due to the constantly active testing and finally TDD can decrease the overall development time. Despite the advantages TDD presents, there are still some problems with the development method as well which include the difficulty in applying it to some development projects and how it can cause the developers and testers to miss the big picture leading to classic missing the forest for the trees type of situation.

4.5 Usability testing

When developing software, it is important to evaluate the user friendliness and appeal of the product during the development process. Usability testing is the way to perform valuation on how well the software might appeal to users.

Usability testing is effective for collecting data that can be utilized for repairing or for software improvement. Allowing users to experience the software and offer feedback in notes grants the development team qualifying and quantifying data about the software.

In the past during the era of waterfall method, the deployment and utilization of usability tests were difficult as they were a resource heavy to perform and were often executed late in the development cycle. Usability testing required space to perform the test, equipment for the testers and major investment from development time for setting up, executing and analyzing the results.

As the modern software development now utilizes agile working methods, most of which require active interaction with customer, implementation of usability testing is much easier than before and no longer as heavy investment in resources. Customer is pulled into the development right from the start and can access and try out the software at different points of development thus offering valuable user experience information at multiple points of development process.

5 TOOLS, PROGRAMS & METHODS

The selection of tools for the creation of revision application has been influenced not only by the restrictions set for the application but also based on the past user and development experience. Following set limits, the selected tools allow for development onto Android platforms utilizing C#-coding language.

5.1 Visual Studio IDE

The most standard and widely used IDE for designing C#-code and Windows based software is Microsoft Visual Studio which makes development easy with its clean interface and with the possibility to utilize different kinds of modules and libraries for desired functionalities. Visual Studio has been selected as the main coding and development tool for the application project based on past user experience and because of its good customer support and expandability options with downloadable modules and libraries.

Another reason for choosing Visual Studio and by that extension C#-code as main coding tools is because C#-language offers great functionality for utilizing REST services. The capability to utilize REST API and other similar services are often necessary for functionalities that require retrieving data from online databases which are widely used in modern software instead of internal ones.

5.2 Xamarin

Xamarin is an expansion module that can be installed with Visual Studio during initial installation or afterwards which grants the possibility for multi-platform development. Currently Xamarin supports simultaneous development for Android, iOS and Windows platforms but also provides a way to code for each platform separately with C# language. Visual Studio Xamarin has proven to be an effective tool for cross-platform designing and development via experience and has thus been chosen as the go-to tool.

It is important to note however that there are some differences with coding some functionalities between developing the application to multiple platforms at once and developing the application for one specific platform using Xamarin. Because of this, it is recommended to check related official online documentation about the matter to avoid inconsistences within the code.

5.3 **Android Studio IDE**

While the application itself is written in C#-code, Java based IDE for Android applications known as Android Studio is among the selected tools for its compatibility with Visual Studio which allows the utilization of proper Android simulators for testing the application. Android emulators however are not able to imitate every function available on physical devices and as such the tools provided by Android Studio into Visual Studio are to be used for initial testing, but more accurate, conclusive testing will be done on actual Android mobile device.

5.4 **Development methods**

SCRUM was chosen as the preferred development method based on previous user experience and because of its proven effectiveness. By following SCRUM methodology development team can respond to specification changes more effectively and the weight on collaboration ensures the team is constantly keeping itself up to date on the progress of the development.

Other considerable option for development method is Kanban methodology due to its swift delivery rate. Utilizing Kanban also ensures that developers in the team do not end up taking more than they can chew in terms of workload. With Kanban method delivering results one item at a time it will also be easier to keep tabs on what has been completed and what has not.

5.5 **Testing methods**

Testing methods for the application during its development were chosen based on how time and resource efficient they are to execute alongside coding for limiting the risk of extended development time caused by separate extensive testing. Because of this, the chosen test method type for the application project will be dynamic black-box testing.

TDD has been excluded as an option as the small scale of the project with its restrictions could prove too difficult to implement TDD effectively. TDD is to be considered for implementation later in the development process when application is going to be expanded with more content and possible features.

To test the effectiveness of the application and thus its practical value, usability testing is to be utilized during development phases by collecting user feedback from the team and potential client during team and client meetings. The final usability test should be performed outside of the development team during an actual entrance exam situation with students or then by replicating circumstances trough a dummy entrance exam.

6 EXPANDABILITY OF THE APPLICATION

One part of modern development process outside of embracing the ideals of adaptability is also to support expandability and the further development of software. In the past, software was designed based on specifications set right at the start showing no reason for further development, which often led the software of the time to be non-expandable when shifted or given to another developer.

In modern software development, software expandability is encouraged through commenting the created code simply and cleanly so that any later developer can get an idea what certain parts of the code are meant to do but also by advising on using MVC design model (Model-View-Controller). In MVC model, the code is divided so that the functionalities that handle data and other background operations are under Model, parts that handle the visual elements under View and the parts of the software that include processing user inputs and other similar data are under Controller. By dividing the parts of the code accordingly to MVC model, software functionalities become much easier to inspect and MVC model allows other developers to pick up on what's been made.

Expandability is also achievable by designing software to be either mostly or entirely modular. Modular design focuses on creating each functionality of the software to be as independent as possible so that functions can be potentially reused at different parts of the code. Designing functionalities to be modular also increases functionalities reusability value, makes them easier to repair and upgrade and allows for better customization.

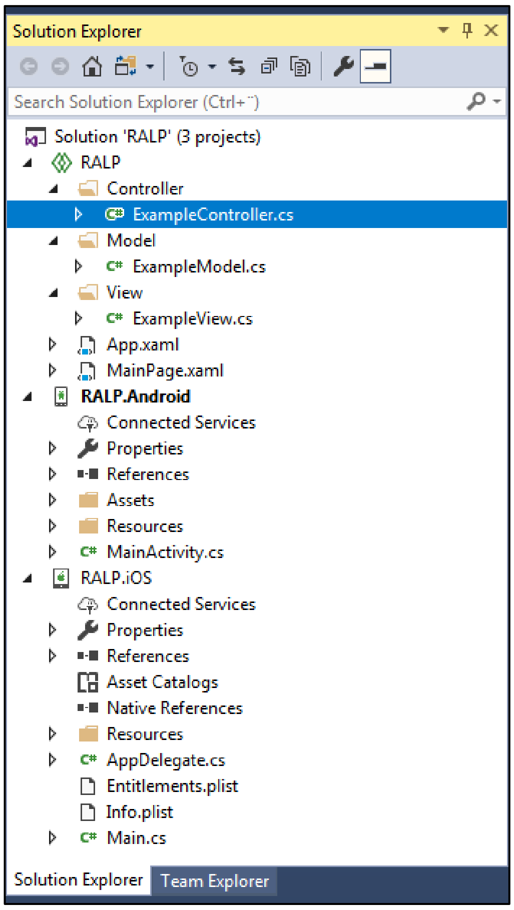
While modular functions may be independent from each other, it is required that all these modules use shared interface to ensure modules overall functionality in cases some functions end up overlapping each other. In the past, modular design was not a favored approach to improve software expandability as interfaces were often difficult to use and wrap head around and required higher than average programming skills.

7 RESULTS

By following and utilizing the tools and methods introduced within the thesis, it was possible to create a documentation according to the set specifications; clearly written application plan on how to develop a logical problem-solving application for Android devices using C#-coding tools. While the application plan certainly presents a clear, potential way for creating a practical execution, developing said practical execution was deemed too challenging within the restricted time limit there was for working with the thesis.

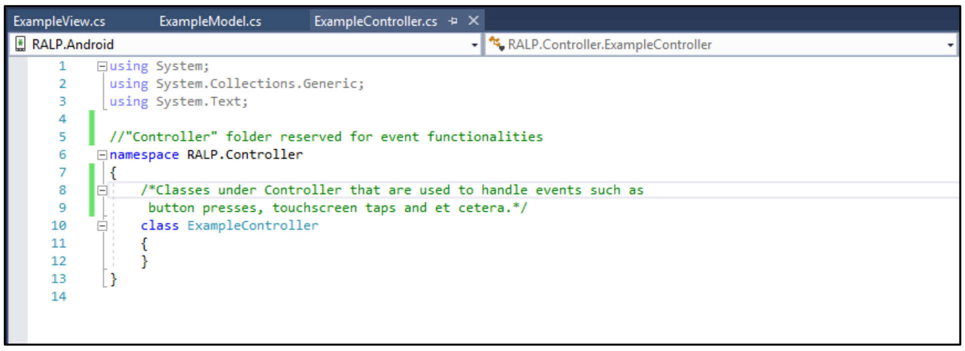
Writing the application plan proved to be relatively easy since many of the elements required in such plans were inspected within the thesis but at the same time, finding a proper template for such document was a bit difficult and as such it was opted to create one by myself. Despite the template being hand made for the most part, its structure came out professional enough to be warranted as an official application plan document.

Even though no demo application was created during this thesis due to time restrictions, it was still possible to set up a project template based on the application plan. Images trough 5 to 8 offer visual examples how the structure of the project could look like when following the plan.



```
Solution Explorer
Search Solution Explorer (Ctrl+¨)
Solution 'RALP' (3 projects)
◢ RALP
    ◢ Controller
        ▷ ExampleController.cs
    ◢ Model
        ▷ ExampleModel.cs
    ◢ View
        ▷ ExampleView.cs
    ▷ App.xaml
    ▷ MainPage.xaml
◢ RALP.Android
      Connected Services
    ▷ Properties
    ▷ References
    ▷ Assets
    ▷ Resources
    ▷ MainActivity.cs
◢ RALP.iOS
      Connected Services
    ▷ Properties
    ▷ References
      Asset Catalogs
      Native References
    ▷ Resources
    ▷ AppDelegate.cs
      Entitlements.plist
      Info.plist
    ▷ Main.cs
Solution Explorer | Team Explorer
```

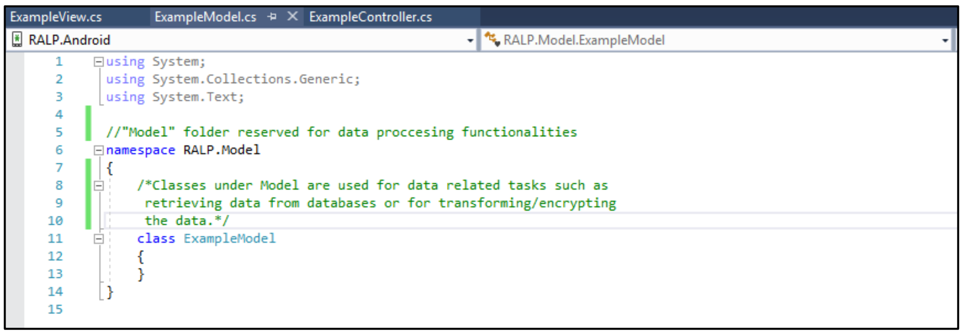
Picture 5. Example structure for the project.



```
using System;
using System.Collections.Generic;
using System.Text;

//"Controller" folder reserved for event functionalities
namespace RALP.Controller
{
    /*Classes under Controller that are used to handle events such as
     button presses, touchscreen taps and et cetera.*/
    class ExampleController
    {
    }
}
```

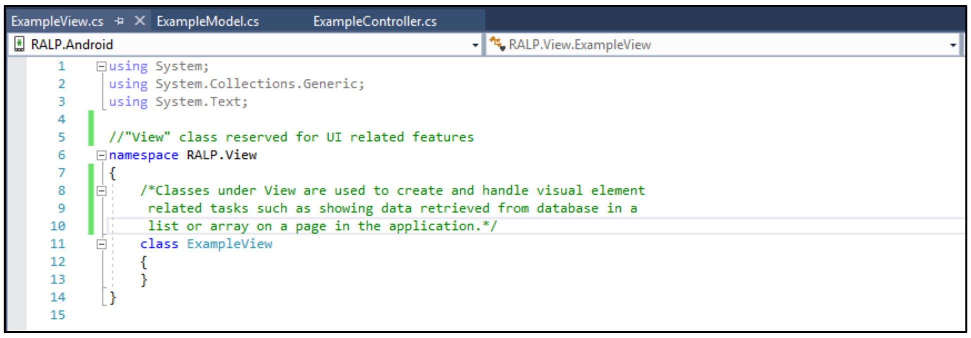
Picture 6. Explanation on Controller folder and for classes under it.



```
using System;
using System.Collections.Generic;
using System.Text;

//"Model" folder reserved for data proccesing functionalities
namespace RALP.Model
{
    /*Classes under Model are used for data related tasks such as
     retrieving data from databases or for transforming/encrypting
     the data.*/
    class ExampleModel
    {
    }
}
```

Picture 7. Model folder and its classes explained.



```
using System;
using System.Collections.Generic;
using System.Text;

//"View" class reserved for UI related features
namespace RALP.View
{
    /*Classes under View are used to create and handle visual element
     related tasks such as showing data retrieved from database in a
     list or array on a page in the application.*/
    class ExampleView
    {
    }
}
```

Picture 8. What View folder is for in the project.

The demonstrated structure is only one of many ways to create and maintain cross-platform application project while still sticking true for the written plan. UWP is left out from demonstration structure as UWP requires that the PC used for development must have Windows 10 Operating System installed which is the same OS that Windows mobile devices have installed on default.

8 CONCLUSIONS

The focus of this thesis was to investigate and as a result present a potential way to develop a revision application focusing on logical problem solving for Android devices in the form of application plan and the overall result is satisfactory. The methods, tools and concepts introduced within the thesis were heavily based on the past positive use experience and own intrigued research.

The theoretical portion of the thesis focused on investigating and presenting the principals and restrictions that modern software developers must face and respond but it also introduced some of the tools and options developers have available for their projects. The practical portion of the thesis was the creation of an application plan based on the topics discussed and explored in the theory portion.

While it could have been possible to perhaps create a practical demonstration based on the application plan, due to time constrains the focus was kept on writing the plan. Based on the plan it was, however, possible to set up and demonstrate an example project template from which this kind of project could be built from.

While writing the thesis and the application plan, it became very clear that there are many elements and requirements to keep in mind while developing applications and software with modern software development methods. Particularly researching and writing down differences in design mentality in traditional and mobile development was very enlightening. Overall, working with the thesis has cleared and helped to understand modern software development in more detail than before and it shall no doubt prove to be invaluable for future career paths.

REFERENCES

Anonymous. (2018). Interview with Junior Software Developer 20.2.2018. Ambientia Oy.

Dotterweich, A. (2016). What is Regression Testing? Blog publication on 05.07.2016. Retrieved on 26.2.2018 from https://rain-forestqa.com/blog/2016-07-05-what-is-regression-testing/

IndiaBix. (n.d.). Logical puzzles. Retrieved on 02.19.2018 from https://www.indiabix.com/puzzles/logical-puzzles/

Kasurinen, J. P. (2013). *Ohjelmistotestauksen käsikirja*. 1st print. Jyväskylä: Docendo Oy.

Lotz, M. (2013). Waterfall vs. Agile: Which Methodology is Right for Your Project? Retrieved on 02.15.2018 from https://www.seguetech.com/waterfall-vs-agile-methodology/

McKay, D. R. (2017). Are You a Problem Solver – See Why You Need This Essential Skill. Retrieved on 01.18.2018 from https://www.the-balance.com/problem-solving-525749

Mikkonen, T. (2004). *Mobiili-ohjelmointi*. Helsinki: Talentum.

Nylund, T. (2017). Mobile Programing Modules Cross-Platform Development materials, Moodle. Häme University of Applied Sciences. Retrieved on 01.13.2018 from https://moodle.hamk.fi

Santos, J. M. D. (2017). XP, FFD, DSDM, and Crystal Methods of Agile Development. Retrieved on 01.19.2018 from https://project-management.com/xp-fdd-dsdm-and-crystal-methods-of-agile-development/

VersionOne. (n.d.). Agile 101 General Learnings. Retrieved on 01.19.2018 from https://www.versionone.com/agile-101/

Appendix 1

Structured Interview in Finnish

KYSELY

Tämä kysely on osa opiskelija Juho Jauhiaisen opinnäytetyötä. Vastatkaa kysymyksiin totuuden mukaisesti ja mahdollisimman kattavasti. Nimet ja vastaukset käsitellään anonyymisti, ellei toisin toivota. Aikaanne ja vastaamistanne arvostetaan suuresti.

1. Nimenne

2. Yrityksenne & työtehtävänne

3. Mitä pidätte nykyaikaisen mobiilikehityksen haasteina? Mitä mobiilikehittäjän pitää ottaa huomioon?

4. Mitä ketteriä tai muita työmenetelmiä olette käyttäneet/suosineet projekteissanne?

5. Sovellusten kyky laajentua on usein mainittu tärkeäksi elementiksi nykyaikaisessa sovelluskehityksessä. Miten tämä on kokemuksestanne nähtävissä ja toteutettavissa mobiilisovelluksissa?

6. Mikä on ollut lähestymistapanne testausmenetelmiin kehityksen aikana? Oletteko hyödyntäneet Test Driven Development kehitystä?

7. Kuinka usein ja miten olette hyödyntäneet käyttäjä pohjaista testausta (Usability testing) projekteissanne?

HAMK Häme University of Applied Sciences

Revision Application for Logical Problem-Solving

RALPS-Project Application Plan

[Team Member Name]
[Team Member Name]
[Team Member Name]
[Team Member Name]

Juho Jauhiainen
21.2.2018

VERSION CONTROL

Version	Date	Changes	Done By

Contents

1. Introduction

RALPS-Project and the application associated with it are result of a thesis work in Business Information Technology. The resulting application that this project and application plan produces can be used for educational purposes, both for the limited subjects introduced in this plan and potentially even to other areas of study.
For the highest efficiency in resource and project management, this project is to be done with maximum of four-man team as any higher team member count could lead to problems in assigning tasks and managing materials everyone produces.
This application plan shall demonstrate and determine the functions and limitations of the applications as well as introduce the development and testing methods that are used for the project.

2. Terms

Some of the more important terms used in this application plan:

IDE	Integrated Development Environment, tool used for coding software.
C#	Windows based coding language.
MVP	Minimum Viable Product, the most primitive demo version of the product.
TDD	Test Driven Development, development model that heavily utilizes testing.
API	Application Programming Interface, collection of methods that can be used between software components.
REST	Representational State Transfer, architectural style used to develop Web services in software.

3. Application Goals & Functions

This chapter introduces and explains the overall purpose, background and functionalities of the application.

3.1. Description of the Application

The RALPS-projects application goal is to offer an efficient and quick way to perform last minute revision on any subject matter with smartphone and other mobile devices. However, some limitations and requirements have been set to keep the development as organized and manageable as possible.

These restrictions are as follows:

- The application is made for Android platform
- Application is to be coded with C# language
- Application must allow for expandability
- Revision material will be limited to logical problem solving

3.2. Background

The application plan is the result of a thesis work that focuses on exploring potential way to develop this kind of application. The thesis subject matter and by that extension this project was born from own personal experiences in entrance exams. The limitations and restrictions of the application also stem from the thesis.

3.3. Functions

At launch window, user will be given a change to choose a subject which they can then revision although for this plan only default option will be logical problem-solving. Idea is to have the application be expandable so that future developers can addition more subjects into the application if they follow established design and structure.
Initial approach and structure to revision subject is to present the user with series of questions which have multiple choice answer option and at the end user can compare their answers for the right ones. With some questions it could be considered possible for user to write their own free-word answer.
User can also save the questionnaire results into a separate list or memo in the application from which it's easier to review the answers and results for quick revision. This option initially could be a button in the launch view but would only be visible after something has been saved to the memo or list.
The application approaches the rehearsal of logical problems via logical reasoning pattern puzzles which are often seen in exams and interviews because of their simple nature and easy implementation to the application. Logical reasoning puzzles are to be initially utilized as embedded resources in the application assembly but later in the development for the sake of supporting expansion this can be changed to utilizing external database.
Logical reasoning puzzles are to be displayed so that user is presented a collection of patterns and the user must then select from three choice options the one they think should go to the end of pattern series.

4. Tools

The chosen development environment for this project is Visual Studio which is the most commonly used C#-based development IDE which also has options to develop to Android and even Apple/iOS platforms besides the default Windows platforms with Xamarin add-on that allows for cross-platform development. Visual Studio has also been chosen as development tool based on C#-code languages capability to utilize REST API services.
Android Studio which is the Android and Java focused counterpart of Visual Studio is also going to be utilized but mainly for it's good quality device emulator functionalities that will be useful during testing. However, as Android simulators are still incapable of displaying or emulating certain physical device functionalities such as GPS functionalities, an actual Android device will be used for testing the application.

5. Development Methods

For the development during this project, chosen agile development method will be SCRUM which enables the developers to be in a constant and active feed-back loop with the rest of the team and other sectors. As such, the team will be expected to have daily meetings and backlog refinements with the customer during the project so that throughout the development every person has knowledge on the application itself and its development progress.
Another option as a development method is Kanban in which results are delivered one at a time per feature instead of a bundle like in SCRUM methodology. Another reason for considering Kanban as secondary development option is based on its encouragement to focus on features that hold most value for both the product and the client and as such it allows faster delivery on core features. This focus on singular tasks one at a time also makes keeping track of the development rather easy.

6. Testing Methods

Black-box testing is selected as primary testing type for the development due to the applications scale. Glass-box testing is advised to be used alongside black-box testing when application is going to be expanded outside of the restrictions described in this plan.
TDD is excluded as development and testing method as it has been deemed too difficult to implement into a project of this small scale.
Usability testing is performed during development via team and client meetings. Final usability testing is best to be performed with students in an actual entrance exam situation or then imitate the circumstances via dummy exam scenario.

7. Results

The expected result for this project is at the very least a MVP of the application that can demonstrate the core functionalities, but it would be preferable to achieve a proper demo version of the application that could also demonstrate some signs of expandability.

8. Risks

Risk	**Consequen-ces**	**Remedy**	**Likelihood (P) Scale 1 to 5**	**Effect (I) Scale 1 to 5**	**Risk value (P*I) MAX = 25**

9. Conclusions

This document has presented the limitations and requirements of the project and the functionalities of the application as they have been inspected and re-searched in the associated thesis work. It introduces one potential way of devel-oping desired revision application based around the set restrictions for its devel-opment. This application plan is aimed for future Business Information Technol-ogy students to use as base for ICT or other similar programming project but also for those developing educational software.