HD Research Report

COS30017 - Software Development for Mobile Devices

Daniel Parker 971328X

November 21, 2014

Abstract

This research report seeks to show the findings of using the RAPPT Android prototyping tool as an initial design and construction tool for Android app development. More specifically it hopes to prove that the use of such a prototyping tool can significally shorten the development timeframe of an app from planning to alpha-level executable. The report also covers any pitfalls of using the tool and tries to identify areas of improvement for the tool to increase it's viability in mainstream application development.

0.1. Introduction

0.2. Method

The goal for this research method was to closely follow a 38 hour week proper development cycle, as one would expect a professional developer to do in the commercial environment. The 38 hour weeks are worked by a single developer only.

1. Conception of app idea happens prior to the timed process as it isn't taken to be an important factor into this study.
2. The date that planning and design begins on is recorded. The time that this takes to complete is also insignificant when assessing the prototyping tool, however the importance of this step occurring is paramount due to it laying the foundation for the developer to continue smoothly onto the prototyping stage and not mistakenly label the planning and design stage as part of the prototyping stage, which it is not.
3. Once planning and designs are complete, the date is recorded as the date that prototyping begins.
4. The app is prototyped using RAPPT as many times as needed until the developer feels they have a base from which they have achieved all they can using the prototyping tool. In other words, if the prototyping tool cannot implement anymore of the features or layouts of the app then the prototyping should cease. The date is recorded for when this occurs.
5. Record the date as the start of extending/non-prototype development. The developer should prioritise the main features of the app above other aspects and ensure that they are implemented on top of the codebase supplied by RAPPT. Make notes of the areas of the app that were made easier to develop on by the prototype and those areas that were more difficult to continue developing on.
6. Once all major/critical components of the app are functional and stable the developer should improve the visual, usability and overall stability of the app until they feel it has reached an alpha release level. That is, there may be bugs present, however the main app features are functional and should be usable by initial-uptakers/testers. Record this date as the termination of alpha development.
7. Run 'lines of code' counter on the original prototype and the final alpha source code to see how much was generated vs. how much was an extension of the prototype. Estimate based on lines of code written and time taken how much time was saved by the prototype tool.

0.3. Results

	Java	**XML**	**Total**
Blank Project	0	18	18
Prototype	384	335	719
Alpha	1668	867	2535

Table 0.1.: Source lines of code

Phase	**Days**	**Hours**
Prototyping	1	7.6
Extending	7	53.2
Total	8	60.8

Table 0.2.: Time worked on each section (rounded to the nearest day)

0.4. Analysis

The results gathered can be used to calculate the efficiency of using the prototyping tool versus starting a project from scratch. The formula for calculating efficiency is said to be:

$$(Work\ Output)/(Work\ Input) * 100\%$$

Where '*Work Output*' is said to be only usable output and doesn't include wasted output.

For the purpose of this experiment we assume the Work Output to be 100% usable as it is estimated to be very close to 100%. The work input and output are calculated as time values. The work input time is the hours spent manually programming or prototyping. The work output is an estimation of the amount of time that it would have taken to write what was generated from the prototype instead of writing the prototype code, plus the time it took to extend the program to alpha release stage. We also assume that most of the prototype code remains in the prototype and minimal amounts have been removed.

Percentage of prototype code in alpha codebase

$$719\ LOC/2535\ LOC \times 100 = 28.4\%$$

NOTE: LOC = Lines of Code

Percentage of time that prototyping took

$$7.6\ hrs/53.2\ hrs \times 100 = 14.3\%$$

From these two calculations it's seen that 28.4% of the code was able to be 'written' in 14.3% of the total time due to the prototyping tool.

Time it would have taken to write an equivalent amount of code to what the generated prototype contained.

$$Manually\ written\ code = 2535\ LOC - 719\ LOC = 1816\ LOC \tag{0.1}$$

$$Manual\ code\ time = 53.2\ hours \tag{0.2}$$

$$\frac{719}{1816} = \frac{x}{53.2} \tag{0.3}$$

$$x = 53.2 \times \frac{719}{1816} \tag{0.4}$$

$$x = 21\ hours \tag{0.5}$$

Efficiency

$$(21\ hours + 53.2\ hours)/60.8\ hours \times 100\% = 122\%$$

0.5. Discussion

These results show that there is a definite increase in efficiency to using the RAPPT prototyping tool as the total efficiency is estimated to be at 122 percent. There are however some considerations from the process of using the tool that should be mentioned. The tool is intended for professional use, so the observed efficiency in the case of this experiment may be much higher than if the tool was being used by someone with many years in Android development experience compared to my 3 or so months of experience.

There are areas of feature development that the prototyping tool already performs well in, but there are also some issues with generated code being slightly outdated and needing modifications as well as features that are just not able to be implemented yet in the prototyping tool. Some of these include;

- The generated error dialog using an older 'Holo' theme rather than leaving the theme selection to the currently selected app theme.
- ListView and List adapters are outdated and should be replaced with RecyclerView and RecyclerView.Adapter which makes the process of creating views a lot easier.
- Network error handling seems limited and does not allow for easy integration of response interceptors for bad requests.
- App theme uses older 'Holo' theme and could be updated to use the AppCompat themes.

- Generated layouts are still very primitive and limited in their complexity. Most of it was thrown away and replaced with custom code.
- The process of actually prototyping is very cumbersome and requires a repetition of writing dsl, generating a project, downloading it, opening it, and then compiling it to see if it is correct. A single cycle can take up to 10 minutes making it quite frustrating.

  The positive of using the prototyping tool has been that very early in the development process, the issues arising from a design can be realised and quickly refined. This process is much more cumbersom and can take a lot longer if done at the manual coding phase of development. There are other benefits to being able to generate interactive prototypes, including initial usability testing and validation of navigation flows in the app as well as testing other software such as server APIs. This is known across all software development platforms and is taught in engineering courses as an important point to managing software development. The cost of making changes increases as the project progresses, and therefore the best time to be making changes is early in the prototyping phases while the cost of doing so both in terms of time and money are still relatively low.

  Chapter 4 of 'Building Mobile Experiences (Bentley, F, Barrett, E)' explores the concept of 'Rapid Iterative Prototyping'. They suggest that a prototype should only implement what is absolutely necessary for the app to function, and not to include smaller less important features that one would normally expect from a publicly available application. Another important point made from their publication is that a prototype should be built to test the experience of using the app rather than the technologies that the developers intend to build it with. This ties back into earlier points that having a prototype built early in the process, allows for the experience to be tested and the prototype to be refined when the cost to do so is low.

  There have been many other protyping and mockup tools built and made available over the past few years, all offering different levels of fidelity in their prototypes. It would be an interesting study to compare the top tools to RAPPT and see the level of efficiency that the competing tools have. MIT have developed a system similar to Swinburne University's RAPPT, however with their tool, the creation is done using a visual editor and blocks. This approach is possibly also quite viable as is testament by the millions of generated apps by their App Inventor system.

0.6. Conclusion

In conclusion, the RAPPT prototyping tool does increase the efficiency of developing an Android application, it also facilitates the developers to exercise a commonly followed software engineering principle, that is to make changes earlier in the development timeline rather than later. In saying this, however the actual process of prototyping using RAPPT and the actual generated code could be refined and improved to further increase the benefits of using the tool.

0.7. References

1. *What is the formula for calculating efficiency?*, Accounting Tools , viewed 21 November 2014, `http://www.accountingtools.com/questions-and-answers/what-is-the-formula-for-calculating-efficiency.html`.
2. Barnett, S, Vasa, R, Grundy, J 2014, *Rapid Prototyping of Mobile Apps*, VL/HCC.
3. Bentley, F, Barrett, E 2012, *Building Mobile Experiences*, Cambridge, Mass: MIT Press.
4. *About Us/MIT App Inventor*, MIT, viewed 21 November 2014, `http://appinventor.mit.edu/explore/about-us.html`

0.7.1. Tools used

CLOC - Tool to count the lines of source code in a directory not including comments or non-source files. `http://cloc.sourceforge.net/`

A. Raw Results

A.1. Counted Lines of Code

```
      25 text files.
      25 unique files.
       0 files ignored.

http://cloc.sourceforge.net v 1.62  T=0.24 s (103.7 files/s, 3716.8 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Java                            13            140              0            384
XML                             12             36              1            335
-------------------------------------------------------------------------------
SUM:                            25            176              1            719
-------------------------------------------------------------------------------
```

Figure A.1.: CLOC results for the prototype project

```
      59 text files.
      59 unique files.
       0 files ignored.

http://cloc.sourceforge.net v 1.62  T=0.67 s (88.5 files/s, 4819.2 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Java                            29            403            199           1668
XML                             30             62             15            867
-------------------------------------------------------------------------------
SUM:                            59            465            214           2535
-------------------------------------------------------------------------------
```

Figure A.2.: CLOC results for the alpha project

```
       4 text files.
       4 unique files.
       0 files ignored.

http://cloc.sourceforge.net v 1.62  T=0.10 s (41.4 files/s, 258.7 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
XML                              4              5              2             18
-------------------------------------------------------------------------------
SUM:                             4              5              2             18
-------------------------------------------------------------------------------
```

Figure A.3.: CLOC results for a blank new project