International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org, editorijaiem@gmail.com
Volume 2, Issue 12, December 2013 ISSN 2319 - 4847

ResPCT – A new Software Engineering Method

Adnan Shaout[1] and Cassandra Dusute[2]

The University of Michigan – Dearborn
The Electrical and Computer Engineering Department, Dearborn, MI, USA

ABSTRACT

While there are many software processes currently available, they are usually focused on large projects with multiple programmers. True, there are processes geared towards one-man teams, however, there is still the assumption that the project in question is a larger one. This paper introduces ResPCT, a software process geared toward the one-man team working on a small to medium sized project. ResPCT requires four phases, Research, Prototype, Create and Test. ResPCT was applied to various applications and showed a great success as a new software method.

Keywords: Software Process, Application Software, Design for Testability, Software Testing, Prototypes, Small software developers

1. INTRODUCTION

Processes help guide the outcome of software. Without one, the probability of failure significantly increases. As software developers, many existing software processes did not apply to our needs. Sometimes a software engineer may work on small projects that do not require much organization, and requirements are more often than not few and easy to understand.

One of the biggest problems software engineers face with current processes is that they seem to be time consuming when what is needed is a process that can be followed which is quick and powerful.

ResPCT gives a basic guideline for a software engineer to start a new project. It allows getting the necessary information that is needed while remaining time efficient. ResPCT also gives a process framework that can be customized based on the type of project.

ResPCT includes four of the fundamental phases necessary for any software process. These phases are as follows:

- **Research** – Requirements gathering. Here, you learn more about what the project is and what your client wants and needs.
- **Prototype** – Before actually starting development, it is important to have a starting point. This phase of the process allows you to create a working prototype of the project. A prototype can include drawings on a napkin, basic wireframes, or a fully-functional design. Once this prototype is approved, then development can begin.
- **Create** – Develop your application with testing in mind. Refer to the prototype for guidelines on how the project should be completed and what requirements are necessary.
- **Test** – Write specific test cases to robustly test your program. Do not change the tests to make them pass. Change the code to pass the tests.

The ResPCT model is very similar to the Waterfall and Sashimi models. In fact, ResPCT is originally derived from using these models. However, the two models mentioned were limiting in their own ways. For example, with the Waterfall model, you are unable to move backwards through the phases. With the Sashimi model, while you can move backwards, there are extra phases that may not be relevant for small projects [1].

Like the Sashimi model [1] ResPCT can move back and forward throughout the different phases. ResPCT uses a technique called the Loop Effect, which is a way of returning to previous phases through other phases after matching certain criteria (much like a loop).

The Loop Effect works by returning to a previous phase, checking the necessary items, and then returning to the next previous phase, if applicable. For example, as a developer if you are currently in the Create phase when you notice that one of the requirements is set to endanger the delivery date. To fix your issue, you decide to go back to your customer and suggest an alternative approach. To move from Create to Research, you would first start by returning to the Prototype phase to check if there is anything in the Prototype that could offer an alternative. From there, you would return to the Research phase, suggest your alternative to the client and on approval move back through the phases. Figure 1 gives the basic diagram of how the Loop Effect works.

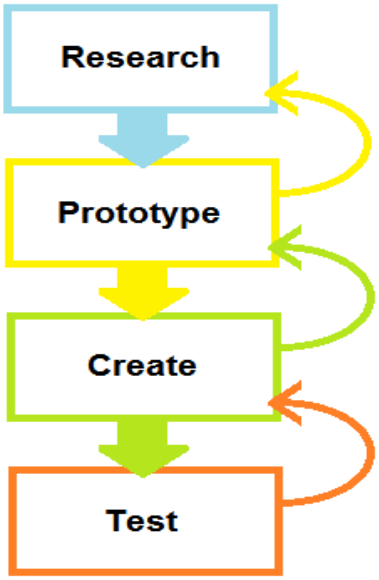

Figure 1. Loop Effect used in ResPCT.

2. DEVELOPMENT WITH ResPCT

To test the effectiveness of ResPCT, the process was applied to the development of an iPhone application. The procedure used by Chang [2] to create a mobile health application was used as a reference in developing with ResPCT. Chang relied heavily on the research stage with their application. Chang project [2] included a basic development structure on which this customized version of ResPCT was built.

2.1 Research

Research for the iPhone application started with requirements gathering. Requirements for this application included the following:

- Create a working prototype of the application
- Include 5 movie questions, ranging from easy to hard
- Have something happen whenever a wrong answer is selected
- Allow user to go back to a previous question
- Allow user to skip a question
- Allow user to answer the questions again after they have all been answered
- Test the application thoroughly. Tests should include the following:
  - Answering a question correctly
  - Answering a question incorrectly
  - Returning to a previous question
  - Starting the game
  - Starting the game from the finish page
  - Skipping a question
- Finish the application project within a three weeks time frame.

The time frame made it easy to assign time to phases accordingly. Each of the following phases (prototype, create and test) were allocated a week to be completed. If one phase was finished earlier, then move to the next phase, giving more time to it if needed.

To keep a record of requirements, a program called iRise was used [4]. iRise is a robust prototyping tool. It can be used to prototype websites, applications and other computer programs. The requirements were added to the prototype so that they were easily accessible.

2.2 Prototype

A prototype is an example of how a program should work. A prototype gives the developer a starting point, along with a visual of some, if not most, of the requirements. iRise was used to create the prototype. The program was used to create a fully-functional prototype that allowed the user to experience how the application would essentially work.

The process started by creating a basic scenario. From there, the pages were added as needed, along with the logic each question needed. Figure 2 shows how this flow works. By doing this, the pages and items needed were made in the form of placeholders.

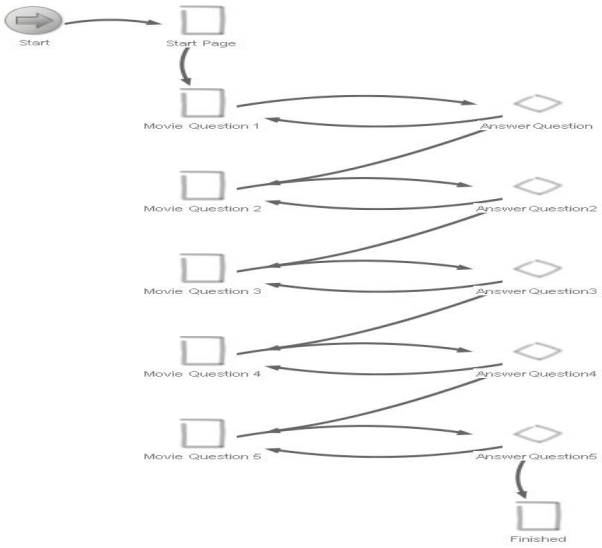

Figure 2. Flow Chart of the Prototype.

From there, proceed to the pages and built them to fit the idea of the design. Each question page includes a header, question, four possible answers, a return button and a skip button. Figure 3 shows a sample of how the prototype was built.

After the pages were built, the formatting and necessary logic for the main application was added. When a user runs through the prototype, they can answer all of the questions, and depending on whether the question was answered correctly or not, they would see an action. The logic statements for the questions were simple if-else decisions, which are detailed in Figure 4.

These logic statements, called decisions in iRise, are activated when the user clicks on an answer. The value of the answer is sent via a clipboard (or memory) value to the decision logic and compared with the correct answer. If the question was answered correctly, then the decision will direct the user to the next question. If it was answered incorrectly, then the decision will direct the user back to the previous question, and the user will be able to answer the question again.

Once the prototype was finished, the process goes through all of the pages to make sure they did work like they were supposed to. If a different client is there, then send the prototype to the client and ask that they run through the prototype. If they had any changes or the prototype was missing any requirements, then they could add comments to the page with the errors. From there, the necessary changes would be made until the client approves the prototype, and then move onto the next phase of ResPCT which is Create.

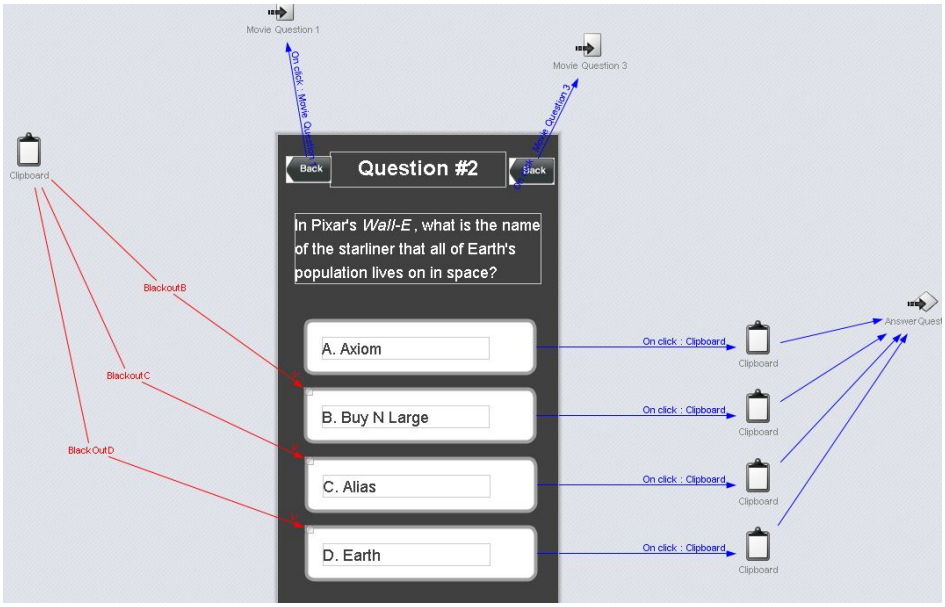

Figure 3. Sample from MovieGame Prototype.

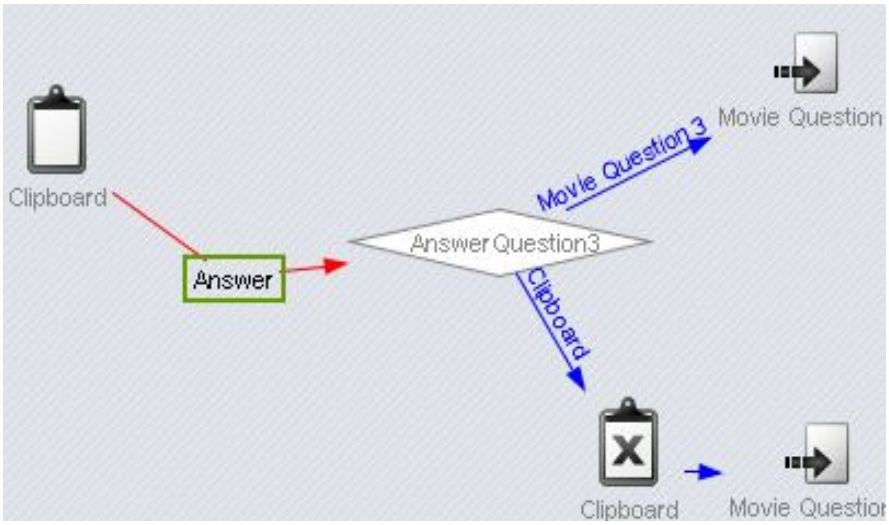

Figure 4. Prototype Logic.

2.3 Create

With the prototype in hand, the development of the iPhone application will start. The application was designed and developed relying on the prototype from the previous phase for guidance. Xcode [5] was the program used to create the application. This software is a free developer tool available for Mac OS used for creating iOS and Mac OS applications. It comes as a toolkit which includes what you need to start creating applications along with an iOS simulator.

This phase of the process started in a similar way as the prototyping phase. It started by adding seven pages to the application storyboard (a sort of white board Xcode used to build pages). From there, the elements that are required were added on each page based on the prototype. Figure 5 shows the MovieGame application in Xcode and running in the simulator.

During development, a problem with one of the elements in the prototype did come up. Originally, the prototype changed the color of the answers when the question was answered incorrectly. Due to time constraints, a change to wrong answers to produce a pop-up instead was made.

To do this, the Loop Effect was used by first going back to the prototype phase and changing the wrong answer to produce a pop-up instead of changing the color of the button. Then the process went back to the Research phase to check the requirements. The requirement said to have something happen when a wrong answer was chosen. Since a pop-up is an action, it fell within the requirements. With this, the process moved back to the prototype, made sure the pop-ups worked, and then implemented the pop-ups in the create phase.

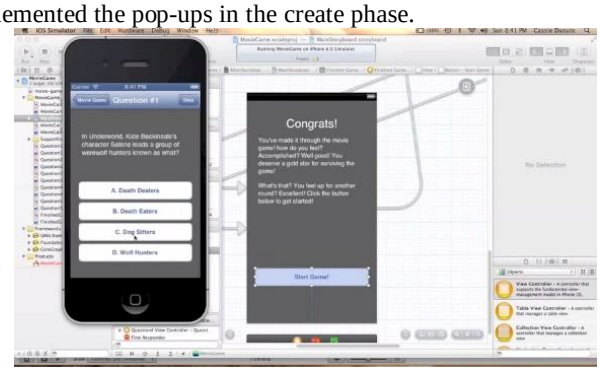

Figure 5. MovieGame Application in Xcode.

2.4 Test

One of the requirements of this project was to include vigorous testing. With any process, testing and quality assurance should be one of the most important steps. The Test phase is the last step when working with ResPCT.

For the MovieGame application, a testing platform called Frank [7] was used. Frank is made up of Cucumber and Ruby and is used to automate iOS application testing. Frank uses given-when-then statements hooked up to Ruby code to run the automated tests. A given-when-then statement is a starting condition (given), an action that causes a change (when), and then the result of that action (then). Here is an example of a given-when-then statement taken from the tests written for the MovieGame application.

- Given I am on the "Movie Game" page
- When I press the button "Start Game"
- Then I should see "Question #1" page

The quotations around Movie Game, Start Game and Question#1 turn those statements into variables. The Ruby files – known as step definitions – will look for these statements to match to the specific step. From there, it will pass the string within the quotes to the ruby function. Figure 6 shows few examples of these step definitions.
Each step definition does a specific task. To build each step definition, you first need to know what task it is going to do. Then, you need to know what type of view you will need. There are many different views – UIRoundedRectButton, UILabel, and UIAlertButton are just a few. Frank comes with a tool called Symbiote, which displays the application in a browser with a list of the accessible views. Finish building your step definition can be done using Symbiote.
After all of the step definitions were created, then ran the tests against the application. If a test failed, then first make sure that the test was correct, and if it was, then go to the application and fixed what needs to be fixed. Once all the tests are passed, then the MovieGame application was complete.

```
When /^I press the button "([^\"]*)"$/ do |mark|
    sleep 2
    touch("view:'UIRoundedRectButton' marked:'#{mark}'")
end

        When /^I see the "([^\"]*)" pop up$/ do |mark|
    sleep 2
    check_element_exists("view:'UILabel' marked:'#{mark}'")
end

When /^I press the alert button "([^\"]*)"$/ do |mark|
    sleep 2
    touch("view:'UIAlertButton' marked:'#{mark}'")
end
```

Figure 6. Sample tests from the MovieGame step definitions.

2.5 Evaluation

By using ResPCT as a guide for creating the application, the project within the timeframe was finished as required. ResPCT helped keep everything in order and on track. As a small program developer, you may have no time to use a large, elaborate process. ResPCT is designed for small program developer people to stand-in as a guideline for getting everything important finished. Based on the result of the iPhone application, ResPCT did the application successfully.

3. RELATED WORK WITH ResPCT

To further extend ResPCT capabilities, it was decided to test how ResPCT works with adding requirements to an already finished iPhone application. The MovieGame application was taken and decided to localize it. To localize an application simply means to make it available in other languages. Based on a report conducted by Distimo, an application analytics company, localization not only boosts revenue and downloads; it also makes the application accessible to non-English speaking users [3].
The phases for localizing an application are as follows:

- **Research:** To start, choose what languages would be most beneficial. It was decided to translate the application into Chinese and Spanish because they are two of the most used languages, and French because we are close to a French-speaking country (Canada). From there, the MovieGame script was translated into these three languages.
- **Prototype:** Since the prototype for the application was already built, then a couple of pages with the translated text were updated.
- **Create:** English was added as the base language for the application. Then, under settings, the other languages that the application would be available in were added. This created an English script, or a file of all of the text in the application, for each added language. Once added, the scripts with the translated languages were edited. For the pop-ups, three string files were added – one for each language – that translated the error message on the pop-up. Then on each page, the pop-up that link to the string file was changed.
- **Test:** Testing for this was done manually. Since it was known that the application had passed all previous tests, we wanted to make sure the application changed languages once the settings were changed. To test this, settings on the iPhone simulator was visited, then the language to one of the three added was changed, then the MovieGame

application was opened. When opened, the MovieGame was translated and displayed the language that was chosen under settings. Figure 7 shows a screenshot for the localized application.

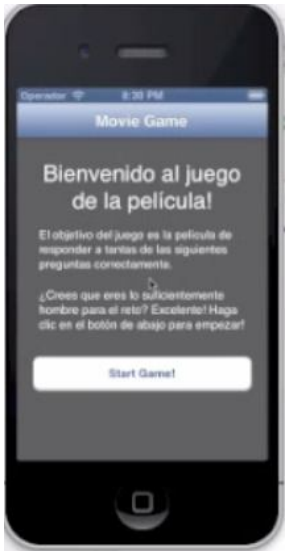

Figure 7. Localized MovieGame Application screenshot

ResPCT was also applied to a web project for a Pilates and postural health website. The goal of the project was to redesign an existing website that was previously managed in FrontPage.
The phases for redesigning the website were as follows:

- **Research:** Requirements for the Pilates website included three main things: a new design, an easy content management system that the client could use to make its own changes, all of which need to be completed in a short time line.
- **Prototype:** The prototype started by building a template in WordPress, and creating the first three pages of the website.
- **Create:** Once the client approved of the prototype, then moving existing content over to the new website would start.
- **Test:** After all of the pages and content were moved over, the website was tested by clicking through all of the pages, testing all of the links and made sure all of the images were loaded. A testing product called Selenium [6] to automate the testing process was used.

4. CONCLUSION

For program developers, it is know that it is important to have a process. Whether it is a well-known process, or a self-made one, it helps the way that programs are developed and projects are completed. ResPCT assumes that the developer works on personalizing the process to fit their own needs as well as improving themselves and the process over time. ResPCT itself is a framework process to be interpreted as the user sees fit.
From testing, it was found that ResPCT can be applied to a multitude of software projects, including applications and websites, from the beginning of the project or with a completed project. It is hoped that ResPCT can be applied by developers that are having problem of finding a suitable software process for their needs. ResPCT is most suitable for small to medium software projects.

REFERENCES

[1] "Waterfall Model" http://www.waterfall-model.com [Accessed: November 13, 2013]. (General Internet site)
[2] Chang, T., " Food Fight: A Social Diet Network Mobile Application," Berkeley: University of California, 2012.
[3] Van Agten, T., "*The* Impact of App Translations," Utrecht: Distimo, 2012.
[4] "iRise," iRise: http://www.irise.com [Accessed: November 13, 2013]. (General Internet site)
[5] "Xcode Tools you'll love to use," https://developer.apple.com/technologies/tools/ [Accessed: November 13, 2013]. (General Internet site)
[6] SeleniumHQ, "Web browser automation," http://www.seleniumhq.org/ [Accessed: November 13, 2013]. (General Internet site)
[7] "Testing with Frank, " http://www.testingwithfrank.com/ [Accessed: November 13, 2013]. (General Internet site)

Dr. Adnan Shaout is a full professor in the Electrical and Computer Engineering Department at the University of Michigan – Dearborn. At present, he teaches courses in fuzzy logic and engineering applications and computer engineering (hardware and software). His current research is in applications of fuzzy set theory, embedded systems, software engineering, artificial intelligence and cloud computing. Dr. Shaout has more than 30 years of experience in teaching and conducting research in the electrical and computer engineering fields at Syracuse University and the University of Michigan - Dearborn. Dr. Shaout has published over 140 papers in topics related to electrical and computer engineering fields. Dr. Shaout has obtained his B.S.c, M.S. and Ph.D. in Computer Engineering from Syracuse University, Syracuse, NY, USA, in 1982, 1983, 1987, respectively.

Cassandra Dusute is a graduate student in the College of Engineering and Computer Science at the University of Michigan - Dearborn