CrunchTime: A Case Study in Software Engineering on the iPhone

Keyvan Nayyeri, Robert Porter, and Faranak Sorooshian
Department of Computer Science, College of Sciences, The University of Texas at San Antonio

Introduction

Software Engineering is the study of designing, implementing and modifying software to be more efficient, bug-free and cost-effective. To that end, our study was to bring together a group of individuals with diverse backgrounds and use software engineering concepts to implement a well-designed and easily modifiable project for the Apple iPhone platform with the goal of analyzing these Software Engineering concepts.

This development platform was chosen in part because all developers on the team had no prior experience on this platform, so the team had to learn to work together to learn and implement this project.

The project chosen was a Reverse Polish Notation (RPN) calculator, which uses a stack to store and display numbers as well as postfix notation to operate on the operands.

During the process, we created functional specifications to properly understand the goals before implementation and used a waterfall method to implement our project. This careful planning proved fruitful as we were able to implement the core design of our project on time and added several additional advanced features to the calculator due to the modular design and good planning.

Goals

We had the following major goals in mind by creating this calculator:

- Develop a project to benefit the community
- Develop a project as a team
- Learn to work as a team
- Learn to use good software engineering practices
- Learn to develop for the iPhone platform
- Build a good sample for iPhone development

Requirements

At a minimum, the requirements were to implement a basic RPN calculator with user key entry, stack display, and the four basic arithmetic operations. Therefore, these requirements were defined as follows:

- Keypad for entry including 10 digits, decimal, +/-, arithmetic operators, ENT, DEL and CLR
- Display of numbers being entered in temporary storage area
- Display of values on the stack
- Allow user entry of numbers into temporary storage
- When ENT key or operator key is pressed, the value in temporary storage is pushed onto the stack
- When an operator key is pressed, it will pop the last value(s) from the stack, perform the operation, and push the result back onto the stack

Functional Specification

We analyzed what the user would need in an RPN calculator and from this, defined three basic levels for our functional specification.

Level 1 - Core functionality

- Double numbers
- Scrollable stack
- Basic mathematical operations

Level 2 - Not required functionality, but should be available

- Pop up errors
- Aesthetically pleasing interface
- Display rotation (not implemented)
- Scientific notation

Level 3 - Not required functionality, but "nice to have" if available

- Additional mathematic operations
- Use iPhone virtual keyboard (not implemented)
- Toggle decimal and scientific display of numbers
- Store values in memory (not implemented)
- Formula definition (not implemented)
- Remove values from the stack

An early implementation of CrunchTime with the core functionality

We divided the project into four components that were implemented by each of the team members separately and combined together using communication mechanisms in Object-Oriented design.

Architecture

This project used a combination of multiple architectural styles.

- CrunchTime is mainly built on top of a waterfall design approach.
- Front-End, Back-End style is used for the User Interface.
- Object-Oriented style is brought in to work with a bottom-up approach to design and implement smaller modules.

Component Design

We split our project into four main components each of which assigned to a team member.

User Interface Front End

- Design and implementation of the View and Controller
- Get input keys and pass them to User Interface Back End
- Get validated string from User Interface Back End and display it

User Interface Back End

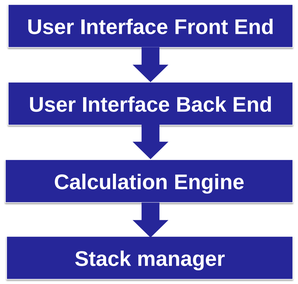
- Validate numeric entries
- Perform actions for non-numeric entries

Calculation Engine

- A bridge between User Interface and stack
- Provide interfaces for stack access
- Get operators and operands and return the result
- Some validations on data

Stack Manager

- Abstract away details of stack and value storage
- Internalize data structure representation
- Provide methods for managing stack



Component Design of CrunchTime

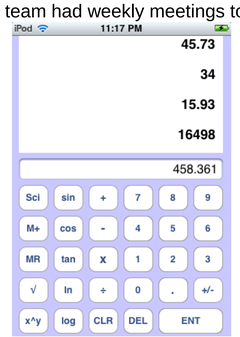
We divided the project into four components that were implemented by each of the team members separately.

Development Process

We used a combination of individual programming and pair programming from Agile Software Development (where two group members program together) to build CrunchTime.

Also we used source control (Mercurial) and bug tracking (FogBugz) systems to facilitate the communication and keep everyone on track.

The team had weekly meetings to discuss the problems and plan.



Final implementation of CrunchTime

Project could be delivered on time with a rich set of features proposed to be included. Many of the bugs were resolved and user exceptions were handled appropriately. The final result is a practical calculator for end users.

Successes

We could deliver CrunchTime on time after almost three months of development. We believe that there were some main reasons for our success.

- A good design plan
- Frequent communication between team members
- Strong collaboration
- Incremental approach in development
- Risk management
- Careful selection of Software Engineering techniques
- Using a combination of different techniques as necessary

Challenges

We also had some minor challenges during the development process, which can be classified as either technological or social challenges.

- Major differences between Apple development platform and other platforms
- Difficulty of programming in Apple platform in comparison with other platforms available
- Complicated process of debugging on Apple platform
- Lack of enough documentation for learning
- Issues with Mercurial as the source control system
- Difficulties to find the best control for stack display
- Availability of all team members as full-time or part-time students with different schedules

Conclusion

Working on a project based on the Apple iPhone platform, we could apply some common techniques and practices of Software Engineering in action, learn about iPhone development, train our team work skills, and evaluate the effectiveness of the techniques and technologies that we used.

Dividing the project into four independent components helped us split the work among team members and manage the communication problems easily. Besides, we used a combination of multiple Software Architectures and applied Agile Software Development techniques to develop the project in a smooth way.

We succeeded in implementing all our major goals and delivering the project on time, however, there were some minor challenges in regards to Apple development platform that we used that were naturally inevitable.

Research Team

Keyvan Nayyeri, Robert Porter, Neil Gandhi, and Faranak Sorooshian contributed to this project.