CMPT 276 Term Project HW5
SFUnwind - Quality Assurance Plan

*"**SFUnwind:***

"Psychotherapy-based interventions for the prevention and management of panic disorders"

Adam Badke

Berke Boz

David Magaril

Joseph Zhou

https://sites.google.com/view/sfuwind/

Table of Contents

1. Coding Standards

All SFUnwind source files must have a standard header of the following format:

```
//
// FILENAME.swift - BRIEF FILE DESCRIPTION
// SFUnwind
// Project Group 5: SFU CMPT 276
// Primary programmer: NAME #STUDENT_NUMBER
// Contributing Programmers: NAME #STUDENT_NUMBER
// Known issues:
// - BRIEF ISSUE DESCRIPTION
// - BRIEF ISSUE DESCRIPTION
//
// Note: All files in this project conform to the coding standard included in the SFUnwind HW3 Quality Assurance Documentation
```

Additionally, all SFUnwind source files are to conform to the following coding standard:

Identifiers (ie. Variable, class, method and file names):

- All identifier names must be meaningful and self-descriptive
    - Eg. theViewController
- All class names are to be capitalized
    - Eg. PanicAlertViewController
- The first letter of variables, methods and filenames are always lowercase
    - Eg.alert01.txt
- Subsequent words within a name must be capitalized using camel case
    - Eg. timerArray

Comments:

- All methods must be prefaced with a brief comment explaining the methods purpose, the argument parameters and return value
- Code should be commented throughout to ensure easy reading for other programmers

- All GitHub check-ins must have meaningful and descriptive comments.

Indentation

- Spaces are to be used for indentation, as per the XCode standard
  - Each subsequently nested block of code is to be indented by 4 spaces
- Opening braces are to be on the same line as the condtion or header
  - Eg:

```
override func viewDidLoad() {

  /** Method body **/

}
```

2. Software Tools and Automatic Unit Testing

During the development of SFUnwind, the following software tools will be used for development and automatic unit testing:

- The XCode IDE and XCTest (versions 8.2.1) will be used to perform and manage automatic unit testing of the SFUnwind application.
  - Test classes will be managed within the project and its GitHub repository.
- The XCode IDE iPhone/iPad simulator tools, as well as physical iOS devices will be used to perform manual unit, integration and system testing throughout development.
- Each programmer is responsible for writing and verifying automated unit tests for each individual system components they write.
  - Automated unit tests are to be written for each program method that cover both allowable and invalid ranges of input/output.
  - Unit tests will check correctness of return values, as well as ensuring any internal values are correctly configured
  - Each programmer must complete unit testing and fix any bugs before merging a component into the main branch.
- Each programmer is responsible for writing and verifying automated integration tests of all coupled functions within a specific SFUnwind feature screen when integrating (unit-tested) components into the main branch of the SFUnwind GitHub repository.
  - Automated integration tests will be written to cover all allowable ranges of input/output possible for each program method.

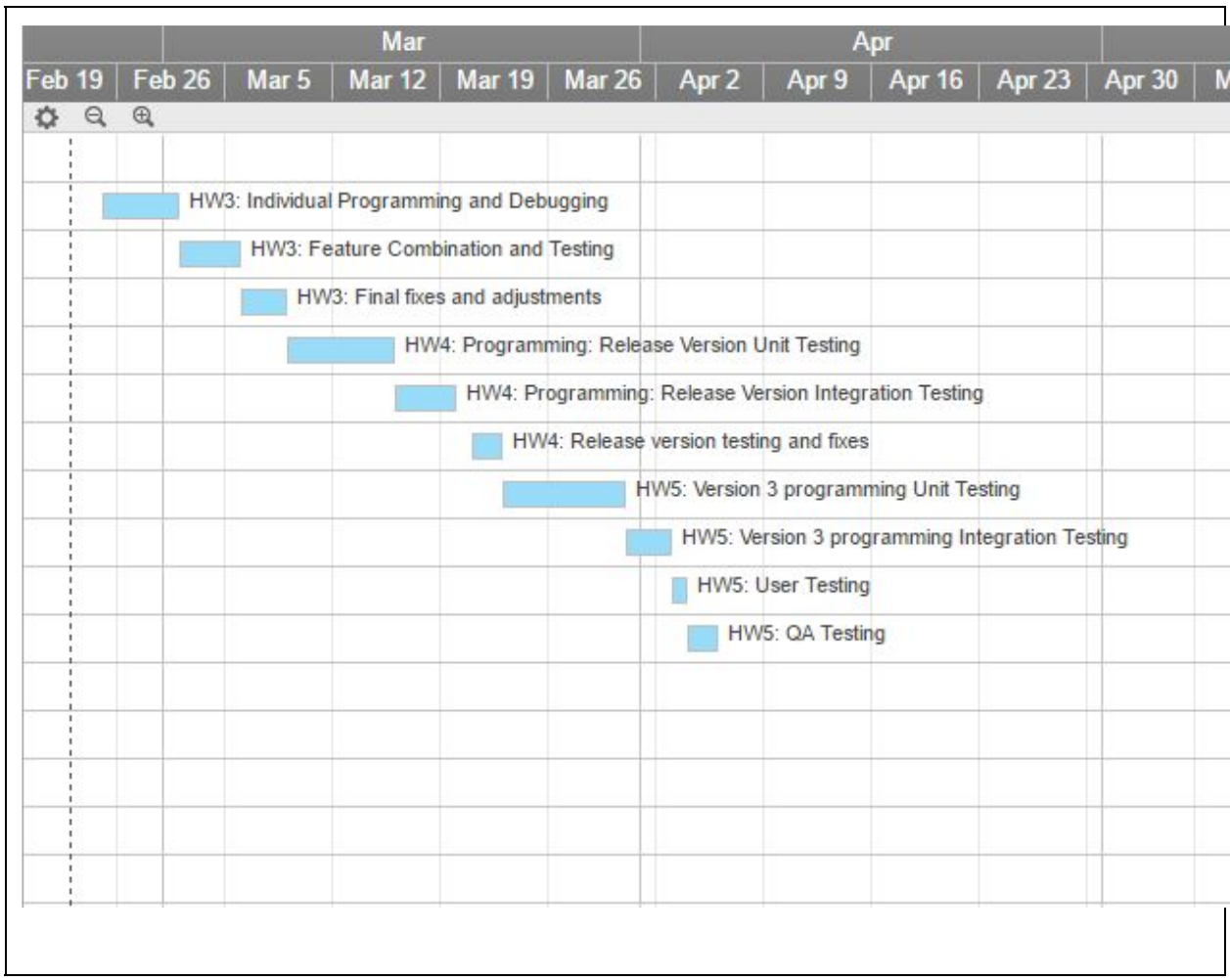
2. Internal Deadlines

Testing of SFUnwind is to be interleaved throughout the entire production cycle:

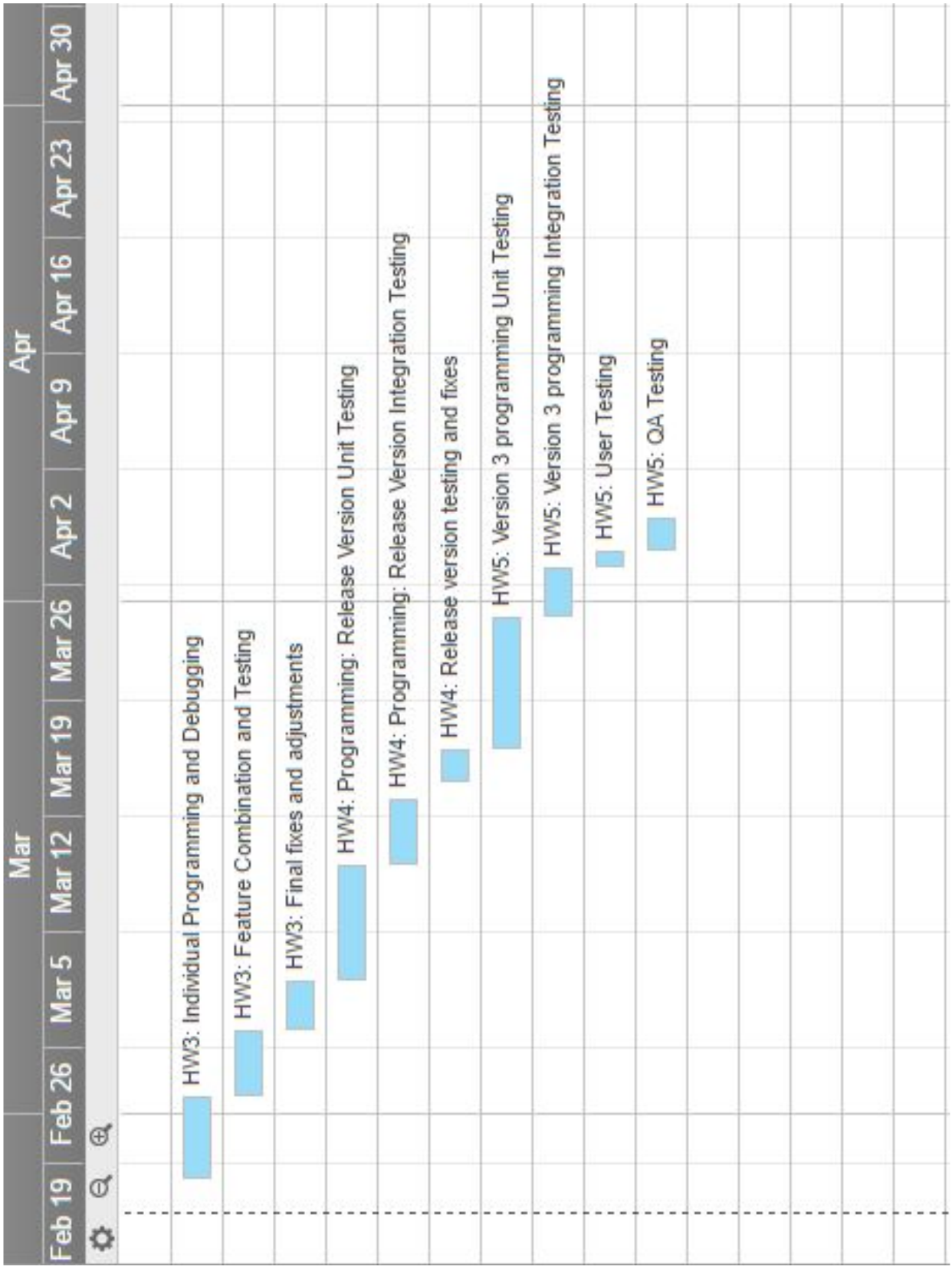
Project Task	Testing Component	Duration	Start	Finish
HW3: Individual Programming and Debugging	Unit Testing	5 days	Sat Feb 25	Wed Mar 1
HW3: Feature Combination and Testing	Integration Testing	4 days	Thurs Mar 2	Sun Mar 5
HW3: Final fixes and adjustments	System Testing	3 days	Mon Mar 6	Wed Mar 8
HW4: Programming: Release Version	Unit Testing	7 days	Thurs Mar 9	Wed Mar 15
	Integration Testing	4 days	Thurs Mar 16	Sun Mar 19
HW4: Release version testing and fixes	System Testing	3 days	Mon Mar 21	Wed Mar 22
HW5: Version 3 programming	Unit Testing	8 days	Thurs Mar 23	Thurs Mar 30
	Integration Testing	3 days	Fri Mar 31	Sun April 2
HW5: User Testing	User Acceptance Testing	1 day	Mon April 3	Mon April 3
HW5: QA Testing	System Testing User Testing	2 days	Tue April 4	Wed April 5

- *All dates in 2017 - Gantt chart below.*

Internal Deadlines Gantt Chart:



Internal Deadlines Gantt Chart(Print Enlargement):



3. User Acceptance Testing & Results

As stated in the internal deadlines above:

Project Task	Testing Component	Duration	Start	Finish
HW5: User Testing	User Acceptance Testing	1 day	Mon April 3	Mon April 3

User Acceptance Testing Details:

- User Acceptance Testing will performed on a physical iOS device

User Testing Details:

- General user testing will be performed with 6 volunteers selected from the SFU student body, who have no prior experience with the SFUnwind application.
    - Tests will be performed between April 4th and 5th, 2017
- Berke Boz and Joseph Zhou are responsible for user testing
- User Testing will performed on a physical iOS device

Specific User Testing Procedure:

- Users will be given a short briefing and overview of the intentions of the user test, and a brief, high level description of the goal of the SFUnwind application
- Users will be instructed to perform each of the tasks included in the task list below.
    - No specific coaching or instruction will be provided to test subjects
- The SFUnwind team member performing the test will observe the user, taking notes of any issues they encounter
    - Both usability and technical issues will be recorded

Test Task List:

Task #	Requirement number	Task Area	Task Description
1	1.1	Menu	Navigate between all 4 feature screens.
2	1.2		Check if active feature screen is highlighted
3	3.4		Check if Camera functionality works
4	3.3	Grounding Exercise Feature	Capture 5 images: See/Touch/Feel (15 images total).
5	3.3		Capture 4 images: See/Touch/Feel (12 images total).
6	3.3		Capture 3 images: See/Touch/Feel (9 images total).
7	3.3		Capture 2 images: See/Touch/Feel (6 images total).
8	3.3		Capture 1 image: See/Touch/Feel (3 images total).
9	3.1		Access the help screen.
10	3.2		Check displayed photo panel
11	3.2.1		Check if thumbnail photo panel's length adjusts itself after each image taken
12	3.5		View end of exercise gallery.
13	3.5		Restart exercise.
14	4.1	Positive Affirmation Feature	Display Help Screen
15	4.4		View Mantra
16	4.3		Delete Mantra.
17	4.3.1		Check if user is prevented from deleting all Mantras
18	4.5		Display Next/Previous mantra.
19	4.2		Add custom mantras.
20	4.6		Enable Mantra Notifications

21	4.6.1		Change Mantra notification frequency
22	2.3	Square Breathing Feature	Trace through animated lines until the timer reaches to 0:00.
23	2.3.1		Test if vibrations work while tracing animation
24	2.1		Reset the timer.
25	2.2		Press Info button.
26	2.4		Check current session time spent interacting with the Square
27	2.4.1		Reset current session time spent interacting with the Square
28	2.5		Check total time spent doing exercise
29	2.6		Access the stats screen
30	5.1	Panic Alert Feature	Access Help Screen
31	5.2		Check if Panic Alert Contacts feature is available
32	5.2.1		Create panic alert contact.
33	5.2.2		Edit panic alert contact.
34	5.2.3		Delete panic alert contact
35	5.2.4		Send Panic Alert to selected contact
36	5.3		Press SFU Counselling Service Website Link.
37	5.4		Press SFU Counselling Service phone number

Suitable iOS Testing Devices:

- iPhone 4
- iPhone 4S
- iPhone 5
- iPhone 5C
- iPhone 5S
- iPhone 6
- iPhone 6S
- iPhone 6S Plus
- iPhone SE
- iPhone 7
- iPhone 7 Plus
- iPhone 6 Plus
- iPod Touch 4G
- iPod Touch 5G
- iPod Touch 6G
- iPad 2
- iPad 3
- iPad 4
- iPad Air
- iPad Air 2
- iPad Mini
- iPad Mini 2
- iPad Mini 3
- iPad Mini 4
- iPad Pro

User Test Results:

Tester #1
• 5-4-3-2-1 Grounding: “Feels like a lot of work, not attractive enough for me to finish. I wouldn’t want to finish something while already under stress.”   ○ Tester switched to next screen in the middle of capturing, after he took ~15 photos • Positive Affirmation: “This is fun and useful for preparing for big future events. It’s like an AI friend to cheer you up at a certain time.” • Panic Alert: “This would be useful when they are really freaking out, specially for international students who don’t have family here but only few friends. It’s like an emergency contact” Notes: Figured out how to use the app by himself without instruction
Tester #2
• Seemed to like the Positive Affirmation feature the most. Played with the various features on this screen for over 20 seconds • Spammed image capture in the Grounding Technique screen • Suggested having detailed tutorials • Liked the app in overall
Tester #3
• Tester expressed they had suffered panic attacks in the past • Said he feels much more relaxed and calm after using the app • Added “This app made my day”
Tester #4
• Tried to press dots in Square Breathing Exercise

● Found help button ● Didn't read whole text in help instructions ● Tried to swipe text input box in Panic Alert Edit Screen ● Said the places of start and statistics button were confusing
Tester #5
● Achieved every goal without an issue ● Asked for camera focus functionality
Tester #6
● Panic Attack Sufferer ● Was very positive about everything ● Really liked the send panic alert feature
General Comments:
● "I could see this being really useful when you feel anxious or if a sudden panic attack was coming on - Like right before interviewing or presentation, for example." ● Navigation: Almost all users were able to navigate between features ● UI: Users typically liked the color scheme

4. Integration Testing

- SFUnwind programmers are to integrate changes as frequently as possible to ensure problems are identified early:
    - New code functionality must be integrated every 3 days at maximum (from initial file creation or GitHub checkout of existing file).
        - Programmers must never perform “big bang” integration.
    - Programmers must never lock others out of project critical files on GitHub whenever possible
- The SFUnwind will be Integration Tested with a “bottom-up” testing approach:
    - Low level components are to tested prior to higher level classes.
    - Test class drivers will be written to link coupled lower level components together for higher level for testing.
- As there is no data or functional dependency between any of the SFUnwind application feature screens modules, tests of components related to each of the 4 application features can be performed in any order.
- All code is to be integrated and tested on a local machine by its programmer prior to merging changes to the main branch on project GitHub repository.
    - Every module must be checked by its programmer to ensure it has been implemented correctly, and that it functions well with the other modules in the application.
- No code will be integrated into the Github branch without first being unit tested, and then integration tested.
    - Automated integration testing will be done using XCode’s XCTest
    - Manual UI testing will be performed using the Xcode simulator where possible.
    - Manual UI testing will be performed on an iOS device twice per week.
    - An iOS device will be used for testing all features that cannot be tested in software (Eg. Text messaging, camera functionality etc)

5. Complexity Measurements: Tracking and Estimations

The complexity of the SFUnwind application will be tracked to measure the following measurement metrics:

- Number of files in the main branch.
- Number of lines of code in the main branch.
- Number of classes in the main branch.
- Number of known issues/bugs.

Note: *Only files contained in the GitHub branch will be considered for these measurements.*

Methods for Tracking Complexity:

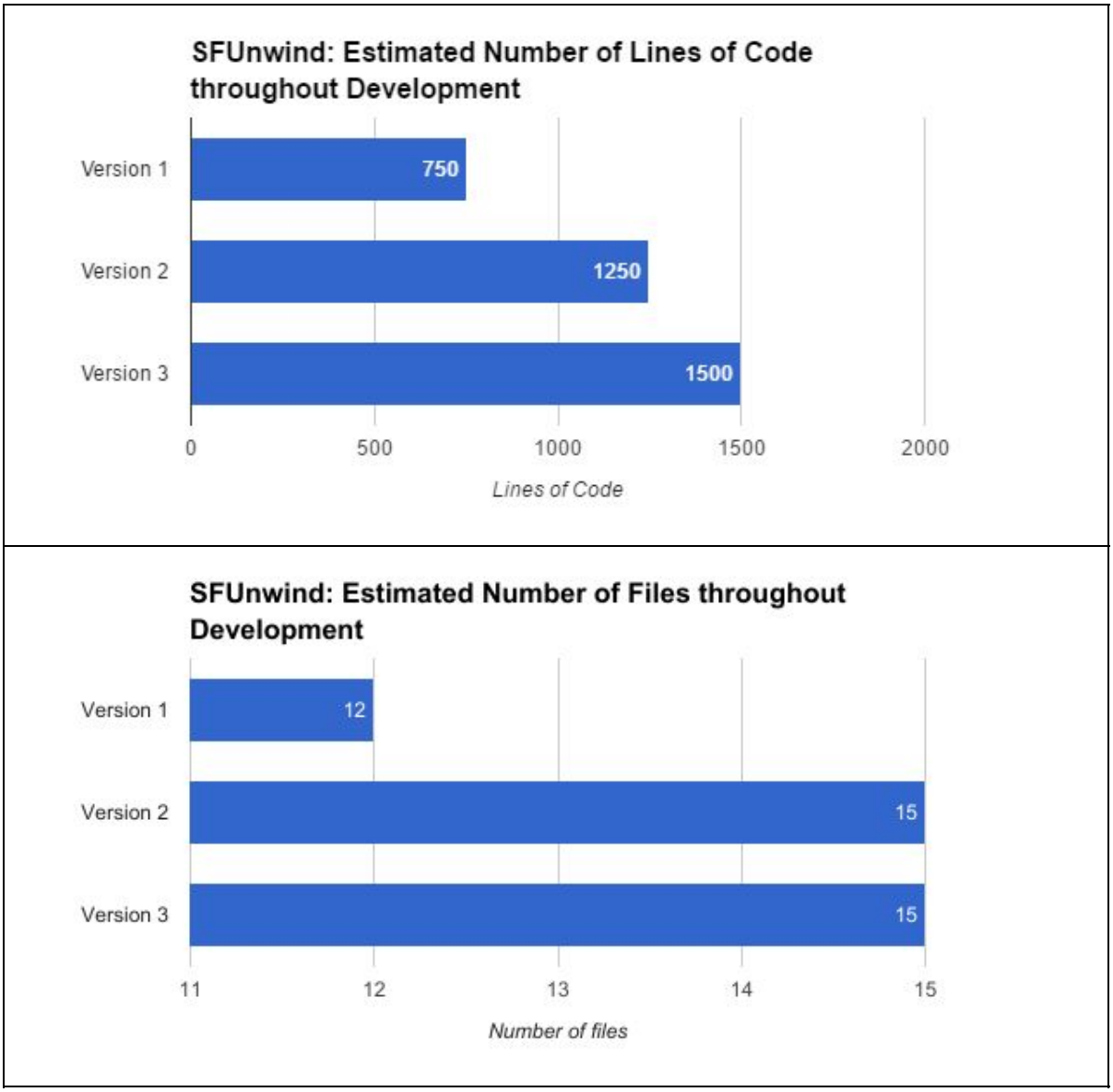
- The open source, command line operated CLOC utility ("Count Lines Of Code") will be used to monitor:
  - The total number of lines of code within all SFUnwind project files.
  - Number of classes will be tracked using a command line program called as "CLOC".
- The GitHub repository statistics will be used to monitor the number of files in the project
- A shared Google spreadsheet document will be used to track the number of known issues/bugs:

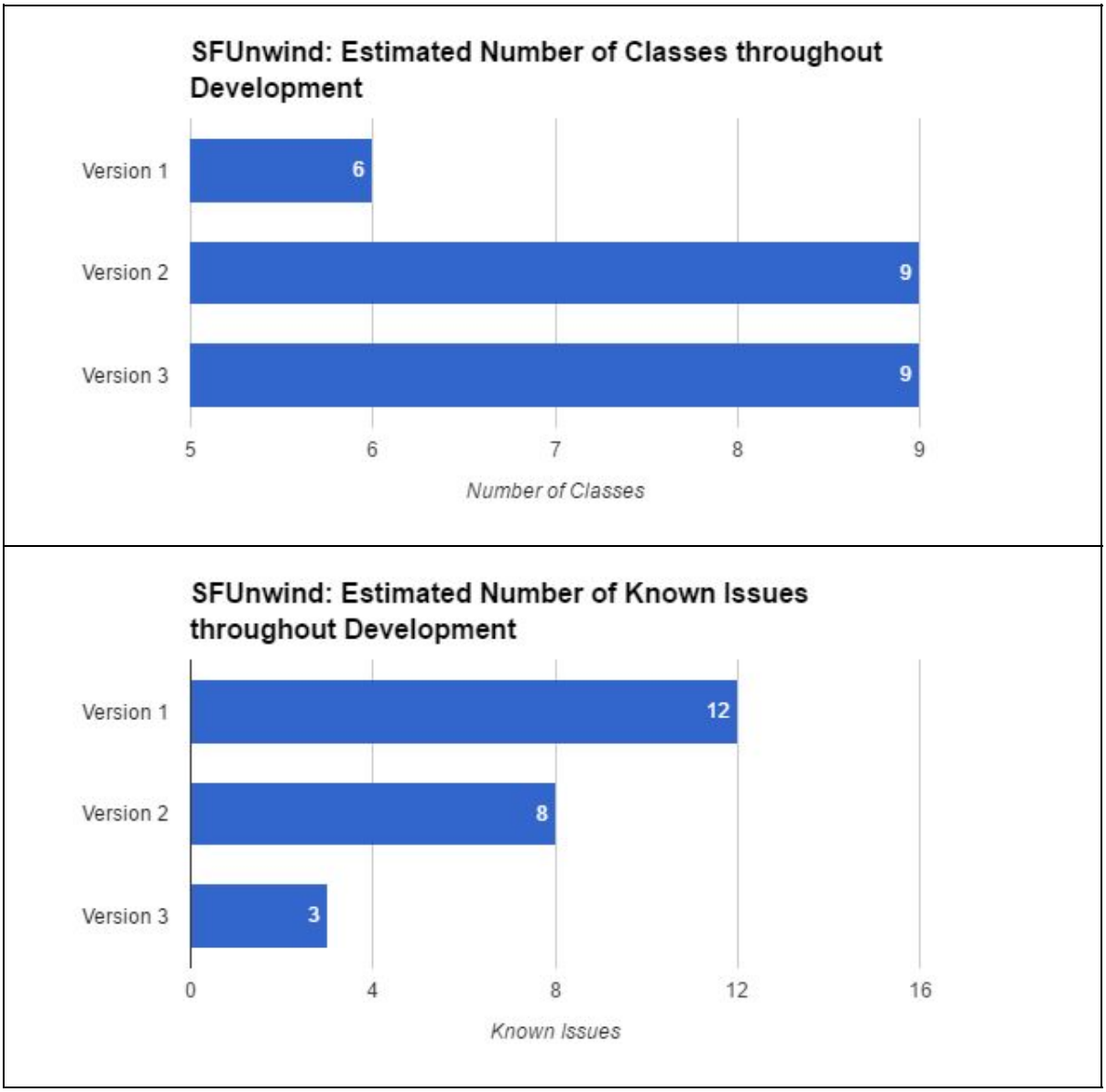
https://docs.google.com/spreadsheets/d/15kFl4Z1oPF4_ijJSjHlm77Es5uLbdm7QkaWrpinhJkk/pubhtml

Estimated project complexity:

Note: *Please see the appendix at the end of the document for actual version 1 statistics*

	Version 1	Version 2	Version 3
Lines of Code	~750	~1250	~1500
Number of Files	12	15	15
Number of Classes	6	9	9
Number of known issues/bugs	12	8	3





Spreadsheets used for these charts can be found here:
https://docs.google.com/spreadsheets/d/15kFI4Z1oPF4_ijJSjHlm77Es5uLbdm7QkaWrpinhJkk/pubhtml

6. Other Quality Processes

- Each programmer is responsible for unit testing their own code prior to check-in. All code must pass all unit tests before merging is allowed.
- Each programmer is responsible for integration testing their code after check-in. All code must pass all integration tests, or merging must be reversed.
- All programmers must maintain communication via Slack and Trello during all code merges, to alert their team-mates of incoming changes.
- A weekly smoke test of all submitted work in the GitHub branch will be performed twice per week, by compiling and testing the build on a physical iOS device.
- GitHub will be used to synchronize and merge all code, to ensure that the project has the ability to roll back if severe bugs are introduced.
- All GitHub check-ins must have meaningful and descriptive comments.

Citations

[1]*IEEE Standard for Software Unit Testing*," in ANSI/IEEE Std 1008, 1987.

[2] W. S. Humphrey. *Introduction to the personal software process*. Boston: Addison-Wesley Longman Publishing Co., 1997

[3]C. Harra, "35 Affirmations That Will Change Your Life", *The Huffington Post*, 2017. [Online]. Available: http://www.huffingtonpost.com/dr-carmen-harra/affirmations_b_3527028.html. [Accessed: 07- Mar- 2017].

[4]T. Allan, "Panic Attacks Positive Affirmations", *Free Affirmations.org - The world's largest collection of free positive affirmations*, 2017. [Online]. Available: http://www.freeaffirmations.org/panic-attacks-positive-affirmations. [Accessed: 28- Mar- 2017].

[5]"Positive Affirmations", *No Panic*, 2017. [Online]. Available: http://www.nopanic.org.uk/positive-affirmations/. [Accessed: 28- Mar- 2017].

[6]"Zen & Anti-stress Mandalas - 100% Mandalas Zen & Anti-stress", *Free-mandalas.net*, 2017. [Online]. Available: http://www.free-mandalas.net/themes/zen-anti-stress/. [Accessed: 28- Mar- 2017].

[7] keywordSuggest.org, *Mental Illness*. 2017 [Online]. Available: http://keywordsuggest.org/gallery/552451.html [Accessed: 02- Apr -2017]

[8] Vector News, *Fear of the unknown*. 2017 [Online]. Available: http://vectornews.eu/news/society/33344-fear-of-the-unknown-common-to-many-anxiety-disorders.html [Accessed: 02- Apr -2017]

[9] BitNo, *Anxiety*. 2017 [Online]. Available: https://i0.wp.com/www.bitno.net/wp-content/uploads/2015/03/shutterstock_56941108.jpg [Accessed: 02- Apr -2017]

[10]Creative VIP, *iPhone 7 Template*. 2017 [Online]. Available: https://creativevip.net/resource/iphone-7-mockup/ [Accessed: 02- Apr -2017]

[11]*Rain On Dry Leaves Sound*. 2017 [Online]. Available: http://www.orangefreesounds.com/rain-on-dry-leaves-sound/ [Accessed: 02- Apr -2017]

[12]B. McFerrin, *Don't Worry Be Happy*. 2017.

Appendix: SFUnwind Version 1, 2 & 3 Testing Results & Statistics

1. SFUnwind Version 1, 2 & 3: QA Automated Test Details

Note: All automated tests were programmed using XCode's XCTest. Test classes can be found within the /SFUnwindTests directory of the SFUnwind project.

Test Class:	**SFUnwindTests.swift**
XCTest Tester Class:	**SFUnwindPageViewController.swift**
Automated tests:	
SFUnwind Target Function:	***XCTest Test Function:***
// Called the first time this view controller loads: func viewDidLoad()	func testViewDidLoad()
func presentationCount()	func testPresentationCount()
func presentationIndex()	func testPresentationIndex()
Manually Tested UI Functions:	
SFUnnwind Target Function:	**Manual Test Details:**
// Load the main storyboard func VCInstance()	Load application and verify the storyboard is visible
// Move to previous page: Load the previous view controller func pageViewController()	Swipe left, visiting all application pages and returning to the first page again
// Move to next page: Load the next view controller func pageViewController()	Swipe right, visiting all application pages and returning to the first page again

Test Class:	**PanicAlertViewController.swift**
XCTest Tester Class:	**PanicAlertViewControllerTests.swift**
<u>Automated tests:</u>	
SFUnwind Target Function:	***XCTest Test Function:***

// Contact $: Create/send buttons ($ = 1 to 5)
func contact$CreateSendBtn()
 	
// Contact $: Edit buttons ($ = 1 to 5)
func contact$EditBtn()
	testContactCreateSendEditBtns()

// Called once when the PanicAlertViewController.swift object is first initialized func viewDidLoad()	func testViewDidLoad()
// Initialize the alert list. func initializeAlertList()	func testInitializeAlertList()
// Load alert text data stored in a txt file func getStoredAlerts()	func testGetStoredAlerts()
// Handle cancel button func contactPickerDidCancel()	func testContactPickerDidCancel()
// Handle contact selection func contactPicker()	func testContactPicker()
<u>Manually Tested UI Functions:</u>	
SFUnwind Target Function:	**Manual Test Details:**
// Display the contact select screen func displayContactSelector()	Navigate to Panic Alert screen. Tap “Create” or “Edit” to create/edit a new contact. The contact selector will be displayed.
// Save alert contact text data to a txt file func setStoredAlert()	Navigate to Panic Alert screen. Tap “Create” or “Edit” to create/edit a contact. Use the iOS interface to select a contact. This will trigger the setStoredAlert() function
// Handle the create/send button functionality func handleCreateSendBtn()	Navigate to Panic Alert screen. Tap “Create” to display the contact selector. Tap “Send” to send a pre-saved alert via the iOS system text message app.
// Send a pre-formatted alert message using the iOS messenger app func sendAlertMessage()	Must be tested on physical device: Sending messages does not work on simulator. Navigate to Panic Alert screen.

	Tap “Send” to send a pre-saved alert via the iOS system text message app. The iOS text app will be presented.

Test Class:	**PanicAlertPopupViewController.swift**
XCTest Tester Class:	**PanicAlertPopupViewControllerTests.swift**
<u>Automated tests:</u>	
SFUnwind Target Function:	***XCTest Test Function:***
// Called once when the view is loaded override func viewDidLoad()	// Test the viewDidLoad() function: func testViewDidLoad()
<u>Manually Tested UI Functions:</u>	
SFUnwind Target Function:	**Manual Test Details:**
// UITextView Delegate function: Catch newline characters, and close the keyboard: func textView()	Navigate to the Panic Alert Screen. Create or edit an existing panic alert. Input a panic alert message, then press the “done” key on the keyboard. The keyboard should close.
// Handle the transition back to the PanicAlertViewController func prepare()	Navigate to the Panic Alert Screen. Create or edit an existing panic alert. Tap the “Save” button. Return to the edit screen by selecting “edit” for the contact. The alert should be saved as input. Press the “delete” button. The alert should be cleared.
// Handle the user tapping outside of the text field func closeKeyboard()	Navigate to the Panic Alert Screen. Create or edit an existing panic alert. Input a panic alert message, then tap an area outside of the text input field. The keyboard should close.
// Handle contact selection: public func contactPicker()	Navigate to the Panic Alert Screen. Create or edit an existing panic alert. Tap the “Recipient” field. The contact picker should appear.
// Update the UI when the user has made a new selection func updateUI()	Navigate to the Panic Alert Screen. Create or edit an existing panic alert. Tap the “Recipient” field. The contact picker should appear. Select a contact. The contact name should appear in the “Recipient” field

// Check the user has selected a contact name + number and input a message before allowing them to save func shouldPerformSegue()	Navigate to the Panic Alert Screen. Create or edit an existing panic alert. Without selecting a recipient or inputting an alert message, press the save button. The application should not allow you to save.

Test Class:	**GroundingExerciseViewController.swift**
XCTest Tester Class:	**GroundingExerciseViewControllerTests.swift**
<u>Automated tests:</u>	
SFUnwind Target Function:	***XCTest Test Function:***
// This function is called once the GroundingExerciseViewController.swift object is first initialized func viewDidLoad()	func testViewDidLoad()
// This function is called every time the GroundingExerciseViewController is viewed func viewDidAppear()	func testViewDidApeear()
// This function is called every time the reset button is pressed func resetButtonAction()	func testResetButtonAction()
// This function is called when the user pressed the appropriate button func cameraButtonAction()	WITHOUT PHONE: func testCameraButtonAction()
<u>Manually Tested UI Functions:</u>	
SFUnwind Target Function:	**Manual Test Details:**
// This function is called when the user pressed the appropriate button func cameraButtonAction()	WITH PHONE: Navigated to the 5-4-3-2-1 Grounding Exercise screen by swiping the screen to the left. Tapped the capture button 45 times, seperated by a 0.5 second interval. The grid will be displayed with animations.
// This function is called when the user pressed the appropriate button func cameraButtonAction()	WITH PHONE: Navigated to the 5-4-3-2-1 Grounding Exercise screen by swiping the screen to the left. Tapped the capture button once, picture taken is shown at the top left with animations.

// This function is called when the user pressed the help button func prepare()	WITH PHONE: Navigated to the 5-4-3-2-1 Grounding Exercise screen by swiping the screen to the left. Tapped the white help icon at the right. Confirmed a passed test by comparing the message displayed with the message stored in the project's text file.

Test Class:	**PositiveAffirmationViewController.swift**
XCTest Tester Class:	**PositiveAffirmationViewControllerTests.swift**
Automated tests:	
SFUnwind Target Function:	***XCTest Test Function:***
// Called once when the PositiveAffirmationViewController.swift object is first initialized func ViewDidLoad()	func testViewDidLoad()
// Called when the user selects the previous button func Previous()	func testPrevious()
// Called when the user selects the next button func Next()	func testNext()
// Called once user selects the notification button func scheduleNotification()	func testNotifications()
// Called once user selects the create mantra button func Create()	func testCreateMantra()
// Called once user selects the create mantra button func DeleteAlert()	func testDeleteMantra()
//titleForRow, set up the text name for each options func pickerView()	func testPickerViewTitle()
//didSelectRow, hide and enable some view picker at certain cases func pickerView()	func testPickerViewDidSelect()
//hide the view picker after editing func textFieldDidBeginEditing()	func testTextFieldDidBeginEditing()
Manually Tested UI Functions:	
SFUnwind Target Function:	**Manual Test Details:**

//User creates a mantra func Create()	WITH PHONE: Navigated to the Positive Affirmations Feature by swiping right two times. Tapped the “Create Mantra” button and entered a random string (not empty.) Passed test by confirming the central text is equivalent to the entered text.
//User deletes a mantra func DeleteAlert()	WITH PHONE ON FIRST RUN: Navigated to the Positive Affirmations Feature by swiping right two times. Tapped the “Delete Mantra” button. The deleted mantra can no longer be found by selecting “Previous” and “Next” buttons, passing the test.
//User selects help button func prepare()	WITH PHONE: Navigated to the Positive Affirmations Feature by swiping right two times. Tapped the white help icon at the right. Confirmed a passed test by comparing the message displayed with the message stored in the project’s text file.
//User sends a notification func scheduleNotification()	WITH PHONE WITH SIM CARD: Navigated to the Positive Affirmations Feature by swiping right two times. Tapped the “Create Mantra” button and entered a random string (not empty.) Selected “Never”, and configured the frequency to “Hourly”. Then configured the minutes displayed as “00” to the minute arriving after current one (was 09, configured to 10.) Waited less than or equal to 1 minute, a notification appears with a message matching the mantra contents displayed on screen.

Test Class:	**SquareBreathingViewController.swift**
XCTest Tester Class:	**SquareBreathingViewControllerTests.swift**
<u>Automated tests:</u>	
SFUnwind Target Function:	***XCTest Test Function:***
// Called once when this object is first instantiated override func viewDidLoad()	func testViewDidLoad( )
// Save the seconds value of the timer to the device func saveSecondsTimer()	func testSaveSecondsTimer( )

// Save the minutes value of the timer to the device func saveMinutesTimer()	func testSaveMinutesTimer( )
// Load the seconds data from the device func loadSecondsTimer()	func testLoadSecondsTimer( )
// Load the minutes data from the device func loadMinutesTimer()	func testLoadMinutesTimer( )
// Handle the timer as it is being displayed on the screen: func timeManager()	func testSessionSecondsTimerManager( ) func testSessionMinutesTimeManager( )
// Checks if restart button is loaded func restartButton()	func testReStartButtonInitialized( )
// Navigate between four main circles in correct order func squareOrderManager(currentCircle:Int) -> UIImageView	func testSquareOrderManager
// Stop timer and animation when view is changed override func viewDidDisappear(_ animated: Bool)	func testViewDidDisappear()
// Set boolean used for restartButton to default value when view is changed to SquareBreathing screen override func viewDidAppear(_ animated: Bool)	func testViewDidAppear()
// Navigate between four main inner circles in correct order func innerOrderManager(currentCircle:Int) -> UIImageView	func testInnerOrderManager()
// Test if the function tracks just last 10 values func saveRecentSessionTracker()	func testSaveRecentSessionTracker()
// Tests if sync function calculates statistics correctly func syncStatistics()	func testSyncStatistics()
// Tests valid and invalid data func scaleAnimationManager()	func testScaleAnimationManager()
<u>Manually Tested UI Functions:</u>	
SFUnwind Target Function:	**Manual Test Details:**
func rotateBG(targetView: UIView, duration: Double = 1.0)	Start application. Check if background image rotates over time
func scaleAnimationManager()	Start application, and pressed “Start” button. Checked if circles expanded and shrinked over time in counterclockwise order.

	Check if device vibrates when circle is scaled to double size

Test Class:	**StatisticsViewController.swift**
XCTest Tester Class:	**StatisticsViewControllerTests.swift**
Automated tests:	
SFUnwind Target Function:	***XCTest Test Function:***
// This function resets all previously written statistics data func restartAllButton()	func testRestartAllButton()
Manually Tested UI Functions:	
SFUnwind Target Function:	**Manual Test Details:**
// This function loads when StatisticsViewController appears on screen func viewDidLoad()	Check if graphs are drawn. Check if the number of graphs is equal to total session only when totalSession is less or equal to 10

Test Class:	**HelpViewController.swift**
XCTest Tester Class:	**HelpViewControllerTests.swift**
Automated tests:	
SFUnwind Target Function:	***XCTest Test Function:***
// This function tests the help view loading func viewDidLoad()	func testViewDidLoad()
// This function populates of display text. func populateHelpText(screenNumber: Int)	func testPopulationHelpText()
Manually Tested UI Functions:	
SFUnwind Target Function:	**Manual Test Details:**
// This function populates of display text. func populateHelpText(screenNumber: Int)	Start application. Click on the white info button. Check to see if contents of new display matches the content within the text file for Square Breathing.

SFUnwind bug tracker document can be found here:
https://docs.google.com/spreadsheets/d/15kFl4Z1oPF4_ijJSjHlm77Es5uLbdm7QkaWrpinhJkk/pubhtml

6. SFUnwind Version 1, 2 & 3: Actual Complexity Measurements

	Number of files in main branch	Number of lines of code in main branch	Number of classes in main branch	Number of known issues/bugs
Version 1	37	993	13	1
Version 2	39	1773	15	1
Version 3	60	2679	17	0

SFUnwind bug tracker document can be found here:
https://docs.google.com/spreadsheets/d/15kFl4Z1oPF4_ijJSjHlm77Es5uLbdm7QkaWrpinhJkk/pubhtml