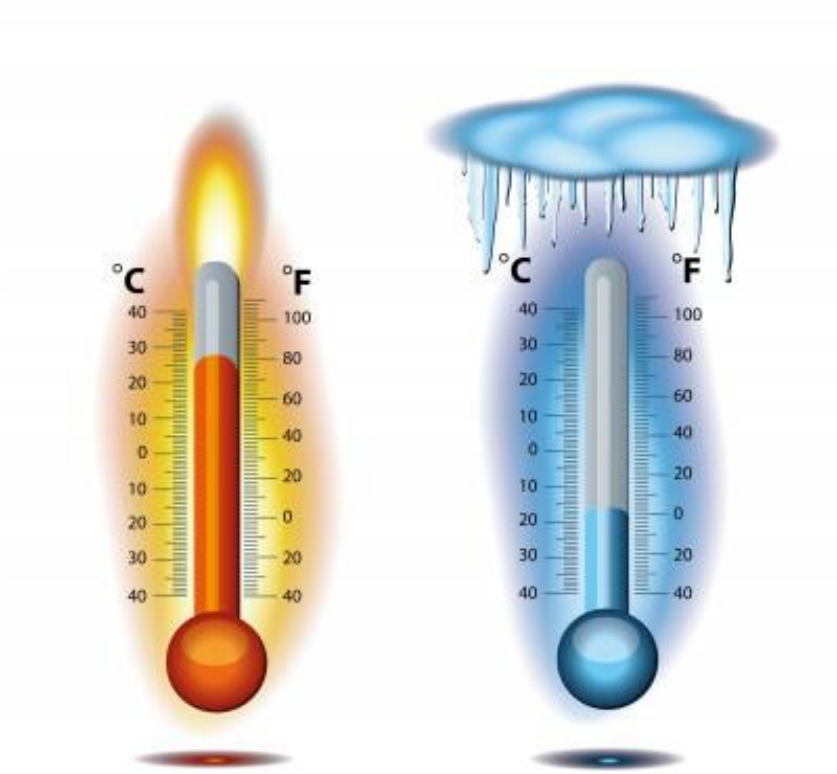

Thermometer

Fahrenheit to Celsius Calculator

Software Quality Assurance (SQA) Plan
By Zailyn Tamayo

Date: January 2015

Signature Page

Prepared by: _____ Date:_____
(Zailyn Tamayo)

Reviewed by 1: _____ Date:_____

Reviewed by 2: _____ Date:_____

Approved by : _____ Date:_____

Document Change Record

Revision	Description of Change	Approved By	Date

Table of Contents

1. Purpose and Scope……...……...……...…5
1.1. Purpose……..……..……..5
1.2. Scope……...5
1.3. Project Summary……..5
1.4. Quality Goals and Expectations……...5
2. Reference Documents……...……6
3. Management……6
3.1. Management Organization……...…6
3.1.1. Project Management……...6
3.1.2. Assurance Management……..7
4. Standards and Metrics……..7
4.1. Standards……..7
4.2. Metrics……..7
5. Verification……..……....7
5.1. Software Reviews……....8
5.1.1. Purpose……...8
5.1.2. Group Walkthrough……8
5.1.3. Managerial Review……8
5.2. Inspections……8
5.3. Audits……..……9
6. Validation…….……9
6.1. Project Testing……..9
6.1.1. Unit Testing……..…9
6.1.2. User Acceptance Testing……9
6.2. Product Testing……10
6.2.1. Functional/Positive Testing……...…10
7. Problem Reporting and Corrective Action……10
8. Tools, Techniques and Methodologies……...10
9. Training……...……...11
10. Appendix……...……...11
10.1. Design Specifications……...……11
10.2. Software Code……..12

Fahrenheit to Celsius Calculator – Project

1. Purpose and Scope

1.1. Purpose

The purpose of this Software Quality Assurance (SQA) Plan is to establish the goals, processes, and responsibilities required to implement effective quality assurance functions for the Fahrenheit to Celsius Calculator project.

The Fahrenheit to Celsius Software Quality Assurance Plan provides the framework necessary to ensure a consistent approach to software quality assurance throughout the project life cycle. It defines the approach that will be used by the Quality Assurance Manager and Quality Assurance (QA) personnel to monitor and assess software development processes and products to provide objective insight into the maturity and quality of the software.

1.2. Scope

The purpose of SQA is to ensure that the software developed does not deviate from the original intended product as defined in the Project Summary document and specified in detail in the Fahrenheit to Celsius Calculator – Design Specifications document. SQA is also concerned to identify any errors, omissions, inconsistencies, and alternatives, enhancements or improvements that can be made at any stage of development.

1.3. Project Summary

This SQAP is for a simple Fahrenheit to Celsius calculator program. The program converts inputted temperatures into either Celsius or Fahrenheit. This program was written using C++; it uses a windows form that contains three buttons and a text box. The text box is used for the user to input a number. The first button to the left is titled “C to F”, which converts Celsius to Fahrenheit. The middle button is titled “F to C”, which converts Fahrenheit to Celsius. The last button is title “Clear”; this button clears the text box.

1.4. Quality Goals and Expectations

The four most important qualities that this program should have are simplicity, ease of understanding, usability and consistency. The aim is to produce a simple

program that does a very specific task. Anyone should be able to understand how to use the program by looking at it. The calculator be easy to use and the results need to be accurate and consistent.

2. Reference Documents

- IEEE STD 730-2002, IEEE Standard for Software Quality Assurance Plans
- Fahrenheit to Celsius Calculator – Project Plan
- Fahrenheit to Celsius Calculator – Design Specifications
- Fahrenheit to Celsius Calculator – Program Code

3. Management

This section describes the management organizational structure, its roles and responsibilities, and the software quality tasks to be performed.

3.1. Management Organization

The overall responsibility for defining the scope of the quality assurance system assigned to the Quality Assurance (QA) Manager. The QA Manager is also in charge of the implementation of quality assurance system. The QA Manager's proposals for Quality Assurance are verified internally by Internal Verifiers, and externally by an External Examiner.

3.1.1. Project Management

The QA Manager will be responsible for approving:

- The Fahrenheit to Celsius Calculator – Design Specifications and Code
- The overall time scale for the project
- The choice of software development tools and techniques utilized
- The selection of project teams

- The training of project teams

3.1.2. Assurance Management

The QA Manager provides Project Management with visibility into the processes being used by the software development teams and the quality of the products being built. The QA Manager is assigned to the Fahrenheit to Celsius Calculator project and maintains a level of independence from the project and the software developers.

In support of software quality assurance activities, the QA Manager has assigned and secured Quality Assurance personnel from the pool of available QA trainees to coordinate and conduct the QA activities for the project and report back results and issues. Additional support personnel will also include Internal and External Verifiers as mentioned under Management Organization.

4. Standards and Metrics

4.1. Standards

- Documents – Microsoft Word 2010, shared drive folder
- Code – Visual C++
- Testing – IEEE Standard for Software Test Documentation

4.2. Metrics

- Source lines of code
- Bugs per lines of code
- Program execution time
- Program load time
- Maintainability index

5. Verification

CMMI defines verification as “confirmation that work products properly reflect

the requirements specified for them. In other words, verification ensures that 'you built it right'". Verification means confirming that something conforms to its documented specifications, standards, regulations, etc.

5.1. Software Reviews

5.1.1. Purpose

This section identifies the number and type of reviews that will be performed. It describes the artifact types to be reviewed as well as the format of the reviews that will be conducted. These reviews have been approved by the QA Manager and accounted for in project planning.

5.1.2. Group Walkthrough

During walkthroughs, someone other than the author goes through every sentence of the Fahrenheit to Celsius Calculator – Design Specifications and every line of the Fahrenheit to Celsius Calculator – Code. This will be done in the form of a group meeting review. Two Quality Assurance personnel will be chosen by the QA Manager to take part in the review. Both QA's will submit a review report to the QA Manager listing any defects that were uncovered or any suggestions for improvement. The QA Manager will review the reports, unsure any uncovered defects are corrected and at that point decide with another peer review is necessary.

5.1.3. Managerial Review

A managerial review will be conducted by the QA Manager after the peer review process is deemed completed. The QA Manager will ensure that the right product was built and look over the product for any possible problems. It is essential during this review for the QA Manager to ensure that no required information is missing from the product before approving further progress.

5.2. Inspections

Due to the length and simplicity of the Fahrenheit to Celsius Calculator program, only one software inspection will be conducted. The inspection conducted will be an Acceptance Testing Readiness Inspection. A QA assigned by the QA Manager will conduct this inspection and turn in an inspection report with a pass or fail outcome result.

5.3. Audits

Due to this being a small project, only three phase-end audits will be conducted. A project initiation audit will be conducted to ensure that the project is initiated in accordance to the set process. A software construction audit will take place after software construction is completed. This will ensure that all software reviews and testing of the developed code were completed. Lastly, a project closure audit will occur before the project is formally closed to ensure that all project closure activities (document, archiving, etc.) were carried out in conformance to the set process.

6. Validation

Validation is an activity to confirm that all the designed (or required) functionalities are indeed built and are working in adherence with the original specifications (intended us). CMMI defines validation as "confirmation that the product, as provided (or as it will be provided), will fulfill its intended us. The main tool for validating a product is software testing.

6.1. Project Testing

Concurrent with software development and carried out as part of the software development project.

6.1.1. Unit Testing

Unit testing will be conducted by the author of the code and also by a designated QA. The QA Manager will provide the test cases, guidelines and code to the tester. The tester executes all test cases, logs the results (indicating a pass of fail) and turns in the log to the QA Manager.

6.1.2. User Acceptance Testing

The QA Manager designs a test plan and test cases based off of the project requirements. User acceptance testing (UAT) is conducted to prove that the software functions as it is supposed to, in accordance with the requirements. The desirable outcome is to receive a sign-off that the software is acceptable.

6.2. Product Testing

Product testing is usually conducted when the software developed is not for a single customer, but rather for commercial off-the-self (COTS) use. The Fahrenheit to Celsius Calculator was not created for a client and therefore would fall under the COTS category.

6.2.1. Functional/Positive Testing

Functional testing is synonymous with positive testing. Functional testing tests the software to ensure that all its functions are working correctly. This type of testing tests the software under its intended use conditions to make sure that it works as it was designed to work. This type of testing will be carried out by the developer, QA Manager and QA tester.

7. Problem Reporting and Corrective Action

The QA Manager will be in charge of driving any problem reporting and corrective action. As the liaison between the development team and QA team, the QA Manager will receive all review reports, test cases and test logs from the testers and then report any defects or suggestions for improvement to the developer(s). As mentioned, any problem reporting will be produced and collected in the form of review reports, test cases, test logs and any other form of documentation. Corrective action will be spearheaded by the QA Manager, whose function it is to make the developer aware of any problems with the software. The developer corrects any problems by fixing the code. All documentation regarding problem reporting and corrective will be kept in a shared drive folder.

8. Tools, Techniques and Methodologies

- Microsoft Office Suite 2010
- Microsoft Visual Studio 2013

9. Training

QA personnel should have fundamental knowledge in the following areas through prior experience, training, or certification in methodologies, processes, and standards:

- Software Engineering
- Software Quality Assurance
- Reviews, Inspection and Audits
- Software Testing
- ISO 9000
- CMMI

10. Appendix

10.1. Design Specifications

- The GUI is a windows form with a text box and three buttons.
- The calculator has a text box where the user inputs a number.
- The first button to the left is titled “C to F”, which converts Celsius to Fahrenheit.
- The middle button is titled “F to C”, which converts Fahrenheit to Celsius.
- The last button is title “Clear”; this button clears the text box.

How the Program Works:

- User inputs a numerical value in to the text box, where is it displayed.
- User clicks either “C to F”, “F to C”, or “Clear” button.
- Clicking the “C to F” button performs the calculation: ((inputted value * 9) / 5) + 32.
- Clicking the “F to C” button performs the calculation: ((inputted value – 32) * 5) / 9.

- Clicking the “Clear” button clears the text box at any point.
- Clicking in the text box and deleting the inputted text also clears the display.
- Clicking “C to F” or “F to C” updates the text box display with the results of the calculation.
- The convert buttons can continue to be pressed and the display will continue to update the results.
- Inputting anything other than a numerical value will result in an error.
- The “X” button exits the program.

10.2. Program Code

```
#pragma once


namespace Ex08 {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Summary for Form1
	/// </summary>
	public ref class Form1 : public System::Windows::Forms::Form
	{
	public:

		double valueC, valueF;   // Value entered by user
		double resultF, resultC;  // Result

		Form1(void)
		{
			InitializeComponent();
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~Form1()
		{
```

```
                if (components)
                {
                        delete components;
                }
        }
    private: System::Windows::Forms::TextBox^  textBox1;


    protected:


    private: System::Windows::Forms::Button^  Clear;
    private: System::Windows::Forms::Button^  CtoF;
    private: System::Windows::Forms::Button^  FtoC;


    private:
        /// <summary>
        /// Required designer variable.
        /// </summary>
        System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        void InitializeComponent(void)
        {
                this->textBox1 = (gcnew System::Windows::Forms::TextBox());
                this->Clear = (gcnew System::Windows::Forms::Button());
                this->CtoF = (gcnew System::Windows::Forms::Button());
                this->FtoC = (gcnew System::Windows::Forms::Button());
                this->SuspendLayout();
                //
                // textBox1
                //
                this->textBox1->Location = System::Drawing::Point(13, 13);
                this->textBox1->Name = L"textBox1";
                this->textBox1->Size = System::Drawing::Size(259, 20);
                this->textBox1->TabIndex = 0;
                //
                // Clear
                //
                this->Clear->Location = System::Drawing::Point(197, 56);
                this->Clear->Name = L"Clear";
                this->Clear->Size = System::Drawing::Size(75, 23);
                this->Clear->TabIndex = 3;
                this->Clear->Text = L"Clear";
                this->Clear->UseVisualStyleBackColor = true;
               this->Clear->Click += gcnew System::EventHandler(this, &Form1::Clear_Click);
                //
                // CtoF
                //
                this->CtoF->Location = System::Drawing::Point(13, 55);
                this->CtoF->Name = L"CtoF";
                this->CtoF->Size = System::Drawing::Size(75, 23);
                this->CtoF->TabIndex = 4;
```

```
                    this->CtoF->Text = L"C to F";
                    this->CtoF->UseVisualStyleBackColor = true;
                    this->CtoF->Click += gcnew System::EventHandler(this, &Form1::CtoF_Click);
                    //
                    // FtoC
                    //
                    this->FtoC->Location = System::Drawing::Point(104, 55);
                    this->FtoC->Name = L"FtoC";
                    this->FtoC->Size = System::Drawing::Size(75, 23);
                    this->FtoC->TabIndex = 5;
                    this->FtoC->Text = L"F to C";
                    this->FtoC->UseVisualStyleBackColor = true;
                    this->FtoC->Click += gcnew System::EventHandler(this, &Form1::FtoC_Click);
                    //
                    // Form1
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(284, 93);
                    this->Controls->Add(this->FtoC);
                    this->Controls->Add(this->CtoF);
                    this->Controls->Add(this->Clear);
                    this->Controls->Add(this->textBox1);
                    this->Name = L"Form1";
                    this->Text = L"Zailyn Tamayo";
                    this->ResumeLayout(false);
                    this->PerformLayout();

                }
#pragma endregion

       private: System::Void Clear_Click(System::Object^  sender, System::EventArgs^  e) {
                                textBox1->Text = "";
                    }

private: System::Void FtoC_Click(System::Object^  sender, System::EventArgs^  e) {

                    valueF = Convert::ToDouble(textBox1->Text);
                    resultC = ((valueF - 32) * 5) / 9;
                    textBox1->Text = Convert::ToString(resultC);
                }

private: System::Void CtoF_Click(System::Object^  sender, System::EventArgs^  e) {

                     valueC = Convert::ToDouble(textBox1->Text);
                     resultF = ((valueC * 9) / 5) + 32;
                     textBox1->Text = Convert::ToString(resultF);

                }
};
}
```