SE 3XA3: Software Requirements Specification
Module Guide

Team 05, Q-aRt QTs
Elton Schiott schiotek
Emilio Hajj hajje
Liam Duncan duncanla

November 13, 2016

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
08/11/16	1.0	Basic design information

1 Anticipated and Unlikely Changes

1.1 Anticipated Changes

AC1: The range of allowed image sizes and image aspect ratios for the input image is expected to change after testing experimenting with different sizes after revision 0.

AC2: The sizes and selection of levels (number of chars encoded as well as data robustness level) of output QaRt Codes are expected to change with experimentation after revision 0 to deliver the most reliable most convenient elements.

1.2 Unlikely Changes

UC1: The general QR encoding algorithms themselves are not likely to see changes. Therefor within reason parts of the process can be spread across multiple modules with low coupling but which all are aware of the math behind other modules which are part of the process.

UC2: The max number of characters encoded is not likely to change, as our choice was based more on the allowances of the URL standard, rather than the allowances of any QR Code formats.

2 Module Hierarchy

The system consists of 7 modules including 'Main.py'. The modules listed below are the modules that will be implemented in our system.

M1: Behaviour-Hiding Modules

Constants

Draw

Main

Structure

M2: Software Decision Modules

Data

ECC

Matrix

Level 1	Level 2
Behaviour-Hiding Modules	Input Format (Main.py) Library of characters (Constants.py) Output Format (Draw.py) Encoded Interleaved Data (Structure.py)
Software Decision Modules	Encoding and Pading Algorithms (Data.py) Conversion Methods (Data.py) Matrix Data Structure Format (Matrix.py) Codeword Generating Algorithm (ECC.py)

Table 2: Module Hierarchy

3 Connection Between Requirements and Design

We dedicated a module aptly named GUI to cover and contain R3 entirely. We also made the input for the entire program simultaneous to combine couplings with the new GUI (which was not part of the original project). We also decided to fill R4 through user input of desired size and error correction level. This general interface will also cover R9, and will comply with R7.

In the case of input rejection we decided to fulfill the input check as soon as it is recieved from the GUI. To do this we simply layed the light check-code in the main function. This set of checks encompasses solutions to R5, R6, R8 and R15.

To achieve R10 and R14 we minimilized the system by removing the moving-image QR Codes, which made sense considering that our expected application is mostly printed graphics (since a QR Code could be replaced with a linked image on a web page and unclickable digital advertisements are not yet as common). This will not make any remaining creations faster but will certainly cut down on program size.

4 Module Decomposition

4.1 Behaviour-Hiding Module

4.1.1 Main (M1)

Secrets: The format and structure of the input data.

Services: Calls all the modules in a specified order, except for the Constants Module.

Implemented By: Q-aRt Code

4.1.2 Constants Module (M2)

Secrets: Lists of characters/integers/bits associated to certain classifications (alphanumerics, mode indicators, etc..).

Services: Acts as a library or reference tool.

Implemented By: Q-aRt Code

4.1.3 Draw Module (M3)

Secrets: The format and structure of the output data.

Services: Generates a PNG image file representing the data contained in the QR code matrix.

Implemented By: Q-aRt Code

4.1.4 Structure Module (M4)

Secrets: Encoded interleaved data.

Services: Interleaves the generated data codewords and error correction codewords by QR code standards before the data is placed in the matrix.

Implemented By: Q-aRt Code

4.2 Software Decision Module

4.2.1 Data Module (M5)

Secrets: Algorithms used to encode data bits and pad strings. Methods of conversion from str to bin.

Services: Encodes the data bits for a QR code given an input string, version, error correction level, and mode.

Implemented By: Q-aRt Code

4.2.2 ECC Module (M6)

Secrets: Algorithms to deduce error correction codewords, divide message polynomials and obtains the result of the 'exclusive OR' operation on the generator/message polynomials.

Services: Generates error correction codewords for a given version, level and data string

Implemented By: Q-aRt Code

4.2.3 Matrix Module (M7)

Secrets: A matrix containing the final QR Code

Services: Places the data in a matrix

Implemented By: Q-aRt Code

5 Traceability Matrix

Req.	Modules
R1	M1, M3
R2	M1, M3
R3	M1
R4	M2, M4, M5, M6, M7
R5	M5
R6	M1
R7	M1
R8	M5
R9	M1
R10	M2, M4, M5, M6, M7
R11	M1, M3

Table 3: Trace Between Requirements and Modules

AC	Modules
AC1	
AC2	
UC1	
UC2	

Table 4: Trace Between Anticipated Changes and Modules

6 Use Hierarchy Between Modules

Main USES Data, Structure, ECC, Matrix
Data, Structure, ECC, Matrix USES Constant

The process of QR Code generation and image combination is essentially linear and should be completely dictated by the original input. The main method calls all modules

other than Constant as libraries in turn to process the element being created. Constant is a library of raw data, integers and standard values, that are used for various subtasks throughout the linear process. Because of this, the modular heirarchy percolates at both ends into Main at the top and Constants at the bottom, with all other modules on the same center level and all connected to each point of percolation.

Figure 1: Use hierarchy among modules