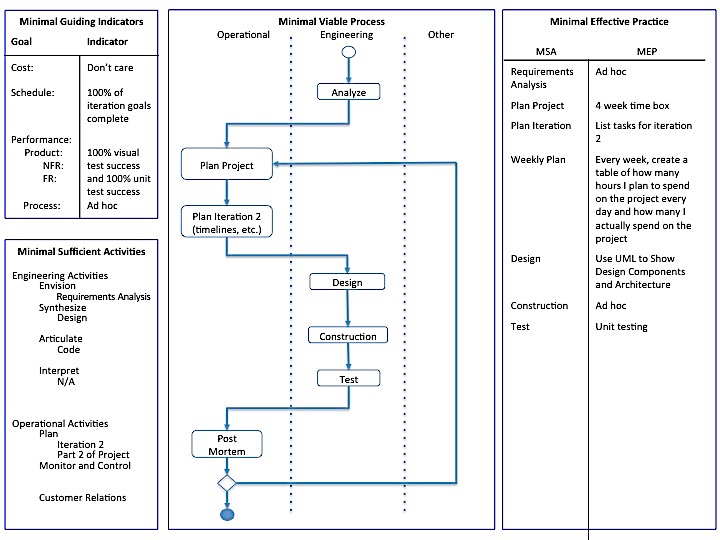
Iteration 5

# Process for Iteration 5



The process has not changed since Iteration 4.

# Requirements Analysis

The requirements have not changed since Iteration 1. They are as follows:

Given a secret image file and two innocent image files, the tool should

* Be able to read in image files and store the pixel information
* Use the extended visual cryptography scheme to encode the secret image pixels into the two innocent images
* Store the encoded images in new image files
  + The filenames and location can be specified by the user. If not, the files are named share1 and share2 and gets stored on the Desktop.

Given two encoded image files, the tool should

* Be able to read in the files and store the pixel information
* Use the extended visual cryptography scheme to decode the secret image from the encoded images (similar to super imposing them)
* The image revealing the secret gets stored in a new image file
  + The filename and location can be specified by the user. If not, the file is named secretMsg and gets stored on the Desktop.

The visual cryptography tool will only work with PNG and JPEG images. The images involved with the encoding process must have the same dimensions. The tool can handle images of any coloring.

# Plans for Project

Iteration 1 (Sept. 11 – Oct. 9):

* Create a graphical user interface
* Get the tool working for strictly black and white images
* Test the tool to check the quality of the encoded shares and the decoded message

Iteration 2 (Oct. 10 – Nov. 6):

* Research visual cryptography schemes with gray scale images

Iteration 3 (Nov. 7 – Dec. 4):

* Implement the visual cryptography scheme with grayscale images
* Begin researching how to modify the current algorithm to handle color images

Iteration 4 (Dec. 5 – Jan. 1):

* Research how to add the ability to encode and decode multicolor images

Iteration 5 (Jan. 2 – Jan. 29):

* Implement the encoding and decoding of color images
* Add unit tests for the components of the visual cryptography tool

Iteration 6 (Jan. 30 – Feb. 26):

* Analyze the tool and look for ways to improve efficiency (performance and memory storage)
* Add features to project to help boost robustness (i.e. add in checks to keep the user from breaking the tool easily)

# Plans for Iteration 5

* Implement the encoding and decoding of color images
* Add unit tests for the components of the visual cryptography tool

# Weekly Plans

Week 1:

|  |  |  |
| --- | --- | --- |
| Day | Expected Hours | Actual Hours |
| Saturday, January 2nd | 0 | 0 |
| Sunday, January 3rd | 0 | 0 |
| Monday, January 4th | 2 | 0 |
| Tuesday, January 5th | 2 | 3 |
| Wednesday, January 6th | 3 | 1.5 |
| Thursday, January 7th | 0 | 0 |
| Friday, January 8th | 0 | 0 |

Week 2:

|  |  |  |
| --- | --- | --- |
| Day | Expected Hours | Actual Hours |
| Saturday, January 9th | 0 | 0 |
| Sunday, January 10th | 0 | 0 |
| Monday, January 11th | 0 | 0 |
| Tuesday, January 12th | 0 | 0 |
| Wednesday, January 13th | 0 | 0 |
| Thursday, January 14th | 0 | 0 |
| Friday, January 15th | 3 | 2 |

Week 3:

|  |  |  |
| --- | --- | --- |
| Day | Expected Hours | Actual Hours |
| Saturday, January 16th | 0 | 0 |
| Sunday, January 17th | 0 | 0 |
| Monday, January 18th | 4 | 4 |
| Tuesday, January 19th | 2 | 1 |
| Wednesday, January 20th | 4 | 3 |
| Thursday, January 21st | 0 | 0 |
| Friday, January 22nd | 0 | 0 |

Week 4:

|  |  |  |
| --- | --- | --- |
| Day | Expected Hours | Actual Hours |
| Saturday, January 23rd | 0 | 0 |
| Sunday, January 24th | 0 | 0 |
| Monday, January 25th | 4 | 2 |
| Tuesday, January 26th | 2 | 3 |
| Wednesday, January 27th | 3 | 3 |
| Thursday, January 28th | 4 |  |
| Friday, January 29th | 3 |  |

# Design

Figure 1: UML Diagram from Iteration 1

Note the design has not changed since Iteration 2.

Before changing the Java files to handle the gray scale images, I exported the PlantUML diagram of the visual cryptography tool. Figure 1 shows the class relations.

In iteration 4, I decided to focus on researching techniques for encrypting color images. The most promising technique I found came from Varalakshmi, R, and Parameswari, and it utilized Visual Information Pixel (VIP) synchronization. VIP synchronization helps hide the secret image pixel information inside the innocent pixels. The process for encrypting a secret image is as follows:

1. Gather and process the two innocent images and one secret image.
2. Half-tone the innocent images using error diffusion.
3. Split the secret image into three images. One image represents only the red concentration of the picture, the second represents the green concentration, and the third represents blue.
4. Perform VIP synchronization on the innocent images and the three secret images.
5. Use error diffusion on the encrypted shares to smooth any pixels that cause the encoded image to look as if they are hiding something.

Decryption for this technique does not require the user to have a computer. The images can be printed on transparencies and stacked to reveal the secret image.

# Construction

Began construction with fixing decryption to XOR the pixels.

Then stubbed out the functions to be made for the new encryption process. The stubs were placed in the encryptImage() method.

Wrote code to split secret image into three images based on red, green, and blue.

Implemented the error diffusion technique

Added VIP synchronization and had unsuccessful test with grayscale images. With the VIP synchronization, I had to change the BufferedImage image type from TYPE\_INT\_ARGB to TYPE\_INT\_RGB to get the encoded images to print to the file properly. The alpha value was never being set to 100% so the images were being printed fully transparent. The decryption worked fine, but the encoded images did not show the cover images and you could see portions of the secret message.

Then for sanity break, fixed the frames to appear in the center of the screen because popping up in the left hand corner really bothered me.

# Test

TBD

# Post Mortem

TBD

# Source Code

MainFrame.java

TBD

EncodeFrame.java

TBD

DecodeFrame.java

TBD

ImageCustomFilter.java

TBD

DirectoryCustomFilter.java

TBD

ExtendedVCS.java

TBD

Pixel.java

TBD