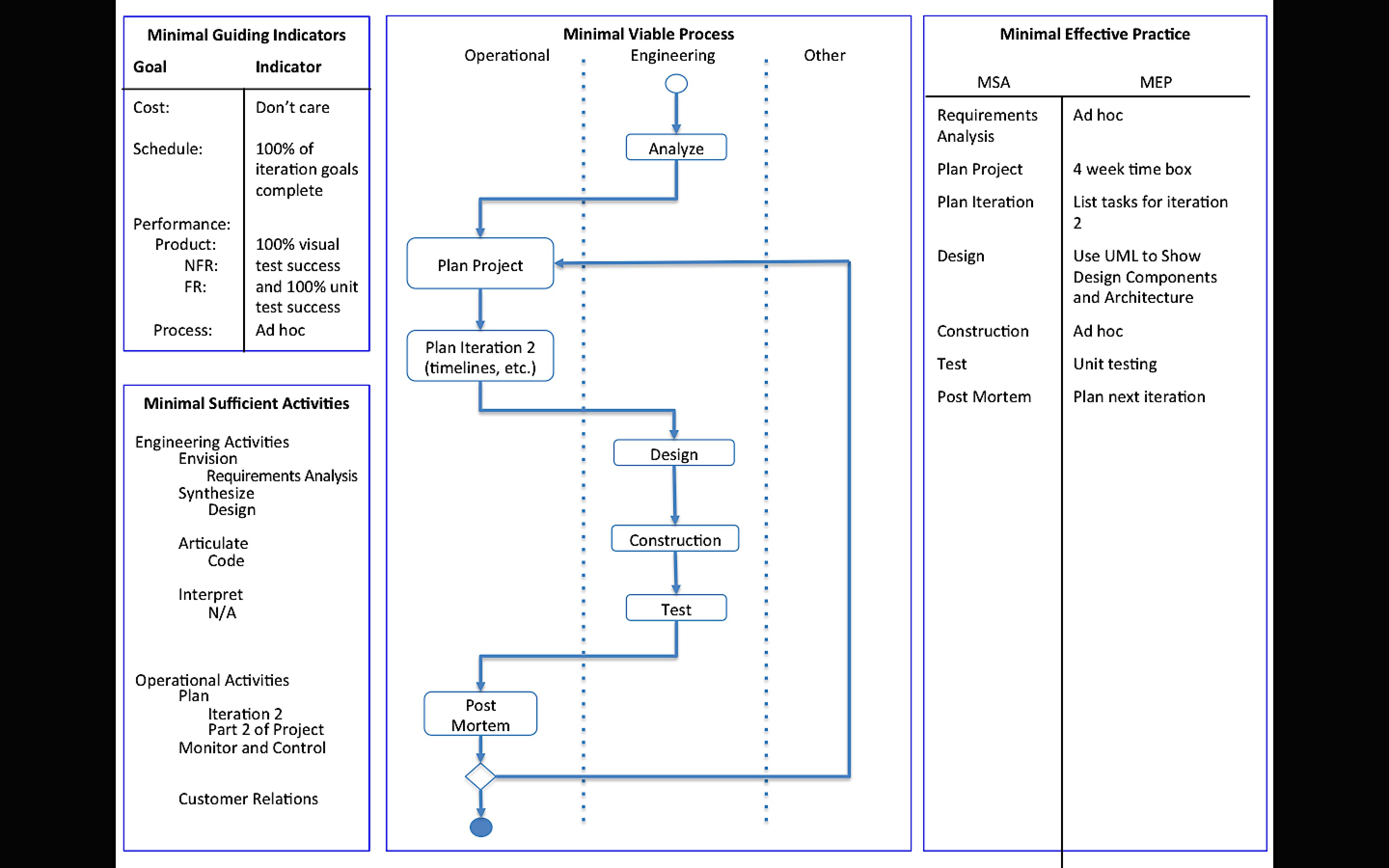
Iteration 2

# Process for Iteration 2



The process above has been slightly updated. The requirements analysis was thorough enough to only be completed once (during the first iteration). Hence, the analysis is not repeated for every iteration. For performance measurement, I will be performing unit tests along with the visual examination of the images. Also, during the design phase, I will be constructing a UML diagram via a PlantUML plugin to show the design components and architecture of the system.

# 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:

* 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:

* Research and implement the visual cryptography scheme with grayscale images
* Add features to project to help boost robustness (i.e. add in checks to keep the user from breaking the tool easily)

Iteration 3:

* Add the ability to encode and decode multicolor images

Iteration 4:

* Analyze the tool and look for ways to improve efficiency (performance and memory storage)

# Plans for Iteration 2

* Research current extended visual cryptography techniques and halftone visual cryptography techniques for handling grayscale images
* Based on the research findings, update the current cryptography scheme to improve the encryption and decryption of grayscale images
* If time permits, add more robustness to the tool, such as input validation

# Design

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.

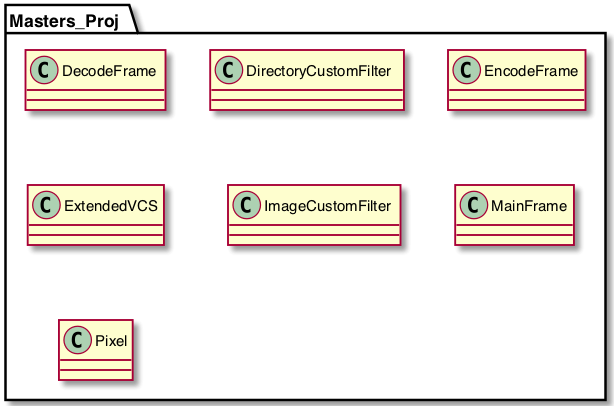


Figure 1: UML Diagram from Iteration 1

The research uncovered multiple ways of hiding gray scale images within other gray scale images. One technique utilizes several forms of visual cryptography. First, it breaks the secret image up into *n* shares. If those *n* shares were super imposed, then the secret image would be revealed. This first step is the visual cryptography scheme with pixel expansion. Then the innocent images need to be generated to have the same dimensions as the secret shares. Finally, you embed the secret shares into the generated images. To decrypt and reveal the secret image, you just have to superimpose the embedded shares. A second technique uses pixel expansion and error diffusing. Some other variations use dithering to group the pixels and treat them as strictly black and white images.

I plan on implementing the first technique mentioned with the pixel expansion and embedding the secret shares into the innocent images.

# Construction

Unfortunately, due to time constraints from my calendar external to the project, I was unable to make progress in the construction of the visual cryptography tool.

# Test

TBD

# Post Mortem

During Iteration 2, …