THE TEAM WITH THE BIG GUNS
Special Weapons, NSWC Crane

Architecture

Andrew Houvener
Matthew Jacobs
Kyle Kopacz
Micah Weaver

May 18, 2009

Contents

1 Introduction

The purpose of this document is to outline the architecture and framework which we plan to use to implement the Small Arms Naval Target Analyzer. It will use different views to highlight the chosen implementation, as well as address the various tactics with which we will attack the architecture. The requirements, supplementary specifications, and use cases for this project can be found in the document Project Requirements Documentation.pdf.

2 Quality Attributes

2.1 Usability

Source: End user
Stimulus: Usability Test
Artifact: System
Environment: Runtime environment
Response: User performs the requested task
Response Measure: User performs the task successfully without requiring assistance

Source: End user
Stimulus: Process an image
Artifact: System
Environment: Run time
Response: A report is generated by the system
Response Measure: The user should not have to utilize the help menu if he has already processed an image in the past.

2.2 Reliability

Source: Environment
Stimulus: The program becomes unresponsive or abruptly terminates while manipulating the image.
Artifact: System
Environment: Run time
Response: The change log kept by the system is saved to disk and recovered at the start of the next program.
Response Measure: The program should be able to trace back through the log to return the user to the most recent change.

Source: Environment
Stimulus: The program attempts to save an image's data, but cannot connect with an Access database.
Artifact: System
Environment: Saving image data
Response: The system will inform the user that data could not be saved.
Response Measure: All data will be lost when the user exits the application if a connection with the

database cannot be established before termination.

2.3 Performance

Source: End user
Stimulus: User selects for system to begin analyzing image
Artifact: System
Environment: Under Normal Operations
Response: Image is analyzed and a list of points of bullet holes is presented.
Response Measure: The entirety of the image processing should occur faster than the user could do it by hand.

2.4 Modifiability

Source: Developer
Stimulus: Add or change a bullet hole detection algorithm
Artifact: System
Environment: Design
Response: Algorithm added to system
Response Measure: An algorithm, unit tested separate to the system, can be integrated with the system in an hour.

Source: Developer
Stimulus: The client requests a change be made to the user interface
Artifact: System
Environment: Design
Response: The change is made to the interface
Response Measure: Purely cosmetic changes made to the interface must be able to be made only to the UI components of the system.

Source: Developer
Stimulus: The client requires a new user interface be created
Artifact: System
Environment: Design
Response: The system utilizes a new interface
Response Measure: The system operates with a new interface. The interface, developed and tested separate to the system, can be integrated with the system in 2 hours.

2.5 Testability

Source: Tester
Stimulus: Field Test

Artifact: System
Environment: Under testing conditions
Response: System produces results identical to a hand analysis.
Response Measure: The system must compute the results faster than can be performed by hand.

2.6 Security

This project has no security concerns. All security related concerns will be handled by NMCI on the files themselves, as anyone with physical access to the program, files, or database will already have the permissions required to use all aspects of the system. Also, because the source code contains nothing sensitive in nature, it will not need to be secured.

3 Patterns and Tactics

In order to achieve the quality attributes listed, we will utilize the following tactics:

3.1 Usability

- The interface will use standard elements common to most computer users, including auto completion fields and drop down menus.
- All items in the interface will be clearly labeled to facilitate ease of use. The judgment of clearly labeled images will be up to the project clients.
- Storyboarded screen shots will be given to the clients for review before building the user interface.
- As each version is developed, changes to the interface will be given to the clients for approval.

3.2 Modifiability

- Implementation will be done through object oriented programming practices.
- The system will be modeled with the model-view-controller (MVC) pattern. As the system will need to evolve with new algorithms and data storage options, and because the user interface needs to be flexible during development, the MVC approach will allow all of these items to be simply altered independent of the rest of the system.

3.3 Availability

- As the program will be a standalone application, it will always be available on any computer where the executable resides.
- If the shared network location where the Access database resides is unavailable, the program will notify the user that a save was not possible and will continue to generate reports.

3.4 Performance

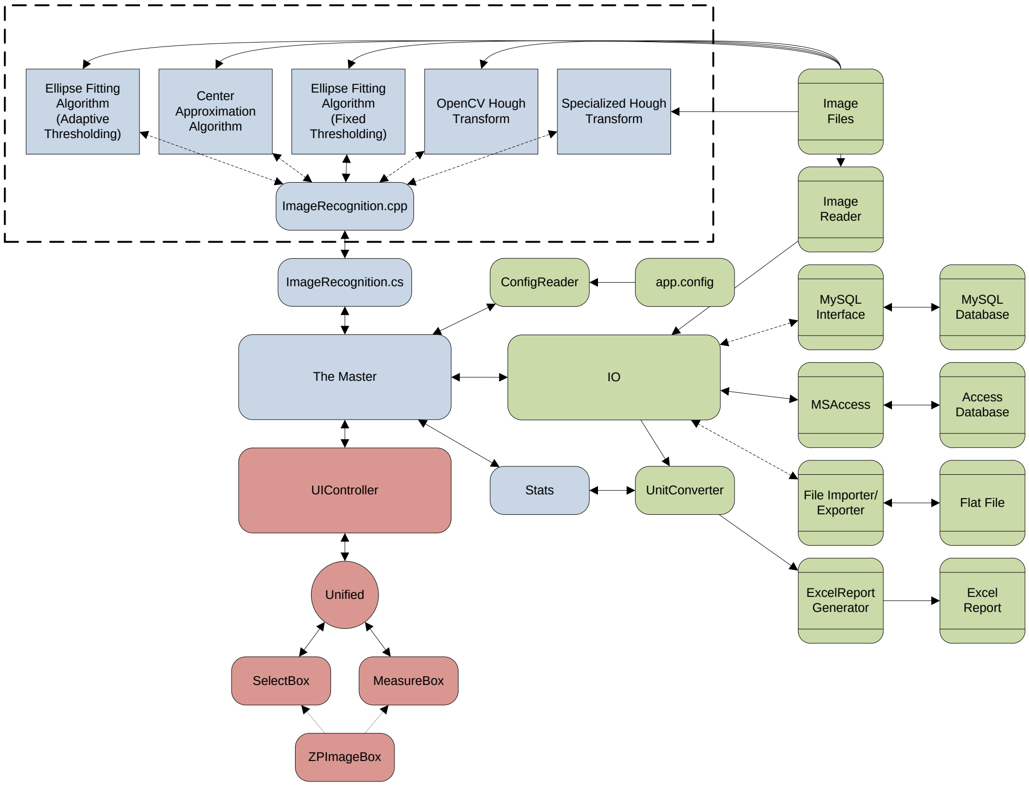
- Images will be analyzed for the best and most efficient calculating algorithm before being processed.
- Images will be analyzed and processed while the user is inputting test data.

3.5 Testability

- The accuracy of all statistics calculated will be tested using unit tests.
- The user interface will be tested using usability testing.
- New algorithms will be tested and utilized following the procedures found in the evaluation portion of the Prototypes via Sprints section found in the Project Plan.

4 Class View

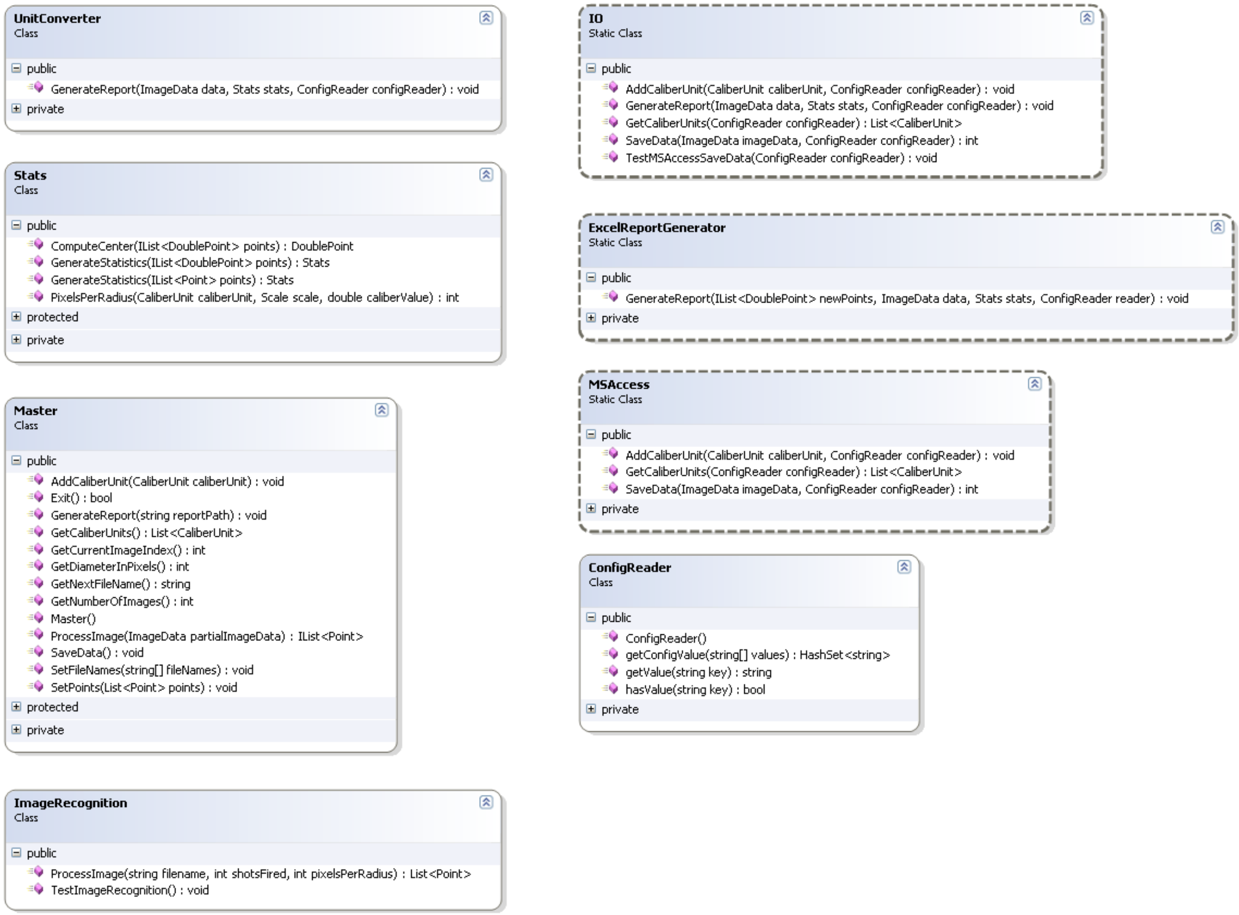
4.1 Primary Presentation



4.2 Element Catalog

The elements of the class view represent the various classes that will be contained within the project. The ImageRecController and algorithms are written in C++[1], as they require OpenCV[4], which is only available for that language. To ease development of the remainder of the system, C#[2] is used in conjunction with version 2.0 of the .NET Framework[3].

The following diagram displays the public methods of the controllers in the project.



```
UnitConverter
Class
⊟ public
    GenerateReport(ImageData data, Stats stats, ConfigReader configReader) : void
⊞ private

Stats
Class
⊟ public
    ComputeCenter(IList<DoublePoint> points) : DoublePoint
    GenerateStatistics(IList<DoublePoint> points) : Stats
    GenerateStatistics(IList<Point> points) : Stats
    PixelsPerRadius(CaliberUnit caliberUnit, Scale scale, double caliberValue) : int
⊞ protected
⊞ private

Master
Class
⊟ public
    AddCaliberUnit(CaliberUnit caliberUnit) : void
    Exit() : bool
    GenerateReport(string reportPath) : void
    GetCaliberUnits() : List<CaliberUnit>
    GetCurrentImageIndex() : int
    GetDiameterInPixels() : int
    GetNextFileName() : string
    GetNumberOfImages() : int
    Master()
    ProcessImage(ImageData partialImageData) : IList<Point>
    SaveData() : void
    SetFileNames(string[] fileNames) : void
    SetPoints(List<Point> points) : void
⊞ protected
⊞ private

ImageRecognition
Class
⊟ public
    ProcessImage(string filename, int shotsFired, int pixelsPerRadius) : List<Point>
    TestImageRecognition() : void

IO
Static Class
⊟ public
    AddCaliberUnit(CaliberUnit caliberUnit, ConfigReader configReader) : void
    GenerateReport(ImageData data, Stats stats, ConfigReader configReader) : void
    GetCaliberUnits(ConfigReader configReader) : List<CaliberUnit>
    SaveData(ImageData imageData, ConfigReader configReader) : int
    TestMSAccessSaveData(ConfigReader configReader) : void

ExcelReportGenerator
Static Class
⊟ public
    GenerateReport(IList<DoublePoint> newPoints, ImageData data, Stats stats, ConfigReader reader) : void
⊞ private

MSAccess
Static Class
⊟ public
    AddCaliberUnit(CaliberUnit caliberUnit, ConfigReader configReader) : void
    GetCaliberUnits(ConfigReader configReader) : List<CaliberUnit>
    SaveData(ImageData imageData, ConfigReader configReader) : int
⊞ private

ConfigReader
Class
⊟ public
    ConfigReader()
    getConfigValue(string[] values) : HashSet<string>
    getValue(string key) : string
    hasValue(string key) : bool
⊞ private
```

Figure 1: Controllers Class Diagram

The following diagram displays the user interface classes in the project.

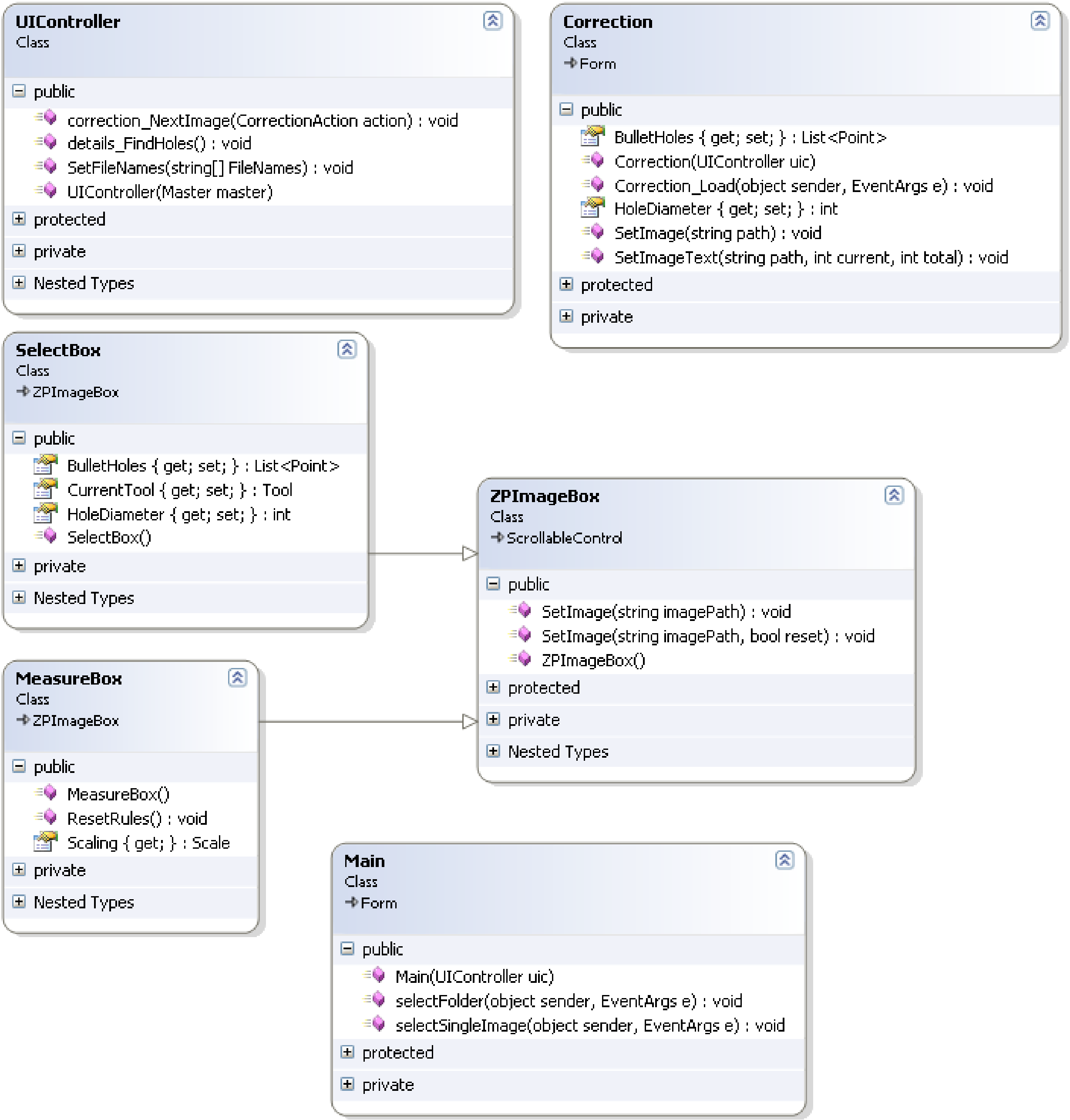
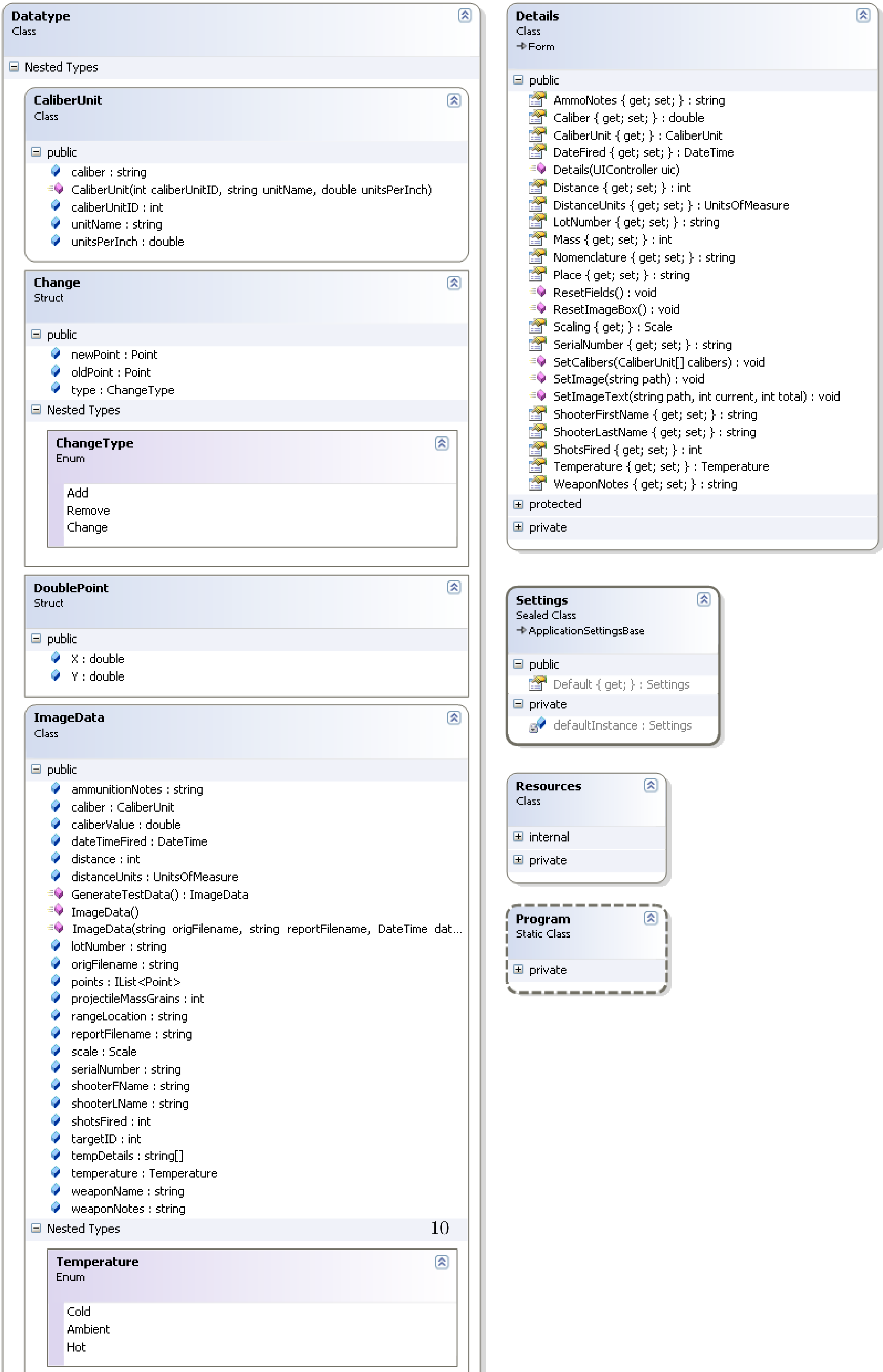

UIController
Class
public
correction_NextImage(CorrectionAction action) : void
details_FindHoles() : void
SetFileNames(string[] FileNames) : void
UIController(Master master)
protected
private
Nested Types
Correction
Class
Form
public
BulletHoles { get; set; } : List<Point>
Correction(UIController uic)
Correction_Load(object sender, EventArgs e) : void
HoleDiameter { get; set; } : int
SetImage(string path) : void
SetImageText(string path, int current, int total) : void
protected
private
SelectBox
Class
ZPImageBox
public
BulletHoles { get; set; } : List<Point>
CurrentTool { get; set; } : Tool
HoleDiameter { get; set; } : int
SelectBox()
private
Nested Types
ZPImageBox
Class
ScrollableControl
public
SetImage(string imagePath) : void
SetImage(string imagePath, bool reset) : void
ZPImageBox()
protected
private
Nested Types
MeasureBox
Class
ZPImageBox
public
MeasureBox()
ResetRules() : void
Scaling { get; } : Scale
private
Nested Types
Main
Class
Form
public
Main(UIController uic)
selectFolder(object sender, EventArgs e) : void
selectSingleImage(object sender, EventArgs e) : void
protected
private

Figure 2: User Interface Class Diagram

The following diagram displays the data types and other utilities in the project.



Datatype
Class

⊟ Nested Types

CaliberUnit
Class

⊟ public
- caliber : string
- CaliberUnit(int caliberUnitID, string unitName, double unitsPerInch)
- caliberUnitID : int
- unitName : string
- unitsPerInch : double

Change
Struct

⊟ public
- newPoint : Point
- oldPoint : Point
- type : ChangeType

⊟ Nested Types

ChangeType
Enum

Add
Remove
Change

DoublePoint
Struct

⊟ public
- X : double
- Y : double

ImageData
Class

⊟ public
- ammunitionNotes : string
- caliber : CaliberUnit
- caliberValue : double
- dateTimeFired : DateTime
- distance : int
- distanceUnits : UnitsOfMeasure
- GenerateTestData() : ImageData
- ImageData()
- ImageData(string origFilename, string reportFilename, DateTime dat...
- lotNumber : string
- origFilename : string
- points : IList<Point>
- projectileMassGrains : int
- rangeLocation : string
- reportFilename : string
- scale : Scale
- serialNumber : string
- shooterFName : string
- shooterLName : string
- shotsFired : int
- targetID : int
- tempDetails : string[]
- temperature : Temperature
- weaponName : string
- weaponNotes : string

⊟ Nested Types

Temperature
Enum

Cold
Ambient
Hot

Details
Class
→ Form

⊟ public
- AmmoNotes { get; set; } : string
- Caliber { get; set; } : double
- CaliberUnit { get; } : CaliberUnit
- DateFired { get; set; } : DateTime
- Details(UIController uic)
- Distance { get; set; } : int
- DistanceUnits { get; set; } : UnitsOfMeasure
- LotNumber { get; set; } : string
- Mass { get; set; } : int
- Nomenclature { get; set; } : string
- Place { get; set; } : string
- ResetFields() : void
- ResetImageBox() : void
- Scaling { get; } : Scale
- SerialNumber { get; set; } : string
- SetCalibers(CaliberUnit[] calibers) : void
- SetImage(string path) : void
- SetImageText(string path, int current, int total) : void
- ShooterFirstName { get; set; } : string
- ShooterLastName { get; set; } : string
- ShotsFired { get; set; } : int
- Temperature { get; set; } : Temperature
- WeaponNotes { get; set; } : string

⊞ protected

⊞ private

Settings
Sealed Class
→ ApplicationSettingsBase

⊟ public
- Default { get; } : Settings

⊟ private
- defaultInstance : Settings

Resources
Class

⊞ internal

⊞ private

Program
Static Class

⊞ private

4.2.1 Master

The master class is the brains of the program. Taking inputs from the dumb user interface, it will take care of delegating the responsibilities to the appropriate sub classes. It will server as the main hub for the program.

4.2.2 Stats

All calculations and statistics needed to be performed by the system will be handled by this class.

4.2.3 ImageRecognition

The ImageRecognition class is responsible for making sure an image is correctly and adequately processed. It will determine which of the available algorithms should be used and what to do with the results. Possible scenarios include choosing an algorithm based on the caliber of the bullet, including holes found by a majority of algorithms, averaging center points found by multiple algorithms, or taking a pipe and filter approach with multiple algorithms. The only image processing that this class may do is to pre-process an image to choose the optimal algorithm.

4.2.4 Known algorithms

The following tested algorithms are listed as classes separate from the ImageRecognition class:

Ellipse Fitting Algorithm The ellipse fitting algorithm is designed to work on smaller caliber holes. It holes skewed in most directions, but will not find holes produced by multiple bullets.

Center Approximation Algorithm The center approximation algorithm follows each group of contours to find the one or more centers possible in that group. This algorithm is ideal for images where holes are produced by multiple bullets and images produced by large caliber bullets.

4.2.5 IOController

The IOController will serve as a class to automatically handle data passed to it by the Mater class. It will determine the correct interface to utilize and perform the tasks required. The IPLImage object is from openCV library, and is used to manipulate images.

4.2.6 ExcelReportGenerator

The report generator is responsible for building and saving Excel reports detailing the results of an image.

4.2.7 UnitConverter

The unit converter converts units from pixels to inches while passing the report generation command from IO to ExcelReportGenerator.

4.2.8 ConfigReader

ConfigReader reads user settings in the configuration file such as the database location.

4.2.9 Data interfaces

The Image Reader, MySQL, Access, and File Importer/Exporter interfaces serve as a conduit for the programmer to easily access whichever file structure is required. They allow for the IOController to easily connect to the resources required. The interfaces indicated by dotted lines are not within the scope of this project's requirements.

MSAccess The MSAccess class connects to a Microsoft Access database and accomplishes saving and loading to the database.

4.2.10 UIController

UIController is the bridge between the Master class and the user interface mechanisms.

4.2.11 Main

Main is the opening window that presents the user with the option to process an image or process a folder of images.

4.2.12 ZPImageBox

ZPImageBox is a zoomable, panable image box that allows the user to navigate the image similarly to Google Maps.

4.2.13 Details

Details is the screen that allows users to input details about an image.

4.2.14 MeasureBox

MeasureBox is the panel that enables measurement of the scale.

4.2.15 Correction

Correction is the screen that allows users to add and remove bullet holes. It contains a SelectBox to actually allow selection.

4.2.16 SelectBox

SelectBox is the panel that enables selection of pixels.

4.2.17 Datatype

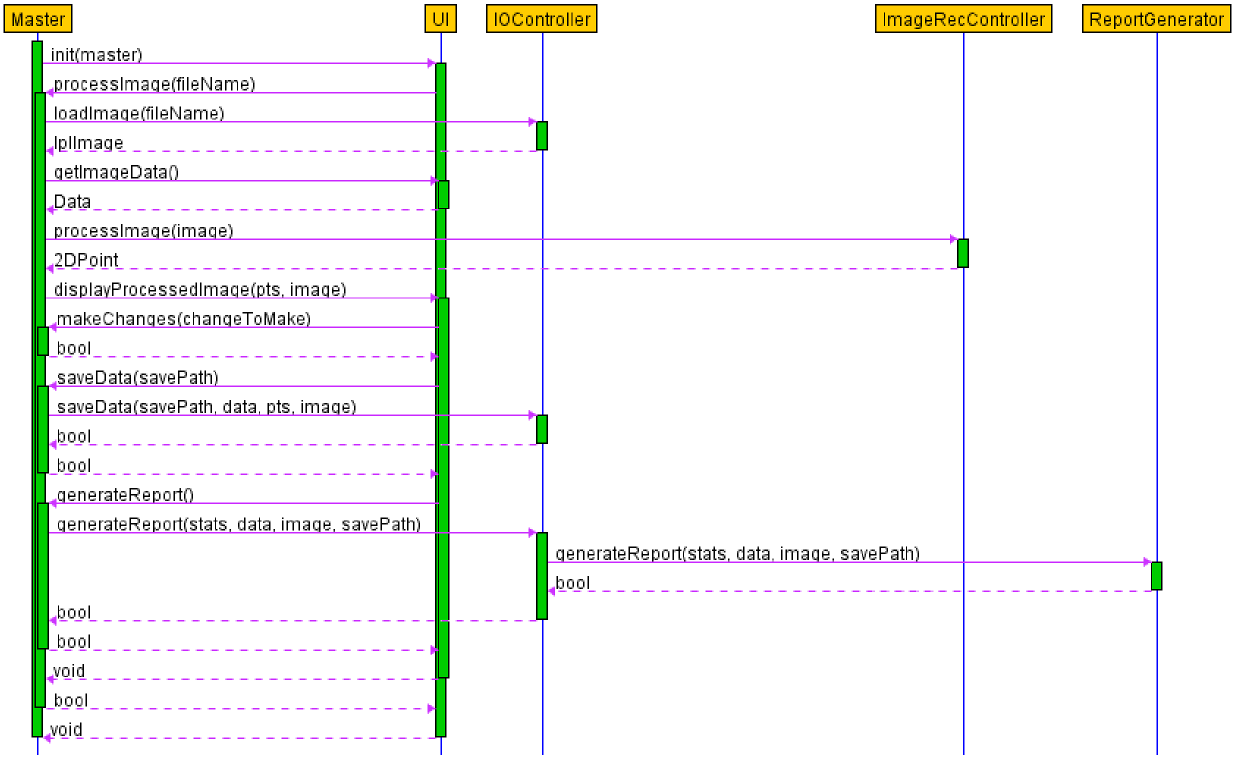
Datatype contains a large number of structures and classes that pass information between various classes.

4.2.18 Program

Program contains the Main() method which causes the entire application to run.

5 Sequence Diagram

5.1 Primary Presentation



5.2 Element Catalog

The primary classes in the sequence diagram represent the same classes as shown in section 4.

5.3 Architecture Background

This view shows how data will flow under a normal sequence by the user.

Glossary

caliber is the radius of a bullet.

openCV is the open source computer vision library used by this project to handle the image recognition portions.

References

[1] "C++." The C++ Resources Network, 16 January 2009, http://www.cplusplus.com/.

[2] "C#." Visual C# Developer Center, 16 January 2009, http://msdn.microsoft.com/en-us/vcsharp/default.aspx.

[3] ".NET Framework" Wikipedia, the free encyclopedia, 16 January 2009, http://en.wikipedia.org/wiki/.NET_Framework.

[4] "OpenCV." Wikipedia, the free encyclopedia, 13 November 2008, http://en.wikipedia.org/wiki/OpenCV.