6 – Emergency: Release 3 Summary

1 Team Members

Jad Khoriaty	1959220	(Project Lead)
Quynh-Anh Ly	6356370	
Walter Chacon	9238662	
Andrew Jia	9774491	
Gregory Fischer-Rush	2605929	
Sahil Nanda	1951815	

2 Project Summary

Our software, Emergency Team Dispatcher, will be used as a tool for a dispatcher to handle first aid teams in the context of a cultural or sport event. Teams can be assigned a sector to cover, or can be assigned onto an intervention to provide first aid services for a patient that requires medical attention. A team can also be used as backup for another team that needs advanced equipment to handle the emergency.

Replacing the old paper-based system, the software will basically be used to move around teams and assign them to interventions. However the software decreases the dispatcher's workload and cognitive load, rendering him more efficient and allowing him to handle a greater flow of information. During emergencies, where time is very limited, the software will enable the whole service to decrease their response time, thus increasing the chances of survival of the critical patient.

It will also document every team's movements and will be able to produce statistics on the overall first aid service provided in order to evaluate its quality. Documentation can either be used for any legal suit that is brought against our client, in order for them to prove that the correct procedure was followed and that there was no delay in the team's response. It can also be used as a marketing tool to prove the efficiency and quality of the service itself.

3 Important project updates

This releases focus was on a severe refactoring of the main program as well as the creation of an accompanying Android application. This is the reason for the lack of story points completed this iteration. We decided it was a better idea to take the time to refactor our current code rather than to proceed with architecture we knew to be less than ideal and, quite frankly, incorrect. Since there was still the development of several complex features to go before the final release, it was unanimously decided that this decision would pay dividends in the long run when you compare time spent now to time saved then.

The main goal of the refactoring was to implement the observer pattern throughout our code. There was many reasons for this, chief amongst them was the fact that the patterns of relations between the various classes, as well as what information was being passed where, had become overly convoluted. It had become very complex to decipher what class was receiving what information from which other class. This problem was largely solved through the use of the aforementioned pattern. This made the code a lot more modular which had the added benefit of being extremely useful when it came time to integrate the desktop software with its Android-based counterpart.

After speaking with the client, it became apparent that an Android application would greatly expedite and simplify workflow. Being able to have the location of the various teams automatically update on the map as well as allowing for an open channel of communication between operator and team members was very useful. As such an Android - based companion to the original software was created. This was the major development of the release.

4 Story Points

4.1 Overall Contributions (Desired individual velocity: 6.5 story points/iteration)

Name	ID	Number of story points contributed
Jad Khoriaty	1959220	95.5
Quynh-Anh Ly	6356370	57
Gregory Fischer-Rush	9238662	45.5
Sahil Nanda	1951815	42
Andrew Jia	9774491	43
Walter Alexander Chacon	9238662	45
AVERAGE:		54.6

4.2 Velocity (Desired velocity: 39 story points/iteration)

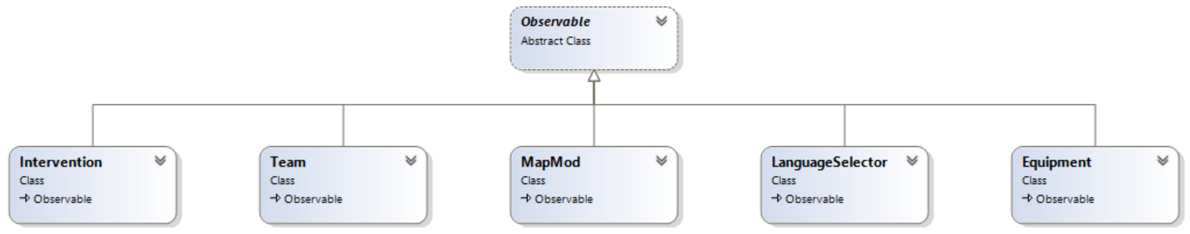
Iteration #	Actual Velocity (Only includes completed tasks)
8	13
Android (Iteration #8)	31
9	42 Not applicable (low due to solution-wide refactoring)
TOTAL:	86

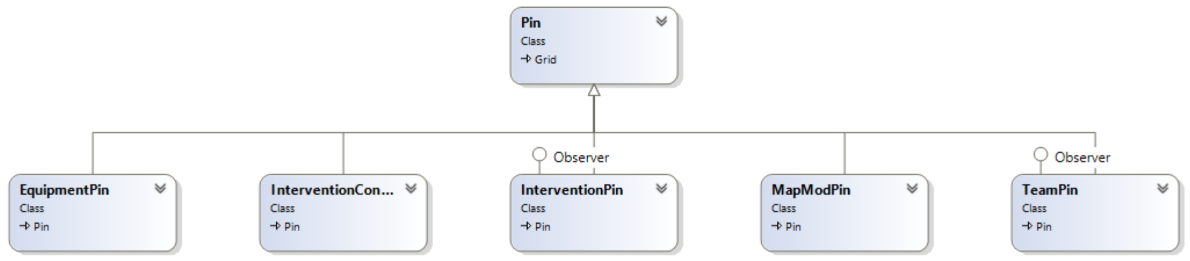
4.3 Notes

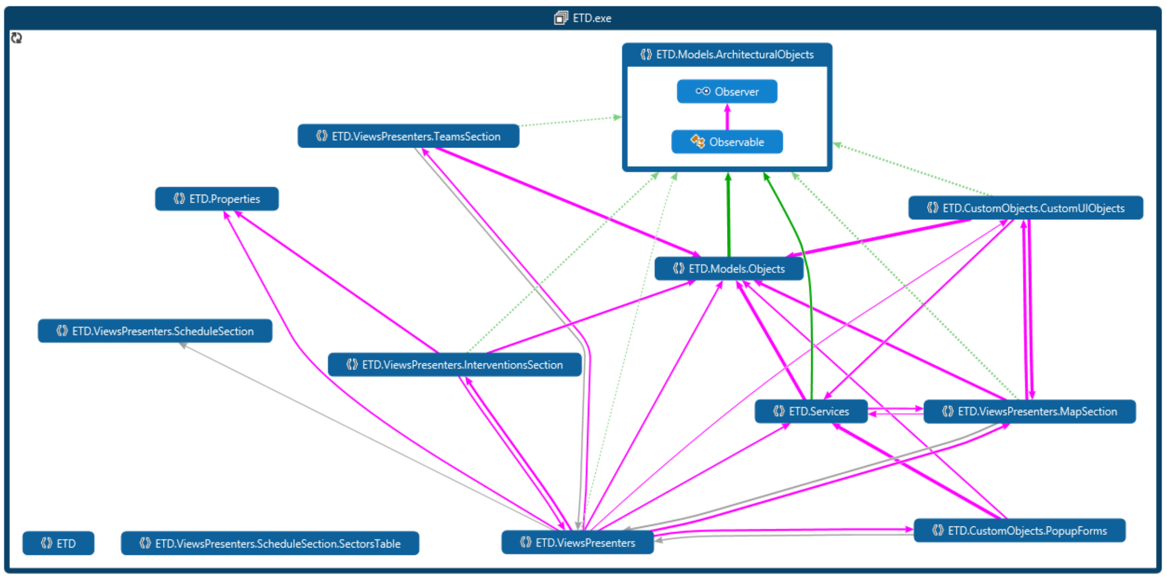
- No story points were completed during Iteration #9 due to the fact that the iteration was used to refactor the whole software. The whole architecture was redesigned and the software modified to reflect that new design. We are now relying heavily on the observer pattern and we have removed many (if not all) bad code-smells in the process of restructuring. Since all of the team's effort were put towards refactoring and restructuring the software, no stories have been completed although some progress was made on some stories in the process of refactoring.

5 ARCHITECTURE

5.1 DIAGRAMS






ETD.exe
ETD.Models.ArchitecturalObjects
Observer
Observable
ETD.ViewsPresenters.TeamsSection
ETD.Properties
ETD.CustomObjects.CustomUIObjects
ETD.Models.Objects
ETD.ViewsPresenters.ScheduleSection
ETD.ViewsPresenters.InterventionsSection
ETD.Services
ETD.ViewsPresenters.MapSection
ETD
ETD.ViewsPresenters.ScheduleSection.SectorsTable
ETD.ViewsPresenters
ETD.CustomObjects.PopupForms

5.2 Discussion

Up till now we have been using an MVP architecture for our project. However we noticed a few illegal interactions between parts of our system. For example MainWindow being able to access the equipment class directly.

Furthermore, we were not correctly following the object oriented principles. Whenever a team, intervention or equipment was created, the request would go through many middlemen before actually being generated. This would violate the Abstraction and encapsulation principles because the actual data implementation of the object can be accessed directly; not restricted to accessors and mutators. To remediate the situation we have decided to completely restructure our project such that it follows the Object Oriented Programming (OOP) principles. We have done so by focusing and using the Observer pattern extensively. Because the observer pattern allows us to maintain a list of dependents (observers), we can easily notify them whenever there is a change in their observed objects state. As such, the UI implementation of the object is hidden to the model. The Observable class defines the operations for registering and de-registering observers to the client; it is not aware of how methods are defined in the Observer interface. Concretely, the observer pattern has been implemented in two levels: An observer can register interest in an object type (i.e. whenever an object of that type is created or deleted, the observer gets notified, “static interest”), and/or it can register interest in an instance of that type (i.e. whenever the state of that instance changes, the observer gets notified, “object interest”).

For example, when the Team object is instantiated, sections that have registered static interest get notified, the map section handles the notification by creating a TeamPin that itself registers object interest in that team. If the team changes status, the TeamPin will be notified and will handle the notification by changing the color of the pin.

Our architecture has evolved greatly over the course of this release thanks to the implementation of the observer pattern throughout the program. This allows for a more modular approach and reduces coupling and increases cohesion by quite a bit.

6 Code

6.1 Code References

File Path	Purpose
ServerHandshakeThread.java: https://github.com/zukoj/SOEN490-ETD/blob/master/Android/MyApplication/app/src/main/java/concordia_university_capstone/etd_v10/ServerHandshakeThread.java	Tests connection with the server using UDP.
Pin.cs: https://github.com/zukoj/SOEN490-ETD/blob/master/ETD/ETD/CustomObjects/CustomUIObjects/Pin.cs	The parent class of all pin objects and handles collision detection. (This has been optimized since last release).
TeamPin.cs:	Sub-class of Pin, contains over-ride methods that implements behavior specific to TeamPins.

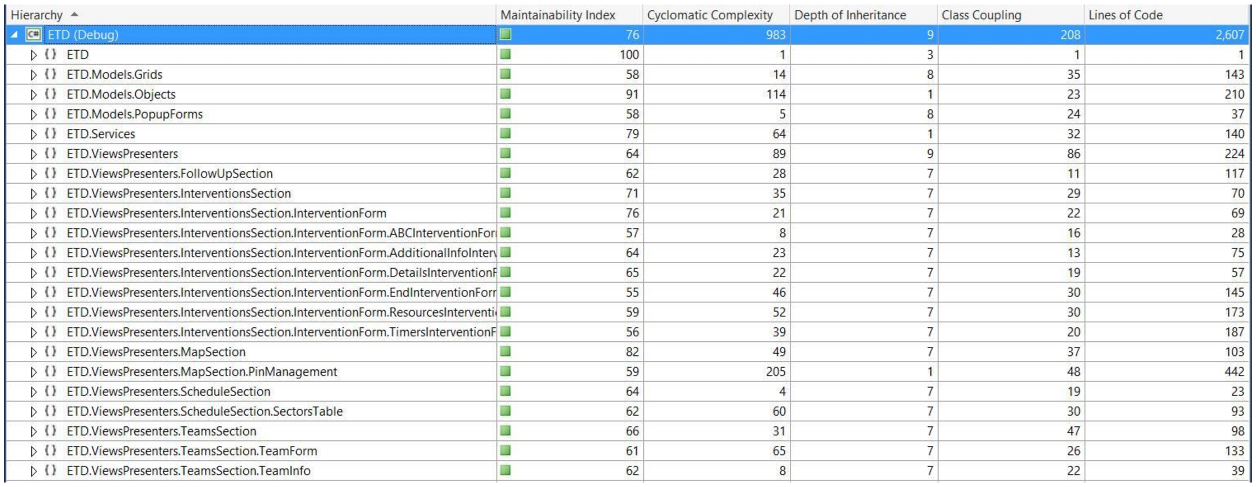
https://github.com/zukoj/SOEN490-ETD/blob/master/ETD/ETD/CustomObjects/CustomUIObjects/TeamPin.cs	
Team.cs: https://github.com/zukoj/SOEN490-ETD/blob/master/ETD/ETD/Models/Objects/Team.cs	Team class demonstrates how the observer pattern works.
Observable.cs: https://github.com/zukoj/SOEN490-ETD/blob/master/ETD/ETD/Models/ArchitecturalObjects/Observable.cs	Allows observers to register interest in either an object or an instance of that object.

Those files should be ignored as they are stubs that will be used for future development of features:

- ScheduleSectionPage.xaml (and .xaml.cs)
- SectorsTablePage.xaml (and .xaml.cs)
- LanguageSelector.cs

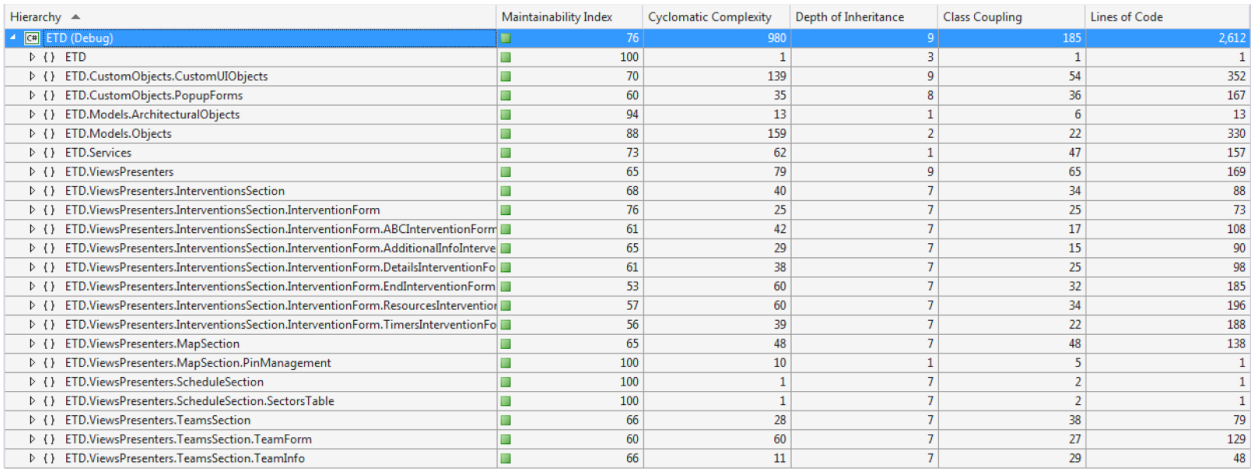
6.2 Code Metrics

Before the refactorization, our code metrics were the following:



Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
ETD (Debug)	🟩 76	983	9	208	2,607
ETD	🟩 100	1	3	1	1
ETD.Models.Grids	🟩 58	14	8	35	143
ETD.Models.Objects	🟩 91	114	1	23	210
ETD.Models.PopupForms	🟩 58	5	8	24	37
ETD.Services	🟩 79	64	1	32	140
ETD.ViewsPresenters	🟩 64	89	9	86	224
ETD.ViewsPresenters.FollowUpSection	🟩 62	28	7	11	117
ETD.ViewsPresenters.InterventionsSection	🟩 71	35	7	29	70
ETD.ViewsPresenters.InterventionsSection.InterventionForm	🟩 76	21	7	22	69
ETD.ViewsPresenters.InterventionsSection.InterventionForm.ABCInterventionFor	🟩 57	8	7	16	28
ETD.ViewsPresenters.InterventionsSection.InterventionForm.AdditionalInfoInter	🟩 64	23	7	13	75
ETD.ViewsPresenters.InterventionsSection.InterventionForm.DetailsInterventionF	🟩 65	22	7	19	57
ETD.ViewsPresenters.InterventionsSection.InterventionForm.EndInterventionFor	🟩 55	46	7	30	145
ETD.ViewsPresenters.InterventionsSection.InterventionForm.ResourcesInterventi	🟩 59	52	7	30	173
ETD.ViewsPresenters.InterventionsSection.InterventionForm.TimersInterventionF	🟩 56	39	7	20	187
ETD.ViewsPresenters.MapSection	🟩 82	49	7	37	103
ETD.ViewsPresenters.MapSection.PinManagement	🟩 59	205	1	48	442
ETD.ViewsPresenters.ScheduleSection	🟩 64	4	7	19	23
ETD.ViewsPresenters.ScheduleSection.SectorsTable	🟩 62	60	7	30	93
ETD.ViewsPresenters.TeamsSection	🟩 66	31	7	47	98
ETD.ViewsPresenters.TeamsSection.TeamForm	🟩 61	65	7	26	133
ETD.ViewsPresenters.TeamsSection.TeamInfo	🟩 62	8	7	22	39

This is how our quality looked like after the refactoring of our code:



Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
ETD (Debug)	🟩 76	980	9	185	2,612
{ } ETD	🟩 100	1	3	1	1
{ } ETD.CustomObjects.CustomUIObjects	🟩 70	139	9	54	352
{ } ETD.CustomObjects.PopupForms	🟩 60	35	8	36	167
{ } ETD.Models.ArchitecturalObjects	🟩 94	13	1	6	13
{ } ETD.Models.Objects	🟩 88	159	2	22	330
{ } ETD.Services	🟩 73	62	1	47	157
{ } ETD.ViewsPresenters	🟩 65	79	9	65	169
{ } ETD.ViewsPresenters.InterventionsSection	🟩 68	40	7	34	88
{ } ETD.ViewsPresenters.InterventionsSection.InterventionForm	🟩 76	25	7	25	73
{ } ETD.ViewsPresenters.InterventionsSection.InterventionForm.ABCInterventionForm	🟩 61	42	7	17	108
{ } ETD.ViewsPresenters.InterventionsSection.InterventionForm.AdditionalInfoInterve	🟩 65	29	7	15	90
{ } ETD.ViewsPresenters.InterventionsSection.InterventionForm.DetailsInterventionFo	🟩 61	38	7	25	98
{ } ETD.ViewsPresenters.InterventionsSection.InterventionForm.EndInterventionForm	🟩 53	60	7	32	185
{ } ETD.ViewsPresenters.InterventionsSection.InterventionForm.ResourcesInterventio	🟩 57	60	7	34	196
{ } ETD.ViewsPresenters.InterventionsSection.InterventionForm.TimersInterventionFo	🟩 56	39	7	22	188
{ } ETD.ViewsPresenters.MapSection	🟩 65	48	7	48	138
{ } ETD.ViewsPresenters.MapSection.PinManagement	🟩 100	10	1	5	1
{ } ETD.ViewsPresenters.ScheduleSection	🟩 100	1	7	2	1
{ } ETD.ViewsPresenters.ScheduleSection.SectorsTable	🟩 100	1	7	2	1
{ } ETD.ViewsPresenters.TeamsSection	🟩 66	28	7	38	79
{ } ETD.ViewsPresenters.TeamsSection.TeamForm	🟩 60	60	7	27	129
{ } ETD.ViewsPresenters.TeamsSection.TeamInfo	🟩 66	11	7	29	48

So our lines of code went down even though there was functionality added as well as an improvement in our class coupling. There was also minor improvements in cyclomatic complexity.

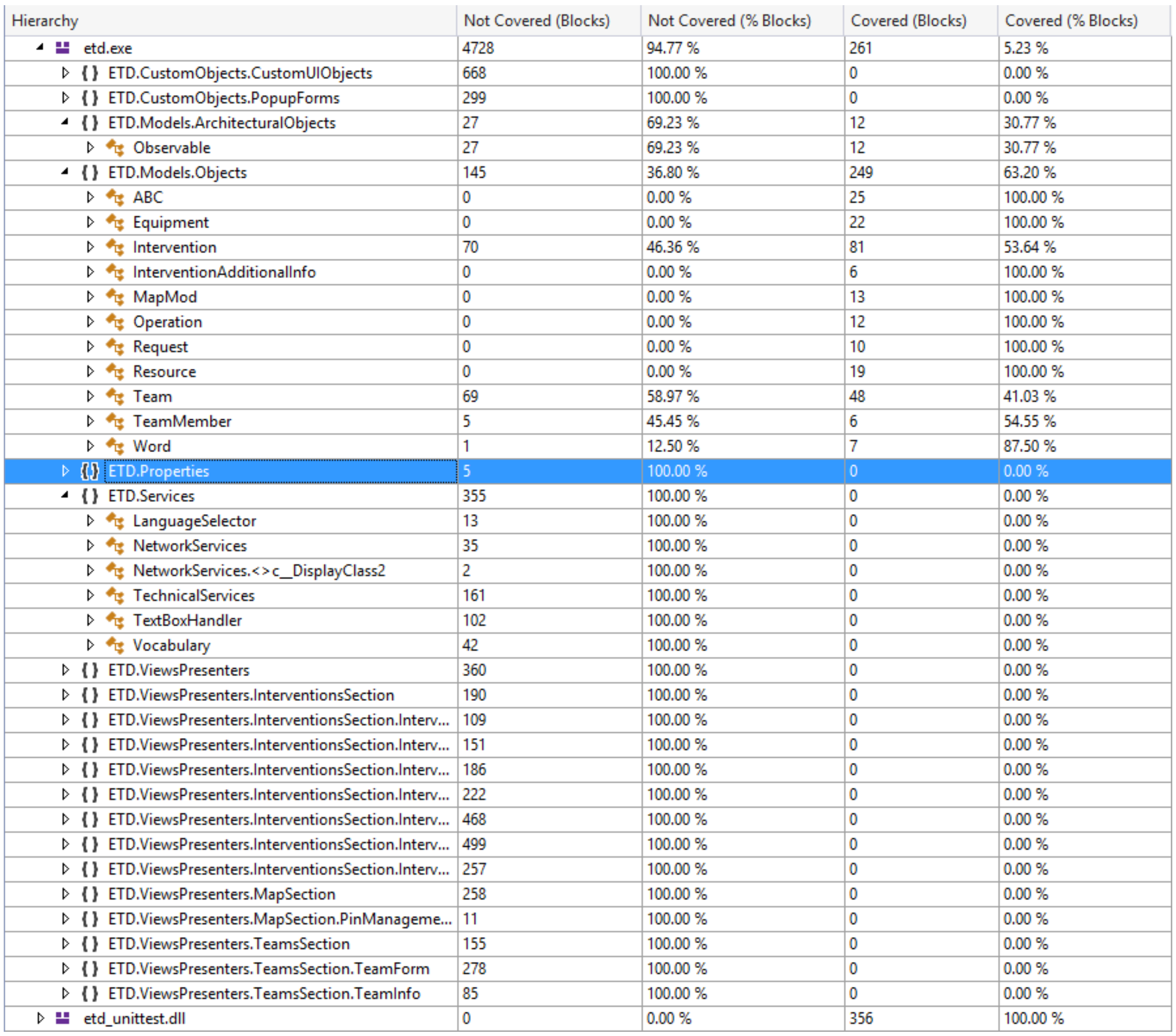
7 TESTING

We did unit tests for all of the classes in the model. We stored our tests in different folders, the Model Objects or targeting different groups of classes together. However some classes and methods outside of the model are missing unit tests because they're calling UI objects and for these, we did black box testing where we test the functionality of the code targeting the specific task. The unusual aspects of the testing approach for unit test is that when we make changes to the classes and functions, we will have to re-write the tests since the ones we wrote originally don't pass anymore so we constantly have to check on the tests and run them to see if they pass or not. We tested each of our completed stories because after the implementation of each functionality, it goes into QA.

For integration testing, we tested the creation of classes where many classes are required and if they are initialized correctly. As an example, inside the TeamTest, we tested the creation of teams with regards to TeamMember.cs and Equipment.cs. We also tested the creation of simple model objects and whether they appear in the UI classes.

Test Class File	**Related Class File**	**Coverage (%)**
ABCTest.cs	ABC.cs	100
EquipmentTest.cs	Equipement.cs	100
InterventionAdditionalInfoTest	InterventionAdditionalInfo.cs	100
InterventionTest.cs	Intervention.cs, ABC.cs	53.64
MapModTest.cs	MapMod.cs	100

OperationTest.cs	Operation.cs	100
RequestTest.cs	Request.cs	100
ResourceTest.cs	Resource.cs, Team.cs	100
TeamMemberTest.cs	TeamMember.cs	54.55
TeamTest.cs	Team.cs, TeamMember.cs, Equipment.cs	41.03
WordTest.cs	Word.cs	87.50
TeamUITest.cs	MapSectionPage.cs, Pin.cs, Team.cs, MainWindow.cs	10.97



Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
etd.exe	4728	94.77 %	261	5.23 %
ETD.CustomObjects.CustomUIObjects	668	100.00 %	0	0.00 %
ETD.CustomObjects.PopupForms	299	100.00 %	0	0.00 %
ETD.Models.ArchitecturalObjects	27	69.23 %	12	30.77 %
Observable	27	69.23 %	12	30.77 %
ETD.Models.Objects	145	36.80 %	249	63.20 %
ABC	0	0.00 %	25	100.00 %
Equipment	0	0.00 %	22	100.00 %
Intervention	70	46.36 %	81	53.64 %
InterventionAdditionalInfo	0	0.00 %	6	100.00 %
MapMod	0	0.00 %	13	100.00 %
Operation	0	0.00 %	12	100.00 %
Request	0	0.00 %	10	100.00 %
Resource	0	0.00 %	19	100.00 %
Team	69	58.97 %	48	41.03 %
TeamMember	5	45.45 %	6	54.55 %
Word	1	12.50 %	7	87.50 %
ETD.Properties	5	100.00 %	0	0.00 %
ETD.Services	355	100.00 %	0	0.00 %
LanguageSelector	13	100.00 %	0	0.00 %
NetworkServices	35	100.00 %	0	0.00 %
NetworkServices.<>c__DisplayClass2	2	100.00 %	0	0.00 %
TechnicalServices	161	100.00 %	0	0.00 %
TextBoxHandler	102	100.00 %	0	0.00 %
Vocabulary	42	100.00 %	0	0.00 %
ETD.ViewsPresenters	360	100.00 %	0	0.00 %
ETD.ViewsPresenters.InterventionsSection	190	100.00 %	0	0.00 %
ETD.ViewsPresenters.InterventionsSection.Interv...	109	100.00 %	0	0.00 %
ETD.ViewsPresenters.InterventionsSection.Interv...	151	100.00 %	0	0.00 %
ETD.ViewsPresenters.InterventionsSection.Interv...	186	100.00 %	0	0.00 %
ETD.ViewsPresenters.InterventionsSection.Interv...	222	100.00 %	0	0.00 %
ETD.ViewsPresenters.InterventionsSection.Interv...	468	100.00 %	0	0.00 %
ETD.ViewsPresenters.InterventionsSection.Interv...	499	100.00 %	0	0.00 %
ETD.ViewsPresenters.InterventionsSection.Interv...	257	100.00 %	0	0.00 %
ETD.ViewsPresenters.MapSection	258	100.00 %	0	0.00 %
ETD.ViewsPresenters.MapSection.PinManageme...	11	100.00 %	0	0.00 %
ETD.ViewsPresenters.TeamsSection	155	100.00 %	0	0.00 %
ETD.ViewsPresenters.TeamsSection.TeamForm	278	100.00 %	0	0.00 %
ETD.ViewsPresenters.TeamsSection.TeamInfo	85	100.00 %	0	0.00 %
etd_unittest.dll	0	0.00 %	356	100.00 %

We primarily focused on developing tests for the Domain part of our project, which is heavily UI-based (almost 85% of the code). For this reason, our measured test coverage is very low. However, when considering only the Domain part, our test coverage becomes 33%. While this is an improvement, it is still not to our satisfaction. Our coverage went down when we started doing the refactoring during the last iteration causing a lot of the tests to need to be reworked. This is something that will be prioritizes during the next week as well as producing UI tests.

8 Story Summaries

Story: As an Android user, I can log in to the Android application using my given Name
https://trello.com/c/8WkXFmPQ/1-1-the-android-application-will-take-the-name-of-the-volunteer-before-transmitting-2
Feature: "Android Application"
Points: 3, Priority: High, Risk: Low
This story involved the creation of a login page to the android application that would get the name of the person using the application and transmit that information to the server and thus on to the main desktop application.

Story: As a desktop user I can retrieve the position of various members via their phone's GPS.
https://trello.com/c/oTbUGI5d/12-5-the-android-application-should-transmit-the-location-even-if-it-is-not-on-the-forefront-of-the-phone-and-even-if-the-phone-is-
Feature:
Points: 5, Priority: High, Risk: Medium
This story involved linking the main desktop application to the server and setting it up to receive location information given to it by said server.

Story: As an Android user I want to keep notifying the server of my position even if I don't have the application in the forefront.
https://trello.com/c/oTbUGI5d/12-5-the-android-application-should-transmit-the-location-even-if-it-is-not-on-the-forefront-of-the-phone-and-even-if-the-phone-is-
Feature:
Points: 5, Priority: High, Risk: Medium
This story involved setting up the application to be able to deliver it's position to the server even if the phone was put on Standby or if another application was opened and placed in the foreground.

Story: As a desktop user I want to resort my teams on the UI by dragging and dropping them.
https://trello.com/c/3cTXi2bd/33-8-resort-teams-with-drag-and-drop-2
Feature:
Points: 8, Priority: Low, Risk: Low
This story involved setting up the user interface to allow for the resorting of the order of appearance of teams in the left-hand portion of the screen.

Story: As a desktop user, I want to be able to use the software in a language I am most comfortable in.
https://trello.com/c/aBoudunF/1-13-language-switching-9
Feature:
Points: 13, Priority: Critical, Risk: Medium
This story involved setting up several resource files, one per language, in order to easily switch between them and remove any hard coded UI text.

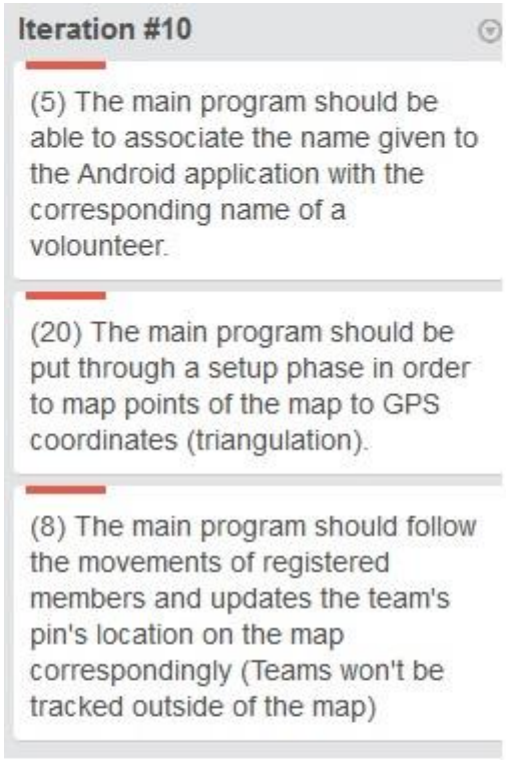
Story: As a desktop user, I want to be able to keep track of any additional requests made including when and who fulfilled them. https://trello.com/c/GC1JmkPd/32-5-request-table-with-follow-up-and-completed-time-5 Feature: Points: 5, Priority: High, Risk: Medium This story involved creating involved setting up a form to track the needed information without obstructing the user's view of the map.
Story: As a desktop user, I want to be able to annotate or be able to add simple figures onto the map in order to add useful information, so that I'll be able to identify certain locations by their annotated name, safety passages, access ramps, or any other pertinent information that increases my efficiency. https://trello.com/c/fTxRqK2Z/16-13-14-annotate-or-add-figures-to-map-40 Feature: Points: 13, Priority: Medium, Risk: Medium This story involved adding the ability to add various shapes over the map without interfering with the team/intervention tracking capabilities.
Story: As a desktop user, I want to be able to save any annotations done on the map in order to reuse them later without having to redraw them. https://trello.com/c/OJeet9yl/37-20-15-save-map-with-modifications-5 Feature: Points: 20, Priority: Medium, Risk: High This story involved finding a way of saving all changes drawn on a map so they can be retrieved later on.

9 STORY MAP FOR NEXT RELEASE

For the next release we will have two teams working in parallel in order to get both of our programs done for the release. One team will be working on the android application while the other will be working on the main project. It is worth noting that all aspects of handling and incorporating the information provided by the app in the main program will count as the android Section.

9.1 Iteration #10 (119 story points, some stories will take more than this iteration):

9.1.1 Android Team (33 story points):



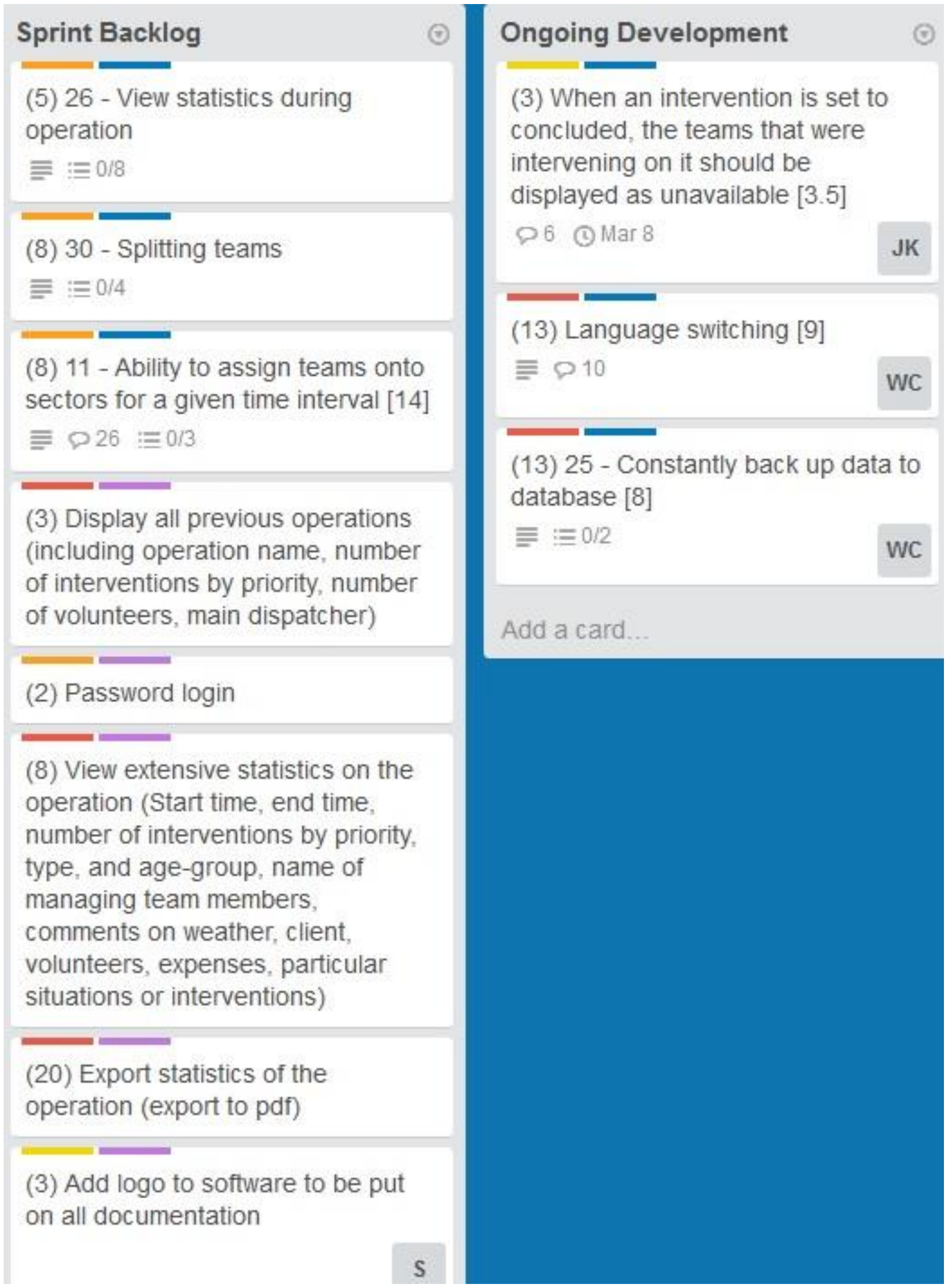
Iteration #10

(5) The main program should be able to associate the name given to the Android application with the corresponding name of a volounteer.

(20) The main program should be put through a setup phase in order to map points of the map to GPS coordinates (triangulation).

(8) The main program should follow the movements of registered members and updates the team's pin's location on the map correspondingly (Teams won't be tracked outside of the map)

9.1.2 C# Team (86 story point, some are expected to bleed into iteration #11):



Sprint Backlog

(5) 26 - View statistics during operation
0/8

(8) 30 - Splitting teams
0/4

(8) 11 - Ability to assign teams onto sectors for a given time interval [14]
26 0/3

(3) Display all previous operations (including operation name, number of interventions by priority, number of volunteers, main dispatcher)

(2) Password login

(8) View extensive statistics on the operation (Start time, end time, number of interventions by priority, type, and age-group, name of managing team members, comments on weather, client, volunteers, expenses, particular situations or interventions)

(20) Export statistics of the operation (export to pdf)

(3) Add logo to software to be put on all documentation
S

Ongoing Development

(3) When an intervention is set to concluded, the teams that were intervening on it should be displayed as unavailable [3.5]
6 Mar 8
JK

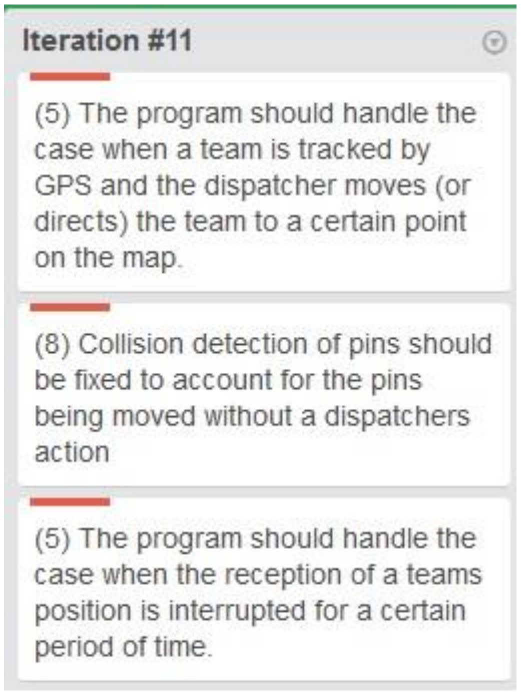
(13) Language switching [9]
10
WC

(13) 25 - Constantly back up data to database [8]
0/2
WC

Add a card...

9.2 ITERATION #11 (66 STORY POINTS, AS WELL AS ANY LEFTOVERS FROM ITERATION #10):

9.2.1 Android Team (18 story points, to be completed as soon as possible so that team can move back to the main project):



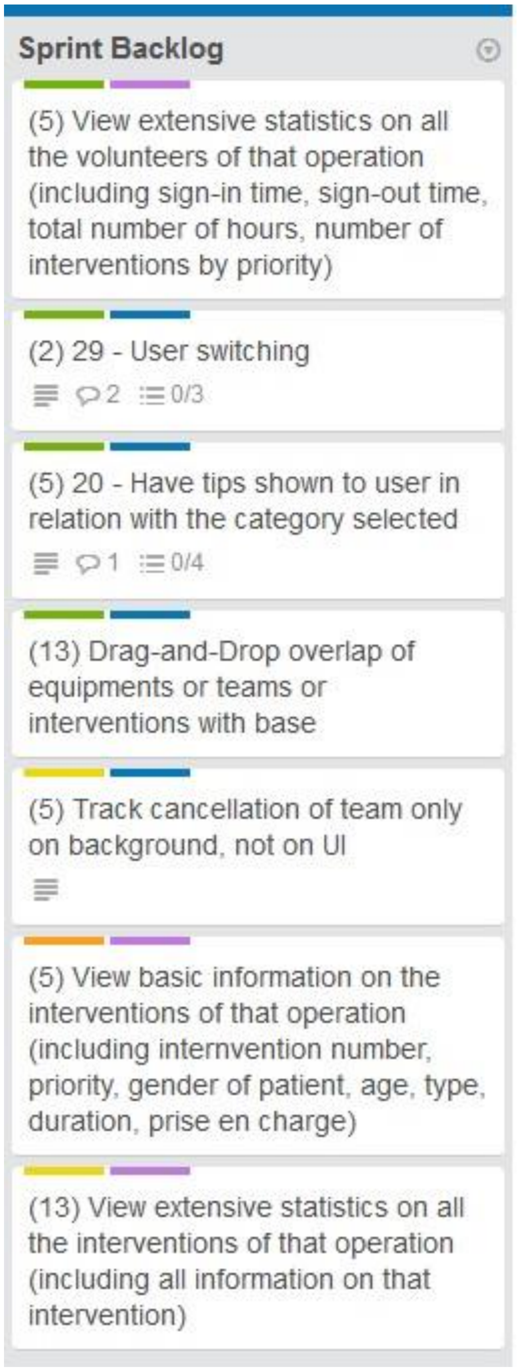
Iteration #11

(5) The program should handle the case when a team is tracked by GPS and the dispatcher moves (or directs) the team to a certain point on the map.

(8) Collision detection of pins should be fixed to account for the pins being moved without a dispatchers action

(5) The program should handle the case when the reception of a teams position is interrupted for a certain period of time.

9.2.2 C# Team (48 story points, as well as any leftovers from iteration #10):



Sprint Backlog

(5) View extensive statistics on all the volunteers of that operation (including sign-in time, sign-out time, total number of hours, number of interventions by priority)

(2) 29 - User switching
2 0/3

(5) 20 - Have tips shown to user in relation with the category selected
1 0/4

(13) Drag-and-Drop overlap of equipments or teams or interventions with base

(5) Track cancellation of team only on background, not on UI

(5) View basic information on the interventions of that operation (including internvention number, priority, gender of patient, age, type, duration, prise en charge)

(13) View extensive statistics on all the interventions of that operation (including all information on that intervention)