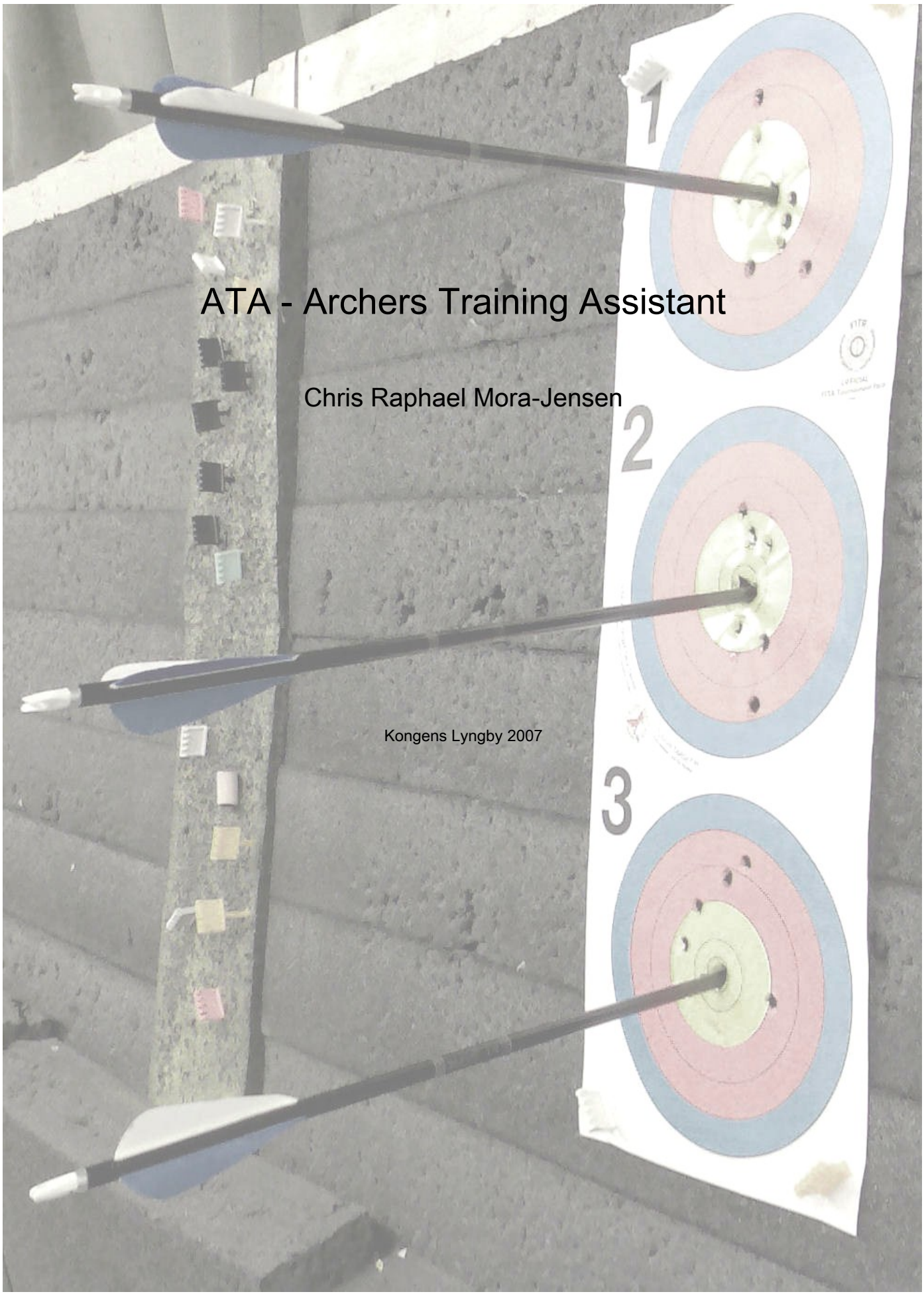


ATA - Archers Training Assistant

Chris Raphael Mora-Jensen

Kongens Lyngby 2007

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

Table of Contents

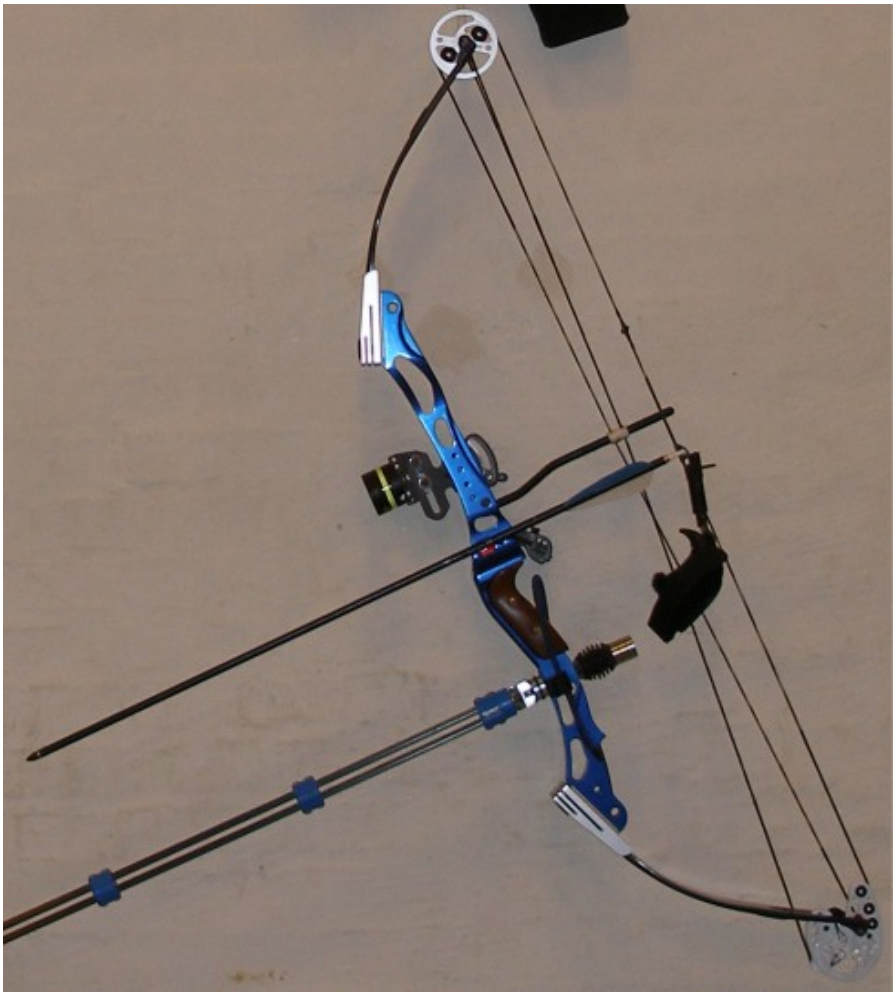
Preface

This thesis reports on the development of an Archers Training Assistant (ATA). The assistant consists of a PDA application and a PC application. The PDA application can be used by archers to track their skills, control a tournament or even as a bow-tuning reference. The PC application can be used as a tool for reviewing saved training sessions. It is also possible to use the PC application to review tournaments or even as a part of the set of tools tournament staff could use.

The total timespan of this project has been 10 weeks and resulted in this report, a PDA application and a PC application. The product is developed as a tool archers could use for tracking their form and finding areas that needs improvement. Several archers (including danish elite archers) has contributed with valuable information and shown great interest in especially the PDA application.

1 Introduction

Compound archery is a very technical kind of archery and very new. The inventor, Hollus Wilbur Allen, patented his invention, which he called “Compound Bow” in the late 1960's. Compound bows are fitted with cables and string and pulleys, their sights are sophisticated ultra fine tunable equipment. Arrow rests constructed so they almost won't touch the arrow when fired. Arrows with weights and special vanes, that will rotate the arrow for a straight flight Mechanical active stabilizers that removes vibration from the bow so it doesn't interfere with the arrow and release aids that can triggered any way you want. On the most basic setup on a compound bow, there are probably 10 different knobs or screws that can be adjusted. Not two sets of equipment are alike and there are as many setups as there are archers. This means that if a bow isn't tuned to perform its best, there isn't a general way to tune it, it depends on the archer using the bow. And when tuning a bow the result is not likely to be visible after 1 or 2 shots or even 10 shots.



Compound Bow

1.1 Aim

The aim of the application is to provide the archer with a tool which can be used to track progress in the performance and form of the archer and as a help to fine tune the bow. However the archer should bare in mind that the application is not the solution to everything. It cannot replace traditional tuning methods, but can provide useful information when tuning the Compound Bow.

Besides the tuning of the bow, the application should also be able to show arrow hit tendency. The goal of every archer is to make their arrows fly the same way, every time. Arrows fired from a compound bow travels with at least 250 km/h (typical at 290 - 320km/h) and are decelerated down to 0 km/h in less than 50 cm. Needless to say that all of this energy somehow affects the

arrow, sometimes they are visible damaged, other times small invisible bends or invisible vane damage makes the arrow fly less than perfect. With the application it should be possible to find those invisible errors by tracking where the arrow hits. Faulty arrows tend to deviate from the flying path of healthy arrows. But only by recording hits over time will it be possible to find the arrow.

Archers uses many hours training for tournaments and in the weeks before an important tournament, many archers train the actual tournament distances and number of arrows used in the tournament. It is therefore useful with a tool that can act as a tournament program with selected distances and number of arrows. This tournament tool could even act as the scorecard used by archers at tournaments, if it could hold scores for several archers. Trainers could use it to keep track of different archers and their form.

When equipment is damage and replaced, the tuning of the new equipment is of course required, but it is also interesting to see if the new equipment fits the archer as good as the old equipment. Many archers save their scorecards from training or a least the end result for this reason, but it would be even better if the exact data was available. By saving training sessions and tournament result and reviewing them on a PC later could provide the archer with information regarding how the equipment fits him or her, and makes it possible to test different setups and see the difference graphically instead of just by scores.

1.2 Target Group

This set of applications is intended for the compound archer who competes and/or wants to keep track of the form, track progress with the equipment and compare different tunings of the bow or the various equipment setups.

2 Requirements

2.1 Problem Definition

With the Aim paragraph as source the following is considered to be the main tasks to be solved.
As a training aid, it is desired to develop a PDA application, which should:

1. PDA
    1. Be able to accept data from an archer after each shooting end
    2. Show various deviations
    3. Hold arrow count and score.
    4. Allow archers to save the collected data.

Besides the training "mode", two tournament settings should be constructed.

5. tournament with several archers, pre-decided distances, pre-decided number of arrows per round and pre-decided number of arrows in total.
6. tournament with one archer, pre-decided distances, pre-decided number of arrows per round and pre-decided number of arrows in total.
7. Scorecards from tournaments should be saved.

As an extension to the PDA application a few PC tools should be implemented as well

2. PC
    1. review saved training sessions.
    2. A tournament control tool to have an overview of a tournament, where archers scorecards can be reviewed and compared.

3 Method

As a development method a combination suited my needs best, as initial method Test-Driven-Development has been used. This gave me a solid foundation but is insufficient when moving on to the graphic user interface. The only prober way to test PDA applications, is by testing on the PDA. It is the only way make sure it behaves the way you want it to behave. It was therefore natural to combine it with an iterated development plan. The TDD implementation is implement into the iteration plan as the iteration 0. TDD is described later in the report.

To complicate it even more, a PC part is also proposed, but the PC part depends heavily on how the PDA functions and which kind of data the PDA is able to supply. This is the reason why the PC part is started later and follows its own iteration plan.

3.1 Iteration Plan

The table below is divided into 10 rows, each row representing one week of the 10 weeks of the project span.

Weeks	Planned	Progress details
1st Week	Analysis (PDA, PC)	Analysis and defining rules for compound bows scoring, research at archery range.
2nd Week	**Iteration 0 PDA** using TDD	Implementing basic functions through TDD, **Iteration 0** for the PDA contains the baseline for training and tournament.
3rd Week	GUI	First simple graphic user interface implemented for training mode on the PDA, short discussion with other archers.
4th Week	**Iteration 1 PDA**, *Iteration 0 PC* Emulator test	First simple graphic user interface implemented for tournament. **Iteration 1** for PDA is usable. Defining needs for the PC application. PC contains basic functionality, *Iteration 0*.
5th Week	**Iteration 2 PDA**, *Iteration 1 PC* PC test of UI, PDA emulator test of UI	Further GUI implemented and tested **Iteration 2**. PC application contains useful functions, PC is now *Iteration 1*. Report begins
6th Week	Report, Shooting range testing, code stop for PDA application, **Iteration 3**	Report, testing PDA application at shooting range, minor fixes, **Iteration 3/Beta**.
7th Week	Report, test PC with real data. Code stop for PC application, *Iteration 2*	Report, testing PC with results collected from shooting range minor fixes PC, PC is now *iteration 2/Beta.*
8th Week	Report	Report
9th Week	Report	Report, brief discussion with other archers as they was introduced the project. Interesting comments
10th Week	Review and finalizing report.	Report review and correction.

4 Risk Management

A range of different risks might arise during the project, some of them have little or no impact while others might stall the project for some time. Below are described some of the major concerns to this project.

4.1 Personal Risks

Personal Risks includes those risks that will prevent me from working on the project. Light illness isn't considered as a threat to the project but serious illness, might result in an extension of the deadline, until recovered.

4.2 Data Risks

Data Risks includes any risks as data loss, hardware failure or circumstances that interferes with the hardwares performance. The most serious of those is the data loss. To prevent data loss backups at least once a day is advised.

4.3 Programing to Hand held Devices, a hidden risk?

Although my experience in programing to hand held devices is very limited, it is not considered a risk that will result in bringing the project to a stand still. I suppose that problems in this area aren't to complex to be resolved by researching the Internet or creating a workaround.

4.4 Hardware Risks

The hand held device makes use of different kind of graphic calculations and rendering, but it is assumed that these wont be to complex for the device available. Should it prove to be a problem possible solutions could either be, reducing the graphics to a device friendly level, or simply test it on a bigger device/newer device.

5 Analysis

The use case scenarios is based on my own experience from training and from my very little experience in tournaments, I have only participated in one tournament so changes for these is not unlikely. The biggest problem with the tournaments however is not my lack of experience but the fact that none of the applications are very likely to be approved as legal tournament score recording application. With that in mind some aspects of the tournament application has been designed with the intended use in clubs and not for official tournaments. In big tournaments names aren't actually used but lane numbers instead and so on. The two following paragraphs are meant to give the reader an idea of the two typical scenarios, it is not meant to be interpreted as use cases or as how it is done every time, archers have their own way of training and both are highly based on own experience. Both are sequences are before PDA application is involved.

5.1 Typical Training Sequence

This is the description of a typical training sequence. It doesn't show the more specialized sequences of how fine tuning of the bow or checking equipment is done.

First bow is assembled and checked with the rest of the equipment. Target face attached to the arrow stop. Archer takes stance and draws the bow, sight and other things like grip and anchor-point is checked. If everything feels good, trigger is pulled, this is repeated until all arrows are fired and ready to be collected. When arrows are collected, score is noted. If the archer finds a general deviation (might not be possible to find a deviation) the sight is adjusted accordingly. When training is done scorecard or total score is saved for tracking progress

5.2 Typical Tournament Sequence

Please note that this description is based on a single tournament experience. Archers are divided into several groups. The number of groups depends on number of archers and number of shooting lanes available at the tournament. Each archer is then assigned to a lane, each lane is typically shared between 3 archers, one archer from each group. First group is called to the shooting line, they shoot the pre-decided number of arrows for the end specified in the tournament description. When done next group of archers take their stand and shoots their arrows, and so on. After all groups have shoot their arrows, all archers walks to their target face (each archer has his or her own target face) and meet with the archers they share their lane with. The archers now writes the score for this end, and tournament staff is given the scores so far. At tournament end total scores are calculated, published and winner celebrated.

5.3 Use Cases

Use Case 1	UCHP1: Practice in general
Description	The general training session, no special goal to be achieved other than registering score.
Actors	Archer
Assumption	PDA application works
Success end condition	Application registers plotted arrows successfully and archers saves session successfully.
Failed end Condition	Nothing registered or not able to save session.
Steps	1 Archer starts application and selects training mode 2 Archer enters, distance, number of arrows per end, face type and arrow size. 3 Repeat until training session is over 3.1 repeat for each arrow 3.1.1 Archer plot arrows at end. 3.1.2 Archer is asked if scores is right 3.1.2.1 if score not accepted, arrow is replotted (3.1.1) 3.1.3 score is added to scorecard and saved in memory with arrow coordinates 4 Review scorecard 5 save training session
Sub variations	none

Use Case 2	UCHP2: tuning sight
Description	One of the more special uses of the application, not really a unique use case since many of the steps are similar to UCHP1, but still not a sub variation. Used to fine tune sight before
Actors	Archer
Assumption	PDA application works, archer is somewhat trained so data entered is reliable
Success end condition	Archer is able to tune sight
Failed end Condition	Application fails to provide useful information and sight isn't tuned correctly
Steps	1 Archer starts application and selects training mode 2 Archer enters, distance, number of arrows per end, face type and arrow size. 3 Repeat until sight is tuned 3.1 repeat for each arrow 3.1.1 Archer plot arrows at end. 3.1.2 Archer is asked if scores is right 3.1.2.1 if score not accepted, arrow is replotted (3.1.1) 3.1.3 score is added to scorecard and saved in memory with arrow coordinates 3.2 check end deviation tab page and set sight accordingly 4 end of sight tuning.
Sub variations	none

Use Case 3	UCHP3: Check for bended arrows
Description	Also a special use with many steps similar to UCHP1. Use case shows how the application can be used to find bad arrows.
Actors	archer
Assumption	PDA application works, archer is somewhat trained so data entered is reliable, arrows marked or numbered individual, archers shots the arrows in the same sequence each time.
Success end condition	Arrow straightness verified
Failed end Condition	PDA fails to show useful information for verifying arrows.
Steps	1 Archer starts application and selects training mode 2 Archer enters, distance, number of arrows per end, face type and arrow size. 3 Repeat until sufficient data is entered 3.1 repeat for each arrow 3.1.1 Archer plot arrows at end. 3.1.2 Archer is asked if scores is right 3.1.2.1 if score not accepted, arrow is replotted (3.1.1) 3.1.3 score is added to scorecard and saved in memory with arrow coordinates 4 read total deviation page
Sub variations	none

Use Case 4	UCHT1: Single archer tournament
Description	Shows the usage of the PDA application when used in training for a tournament
Actors	archer
Assumption	Application works
Success end condition	Tournament training completed
Failed end Condition	Application fails to record tournament scores
Steps	1. Archer starts application and selects tournament mode. 2. Archer enters name and proceeds to tournament setup 3. Archer selects wanted distances and number of arrows in total and per end. Starts tournament 4. Scores are recorded for each distance 5. when tournament completed it is saved.
Sub variations	none

Use Case 5	UCHT2: Multiple archers tournament
Description	Shows the application usage when as a non official tournament, eg in the local club
Actors	archers
Assumption	Application works
Success end condition	Application is able to record scores for each individual archer competing in the tournament.
Failed end Condition	Scores not recorded, no winner found
Steps	1. Archer in charge of registering scores, starts application and selects tournament mode. 2. Archer enter names of competing archers and proceeds to tournament setup 3. Archer selects wanted distances and number of arrows in total and per end. Starts tournament 4. scores are recorded at each end for each archer at the correct distance. 5. Winner found at tournament end.
Sub variations	5a. Winner found at tournament end and tournament results are saved

Use Case 6	UCHT3: Score collector for one or several lanes
Description	The proposed usage at an actual tournament. However it is very unlikely this application will be used for anything else than unofficial score tracking since its not approved by neither FITA or DBSF. This use case scenario extends to UCPT2
Actors	Tournament official responsible for collecting scores, archers
Assumption	PDA application works
Success end condition	Scores recorded successfully and submitted to tournament control
Failed end Condition	Fails to record scores or save scores at end
Steps	6. Tournament staff in charge of registering scores, starts application and selects tournament mode. 7. Staff enter names of competing archers and proceeds to tournament setup 8. Staff selects distances decided for the tournament and number of arrows in total and per end. Starts tournament 9. scores are recorded at each end for each archer at the correct distance. 10. Tournament is saved and saved files are handed over to tournament control (UCPT2).
Sub variations	

Use Case 7	UCPP1:Review Training
Description	Review training saved, various information can be found by analyzing the data.
Actors	Archer
Assumption	Application works, training file available
Success end condition	Training session loaded.
Failed end Condition	File damaged, unable to load training
Steps	1. ATA PC Tool started and user selects training 2. User selects the correct training file at the File Dialog 3. training review can be begin
Sub variations	

Use Case 8	UCPP2:Compare Training
Description	Can be used to compare several training sessions, and check progress
Actors	Archer
Assumption	Application works, training files available
Success end condition	Training sessions loaded.
Failed end Condition	Files damaged, unable to load training sessions
Steps	4. ATA PC Tool started and user selects training 5. User selects the correct training file at the File Dialog 6. training session is loaded. 7. User clicks training button at the main ATA PC Tool application and selects the training session to be compared at the File Dialog. 8. New training session is loaded in a new window
Sub variations	Steps 4 and 5 can be repeated if more sessions are needed for comparison.

Use Case 9	UCPT1:Tournament Compare Single Archer
Description	Used to track an archers progress in training for a tournament.
Actors	archer
Assumption	Tournament files available, application works
Success end condition	Tournament files loaded successfully
Failed end Condition	Files damaged, unable to load training
Steps	9. ATA PC Tool started and user selects tournament 10. User selects the correct tournament file at the File Dialog 11. tournament is loaded and user can load another tournament into the control
Sub variations	

Use Case 10	UCPT2:Tournament Control
Description	Usage of the PC application's tournament tool if used to display all archers total score at a tournament
Actors	Tournament official, or tournament staff in charge of scores
Assumption	Tournament files available, application works
Success end condition	Multiple archers scores loaded
Failed end Condition	Files damaged and unable to load them, unable to display wanted data
Steps	12. ATA PC Tool started and tournament staff selects tournament 13. tournament staff loads one file submitted by the lane staff. 14. tournament is loaded and tournament staff can load another tournament file into the control, files added to the tournament until all lanes loaded into the control. 15. Winner found, tournament can be saved for later use or website publication...
Sub variations	4a Several archers has the same score, tournament officials review scores in detail for the archers in question by clicking on them in the tournament control application.

Use Case diagrams can be found in the appendix

6 Design

6.1 Technology Choice

6.1.1 PDA Definition

PDA's are small digital devices that can be used as an extension of the PC. They can be synchronized with your in box, calender and likewise, but as mentioned they are an extension. They are not meant to be used independently of a PC. They have small processors, limited storage, no actual mouse, no graphics card to handle CPU heavy graphic calculations (some newer PDA's are fitted with a kind of graphics card), limited screen size not only in area but also in resolution and they only have a, on screen keyboard for writing. These facts has heavily affected the choice of technology and development tool. In earlier projects, I have faced the same problem and the best experience has been with Microsoft because their tools offer it all in one package where it all works together. Another possibility would be programing it in Java and running it on a virtual machine on the device; however Java isn't actually available on the device involved in this project, not natively to my knowledge. Third party Java solutions might be available.

With this in mind it is narrowed down to the Microsoft .NET Framework and development tools. Since my experience with these tools has been in the C# language, C# was also the natural choice.

6.1.2 .NET

The .NET Framework is Microsoft's development tool for Windows applications. It gives users the ability to quickly and fairly easy create Windows applications and online solutions. It handles many subjects that could make any programming assignment tedious, such memory management and some hardware specific instructions that could be different from hardware vendor to vendor. All the supported languages are, when compiled, translated into an IL (Intermediate Language) which not only means that the programmer has a wide range of languages to choose from without considering if one language might be a better choice than another for his specific programming job. It also gives great freedom in choosing language and new programmers that are new to .NET can easily find a language that resembles their preferred programing language. The translation into the IL ensures that all applications act the same or functions the same regardless of the language.

6.1.3 Compact Framework

The Compact Framework contains a subset of the .NET Framework. Besides the subset some special items are also available. These special items makes use of features which is only available on hand held devices.

6.1.4 Visual Studio 2005 .NET

Visual Studio 2005 is the latest development tool from Microsoft. It makes use of the .NET Framework and has integrated emulators for more or less any type of compatible device. It supports several languages where C# and VB are the two most commonly used. Its built-in designer is easy to use and allows the user to see the application even before it can run, and changes to the design can easily be applied. It can be used with Microsoft Visio for reverse engineering and documentation. This is only a few features in Visual Studio 2005 but the combination of these features, among others, makes it a very powerful development tool.

6.2 Considerations on programing to the PDA

When writing applications for a PDA, programmers must pay attention to the way applications behave when users push the close button in the top right corner. On a PC, clicking the close button on the top right of the form, will result in the application shutting down and releasing resources.. However, on a PDA, clicking the close button, only closes the form, the application is still running in the background. This means that any information or data entered, is still active and will be shown the next time the application is started. So special attention is needed in handling the closing procedures. Does the user want it to have all the data available the next time he or she powers up the application? Most likely the user are about to start a new independent session, and have no use for old data or at least, the user don't want to write new data on to the active session. To avoid this all data must be flushed or deleted when exiting the application. It behaves like this because PDA's are small digital devices and their purpose is to be available when users push the “on” button. When the “on” button is pushed on the PDA it is because the user want the calender now! Or the email now! And because they lack CPU power it is feasible to have programs running in the background after they are closed, instead of closing it completely and then spent time on loading the program next it is activated. The user will probably access that program again, and access the same data again. However the intended use of ATA is not to access old training sessions. Archers probably wont need old sessions if anything, they want to keep the sessions apart.

Besides the behavior of programs on the PDA, attention to the user interface is needed. A PDA is not much more than a screen, and this screen is nothing like the screen on the desktop or laptop PC. PDA screens are rarely bigger than 4” and has a typical resolution on 240 x 320. there is no a mouse and there is no keyboard by default. All inputs go through the touch screen by using the stylus, which can be used as a mouse. The stylus is also used when interacting with the on-screen keyboard. The keyboard is accessed by clicking the small keyboard icon in the bottom of the screen when it is available. A small on-screen keyboard pops up and it is used by using the stylus to tap/click the appropriate letters. It doesn't calls for high efficiency but it works fine for small notes and likewise but it shouldn't be considered as a primary input option.

This is why parts of the application is designed to accept values without using the keyboard, all values can be entered with out using the keyboard. Another reason why you have to consider if the keyboard is needed: when activating the keyboard, almost half the screen is hidden behind the it, so if it is needed, make sure you don't place anything you might use where the keyboard pops up.

6.3 Design of the GUI for the Applications

PDA

Several parts of the application is designed to accept values without using the keyboard, all values can be entered without using the keyboard (except names for the archers). The reason for this is first of all because it is annoying to activate and tab your way trough the needed data but also because, when activating the keyboard, almost half the screen is hidden behind the keyboard. If it turns out that it is needed, steps has been taken to avoid that anything wont be placed where the keyboard pops up.

Illustration 1: Start Screen

Illustration 1 shows the Start Screen, this screen is kept very simple, two buttons takes the user either to the training session or to the tournament. Instead of two buttons, two separate applications could have been made, but for keeping it together as one project a single entry point was decided. The exit button in the bottom part exits the application completely. It doesn't close the application the way applications normally are closed on PDA's where they remain active in the memory. This button closes the application and releases all resources. Because this is the decided behavior, the default close button in the top right corner has been removed.

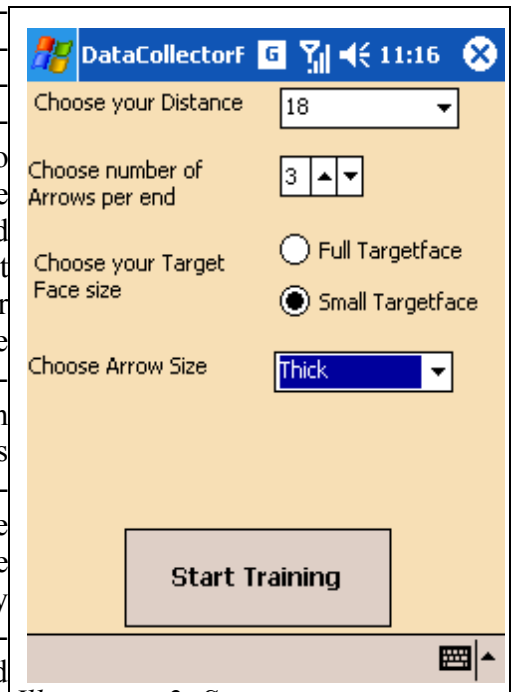


Illustration 2: Setup screen

Illustration 2 shows the setup screen for training, the top drop down list allows the user to select some predefined distances, these distances are the most typical distances used, first solution was a text box where the archer could enter any distance, but testing of the GUI showed that it was annoying to activate the keyboard for entering a distance. This is also the case with the number of arrows per end, so a numericUpDown proved to be a better solution.

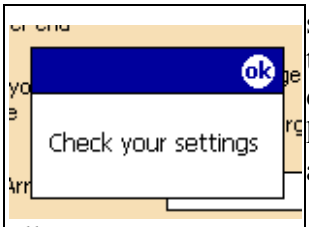


Illustration 3: Error in settings

If the archer fails to enter one or several setting items a simple message box (illustration 3) pops up notifying the archer to check the settings. The program will not proceed before everything is entered correct

The first (Illustration 4 and 5) tab page is used to plot the arrows. The ring in the top left corner shows the arrow size and the archer can drag the ring to the exact location of the arrow.

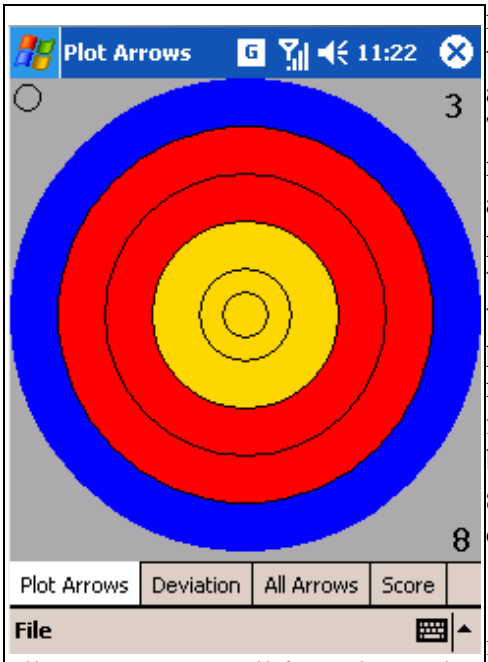


Illustration 4: Small face aka triple face

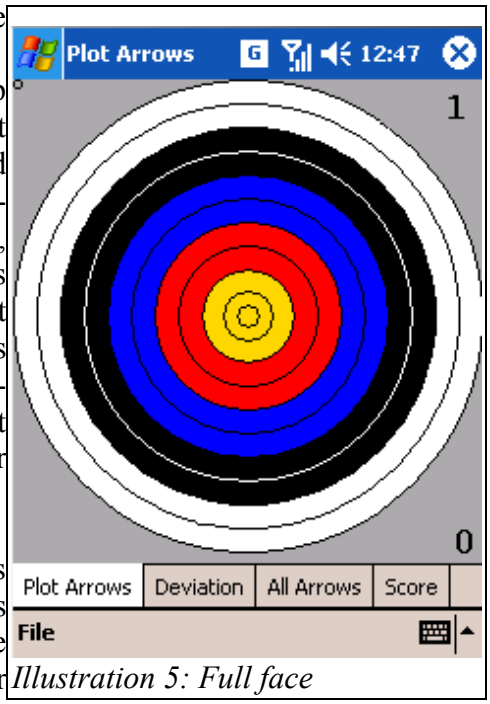


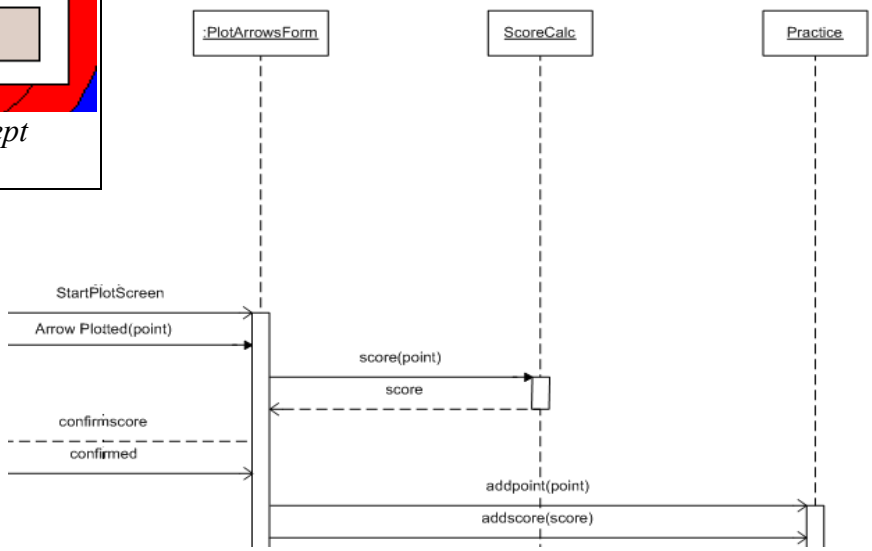
Illustration 5: Full face

The number in the top right corner shows what arrow the archer should plot, and is changed after each approved plot, when number of arrows per end is reached it rolls over and starts from 1 again. The number at the bottom right shows the total number of arrows plotted.

When the stylus is lifted, the coordinates are recorded and the distance from the center is calculated. The distance is used to determine the score.

When a score is calculated a message box (illustration 6) is used to confirm that the arrow was plotted correctly. If the the archer accepts the score, meaning the arrow was correctly plotted, the score and the coordinates are saved. If result isn't accepted, score and coordinates are released and archers can re plot the arrow.

Illustration 6: Accept score Messagebox



Section of SSD for plotting a score. Full SSD and Class Diagrams can be found in the appendix

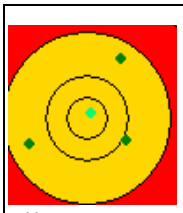


Illustration 7: Deviation

The "Deviation" tab page looks like the front tab-page, but it doesn't allow any input from the archer, instead it shows the arrows plotted this end with dark green dots and the mean deviation of those as a light green dot. This page doesn't allow user inputs and it looks like the front tab page

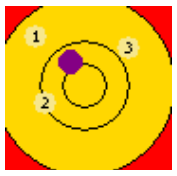


Illustration 8: All Arrows

The "All Arrows" tab page shows how each arrow is behaving individually in total the purple dot show the total overall deviation. This page doesn't allow user inputs either and also looks like the front tab page.

A small menu in the bottom of the screen allows the archer to either save the results or flush all the results.

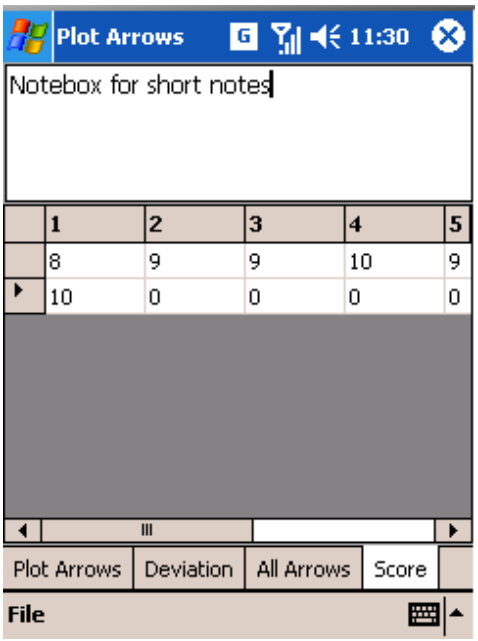


Illustration 9: Score Tab

The last tab page "Score" shows which scores has been recorded during the session, the data grid acting as a score-card, add rows when needed. The top part is a text box for small notes if needed. It is placed at the top because when the on screen keyboard is activated, it fills up the bottom part of the screen and hides whatever is behind it.

The screen in illustration 10 appears when archers wants to save the training session, it is the standard save file dialog box provided by Visual Studio.

The save file dialog for tournament is exactly the same, except that the file type is a tournament file *.tnm.

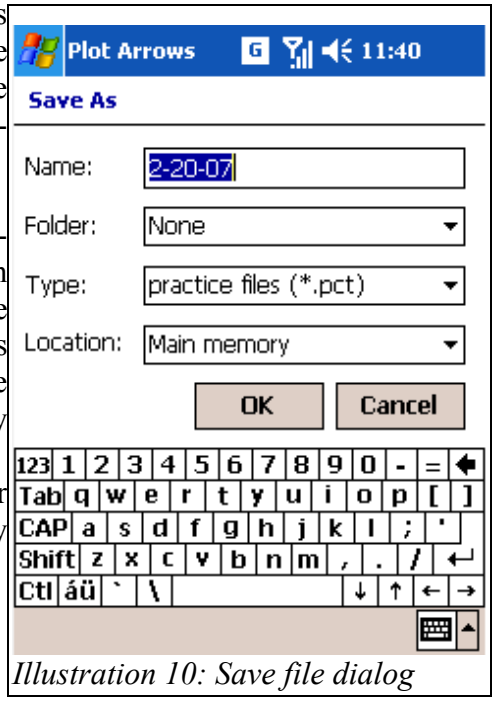


Illustration 10: Save file dialog

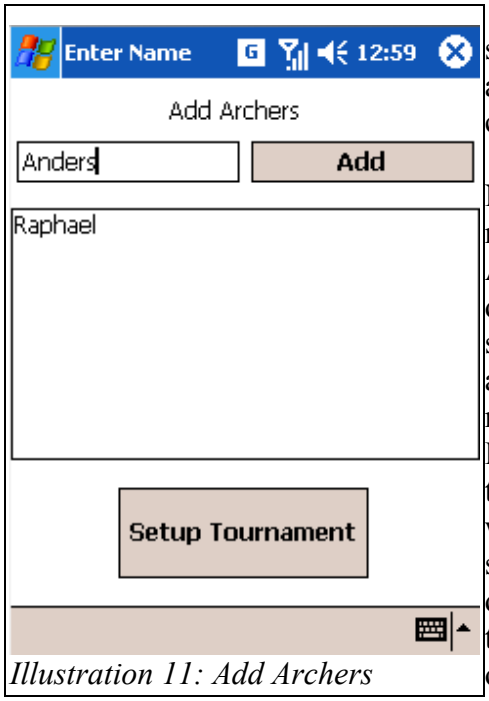


Illustration 11: Add Archers

If users chooses the tournament button at the startup screen, he/she is prompted with a screen where archers can be added (illustration 11) to the tournament. The application accepts single archers.

Next screen is the tournament setup screen. All distances are FITA distances. The setup shown here is known as a FITA outdoor round.

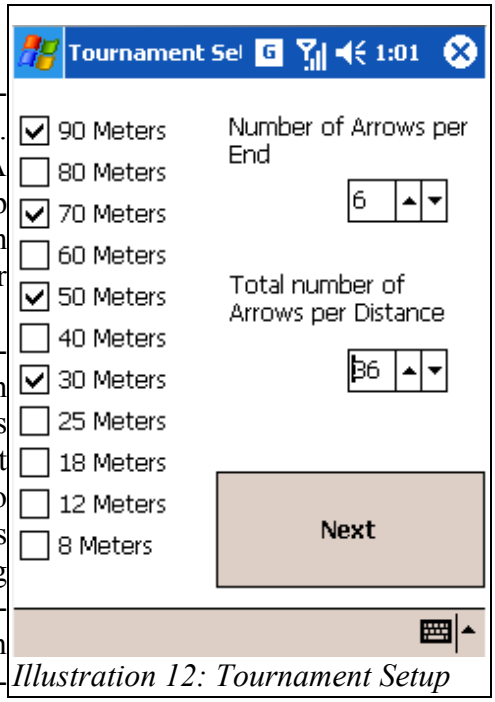


Illustration 12: Tournament Setup

Because archery contest are derived from warfare all tournaments starts with the longest distance and moving to the shorter distances one by one, resembling the enemy getting closer and closer, therefore are distances by default saved with the longest first and shortest last.

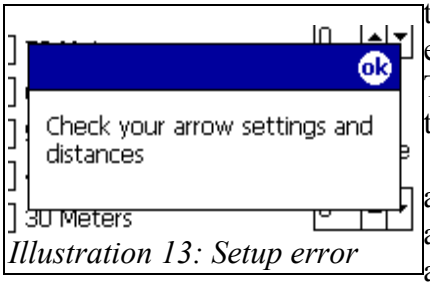


Illustration 13: Setup error

The next button will load the tournament with the archers and distances entered.

If the user fails to enter any distances or leaves out to enter anything in the arrow boxes a message box will pop up and further progress is denied until problem is solved.

Illustration 14 shows the screen where the tournament scores are registered, the two gray boxes in the top shows the the archer and distance in focus, any scores entered are added to this archers scorecard. Bellow these is the scorecard, represented by a datagrid, for the archer and distance in focus. When another archer or distance is selected, text boxes on top are updated and scores loaded accordingly.

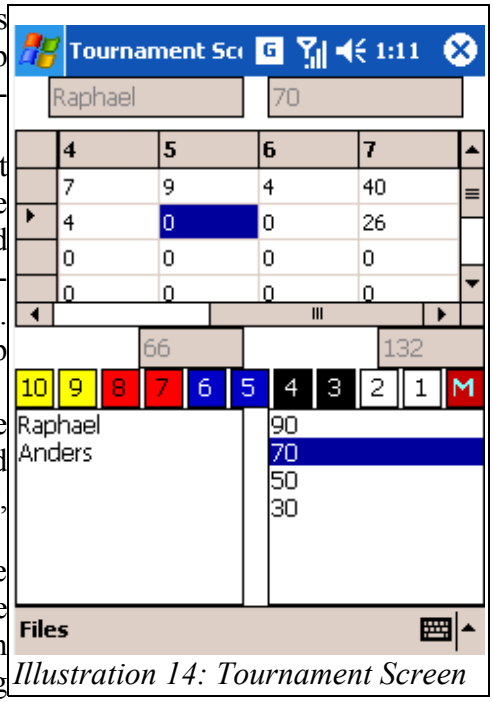


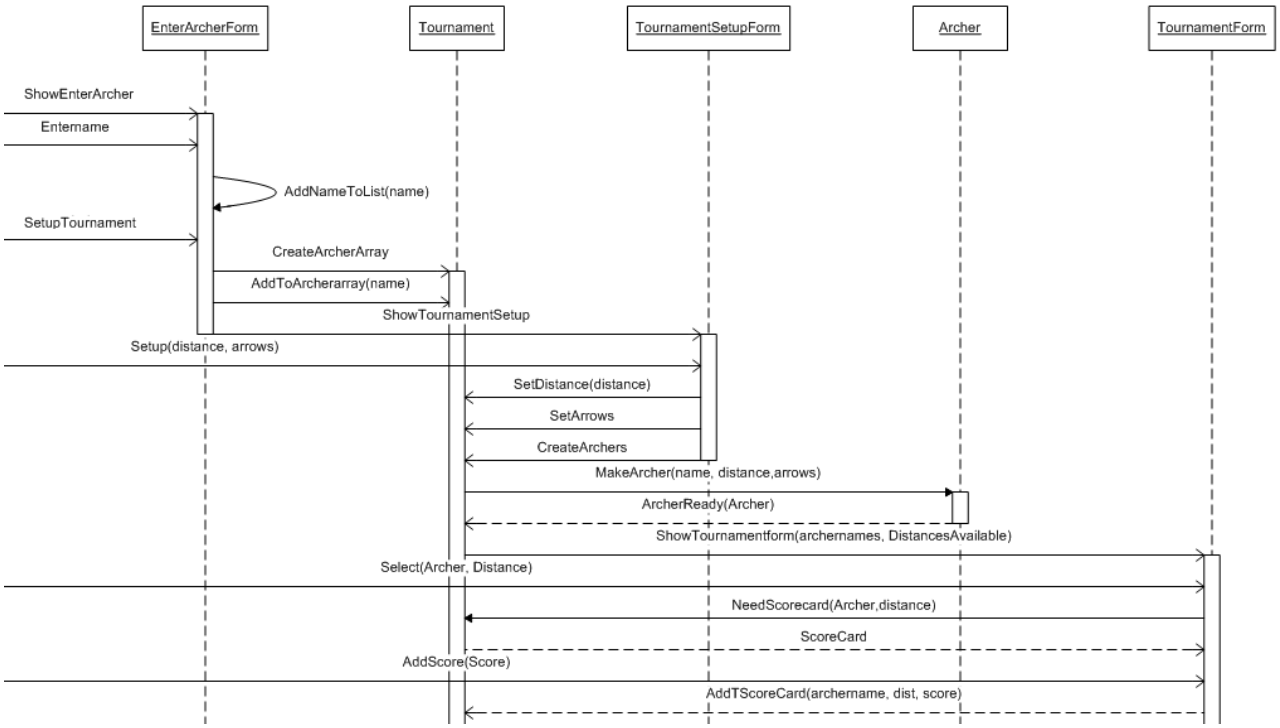
Illustration 14: Tournament Screen

Below the scorecard another two text box are shown, the one to the left shows the total score for the distance in focus and the box to the right shows the total for the tournament so far, for the archer in focus.

Below these text boxes, a line of buttons are placed. These buttons will each add the score shown on them to the score card. Each colored accordingly to the color the score holds on the target face. In the bottom are two list boxes one containing the archers entered in the tournament and one containing the distances selected for the tournament. These are used to switch the focus from one to another simply by tapping the archer or distance

wanted. Finally a menu in the bottom gives the user two options. One: save the tournament, and two: exit the tournament.

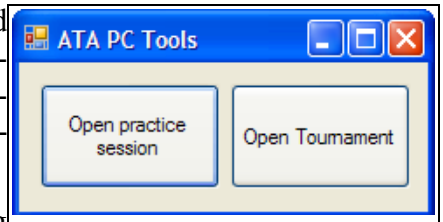
All screen dumps is taken from the emulator; the application executed on PDA has a slightly different color due to the screen. The screen on tested PDA's seems to have a little warmth in them.



SSD section for creating Tournament with archers and distances and adding scores. Full SSD and Class Diagrams can be found in the appendix

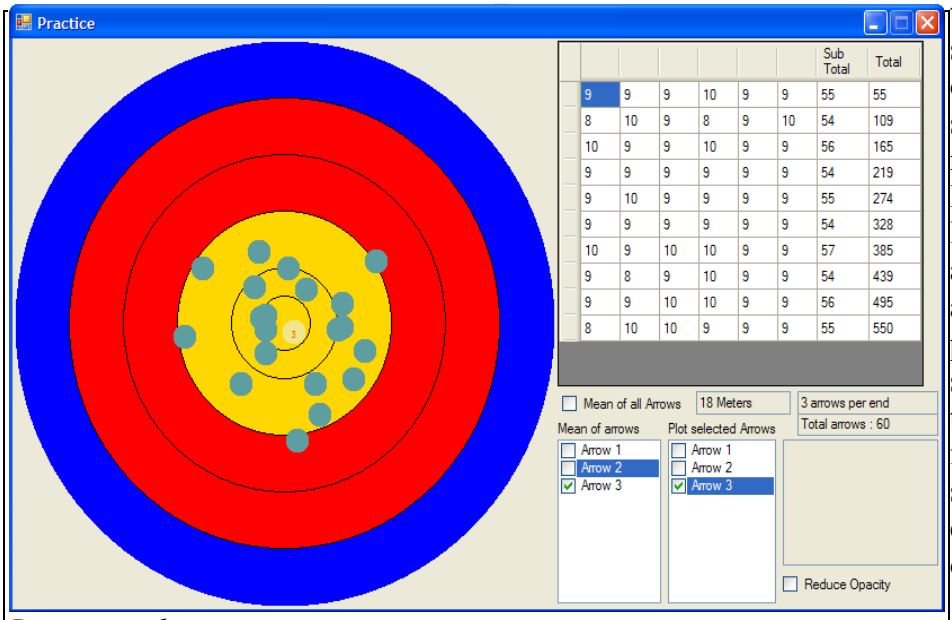
PC- Training

Like the PDA the PC application could have been divided into two separate projects, but again a single entry point was chosen. It is quite simple it consist of only two buttons, one will activate the practice review session, and the other opens the tournament control board.



The PC tool main form.

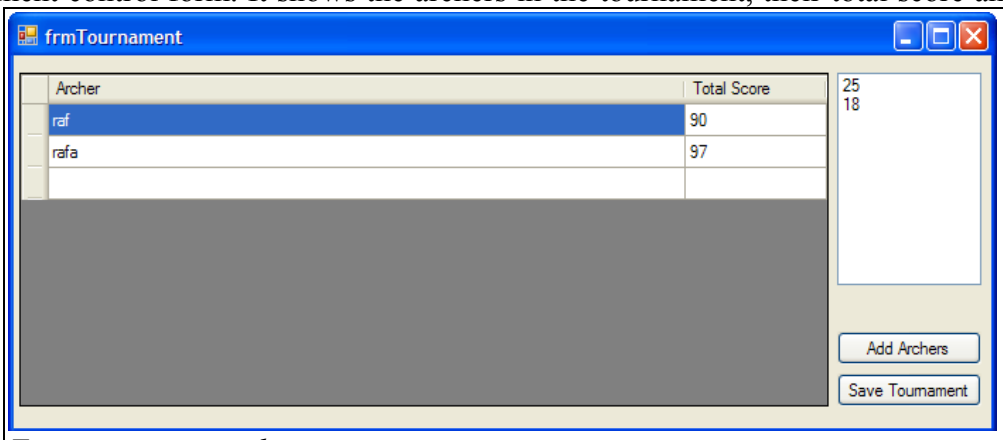
When selecting the practice button, an open file dialog opens up, and user is prompted to select a saved file of the session to be reviewed. When file selected the review form opens. The review form shows the user the scores recorded during training. Different check boxes allows the user to turn on and off different arrows and deviations. It is also possible to open another session and compare them. The opacity of the form can be reduced so several sessions can be compared on top of each other.



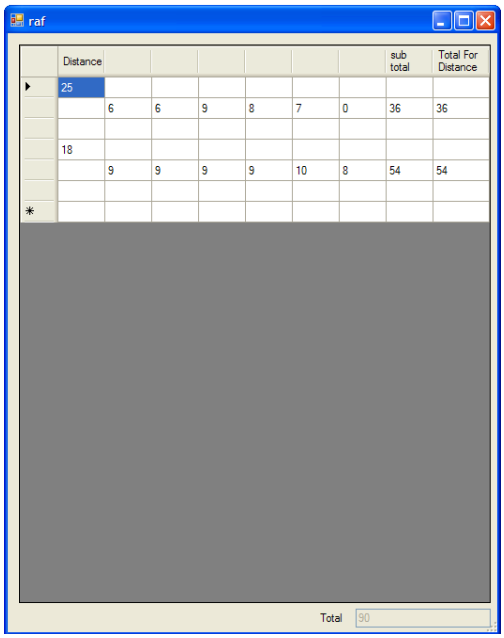
Review application.

PC- Tournament Control

When clicking the Tournament button in the PC Tool main form an open file dialog opens up, and user is prompted to select a saved tournament. The tournament is loaded in to the tournament control form. It shows the archers in the tournament, their total score and what distances was used in the tournament. If wanted, another tournament file can be added if it has the same set-up. The newly combined tournament can the be saved.



Tournament control



Detailed scorecard

When clicking on an archer on the control form a detailed scorecard pops up showing every score at every distance, end totals and distance totals

6.4 File structure

The applications makes use of some custom made files for the saving and loading of both training and tournament. Limitations in the Compact Framework forced me to manually convert complex data types into much simpler data types and manually convert them back again to their original types. Every array or array list and object needed to be converted. The result is files that only contains integers and strings.

The training files:

Data	**Type**	**Remarks**
Number of arrows per end	integer	
Total arrows shot	integer	
Distance	integer	
Arrow size	string	Converted from enum
Scores	string	Converted from int array
Points	string	Converted from point array
Notes	string	
Choice	integer	

The tournament files:

Data	**Type**	**Remarks**	**Repeat**
Number of arrows per end	integer		
Number of arrows per distance	integer		
Number of distances	integer		
Distances	string	Converted from int array	
Number of archers	integer		
Archers name	string		Yes, repeated for every archer
Score for distance	string	Converted from int array	Yes, repeated for every distance per archer

7 Test

Testing the code ensures stability of the application. In the early stages tests has been conducted by the use of a TDD framework, when GUI was implemented the built-in PDA emulator in Visual Studio 2005 was used. Earlier experience has shown that it isn't safe to trust the emulator, it is very important to test it on a real PDA. This is especially important when testing the users interaction with the User Interface. The emulator and the real device might look alike in their GUI but differences in their user interaction occurs as well as in their performance.

Finally the ultimate test is when the application is used in an actual environment. Many experiences has been made at the shooting range that led to a change in the application.

7.1 TDD

Test Driven Development is a powerful development method, where code is tested without running the program. This means that modules, algorithms and functions can be tested independently of the executable program. This method was used with great success in the beginning of the project. The key elements in TDD is write a test (testing what you want of course) and implement the code that will provide passing test. When the test passes the code is done and you move on to next task.

When enough function was implanted to use a GUI, I started using a more manual way of testing, still following the basic TDD processes, I implemented a form where I wrote my tests and displayed the result and validating the result myself. Not quite as fancy as the TDD Framework but just as effective and very important in the testing of the GUI.

When the GUI was advanced enough for actual live test a series of test was performed at the shooting range.

7.2 At the Range

Several test has been carried out at the range in real conditions. As a result of these real life test, a few design changes has been been made. The application works so well so I have recorded every shot I made since the end of January.

The application has been tested as it was developed because progress in the applications development was dependent on well tested functionality before proceeding.

I have managed to find bended arrows with the application, the only thing that hasn't been tested properly is the sight tuning. This is mainly because the sight tuning really only apply when used at long distances, but it is indoor season at the moment. It has been tested at the short range and it is assumed that long ranges wont affect the result or that the use will need to be changed.

The tournament mode has been tested with both single and several archers. And it functions well when used correctly.

7.3 At Home – The ATA PC Tool

The PC Tool has been tested with the data collected from the real conditions tests. Much of

the PC application can be found in the PDA, several classes contains many of the same functions. Therefore has the testing of the PC not been as extreme as it has with the PDA.

7.4 Known bugs

Following bugs has been discovered after code stop

When starting a tournament without entering any names the application crashes. Possible solution is simply a try-catch clause.

When choosing the full face in training the ring representing the arrow is black, this needs to be changed since ring 3 and 4 on the full face are black and the archer cant see the arrow ring. Solution would be to change the color of the ring to gray , light blue or any other color that isn't on the target face.

The application needs some explaining labels from time to time.

8 Conclusion

8.1 Summary

The goal through out the project was to make an application that an archers easily could bring to the shooting range and use as part of tuning equipment and save various sessions and data with out using the small pieces of paper (scorecards) thats normally used. The PDA application fulfills this goal. Archers can save a higher level of data than by just saving the numbers on scorecards, and in detail investigate each training session. The PDA application requires a minimum of input through the keyboard without limiting the usability. The tournament part gives the archer the possibility to train actual tournament conditions and save those for later comparison, or it can even be used as a part of a clubs tournament equipment, again removing the paper-scorecards with hand written scores.

The PC makes it easy for the archer to compare training sessions over time or different setups without guessing or assuming. The complete data set is available for analyzing.

The tournament control gives a quick overview and accepts either values from several PDA's or files saved from different dates for comparison.

The smallest device this application has been fully tested on is the Ipaq h1940. Which is a windows CE 4.20 PDA, it has a Samsung S3C2410 processor running at 266 MHz. The processor is an ARM based processor like the Intel X-scale processor, and since it makes use of the Compact Framework it should run any PDA with running Windows CE 4.20. The screen holds the standard specifications: screen resolution is 240 x 320 pixels with 64K color (65,536 colors, 16-bit).

None of the application are designed in a way that requires special knowledge in their usage and there is no need for high performance specialized hardware.

8.2 Future improvements

There are plenty of ideas for future improvements, one of the most obvious would be the adding of recurve bow scores and some of the other disciplines found in archery such as “Field” for example. Archers that have reviewed the applications also has a lot of ideas, these however tend to address very personal problems. But some interesting ideas came up during a discussing.

At tournaments, archer has to fire their arrows within a certain time frame and special rules apply in this timing sequences. Typical a single beep tells the archer to approach the shotting line followed by two beeps, telling the archer to shoot the arrows, another two beeps signal a cease fire. This is repeated for each group of archers and ended by a long beep signaling archers to collect arrows and note the scores. If a timer was implemented archers could train with time as a constraint like they would encounter at tournaments.

Another interesting improvement involves some hardware from one of the big bow equipment manufactures (Easton). They have made a set of devices, which can be used to map a bows performance in numbers and measure the speed of the arrows when shot through a chronograph. An extension to the applications where archers records arrows hit tendency with PDA together with the

speed measurements for an arrow in flight with the chronograph would strengthen the analyzing possibilities that already lies in this application.

8.3 Short Comings in the application as it is

The device where the application has been tested contains the standard hardware for this version of Pocket PC operating system, and is also programmed to those exact specifications. But some PDA's with the same operating system are fitted with a higher screen resolution. If the application is installed on these PDA's the Compact Framework will ensure that it runs on the device, but the application doesn't make use of the better screen. A suggested extension in the implementation is to let the application take full advantage of the better screen by letting the software be aware of the hardware available, and have all graphics drawn correctly instead of letting the Compact Framework stretch the graphics to the screen size .

Another shortcoming in the project is the missing installation wizard. Installation for the PC is no problem, but creating an installation wizard for the PDA is a long and complicated process that involves editing of .ini files by hand and searching through the registry. But it would be considered a demand if the project should be released to the public.

Appendix

Appendix A :Terms in archery

Anchor-point	The place where the hand pulling the arrow is “anchored” every time for a consistent shot. Typical near the ear or jaw.
Compound Bow	The kind of bow this application is designed for. It comes in many shapes and sizes. In general a compound consists of a string, a cable and pulleys fitted on each limb. These items makes the bow just as hard to pull as a recurve or longbow but holding weight at full draw is much lower, often 35% or less of the pull weight.
DBSF	Danish Bow Shooters Association, the danish version of FITA.
FITA	FITA is the international archery governing organization/federation (Federation Internationale de Tir a l'Arc)
Fps	feet per second, archers quick conversion (if you want it in kph) multiply it with 1.1, eg. 250 fps * 1.1 = 275 kph (real conversion yields 274.32 kph).
Full face	the original target face, consisting of 10 circles with score values from 1 to 10
limbs	the flexible part of the bow
Longbow	this the most primitive kind of bow, it is nothing more than a stick with a string
Recurve bow	a bow derived from the longbow, as the name says, it is has curved limbs on each side of the handle
Small face	See triple face
Target face	The target. (where the pointy end of the arrow goes) <- lame
Triple face	a target face which has 3 separate target faces on but each one contains only the inner 5 circles from the original target face. These target faces are usually used by compound shooters when shooting indoor, because compounds bows are high precision bows and by having one face per arrow, the chance of damaging you arrows is removed
Vane	The tail or back end of an arrow are fitted with feathers or plastic pieces, which has the same function as a rudder on an airplane, control the flight of the arrow.

Appendix B : Implementation (short description of Classes and Methods (PDA)

The function of the PDA is to assist the archer at the shooting range. In its training mode it collects data of not only the score but also where the actual arrow hits the target face. This is valuable data for the top performing archer, it provides a quick overview of the deviation and makes easy to make the needed adjustments to bow and it can also be used to find bended arrows. The tournament settings allow you to setup any kind of tournament with one or several archers shooting at one or several distances. In the tournament mode it will only collect scores.

When the application is powered up the first screen is the startup screen. This is a very simple class it sole purpose is to present the user with the two options training or tournament.

Below are a short description of each class in the PDA application.

Archer.cs

This class is used to represent an archer in a tournament. It is highly flexible since it accepts any kind of tournament, from full FITA round too single distance tournaments. From a logic point of view an archer in a tournament has a name and one or several distances, and each distance has a scorecard.

```
public Archer()
```

This the default constructor of the class

```
public int returnRow(int dist)
```

This returns the current row in the scorecard for the specified distance. It is used by the tournament screen and its function is to keep track of the archers “current cell” in the data grid representing the score card.

```
public int returnColumn(int dist)
```

Same as above but returns the column

```
public void setRow(int dist, int newvalue)
```

When a score is entered, the next cell is in focus. This updates the current row for the specified distance.

```
public void setColumn(int dist, int newvalue)
```

Same as above but updates the column. This method and the 3 above ensures that the data grid always shows the correct current cell even if the user choose another archer.

```
public void SetNameAndSize(string name, int dist, int TArrows)
```

When this is called, the archer is setup with a name and a scorecard for each distances.

```
public void AddScore(int dist, int arrow, int score)
```

Adds a score to the specified distances.

```
public int calcScore(int dist)
```

Returns the total score for the specified distance.

```
public int[] scorearray(int indexdist)
```

Returns the complete array of scores for the specified distance

```
public void addToScore(int index, int score)
```

Adds the total score for one distance to an array of totals.

```
public int returntotalscore()
```

Returns the grand total for the entire tournament

Calc.cs

This class is used by the training mode to perform calculations on Drawing.Points.

```
public static Point(Point[] al)
```

Returns the average point of the point in the array.

```
public static Point[] totalIndividualDeviation(Point[] total, int[] acc)
```

Returns an array of the average or deviation of each arrow in the shooting sequence (the average of all the first arrows, the average of all the second arrows and so forth)

DataCollector.cs

The code in this file has slowly moved out of the class or has been re factored away. It only exist because of a TDD test. It should have been removed long time ago, but it somehow managed to survive.

```
public int MumberofArrows(Arraylist al)
```

Returns the number of arrows shoot

DataCollectorForTraining.cs

This form collects various information from the archer, such as distance, arrows fired each end, target face selection and arrow size.

```
public frmTrainingInput()
```

This is the constructor of the class, it initializes the various components of the form, such as buttons, text boxes, and other controls.

```
private void btnStartTraining_Click(object sender, EventArgs e)
```

Triggered by start button, it validates the information entered by the user, before proceeding to the actual training. Some of the information is used in calculations, some are used in the graphic representations of the

arrows while others are purely used as information to the archer in later reviews. If any fields are left blank a message box will prompt the user and ask the user to check the settings. When everything is entered it proceeds to the next screen and calls `public void kill()`

```
public void kill()
```

This method closes this form.

EnterName.cs

This form is the first screen the user is presented when choosing the tournament mode. It collects the competing archers.

```
public EnterName()
```

Initializes the components of the form.

```
public void btnAdd_Click(object sender, Eventargs e)
```

Adds the entered archers name a list box.

```
public void btnSetupscreen_Click(object sender, Eventargs e)
```

Adds all archers from the list box to a string array before showing the setup screen for tournaments.

```
public void kill()
```

Closes the form.

FileIO.cs

This class handles conversions for the routines that saves either tournament or training. Actual file I/O doesn't take place here.

```
public static string convertPoints()
```

Converts the array of points collected in training, to a string

```
public static string convertScore()
```

Converts the score array to a string

```
public static string distTostring()
```

Converts the array of distances to a string

```
public static string Sorefordist(string archer, int index)
```

Converts the score array for the specified archer at the specified distance to a string.

FullFace.cs

This is how the full face is represented in numbers and coordinates

```
public override void targetface()
```

Overrides the method in the targetfaces.cs class

```
public override Graphics targetface(Graphics pea)
```

Draws the circle which the full face consist of.

```
public override Graphics targetfaceTotal(Graphics pea)
```

Draws the deviation of the arrows on the target face.

```
public override Graphics targetface(Graphics pea, Array ar)
```

Draws the arrow hits and deviation for the last finished end

FullFaceScore.cs

Handles the calculation of the score based on the point of the arrow.

```
public int score(Point p)
```

Returns the score, found by calculating the distance from the center of the target face to the plotted point of the arrow.

JustTest.cs

Class used in very early iterations of the program to test various behaviors of the PDA, or acted as a shortcut for testing plot Arrows. This class should be removed before releasing the software.

```
public justTest()
```

Initializes the few components on the form

```
public JustTest_Load(object sender, EventArgs e)
```

Empty method, earlier used to test graphics

```
public JustTest_MouseU(object sender, MouseEventArgs e)
```

Early method to test “mouse” behavior

```
private void button1_Click(object sender, EventArgs e)
```

Used to test one type of target face

```
Private void button2_Click(object sender, EventArgs e)
```

Used to test another type of target face

PlotArrows.cs

The plotArrows form is the interesting part of the training mode. This form is divided into 4 tab-pages, each one used for one kind of specific information, the first page is the only page used actively by the archer, the following 2 pages provide graphic information on the deviation and the last page shows a scorecard and a text box for short notes if needed.

```
public PlotArrows(int ch)
```

Initializes components

```
private void tbpPlotArrows_Paint(object sender, PaintEventArgs e)
```

Draws the the selected target face to the tab-page and as well as a ring representing the selected arrow size.

```
private void tbpDeviation_Paint(object sender, PaintEventArgs e)
```

Draws the selected target face, a dot showing each arrow from the latest end and a dot showing the deviation for this ends arrows.

```
private void tbpAllArrows_Paint(object sender, PaintEventArgs e)
```

This tab page shows the total of all arrows and the total of the sequence of arrows.

```
Private void PlotArrows_Load(object sender, EventArgs e)
```

Unused, content re factored away.

```
private void tbpPlotArrows_MouseUp(object sender, MouseEventArgs e)
```

Collects the point where the user plots the arrow, if score is accepted score is added to various array lists

```
private void tbpPlotArrows_MouseMove(object sender, MouseEventArgs e)
```

Moves the circle representing the arrow when the user moves the stylus across the screen. This helps the user to plot the arrows more precisely than just tabbing it in with the stylus.

```
private void tbpPlotArrows_MouseDown(object sender, MouseEventArgs e)
```

Redraws the screen in order to remove the arrow circle.

```
private void tbpScorecard_Paint(object sender, PaintEventArgs e)
```

Shows the scorecard tab-page with scores, totals and a notes field.

```
private void menuItem2_Click(object sender, EventArgs e)
```

Duplicate method

```
private void menuItem1_Click(object sender, EventArgs e)
```

Saves the current training session using the binary writer.

```
private void mniReset_Click(object sender, EventArgs e)
```

Resets all values back to null before closing the form.

```
private void PlotArrows_closing(object sender, CancelEventArgs e)
```

Unused event method removed release

Practice.cs

This class holds the two array lists that contains the points of the arrows entered and the score entered when using the training mode.

```
Public static void AddArrowScoreP(Point[] arrows)
```

Adds the points of the arrows from the latest end to the array list holding these values.

```
Public static void AddArrowScore(int[] score)
```

Adds the scores of the arrows from the latest end to the array list holding these values.

```
Public static Arraylist printScore()
```

Returns the array list with scores

```
Public static Arraylist printPoints()
```

Returns the array list containing the points.

```
Public static void ResetScores()
```

Clears the array lists

Program.cs

The file containing the main method. Its got nothing in it besides the Main()

```
static void Main()
```

Main entry point for the application, shows the startup form.

Score.cs

Interface class

```
Public interface Score
```

Interface for the classes containing the actual score calculations, target faces can have different ways to calculate the scores.

ScoreCard.cs

This form has been re factored away, removed before released

```
public ScoreCard()
```

Initializes the components on the form.

Setup.cs

This class collects the data from the training setup form and from the training progress as well. It the data is accessed through get/set methods.

```
public static void resetSettings()
```

Resets all values back to zero.

SmallFace.cs

This is how the small target face (triple face) is represented in numbers and coordinates. It contains the same methods as fullFace.cs, only difference is the graphics of course.

```
public override void targetface()
```

Overrides the method in the targetfaces.cs class

```
public override Graphics targetface(Graphics pea)
```

Draws the circle which the full face consist of.

```
public override Graphics targetfaceTotal(Graphics pea)
```

Draws the deviation of the arrows on the target face.

```
public override Graphics targetface(Graphics pea, Array ar)
```

Draws the arrow hits and deviation for the last finished end

SmallFaceScore.cs

Handles the calculation of the score based on the point of the arrow, the same as full-FaceScore.cs

```
public int score(Point p)
```

Returns the score, found by calculating the distance from the center of the target face to the plotted point of the arrow.

Startup.cs

The startup form which allows the user to select between training or tournament. It is also the last form shown before exiting.

```
public Startup()
```

Initializes the few components on the form.

```
public static void btnTraining_Click(object sender, EventArgs e)
```

Starts the training session

```
public static void shutdown()
```

Unused method – removed before release.

```
public void sd()
```

Unused method – removed before release.

```
private void btnTournament_Click(object sender, EventArgs e)
```

Starts a tournament session

```
Private void Startup_Closing(object sender, CancelEventArgs e)
```

Unused method – removed before release.

```
private void button1_Click(object sender, EventArgs e)
```

Exits the application and releasing all resources

TargetFaces.cs

Abstract class, slowly re factored into a strategy-like pattern

```
public abstract class TargetFaces
```

TDDPractice.cs

This class is a part of the TDD tests used by the nUnit framework, none of the TDD classes are used by the application. TDDPractice.cs is used to test some basic functions in the training mode

```
public void addScorePointsTest()
```

Used to test the storing of points from arrows.

```
public void ArraylistMemPoints(
```

Testing the array list of points.

```
public void addScoreTest()
```

Used to test the storing of each individual score.

```
public void ArraylistMem()
```

Making sure the scores are available

TDDTest.cs

Early test class, many tests has been re factored away or scope changed making the tests un-usable or obsolete.

```
public void ArrowsPerEnd()
```

Testing arrows per end.

```
public void ArrowAverage()
```

Empty method, but used to test the average calculation.

```
public void getsetArrows()
```

Testing some of the get and set methods. The arrows per end in this case.

```
public void getsetDistances()
```

Empty test of the get/set of the distance

```
public void getsetArrowSize()
```

Testing get/set of arrow size

```
public void getsetTotalArrows()
```

Empty method, tested the total arrows shot.

TDDTournament.cs

This class is testing various parts of the tournament part.

```
public void Enternames()
```

Testing the method adding archers

```
public void retrievenames()
```

Testing the retrieve name function. This test is dependent on the Enter-name() method above, test not really needed since this has already been tested in the method above.

```
public void enterdist()
```

Tests if it is possible to enter distances and retrieving them.

```
public void TAarray()
```

Testing the creation of an archer array.

Tournament.cs

The tournament class holds all the information concerning a tournament, distances, number of arrows, competing archers and for so on. In an earlier version it also contained scores, but this was moved to the archer.cs, because although scores are achieved through competing in an tournament, the score is unique for each archer. It besides the get/set methods it contains the following methods

```
public Tournament()
```

The default constructor of the class

```
public static void setArcherlength(int length)
```

Creates two arrays, a string array holding the names of the entered archers and an archer array. Its only created at here, no values entered to any of the arrays.

```
public static void setDistanceLength(int length)
```

Creates an array for holding the distances.

```
public static void addArchers(string str, int index)
```

Adding archers to the archer to the specified position in the array.

`public static string Archer(int archer)`

Return the archer at the specified position.

`public static void addDistances(int dist, int index)`

Adding distances to the distance-array

`public static int returndist(int index)`

Returns the distance at the specified index from the distance array.

`public static void createArcherArray()`

Doesn't really creates the array, it actually adds archer objects to the array of archers and setting up scorecards for each distance as well.

`public static int returnarcherArrayLength()`

As the name says, it returns the length of the archer-array.

`public static void addscore(string name, int dist, int arrow, int score)`

Adds the score for the specified archers at the specified distance at the specified place in at the scorecard.

`public static int[] fetchArray(string name, int dist)`

Returns an array of scores for the specified archer at the specified distance.

`public static int[] SfetchArray(string name, int dist)`

The same as above, the difference lies in the distance, the `int dist` in this case already represents the index of the distances. Should be re factored into something more compact since both methods do exactly the same.

`public static void reset()`

Flushes all values stored in arrays

`public static int[] CCforArcher(string name, int dist)`

Returns the current cell in the scorecard, for the specified archer at the specified distance.

`public static void setCCforArcher(string name, int dist, int row, int column)`

Sets the current cell for the specified archer at the specified distance.

`public static void addtoscorearray(string name, int dist, int score)`

Adds the total score for the specified distance to the archers score array of totals.

`public static int getTotal(string name)`

Gets the overall total score for the specified archer.

TournamentScoreCard.cs

This form is the active form in the tournament mode. It is through this form the user enters scores for each archer.

```
public TournamentScoreCard()
```

Initializes the components on the form and setting up the data grid.

```
private void TournamentScoreCard_Load(object sender, EventArgs e)
```

Triggered by the load event, fills the list boxes with competing archers and distances.

```
private void lbxArchers_SelectedIndexChanged(object sender, EventArgs e)
```

Switch to another archers scorecard.

```
private void lbxDistances_SelectedIndexChanged(object sender, EventArgs e)
```

Switch to the scorecard for the selected distance.

```
private void setValue(int score)
```

Adds the score entered for the archer in focus at the distance in focus, to the archers scorecard.

```
private void UpdateDataGrid( bool valueset )
```

Updates the data grid and the current cell for the archers at the distance in focus.

```
private void dtgTournament_CurrentCellChanged(object sender, EventArgs e)
```

Changes focus to the cell selected in the data grid, mainly used if an entered score needs to be changed.

```
private void btn10_Click(object sender, EventArgs e)
```

Adds the score represented by the button to the archer in focus' scorecard at the distance in focus. Score = 10.

```
private void btn9_Click(object sender, EventArgs e)
```

See above, score = 9.

```
private void btn8_Click(object sender, EventArgs e)
```

See above, score = 8.

```
private void btn7_Click(object sender, EventArgs e)
```

See above, score = 7.

```
private void btn6_Click(object sender, EventArgs e)
```

See above, score = 6.

```
private void btn5_Click(object sender, EventArgs e)
```

See above, score = 5.

```
private void btn4_Click(object sender, EventArgs e)
```

See above, score = 4.

```
private void btn3_Click(object sender, EventArgs e)
```

See above, score = 3.

```
private void btn2_Click(object sender, EventArgs e)
```

See above, score = 2.

```
private void btn1_Click(object sender, EventArgs e)
```

See above, score = 1.

```
private void btnMiss_Click(object sender, EventArgs e)
```

See above, score = 0, archer failed to hit the target face.

```
private void mniSaveTournament_Click(object sender, EventArgs e)
```

Saves everything from the tournament to a binary file.

```
private void mniResetTournament_Click(object sender, EventArgs e)
```

Clears the values in the tournament and closes the tournament form.

TournamentSetup.cs

The form where the tournament is set up with distances and number of arrows

```
public TournamentSetup()
```

Initializes the components on the form.

```
private void btnStartTournament_Click(object sender, EventArgs e)
```

If everything needed to start a tournament is entered this form is closed and the tournament is started. If not user is prompted with a message box and asked to check the settings

```
private bool collectdistances()
```

Collects the chosen distances, if no distances selected a bool set to false is returned invoking the message box mentioned above. Selected distances are added to an array of distances.

```
private bool arrowsSettings()
```

Check the arrow settings. If settings doesn't comply, bool set to false is returned invoking the message box mentioned above.

```
public void kill()
```

Closes this form.

Appendix C : Implementation (short description of Classes and Methods (PC)

The PC application isn't meant to be used actively in the same way as the PDA is used. Since you likely won't have the PC with you at the shooting range theres no need to be able to plot arrows when using the training. And it is not likely that the staff at a tournament would like to drag a PC to each and every target face when using the tournament mode. Instead it is used to get an overview of the saved tournament.

Below are a short description of each class in the PC application.

ArchersInTournament.cs

This class is very the much like the archer.cs class in the PDA application, many of the methods in the class are the same.

```
public ArchersInTournament()
```

Default constructor of the class

```
public void SetNameAndSize(string name, int dist, int TArrows)
```

Has the same function as the method of the same name in archer.cs

```
public int[] scorearray(int indexdist)
```

Returns the score array for the specified distance.

FileConverter.cs

This class contains methods used when converting strings from the saved training and tournament files back to arrays, as well as saving tournaments.

```
public static int[] makescores(string scorestring, int ta)
```

Converts a string of scores to an array containing the scores used in training.

```
public static Point[] makePoints(string point, int ta)
```

Converts a string of points to an array containing the points, is used to plot the arrows.

```
public static int[] setDistances(string distances)
```

Converts the string of distances to an array used in tournaments.

```
public static int[] scoresfromfile(string scores)
```

Converts a string of scores to an array containing the scores used in tournaments.

```
public static string distTostring()
```

Converts an array of distances to a string, used when saving a tournament.

```
public static string Scorefordist(string archer, int index)
```

Converts the array of scores for a specified archer at the specified distance to a string.

frmAtaPcTools.cs

Very simple form which guides the user to either tournament control or training review

```
public frmAtaPcTools()
```

Initializes the components on the form

```
private void btnOpenTraining_Click(object sender, EventArgs e)
```

Opens an open file dialog with a preselected file extension specific for training.

```
private void ofdTrainning_FileOk(object sender, CancelEventArgs e)
```

Triggered by the “ok” button on the file dialog, opens the training review form with data from the selected file.

```
private void btnOpenTournament_Click(object sender, EventArgs e)
```

Opens an open file dialog with a preselected file extension specific for tournament.

```
private void ofdTournament_FileOk(object sender, CancelEventArgs e)
```

Triggered by the “ok” button on the file dialog, opens the tournament control form with data from the selected file.

frmScoreCard.cs

A form showing detailed scoring for an archer in a tournament.

```
public frmScoreCard(string name)
```

Initializes the components and sets the text on the form to the selected archers name.

```
private void frmScoreCard_Load(object sender, EventArgs e)
```

Fills the data grid view representing the complete scorecard with the scores for the archer.

frmTournament.cs

The tournament control form. Shows the distances in the tournament, archers and their total score from the selected tournament file.

```
public frmTournament()
```

Initializes the components on the form.

```
private void btnSave_Click(object sender, EventArgs e)
```

Saves the tournament.

```
private void frmTournament_Load(object sender, EventArgs e)
```

Fills the data grid view on the form and the distance list box.

```
private void dgvOverview_CellContentClick(object sender, DataGridViewCellEventArgs e)
```

Opens a detailed scorecard form.

```
private void btnAddArchers_Click(object sender, EventArgs e)
```

Opens a open file dialog

```
private void ofdAddArchers_FileOk(object sender, CancelEventArgs e)
```

When a file is selected, archers and their score form the file will be added to the tournament.

Practice.cs

This the review form for the saved training files, scores, arrows, various deviation calculations can be reviewed here. Several practice forms can be loaded with different training sessions for comparison.

```
public frmPractice(int index)
```

Initializes the components on the form

```
private void populatedgv(int Index)
```

Fills the data grid view with scores.

```
private void pnlTarget_Paint(object sender, PaintEventArgs e)
```

Paints the target face to the panel.

```
private void clbIndividualArrows_Click(object sender, EventArgs e)
```

Sets selected arrows for plotting on the target face.

```
public Graphics targetface(Graphics pea, int choice)
```

Draws the target face and plots the selected arrows.

```
private void clbIndividualArrows_SelectedIndexChanged(object sender, EventArgs e)
```

Adds or removes individual arrows to the target face

```
private void clbMeanOfArrows_SelectedIndexChanged(object sender, EventArgs e)
```

Adds or removes the mean of the individual arrows

```
private void cbxTotalDeviation_CheckedChanged(object sender, EventArgs e)
```

Adds or removes the dot indicating the total deviation

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
```

Reduces the forms opacity, so the user can compare two training ses-

sions by placing them on top of each other.

Program.cs

The file containing the main method. Its got nothing in it besides the Main()

```
static void Main()
```

Main entry point for the application.

ptcSetup.cs

This class is used when loading a saved training session, it is tied to a form with the loaded values, and really isn't worth describing in detail. All values from the binary writer is stored in here and read by the form it is tied to.

TargetFaces.cs

Class holding the target faces graphic routines.

```
private static Graphics Fullface(Graphics pea)
```

Draws the full face to the graphics object.

```
private static Graphics SmallFace(Graphics pea)
```

Draws the small face to the graphics object.

```
public static Graphics targetface(Graphics pea, int choice)
```

Method used to determine which kind of targetface is used for the selected training.

TournamentClass.cs

This class is a smaller version the Tournament.cs in the PDA application, many of the methods are the same.

```
public static void setArcherlength(int length)
```

Creates two arrays, a string array holding the names of the entered archers and an archer array. Its only created at here, no values entered to any of the arrays.

```
public static void addArchers(string str, int index)
```

Adding archers to the archer to the specified position in the array.

```
public static string Archer(int archer)
```

Return the archer at the specified position.

```
public static int returndist(int index)
```

Returns the distance at the specified index from the distance array.

```
public static void createArcherArray(int index)
```

Doesn't really creates the array, it actually adds archer objects to the array of archers and setting up scorecards for each distance as well.

```
public static int getTotalScore(int index)
```

Gets the total score for the archer at the specified index.

```
public static int[] SfetchArray(string name, int dist)
```

Returns an array of scores for the specified archer at the specified distance.

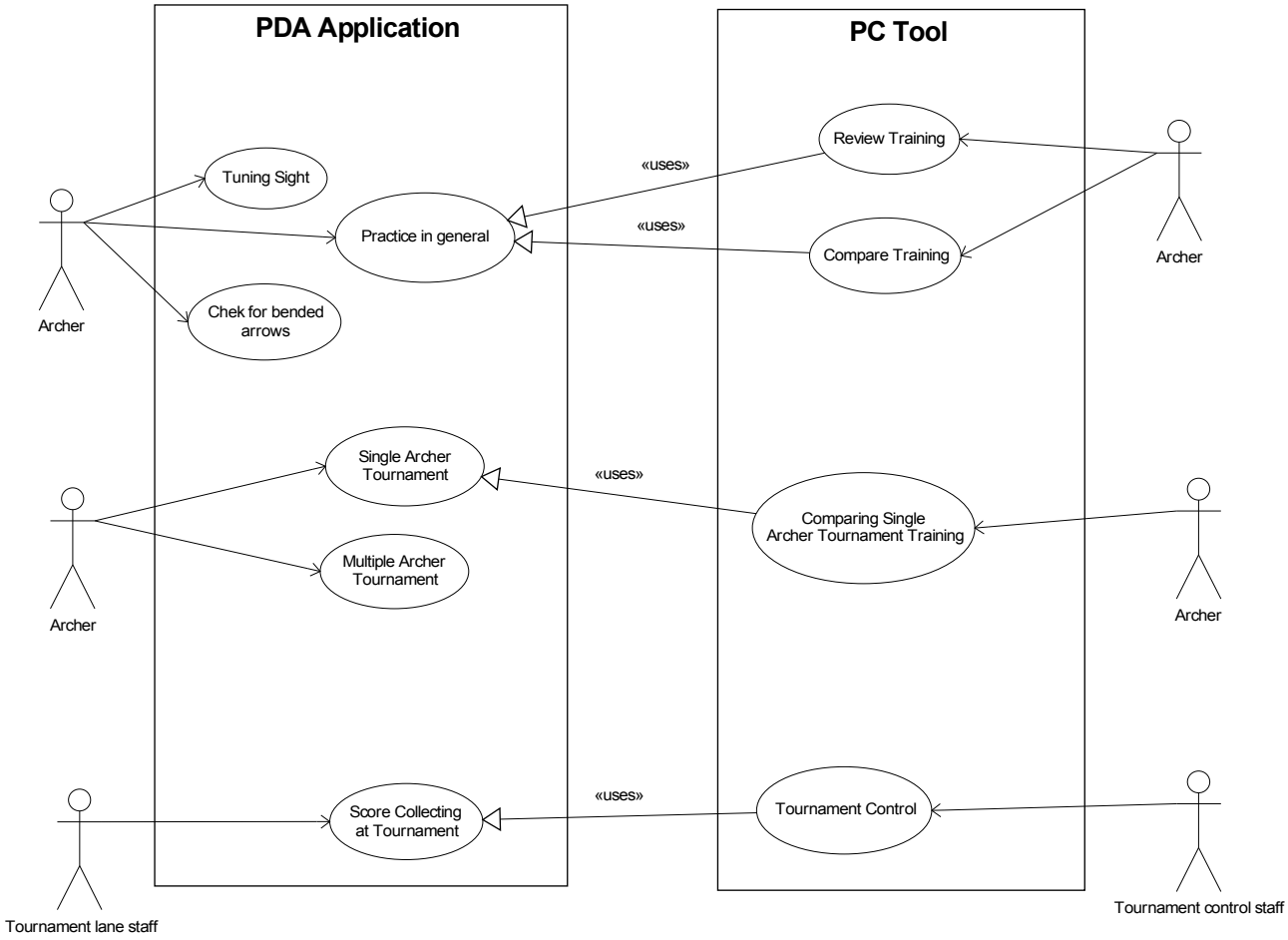
```
public static void addscoreArray(int archer, int[,] convertedScores)
```

Adds the converted score array to the specified archer.

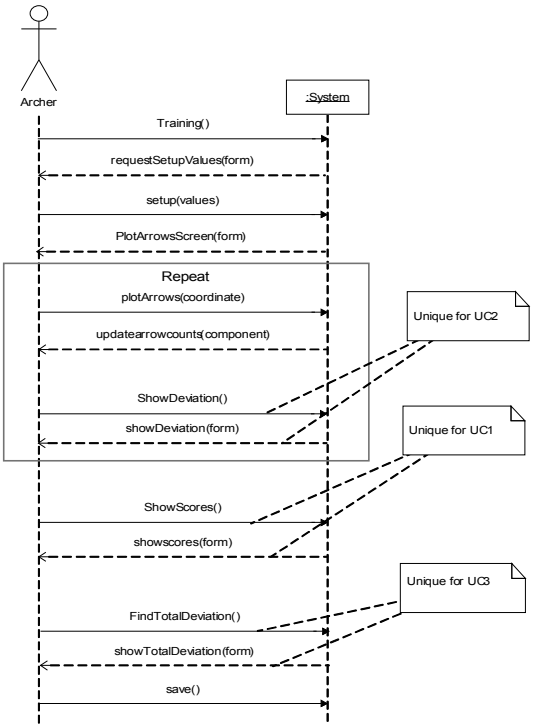
```
public static void resizeTournament(int newlength)
```

Resizes the arrays of archers.

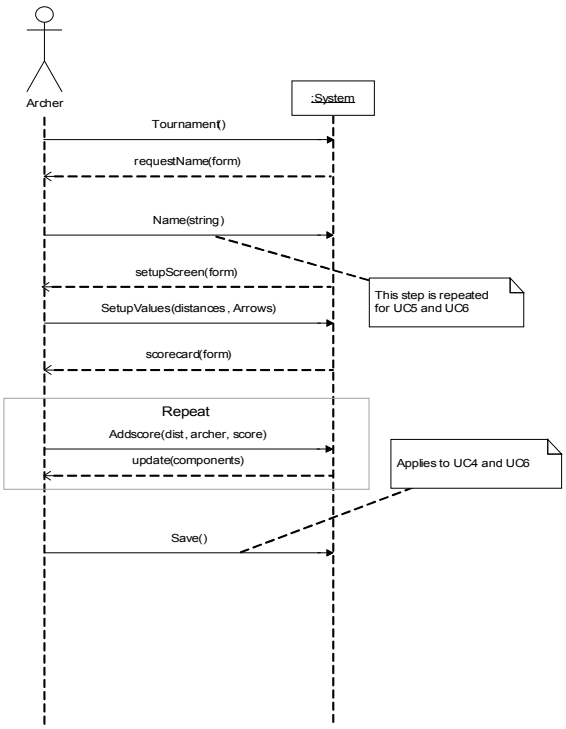
Appendix D:Use Case Diagrams



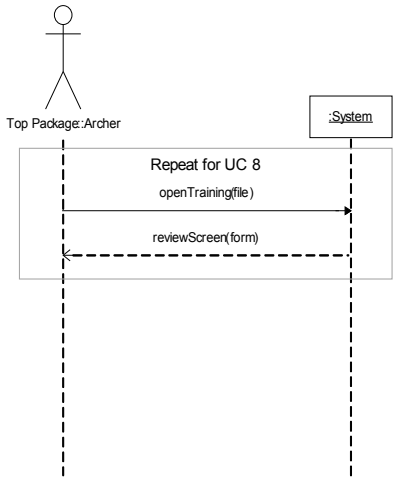
UC1 (UC2, UC3)



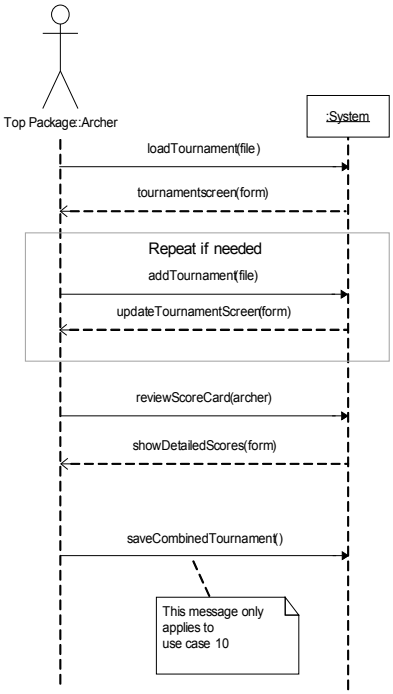
UC4 (UC5, UC6)



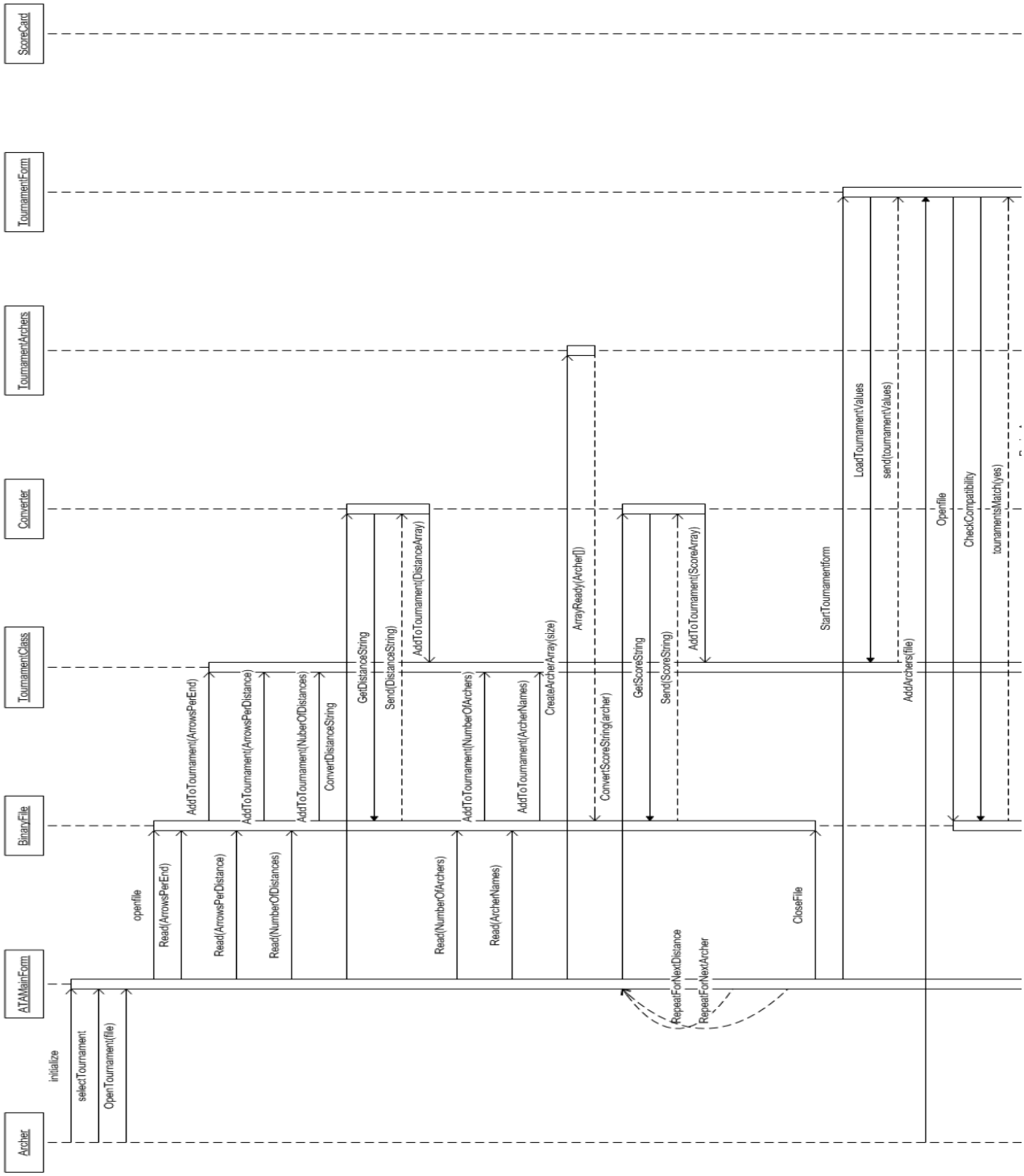
UC7 (UC8)



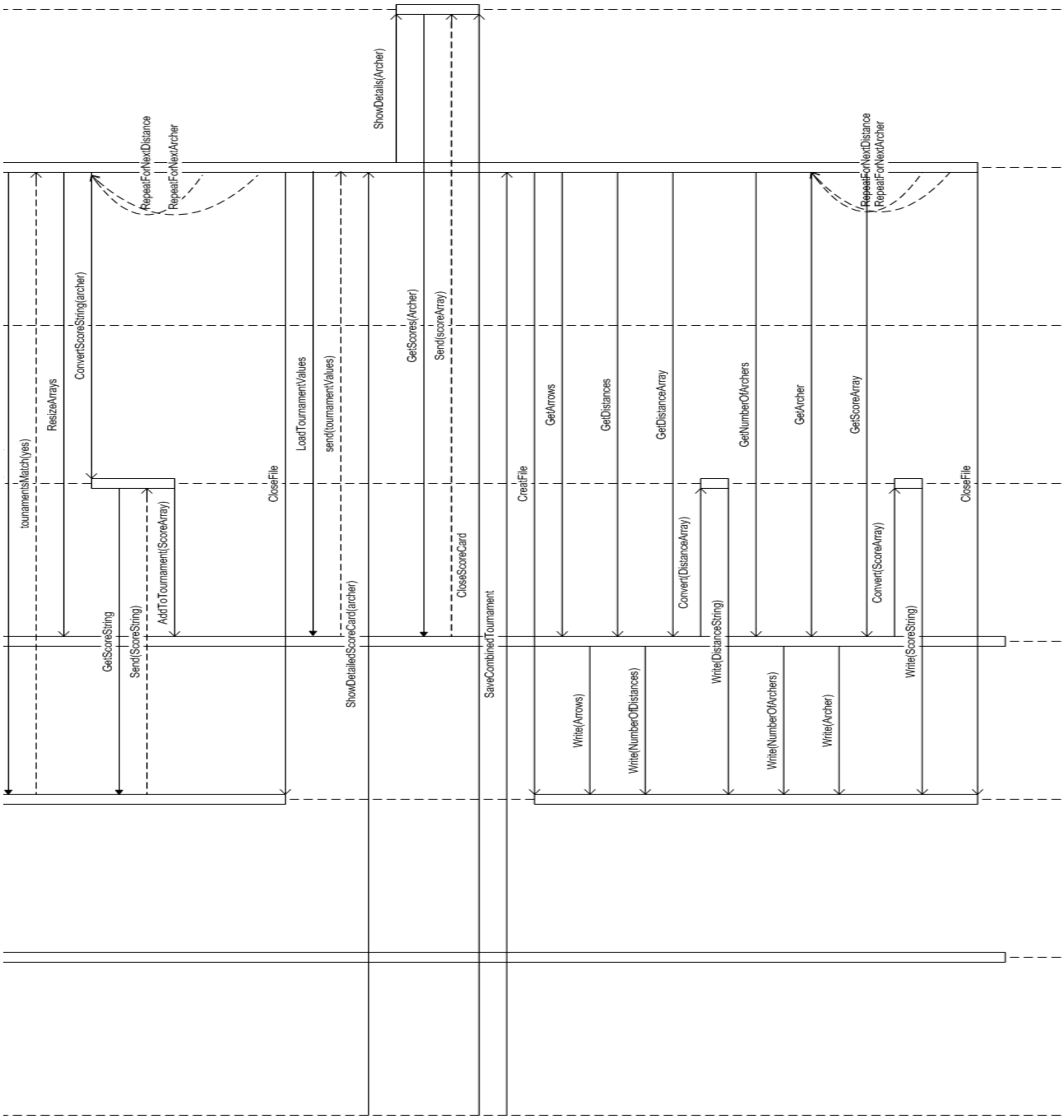
UC9 (UC10)

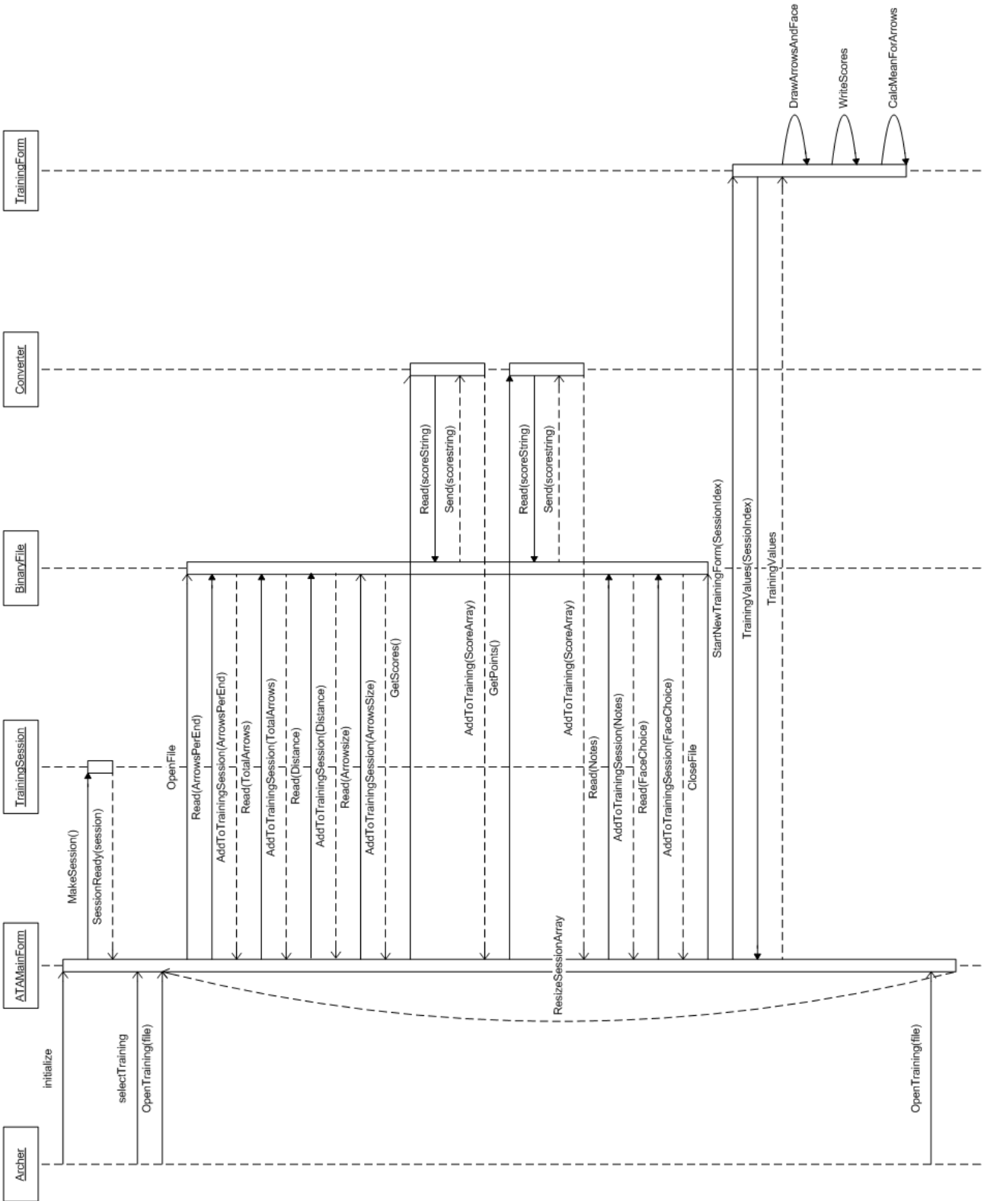


Appendix E: Sequence Diagrams

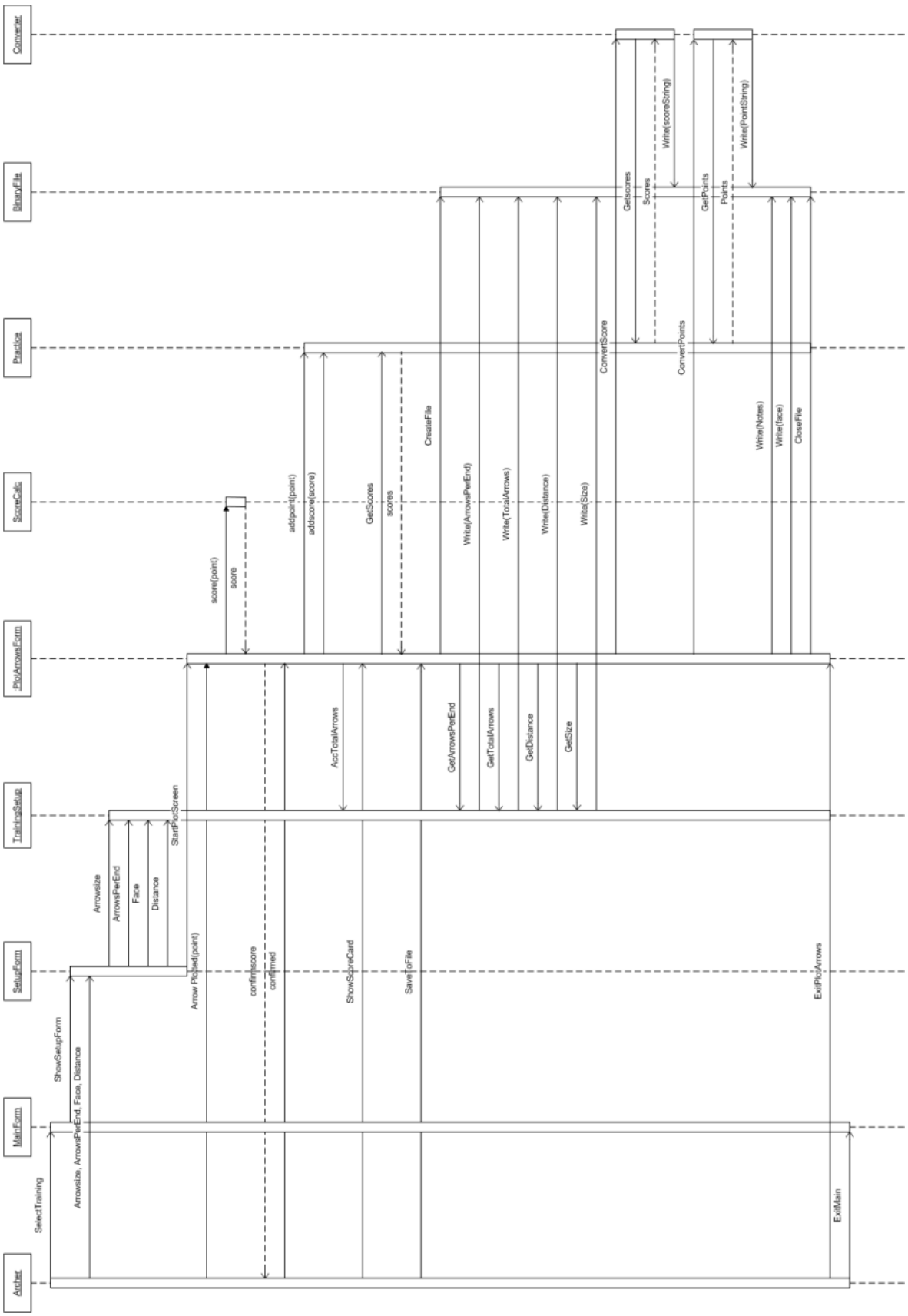


PC Tournament Control -continued on next page

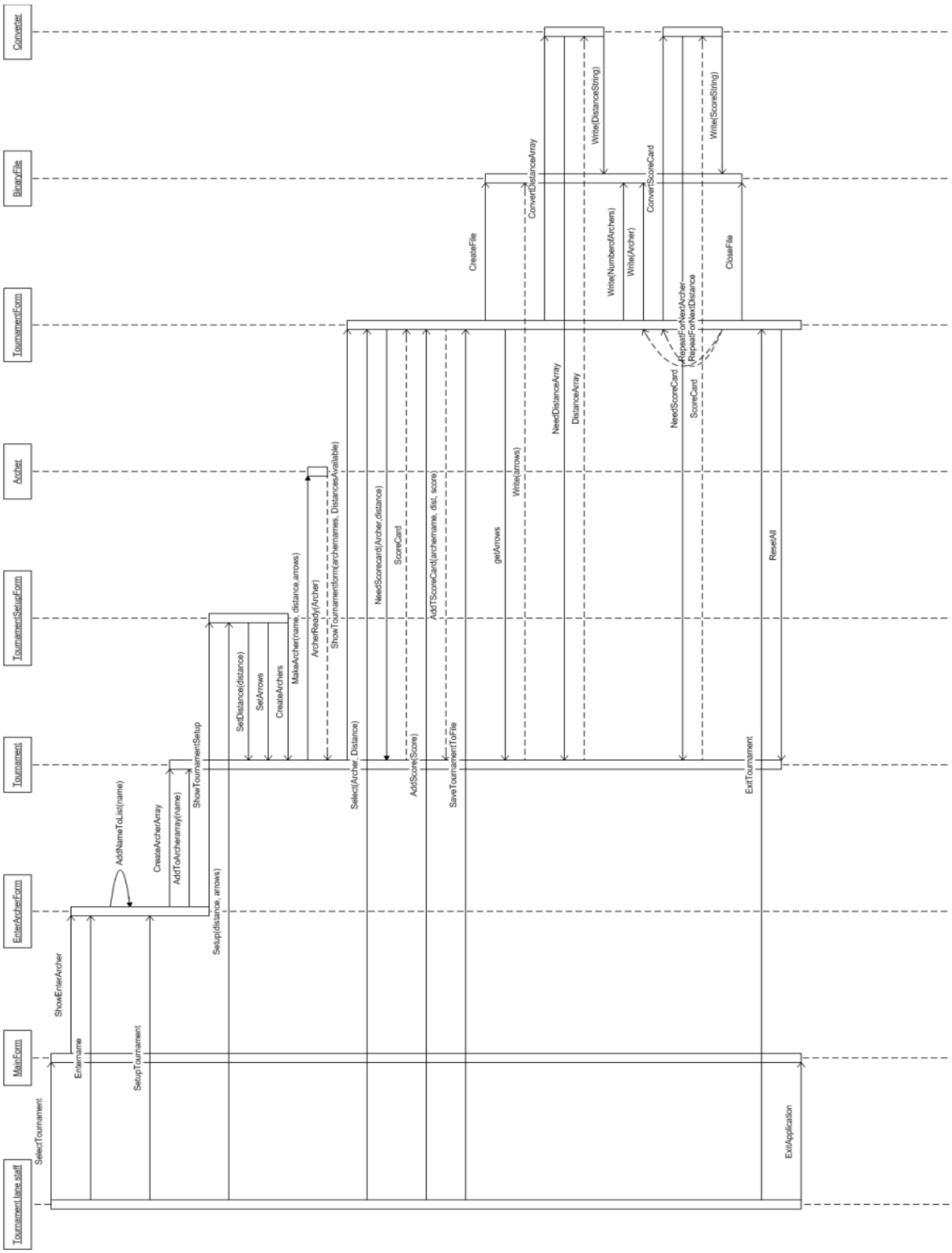

ShowDetails(Archer)
RepeatForNextDistance
RepeatForNextArcher
RepeatForNextDistance
RepeatForNextArcher
tounamentsMatch(yes)
ResizeArrays
ConvertScoreString(archer)
GetScoreString
Send(ScoreString)
AddToTournament(ScoreArray)
CloseFile
LoadTournamentValues
send(tournamentValues)
ShowDetailedScoreCard(archer)
GetScores(Archer)
Send(scoreArray)
CloseScoreCard
SaveCombinedTournament
CreatFile
GetArrows
GetDistances
GetDistanceArray
GetNumberOfArchers
GetArcher
GetScoreArray
Convert(DistanceArray)
Convert(ScoreArray)
CloseFile
Write(Arrows)
Write(NumberOfDistances)
Write(DistanceString)
Write(NumberOfArchers)
Write(Archer)
Write(ScoreString)



PC Training Session Review

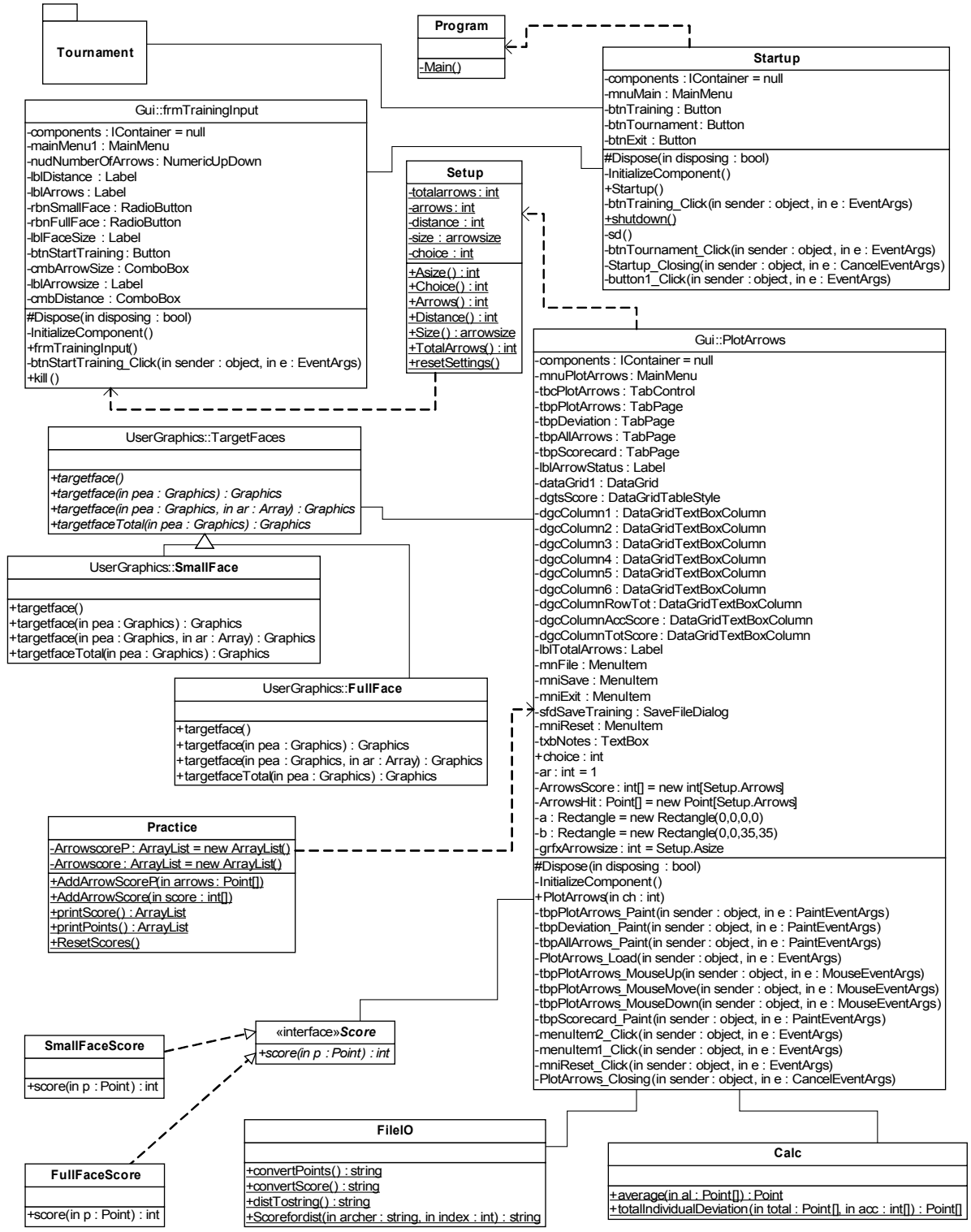


PDA training Session

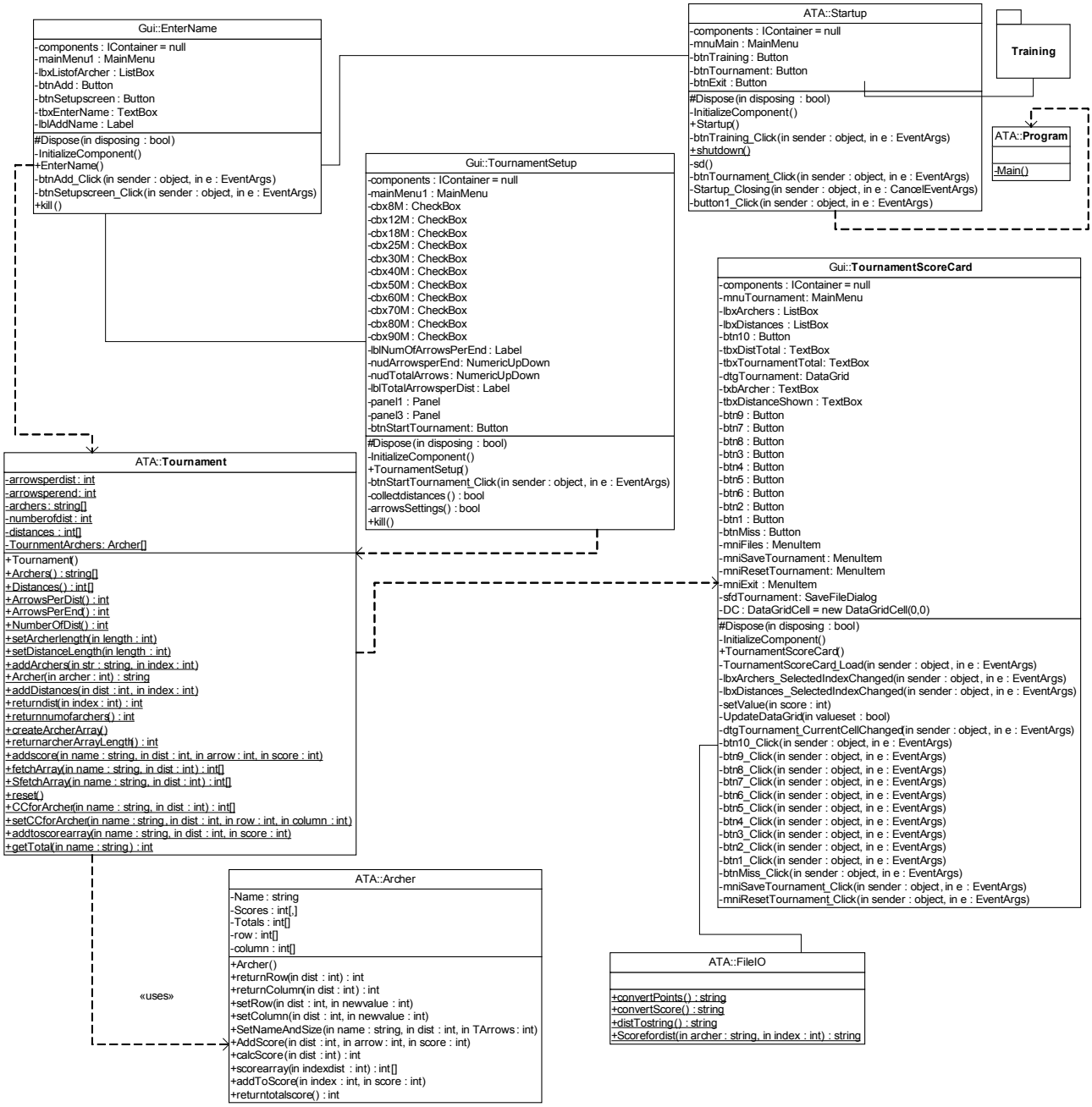


PDA Tournament Session

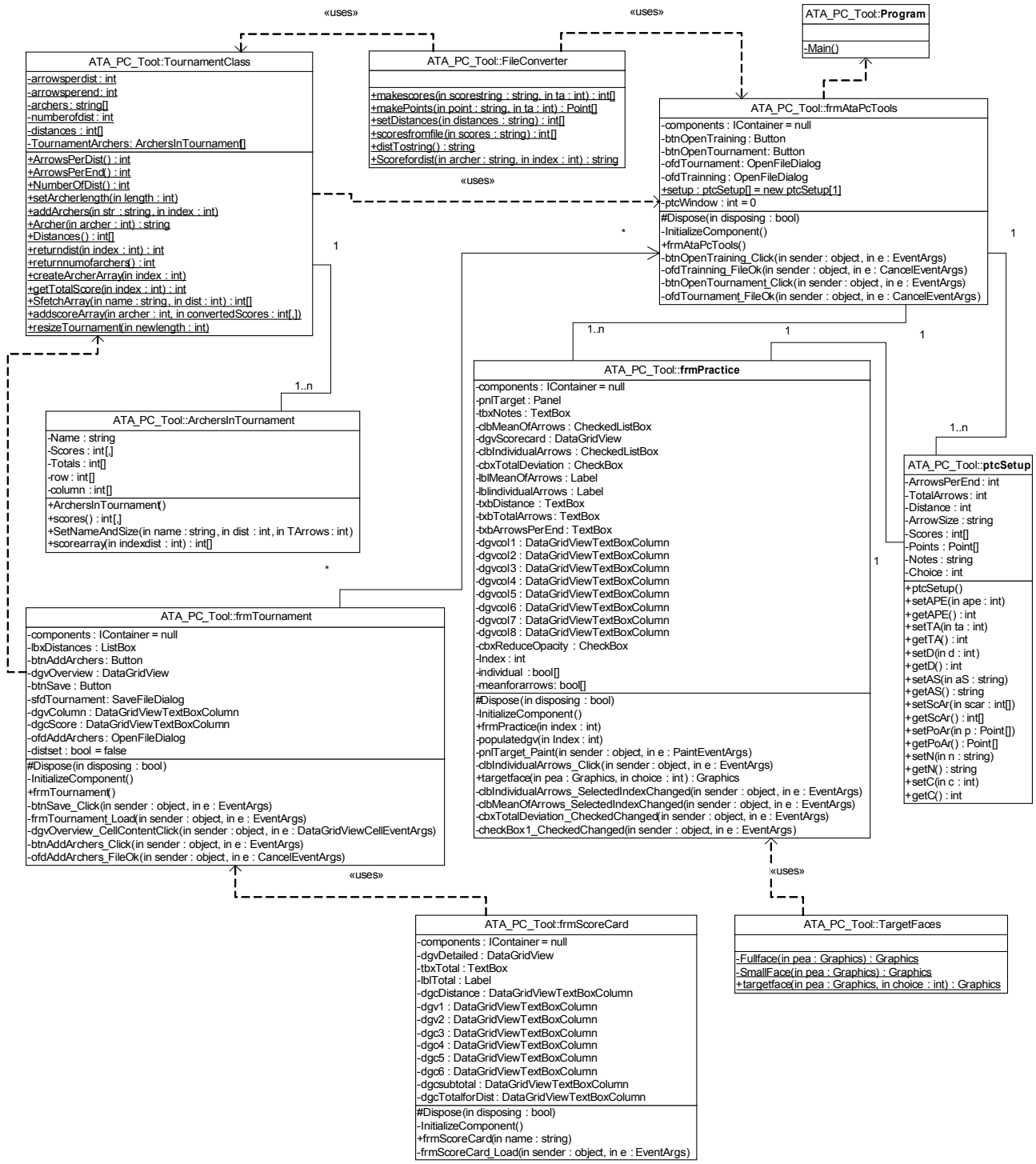
Appendix F: Class Diagrams



PDA Training Class Diagram



PDA Tournament Class Diagram



PC Tool Class Diagram

Appendix G: Literature list

Beck, Kent; Test-Driven Development By Example; Addinson-Wesley; ISBN 0-321-14653-0

Hejlsberg, Anders & Wiltamuth, Scott & Golde, Peter; The C# Programing Language, 2nd Edition; Addinson-Wesley; ISBN 0-321-33443-4

Larman, Craig; Applying UML and Pattern, An Introduction to Object-Oriented Analysis and Design and the Unified Process, Second Edition; ISBN 0-13-092569-1

Newkirk, James W. & Vorontsov, Alexei A.; Test-Driven Development in Microsoft .NET; Microsoft Press; ISBN 0-7356-1948-4

Petzold, Charles; Programming Microsoft Windows With C#; Microsoft Press; ISBN 0-7356-1370-2

Online resources:

http://msdn2.microsoft.com