UiT THE ARCTIC UNIVERSITY OF NORWAY

Faculty of Science and Technology
Department of Computer Science

Soccer Analytic Tool

—

Simen Lomås Johannessen

INF-3983 Capstone Project in Computer Science
December 2013

Abstract

Sports science is an increasingly hot topic and more and more soccer teams are investing time and money into strategic development of big data and sports science. All this to increase the players performance on the field and reduce the risk of injuries. Soccer teams and athletes in general are constantly looking for possibilities to gain advantages over opponents. In soccer your next opponent is analyzed down to the smallest details to find weaknesses and strengths. All this to be able to take advantage of your opponents weak points and limits their strengths.

This project creates and evaluates a system for capturing events relevant for opponent analysis and an interface providing the information for analyzing soccer opponents.

Acknowledgements

Table of Contents

Chapter 1

List of Acronyms

UI User Interface

TIL Tromsø Idrettslag

ZXY ZXY Sport Tracking System

AJAX Asynchronous JavaScript and XML

JSON JavaScript Object Notation

HTML HyperText Markup Language

CSS Cascaading Style Sheets

API Application Programming Interface

REST Representational state transfer

HTTP Hypertext Transfer Protocol

MVP Model View Presenter

DOM Document Object Model

XML eXtensible Markup Language

SPA Single Page Application

List of Figures

List of Tables

Chapter 2

Introduction

2.1 Background

Sports science is an increasingly hot topic and more and more soccer teams are investing time and money into strategic development of big data and sports science [13]. All this to increase the players performance on the field and reduce the risk of injuries. One of the pioneers in this field for a long time AC Milan established in as early as 2002 the MilanLab [5]. The goal for the program was to get better and more concrete information about the players physicality by collecting data over time of players performance. As AC Milan was one of the first to use big data, the use of big data today has exploded. Services like Opta and Prozone serves player statistics, heat maps, and video analytics of players performance. Fans get exposed to these statistics by clubs and broadcasting companies that uses it actively in their coverage of soccer games. The awareness of statistics for clubs and fans has possible never been higher than now [4].

Soccer teams and athletes in general are constantly looking for possibilities to gain advantages over opponents. In soccer, your next opponent is analyzed down to the smallest details to find weaknesses and strengths. All these to be able to take advantage of your opponents weak points and limit their strengths. The players should know what to expect from the opponent team. Detecting typical team plays and player movement of your opponent can help you prepare for the match.

There are several ways to gather information about your opponent; from looking through whole matches to advanced tools, which can highlight key information for you. Some of them are expensive as Prozone [14] which is a complete system for capturing and presenting information. For small soccer clubs with relatively small budgets this can be an expensive investment. Another system is Interplay Sports, which TIL uses for game analytic. A video feed from any match can be used as the input letting you manually tag and describe situations in the match. In a sport where the next soccer match usually is in 3 days tools for filtering out the useless data is valuable.

In the analytic process there are two processes we will look at; first you need to gather the information and secondly is how to present the information gathered.

2.2 Problem definition

This project will develop a system complementing the Muithu and Bagadus systems. Focus will be on soccer opponent analytics, where a data repository needs to be developed capturing important events relevant for this type of analytics. Especially we want to identify the key players offensive in a team. A user interface component providing the core information about the opponent should also be developed and evaluated.

2.3 Interpretation

The project is providing an infrastructure for capturing events like attacks that leads to an attempt on the opponent goal, and presenting this information through a user interface. We are interested in how long it typically takes to capture all relevant events from a single match, as this has to be done manually. A data model that reflects what we want to capture from each attack needs to be developed for being able to get relevant information for being used as an opponent analytic system.

First, a requirement specification shall be developed. Then a prototype developed that fulfills the requirements. At last specialists in the domain of soccer shall evaluate the system.

The design and development processes will be performed in collaboration with staff of the Norwegian soccer club TIL. An end-user comparison of the system against currently available tools will perform a final evaluation, and the result of this evaluation will be used to conclude the project.

2.4 Methodology

The final report of the ACM Task Force on the Core of Computer Science [2] divides the discipline of computing into three major paradigms:

- *Theory*: Theory: Rooted in mathematics, the approach is to define problems, propose theorems and explore to prove them in order to find new relationships and progress in computing.
- *Abstraction*: Rooted in the experimental scientific method, the approach is to analyze a phenomenon by creating hypothesis, constructing models and simulations, and analyzing the results.
- *Design*: Rooted in engineering, the approach is to state requirements and specifications; design and implement systems that solve the problem, and test the systems to systematically find the best solution to the given problem.

For this project the design process seems to be the most suitable out of the three paradigms. The design process consists of 4 steps, which are repeated if tests reveal that the latest version of the system does not meet the requirements.

- *State requirements and specification*: A need or problem is identified, researched, and defined.
- *Design and implement the system*: Data models and a system architecture are designed. Prototypes are implemented.
- *Test the system*: Assessment and testing of the aforementioned prototypes.

2.5 Outline

- *Chapter 2*: Presents some related work to the project
- *Chapter 3*: Describes the requirement specification
- *Chapter 4*: Describes the design and implementation of the system
- *Chapter 5*: Gives a demonstration of the system
- *Chapter 6*: Evaluates and discuss the system
- *Chapter 7*: Concludes

Chapter 3

Related work

In this chapter we present some background related to analytic in the domain of soccer. We look at different types of analysis out there. This spans from pre-match to post-match. Then we go into how data is gathered and presented. For gathering of data there are two main approaches: manually often by humans annotating it, and automatic capturing often by installing receivers and having players wear physical sensors for tracking.

3.1 Types of analysis

In soccer there are several phases where you use analytic to help you gain insight. In this section we will look at the typical use cases of analytic. Not only soccer teams use analytics, but also broadcasting companies and fans widely use statistics to build up under their arguments.

3.1.1 Pre-match

Pre-match you use analytic to find weaknesses and strengths in the opponent team, on an individually level or as a team. You look at your team matched up against your opponent. A typical situation is that the manager gets a video summary of the opponent highlighting the opponents strengths and weaknesses. The coaching staff may use tools like Interplay to gather information and create the video summary. Other software like Prozone[1]

[1]http://www.prozonesports.com/index.html

will also likely be used to break down the opponents offensive and defensive play.

Typically in TV you have pundits bringing you analytic of key battles during the build up to the game. A typical thing is to look at battles like wingers versus backs, midfield clashes or striker versus central defenders. The battles are often illustrated with statistics or with video clips highlighting aspects of the both players game that will be decisive.

Fans are analyzing the game all the time. Except from TV they get exposed to analytic via social media. Opta[2] is company that has taken full advantage of social medias. Figure 3.1 is a screenshot of a tweet from one of the many accounts Opta control.

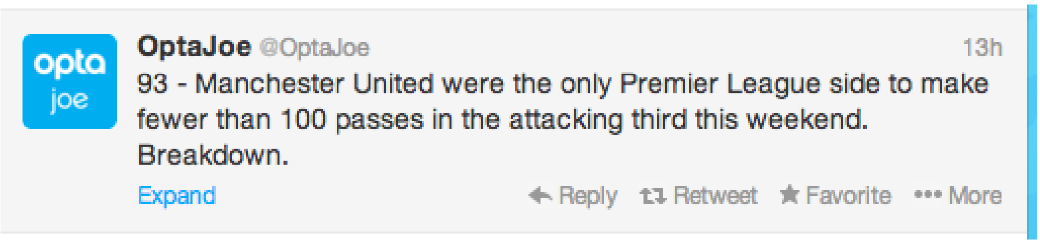


Figure 3.1: A typical tweet from one of Opta's Twitter account

3.1.1.1 In match

During match the coaching staff continuously analyze the match and make adjustment. The players make all the decisions in the end, but the coach is the boss and sets the style of play. An adjustment to the formation can potentially be the tipping point that wins you the game.

Troms IL uses a system Muithu [7] that lets you annotate sequences of a game with entity's like player and comment. This information will then be time synchronized with the video feed. Later, like in at half time or even in the game, you can search on entity's to get the corresponding video feed. It can be anything that has been previous tagged like a player involvement to a team move that was executed perfectly.

Using systems like ZXY Sport Tracking System (ZXY)[3] or MiCoach [13] you can get real time information about a players performance during the match.

[2] http://www.optasports.com/
[3] http://www.zxy.no

Rather than assuming that a player is tiring you can get information metrics that tells you this. You get evidence that the player in fact is fatigued. In soccer you only have 3 substitutions. Making the right ones is crucial in an even match.

Using data gathered from sports data company's like Opta you can get statistics live during the game. It is popular in TV to show statistics like ball possession percent, how far players have run or the number of passes played and so on. As mentioned earlier Opta posts many of their statistics to their social media accounts for fans and media.

3.1.2 Post match

During the post match the coach team goes through the game to evaluate the team performance. This is valuable as you get concrete clips about good and bad involvements on team or player level. You can highlight situations in the match to help players better understand tactical aspects. Interplay Sports[4] is a system used today at Alfheim for this purpose. It is used for analysis of matches in a post-match scenario where you can tag situations in the video.

3.2 Capturing of data

3.2.1 Opta

Opta, one of the big players in the field of sport analytic uses manual input to create their data repository. They have editorial teams across the world that captures data manually for the most popular soccer leagues and other sports. For example to capture statistics for a single match they have 3 humans annotating. The data is captured via an application specifically created for the purpose of capturing data as quick and easily as possible. The editorial teams of Opta need to be able to identify a player in a very short time to be able to keep up with the pace of the game.

Opta capture all types of actions like passes, type of pass, attacks, and interceptions to mention a few. For each action they log, they add a series of description tags like pitch coordinate, timestamp, which player and team.

[4] http://www.interplay-sports.com/

For every single pass they add descriptors if it was a through ball, normal ball or a headed flick on from a long ball. For shots they register the foot it was kicked with, if it was a volley and so on. All this is done while the match is playing. About 1600 individual events are recorded in a standard match. This gives them a very rich database of events and a range of possible queries to run.


```
<Event id=”290575408” event_id=”5” type_id=”1” player_id=”20856” te
outcome=”1” x=”44.6”y=”61.1”
<Q id=”1774596260” qualifier_id=”141” value=”91.6”/>
<Q id=”1429253465” qualifier_id=”140” value=”49.9”/>
<Q id=”1084400575” qualifier_id=”56” value=”Back”/>
</Event>
```


Above is an example of an event registrated in the Opta database. The event has a series of qualifiers describing it. Except from the obvious as timestamp, last modified dates and period we see that the player id, team id, x and y coordinates and the outcome of the event are registrated. Also we see that some extra details are included. In this example the event is a pass from [44.6, 61.1] to [91.6, 49.9] in Optas co-ordination system.

3.2.2 Prozone

Prozone is a video-based system that tries to track players in team sports [14]. Their data capture system incorporates 8-12 cameras, which is strategically positioned throughout the stadium to cover 100 percent of the ground, but with some redundancy in case of a faulty component. They also incorporate the TV-camera feed, which always follows the ball. All cameras are hooked into one server and uploaded at the end of the game before sent to undertake the tracking process. When they system is installed it required some setup before it is operational; the pitch size needs to be incorporated into the system and cameras calibrated. Figure 3.2 shows the camera location on a soccer stadium.

The coding tracking process of the player's movement and actions is a sort of manual process at present time. First each camera feed goes through each own tracking process determining image co-ordinates and continuous trajectories for each player. The output from all the cameras is then combined into one dataset. Going into the algorithm for this is out of the scope of this project. In the final stage the manual work comes into the loop. There has to be a quality control group of operators dedicated for post processing

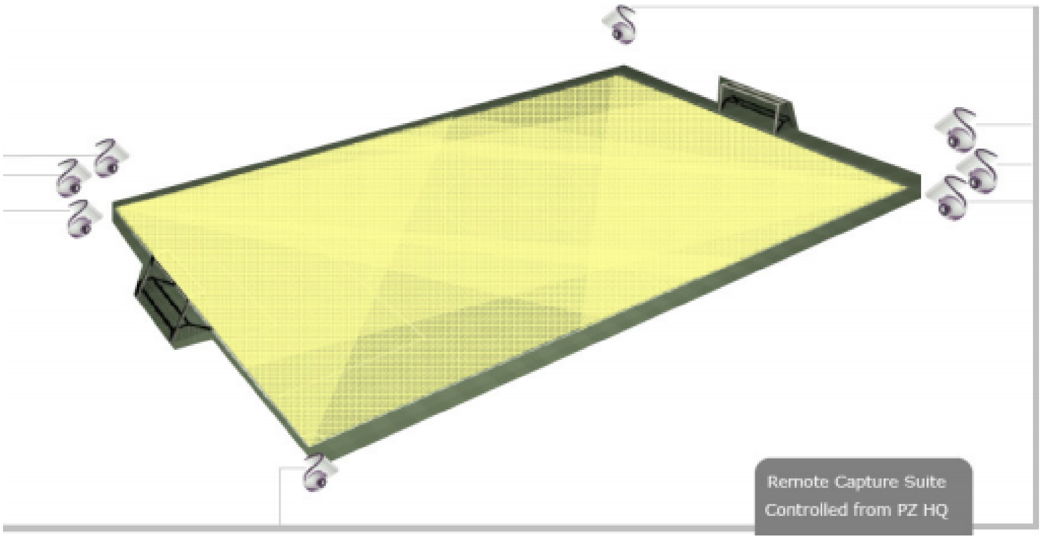


Figure 3.2: The Prozone camera system illustrated. Taken from Prozonesports.com

the game. First, the operators have to map the players identity with their corresponding start location. The operators will then follow the video feed checking that the identification of all players remains constant during the match. The tracking process may run into problems when two players collide or have any other physical contact as it becomes unclear who is who. To map video image co-ordinates to pitch co-ordinates Prozone uses computer vision homography calibration process.

The whole manual operation by the operators is done in own software that helps you minimize the amount of manual work for the operators. The software follows rules created by basic machine learning algorithms to validate and verify the input. A simple example would be if the ball goes out of play the system would know that the next event would be a throw in, corner kick or goal kick.

Prozone claims to be able to track every movement of every player on the pitch with a resolution of 10th of a second. This without using any physical equipment on the players. Di Salvo et al. [1] conducted an empirical evaluation of deployed Prozone systems at Old Trafford in Manchester and Reebook Stadium in Bolton, and concluded that the video camera deployment gives an accurate and valid motion analysis. The data is after a match available through Prozone interface for analysis.

3.2.3 Automatic capturing

A system that uses sensors is the ZXY. The system is used by premier league soccer teams in Norway, including Tromsø Idrettslag (TIL) and Rosenborg BK. Data captured is stored in a Sybase database with each match requiring about 500-700MB storage. The players have to wear a belt around their waist for the system to be able to track their movements. The ZXY system is able to track the players movement very detailed with an accuracy of 0.5m. It has a resolution of 20 samples per second.

The technology behind it relies on a radio-based signaling substrate to provide real-time high-precision positional tracking, also including acceleration and heart rate. An installation of receivers is required for the system to work. The home arena for TIL, Alfheim, is currently equipped with 10 receivers. A receiver tracks a specific area of the soccer field and combined they cover the whole pitch with some redundancy areas. The communication from the belt to the receivers goes on a 2.45-5.2 G Hz frequency radio signal. To compute the positional data the stationary radio receiver uses an advance vector based processing of the received radio signal. The data is aggregated and stored into a relational database. Including the positions of the players ZXY also gives you the step frequency, speed and direction [3].

Figure 3.3: Overview of the ZXY system with receivers placed at the stadium and players wearing sensors. Image from zxy.no

3.2.4 Wrap up

The main problem with tracking systems that uses physical sensors is that usually only one of the teams wears the sensors. This limits the functionality of the system as an opponent analysis system. You only get data for one team and if you have installed it at your stadium it captures only half of the matches.

On the other hand you have the manually systems that require human annotation of matches. These systems have the advantage of being able to track both teams. As they rely on human annotation of some degree they get more rich data as well, but requires strict rules of the semantics of the metadata. There has to be some form of quality control of the data captured to ensure correctness.

3.3 Presenting data

This section gives a short overview of how some systems presents opponent analysis.

3.3.1 Prozone

Prozone comes with several systems to illustrate the data. The most relevant for this project is the opposition analysis system. It includes features like 2D animation, single player analysis, team analysis, pressing analysis, success/direction, player tempo, passing movements, receiving the ball and player events.

On individual level you can get basic statistics like shots, total passes, passing success, crosses, shots on target, balls received, tackles and fouls.

Doing queries on the data can give you all sprints for a certain player. Players in certain areas of the field have more intensive sprints when they first are involved and thus are more vulnerable to injuries. Knowing the actual physical load on players can prevent injuries by regulating the training intensity and amount of time on each exercise.

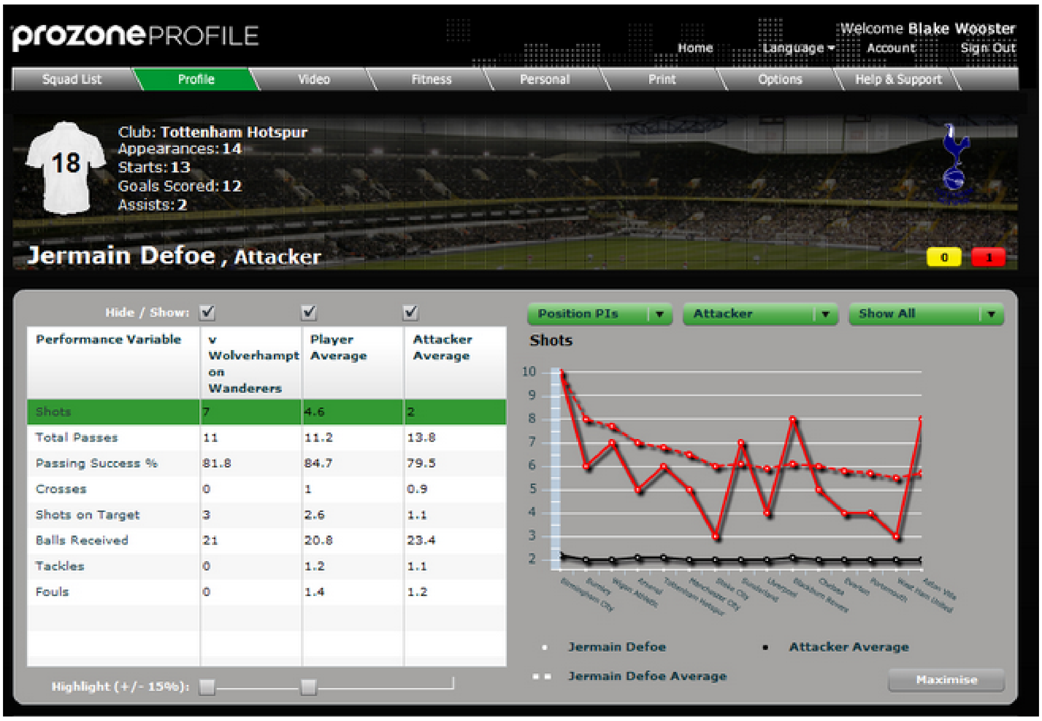


Figure 3.4: The Prozone software - individual player analysis gives you statistics of performance over time. Image from prozonesports.com

3.3.2 ZXY Sport Tracking System

ZXY provides you with a 3D graphic interface showed in figure 3.5. This interface lets you see the players action in real time by reading the data stream to reproduce actions by players. The data is streamed in real time into the database as the match goes on. While watching you can produce timestamps of different events and produce manual input, which is time synchronized with the automatic data. Naturally, you can build your own software on top of the Sybase database. TIL in collaboration with University of Tromsø has made several systems to complement the ZXY software; Muithu[7] and Bagadus[10].

3.3.3 FourFourTwo

FourFourTwo is a monthly magazine about football. They also have a webpage which they update daily. Particular they focus on match analytic with Opta providing them data. All the analyses are created by using their own

Figure 3.5: The ZXY software - tracking of players gives you information of distance covered, average speed and max speed. You also get a heat map of the players movement on the field. Image from zxy.no

developed application Stats Zone. The application is available for everyone and its a popular among blogs writing about soccer. Figure 3.6 shows a typical graphical illustration done with the Stats Zone application highlighting passes in to attacking third of the pitch.

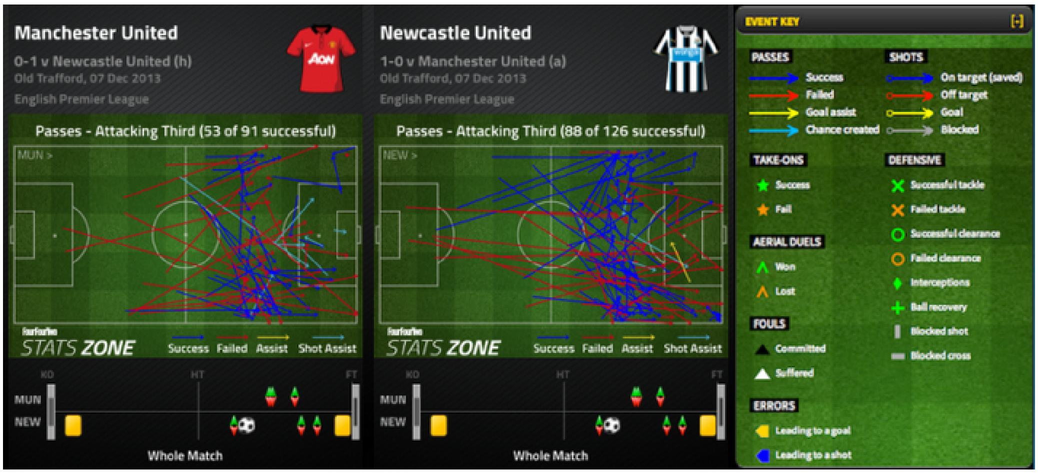


Figure 3.6: Illustration of all passes in to attacking third in the match Manchester United vs Newcastle United. Taken from fourfourtwo.com

Chapter 4

Requirement Specification

This chapter outlines the requirement specification of the system and the process to get there. The process of gathering the requirements was done in collaboration with Tromsø Idrettslag (TIL).

4.1 Overview

The requirements evolved during the process of developing the system. Initially a requirement specification was designed from our perspective. We looked at the different analyzing systems out there and what was in use at Alfheim today. Many systems looks at a single match when analyzing like FourFourTwos application. We are interested to see statistics over time to see if there are patterns of plays that is repeated. The quest of finding which players it is that creates goal-scoring chances is the driving force behind the system. It is easy to check who scores the goals and has the final pass, since these kind of statistics is easily found. Players playing in a more defensive role may have the same influence on the team-results as the top scorer.

Rather than going very wide providing all kind of analyses and statistics we narrowed it down to a very concrete system. The imagined system shall give you the key players in the offensive play of a given opponent soccer team. You shall be able to search on teams and individual players. Additionally you shall be able to see which areas of the pitch players are creating goal chances from. The system also needs a way of capturing data. This shall be an interface that enables you to store successful attacks for any match. We define key players as players that are often the breakthrough player in the build up of the attack by dribbles or passes. Figure 4.2 illustrates involvements of a typical key player we want to identify.

4.2 Capture

A domain model for the captured data is crucial to set as early as possible in the process. A change to the model slows down the development process; captured data has to be re-captured and previous queries on data may fail due to missing fields. The domain model should reflect what we want to get out of the system. It sets the boundary's for which information we can pull from the system afterwards. When defining requirements TIL gave us a domain model they have used for a previous analytic project. The project presented a breakdown of all goals for Manchester United in the 2011/2012 season in Barclays Premier League. This domain model shown in figure 4.1 is used as our foundation for our domain model.

Reg_id	Lag	Kampnummer	Kampdato	Motstander	Kampsted	Mål_nummer	Tid_mål	Omgang	Målscorer	Scoringsmulighet_pr	Målgivende	Antall_i_scoringspos	Corner
1	Manchester United	1	8/14/11	West Bromwich	Borte	1	13	1	Wayne Rooney	30	Ashley Young	2	Nei

Frispark	Skru	Innkast	Straffe	Angrep type	Angrep start	Antall trekk	Målposisjon	Gjennombrudd	Gjennombruddspiller	Innlegg fra	Innlegg til
Nei	Nei	Nei	Nei	Etablert spill	Motstanders banehalvdel	14	S2b	Pasning Bakrom	Ashley Young	I3	S4

Figure 4.1: A screen capture showing some data Troms IL captured in a previous analytic project.

From this we can define what the visioned system should capture from matches. The visioned system should capture from each match every successful attack that leads to an attempt on goal. Then from each attack you capture where the attack started, every pass including from/to zones, and type of attack. At last if there is a breaking point in the attack this should be captured. The breaking point of the attack should be stored with a breakthrough player and what type of breakthrough it was. Figure 4.2 illustrates the different breakthroughs we want to capture.

The visioned system needs a data store. The data should be stored in a database that has a rich query language for being able to support the large range of queries on the data. Good support for aggregate functions is crucial. The data will consist of text and integers.

Additionally an interface for input will be required. This interface should ensure that the input data is correct. For example when defining what type of attack the attack is, only the predefined options should be valid as input.

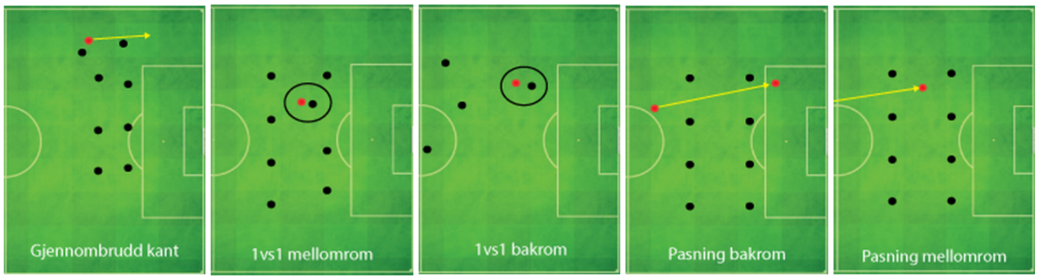


Figure 4.2: Illustrations of different breakthroughs we want to capture.

Capturing every pass in successful attacks requires a positioning system. Here we suggest a more coarse grained dividing of the pitch than x,y co-ordinates like many systems use. We want to look at areas on the pitch The process of defining how to divide the pitch into zones took several rounds of discussion leading to several types. These are shown in figure 4.4.

4.3 Present

The presentation of the data is an important aspect of the system. This will be where the end-users will spend their time. The presentation of data needs to be as simple as possible to understand. The coaching staff is no technical experts and the system is aimed for them to use. Therefor, the system will need to present statistics and other valuable information in a clean and understandable way. In the end it is the 11 players the coach selects that perform all the actions, but the coach sets the style of play and boundaries for players. The goal is to give the coach the opportunity to use the system, using both analytic illustrations and statistics, so he can better reach out to his players with his ideas. The better a coach can sell his ideas to the players the more they will believe in the philosophy, style of play and follow the game plan in games.

We purpose a web interface here, as it is accessible from many devices as long as you have an Internet connection and an up to date web browser.

4.4 Summary

To summarize we need a storage for persisting the data and run queries on, a back-end for transforming client requests to database operations, and an

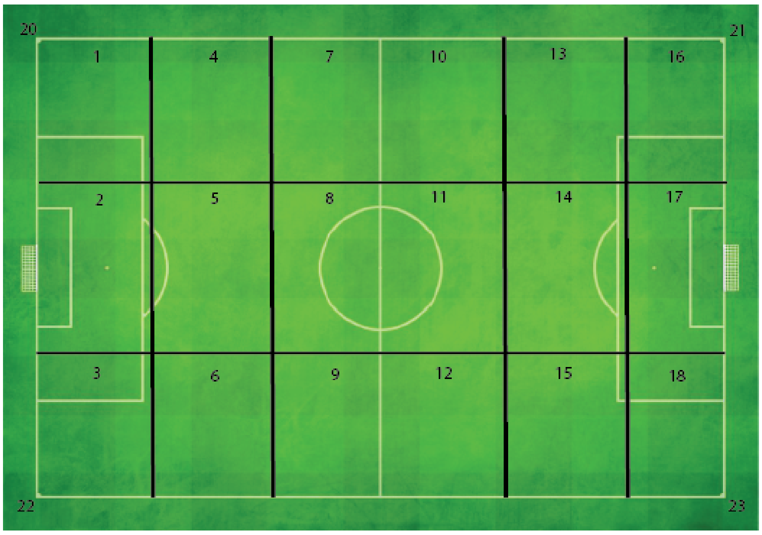


Figure 4.3: Dividing of pitch - the first suggestion had 18 zones

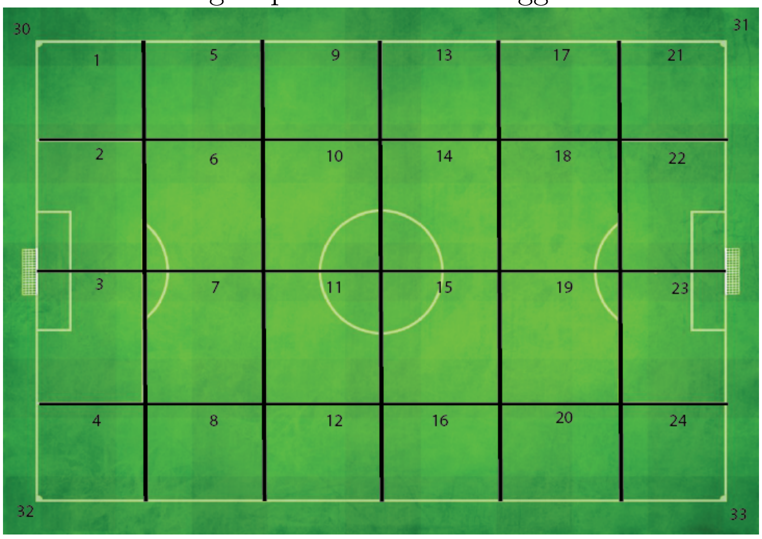


Figure 4.4: Dividing of pitch - the second suggestion had 24 zones

interface for capturing and presenting analytic information. Figure 4.5 shows the conceptual architecture.

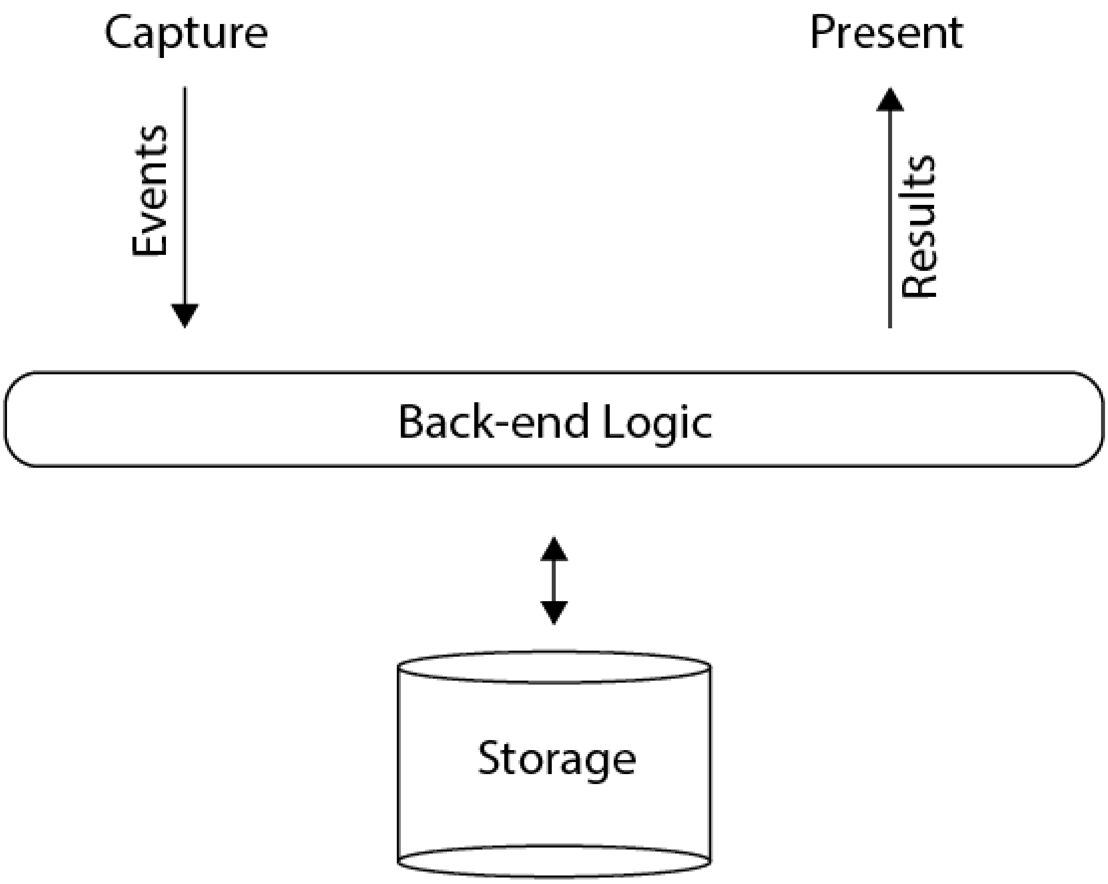


Figure 4.5: Conceptual architecture.

Chapter 5

Soccer Analytic toolkit

5.1 System Architecture

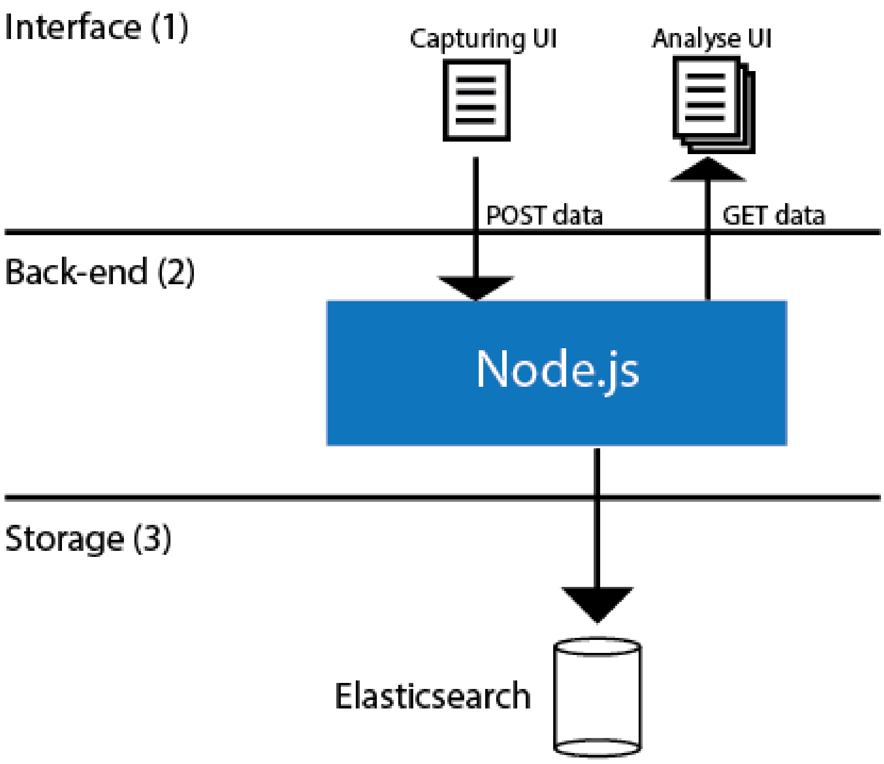


Figure 5.1: Overall architecture of the system

The system architecture can be layered into three layers; client, back-end and database. The control flows from the clients requesting a page or inserting data, to the back-end who communicates with the storage. From the storage the execution cycle is reversed. The storage will execute what its been asked to do by the back-end and return the result to the back-end. The back-end then again returns the result to the client.

5.1.1 Interfaces (1)

The Soccer Analytic tool has one user interface - the web browser interface. Both the capturing and the analytic interfaces can be reached from this single web browser interface. All interfaces are created on the client. The client will get data from the back-end and construct the interfaces dynamically.

5.1.2 Back-end (2)

The back-end is the middle layer between the client and the storage. Its main task is to serve static files to the clients, handle data insertions or handling web request by mapping them to database operations. New data insertions are possible inserted into several database indexes. The back-end will then ensure that all indexes are updated before returning success to the client.

5.1.3 Storage (3)

The storage layers task is to persist data and handle search queries on the data. It consists of several indexes that each stores a part of the domain model.

5.1.4 Domain model

The domain model is based around matches. All attacks are wrapped into a match root. Attacks can be seen as subdocuments of the match document. Inside the attacks all passes lies with other information. This gives an easy way of understanding how all the data is related together. Below is a complete example of how it all is structured. The number of attacks and passes is stripped down to one in the example.

```
{
    "hometeam" : "Tromso",
    "awayteam" : "Rosenborg",
    "score" : "1-0",
    "date" : '2013-15-09',
    "attacks": [
        {
            "time": 4,
            "touch" : 1,
            "team" : "Tromso",
            "breakthrough" : "None",
            "breakthroughPlayer" : "None",
            "typeOfAttack" : "Dodball",
            "attackStart" : {
                "pos" : 17,
                "typeAction" : "Frispark",
                "player" : 403,
            },
            "passes": [
                {
                    "fromPlayer": 403,
                    "toPlayer": 393,
                    "fromPos": 17,
                    "toPos": 23,
                    "action": "CROSS"
                }
            ],
            "finish" : {
                "player": 393,
                "pos": 23,
                "action": "SHOTMISS"
            }
        },
}
```

Figure 5.2 shows the final version of the pitch dividing with a illustration of how an attack is captured. Player 4 plays to player 10 from zone 13 to 14. Player 10 transports the ball until he passes to player 15 from zone 18 to 22. Player 15 tries a finish from zone 22 inside the penalty area. The reason for going for more zones in the penalty area was to be able to see more specific where attacks are finished.

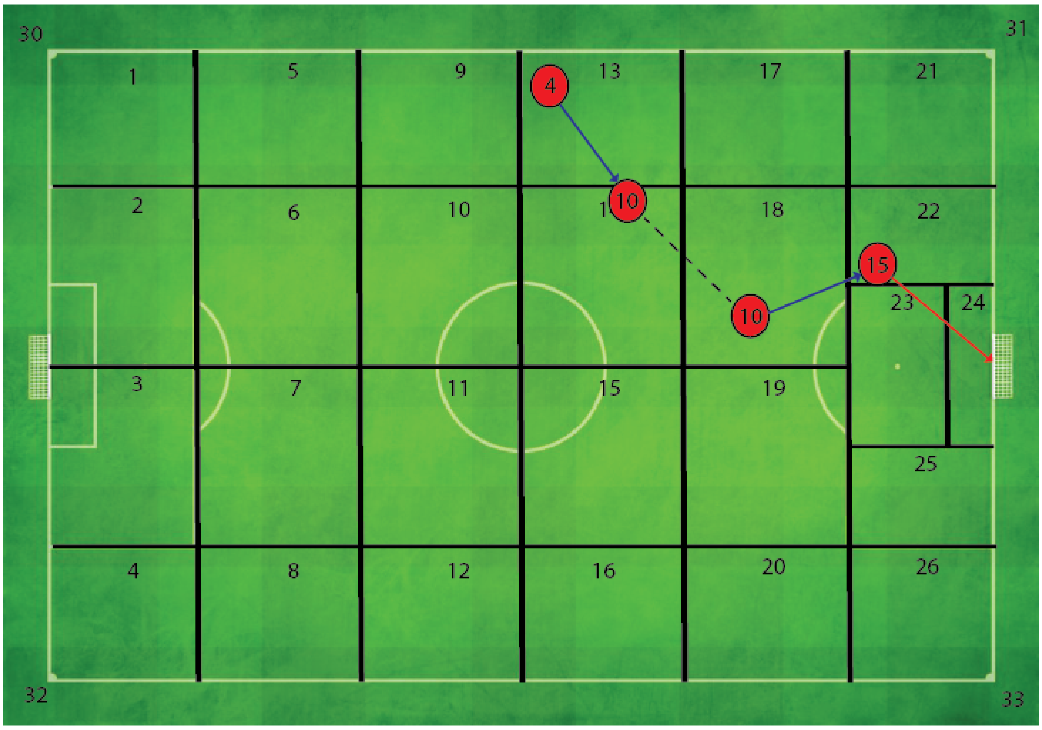


Figure 5.2: Final pitch dividing. More zones have been added to the penalty box area than the initially design.

5.2 Implementation details

Software Stack	
Web server	Node.js
Database	Elasticsearch
Web Interface	HTML5, CSS3
Client Side Technology	Backbone, Bootstrap, jQuery, Mustache, Highcharts, Lightbox, async

Figure 5.3: Software stack

5.2.1 Storage

The data storage is an Elasticsearch database[1]. Elasticsearch is document oriented, schema less and works well with JavaScript Object Nota-

[1] http://www.elasticsearch.org/

tion (JSON)[2]. As our server is built on JavaScript, working with JSON is easy. JSON-objects can be inserted right into the storage and Elasticsearch will map fields and value accordingly, and make it available for search. Elasticsearch takes advantages of embedded documents meaning we can store related data together. An attack is usually made up of several passes these can be stored as an embedded document in the attack document. Then all passes can be retrieved in one query when fetching an attack.

The main reason for using Elasticsearch is its search capability. In the starting phase of the project MongoDB [3]was the storage engine. However as some aggregation queries were hard to figure out how to do, a change to Elasticsearch was made. It should be noted that MongoDB supports map reduce operations. With some extra work the queries causing problems could possible have been done with MongoDB. However, with Elasticsearch you can in a single, simple to write query get counted how many passes all players for a team has played and received, the number of times all players has been the breakthrough-player in an attack, count type of attacks, count the most used zones for passing and finishing and so on. This makes it very easy and efficient to do queries for analyses on teams and players.

Another feature of Elasticsearch is how easy it is to scale it by adding new nodes to your cluster. It is built to scale horizontally out of the box. Elasticsearch uses a sharding technique to spread data across the cluster. Basically, it breaks up the data into smaller chunks spreading it across the cluster of machines to achieve scalability and performance. The configuration setup is highly adjustable letting you specify number of shards and replicas the storage shall keep. Adding a new machine to the cluster will automatically start a reorganizing process of the data to take advantage of the extra hardware.

5.2.1.1 Indexes

Indexes are much like tables in a relational database in the way that they are a container for data. An index stores documents, which is a bunch of key/value pairs, like JSON. You can let Elasticsearch automatic analyze the field data or you can use mappings. For supporting querying on players and other fields where the value is more than one word, you have to tell Elasticsearch that it is a multi field. For example, searching for player name

[3]http://www.json.org/

[3]http://www.mongodb.org/

"Stefan Johansen" when the field is not set as multi field will give out two results and not one, as you would expect.

In our project we have 5 different indexes.

- *Team index*: Holds all the teams.
- *Player index*: Holds all players for every team.
- *Match index*: Holds all data from every match, including all attacks for each match.
- *Attack index*: Holds all attacks in every match.
- *Pass index*: Holds all passes in every attack.

Listing 5.1.4 shows an example of data stored from a match. As may be noted there is data redundancy as some indexes stores the same information. One can argue that storage has become so cheap and if you can use a little bit extra space to gain performance you would do it. In this case it was done to be able to fully support aggregation functions on subdocuments.

5.2.2 Back-end

5.2.2.1 Overview

The back-end is the middleware between the clients and the data layer and is written in Node.js[12] It exposes a Representational state transfer (REST)[4] interface over Hypertext Transfer Protocol (HTTP) shown in figue 5.4 for the client to communicate. A request coming in is transformed to a database query based on the resource it tries to access. Node.js is built on Chromes JavaScript runtime[5] engine. Its a relatively new and unproven platform, but has gained a lot of hype in the computer science community for its lightweight and efficient model. This comes from the even-driven and non-blocking I/O model. Per default, an instance of node runs in a single thread and does not utilize other CPU cores. When programming you need to avoid long, blocking computations as this will block the node process for handling new incoming requests. Our back-end uses non-blocking I/O calls when accessing the database and does minimal computation elsewhere.

[4] http://www.ibm.com/developerworks/webservices/library/ws-restful/
[5] https://code.google.com/p/v8/

5.2.2.2 Routing

The back-end code is structured up in modules. Requests on the URL /players is routed to the Player module for handling. Similar, you have an attack module, pass module, team module, player module and a match module. A very often used URL is /team/:name. A HTTP GET on this URL will be routed to the team module where a query for information about the specified team is run. On answer from the database, the result is transformed before returning it to the client.

Similar, if the client sends new data for a match the server will insert the data into the appropriate indexes. The server will respond will HTTP status code 201 if all goes well or 400 on an error. The server uses HTTP code actively to tell the client the result of requests. For all HTTP POST requests the server will send status code 201, meaning that a new resource is created. For HTTP GET requests the server responds as normal with status code 200, if all went well.

No URL route uses HTTP PUT and DELETE methods. These methods could have been used for deleting or updating attacks or passes. Typically, an operator will do a mistake while capturing new attacks.

In principle, since the data input is generated in the web browser (with JavaScript), it could have been inserted right away into the database without going through an extra middleware. However, this limits us, as we cant combine multiple queries by going through the back-end. Cleaning of data before serving to the client would neither be possible. Normally, you also do validation of the data at the back-end before inserting into your database.

5.2.2.3 API

The Application Programming Interface (API) of the web server is listed in figure 5.4. As mentioned the API follows the REST principles. Optimal, routes like /teams/:name/finalthird should not be needed as all statistics could have been returned when getting team statistics.

5.2.3 Front-end

The front end is a Single Page Application (SPA) [] using Backbone.js[6]as under-supporting library. This means that not every code is necessary loaded

Web server API	
GET /matches	Get all matches
GET /match/:id	Get specific match
POST /match	Post new match
GET /teams	Get all teams
GET /team/:name	Get team statistics
GET /team/:name/finalthird	Get passes into final third of the pitch
POST /team	Post new team
GET /player/:id	Get player statistics
GET /player/:id/breakthroughs	Get breakthroughs for player
GET /player/:id/finalthird	Get passes into final third of the pitch
POST /player	Post new player
POST /attack	Post new attack
POST /pass	Post new pass

Figure 5.4: Overview of the web servers API

at once (HyperText Markup Language (HTML), Cascaading Style Sheets (CSS) and JavaScript), but when needed. Also the whole page does not reload when users navigate around, only parts of it will be updated. This is all possible because of Asynchronous JavaScript and XML (AJAX). AJAX is way for JavaScript running in a browser to make HTTP requests to back-end web servers. Its asynchronously, meaning that the web page is still responsive and navigable during the request[8].

Backbone uses a Model View Presenter (MVP) model to structure the code. With Backbone your views will update automatic when data changes. You use Backbones models and presenters (called views in the documentation of Backbone) by letting your own models and presenters inherit Backbones basic classes. Figure 5.5 shows how the overall architecture of the front-end. In the following sections the architecture of the client and how different concepts is used will be described.

[6]http://backbonejs.org/

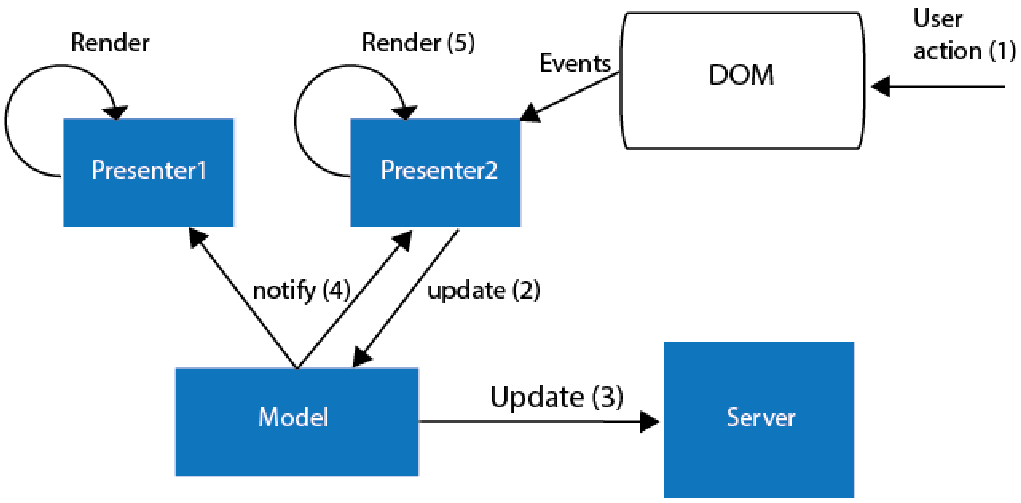


Figure 5.5: Architecture of the client side. The numerating is used to reference different behavior

5.2.3.1 Models

In Backbone you handle data and communication with the back-end with objects called models. A model has mainly two responsibilities; whenever an update on a models data occurs the model notifies the presenters that have subscribed for update events for that particular model (4). The second is that models are responsible for AJAX communication with the back-end (3). An example is when a user registers a new attack for a match. He fills out a form and press the submit button (1). Then a new attack model is created. Calling save on the instance of the model will send an AJAX post request to the server with the models data in the HTTP body (3).

Similar to an attack model we have player model that handles everything around players. When you click yourself into a players profile the model will fetch statistics from the back-end, notify the presenter that the data is ready, and the HTML will be rendered in the web browser.

There is also a match model for fetching and registration of matches, team model for viewing statistics, and a pass model for saving passes. They work it the same way as the models described above using AJAX to communicate with the back-end and notification presenters on data changes.

5.2.3.2 Collections

Collections are a set of models. It has almost the same functionality as a model. You can bind functions to happen on events for any model in the collection. On the client there is two places where collections are used; players and teams. Naturally, players contain many player models and teams collections contain many team models.

5.2.3.3 Presenters

Presenters are the glue between the Backbone models and the view (web page). Presenters can bind own functions to respond to models events. For example on data change event from a model the presenter will call its render function and the User Interface (UI) will updated (4). Each presenter has a own HTML div tag which he renders to. Including to subscribing to events on models the presenter can subscribe for view events that happens in the Document Object Model (DOM), events that the user triggers (1). Following up the example in the previous section on an attack registration. When a user press the submit button the presenter will be notified. He will create a model for the attack and calling the save method on the instance object.

On initialization of a presenter object it is initialized with a model or a collection depending on the view it should render. After the model has fetched its data the presenters render function will be called. This is where the data is inserted into the **??** template explained in more details in the following sections.

5.2.3.4 Views

For an analytic toolkit to be useful a good presentation is critical. Here several helper libraries are used to present the data. Highcharts.js[7]is a JavaScript library for illustrating graphs. A query on team generates a lot of statistics and rather than listing them up they are presented using charts. Using charts gives us the advantage of displaying several numbers for each player by plotting it in the same graph. In figure 5.6 the number of times a player has been involved in all attacks, the number of passes into the final third of the pitch and the number of times a player has been the breakthrough-player is shown.

[7]http://www.highcharts.com/

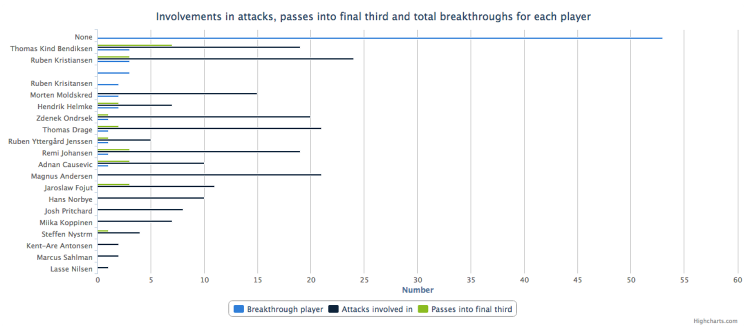


Figure 5.6: Shows how the passing statistic is illustrated on the client by using Highcharts.js

Positional data is created using the new feature of HTML5, canvas element. It lets you draw graphics on the fly in the web page. In this system it is used to create an element that symbol the different zones in our domain model. The team model will fetch data from the back-end containing numbers for all zones that symbols shots taken from that zone. This is then plotted into the respective zones as figure 5.7 is showing.

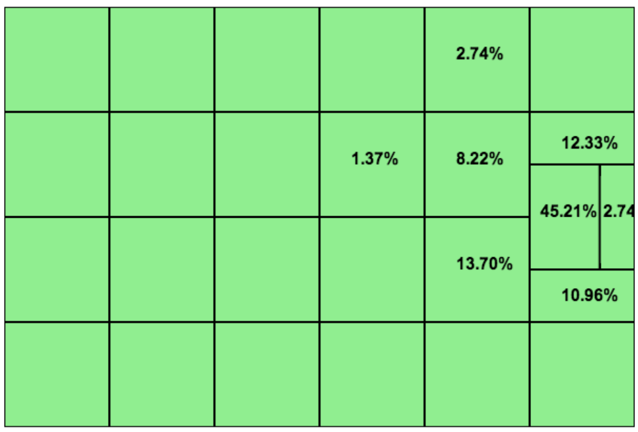


Figure 5.7: Illustrations of which zones the team has finished their attacks from in percent. Team TIL

Backbone comes with a library Underscore.js[8]that makes creating HTML pages with dynamic content easily. When you are rendering new content on the site you can insert data retried from a model dynamically into the HTML. In the example below the input is a an array of objects where each element in the array contains a key value pair 'name' : value. The HTML output of this will be a list of names.

```
{{#players}}
        <li>{{name}}</li>
{{/players}}
```

5.2.3.5 Router

A component not mentioned before is the router. The router is the glue that binds all the other Backbone modules together. It handles the navigation between pages in the application. When users navigate on the page by clicking on links or buttons in a traditional web page the back-end serves a new HTML page. Here, a router module handles this click event. The router model examines the URL request and routes the request to the presenter responsible for that specific route. This is mapped inside the router. The presenter will then call fetch functions on the model to get data from the back-end and render the HTML.

In the page the header is static and will only be rendered once when the user first visits the web page.

5.2.4 Building the database players and teams

Finding squads in a useful format like eXtensible Markup Language (XML) or JSON was harder than expected. On the Norwegian football alliance's website you could download the squads for the current season, but only in PDF format. Current squads is fetched from altomfotball.no, a website by the Norwegian broadcasting company TV2. The fetching process is an own python script meant to run only once to set up the database. For each team the script basically reads the HTML document with all players listed, parse out players name, and sends in to the web server.

5.2.5 Security

Security is not taken into concern. This means anyone getting into the page can post new match data and add attacks. This could have been fixed by requiring a login before getting access to the site.

[8] http://underscorejs.org/

Chapter 6

Demo

6.1 Interfaces

6.1.1 Mockup

Figure 6.1 shows a mockup of the team analysis page that was created before starting developing. There are two main sections in the mockup: team statistics and key players.

In the team statistics section types of attack is illustrated in a graph, there is an overview of different breakthroughs, basic statistics like key players, number of attacks, average possession is listed, and a plotting of which zones attacks are started from.

In the other sections key players and their statistics is listed. You have an overview of where on the pitch the player is involved in attacks, plotting key passes he has played and where on the pitch his ball recoverys are.

6.1.2 Implemented interfaces

6.1.2.1 Home page

The first page you are prompted with is the listing of all matches registered in the database as figure 6.2. A click on match gives you details about that match and prompts you an interface for capturing new attacks if requested. Here you fill out every field in the form.

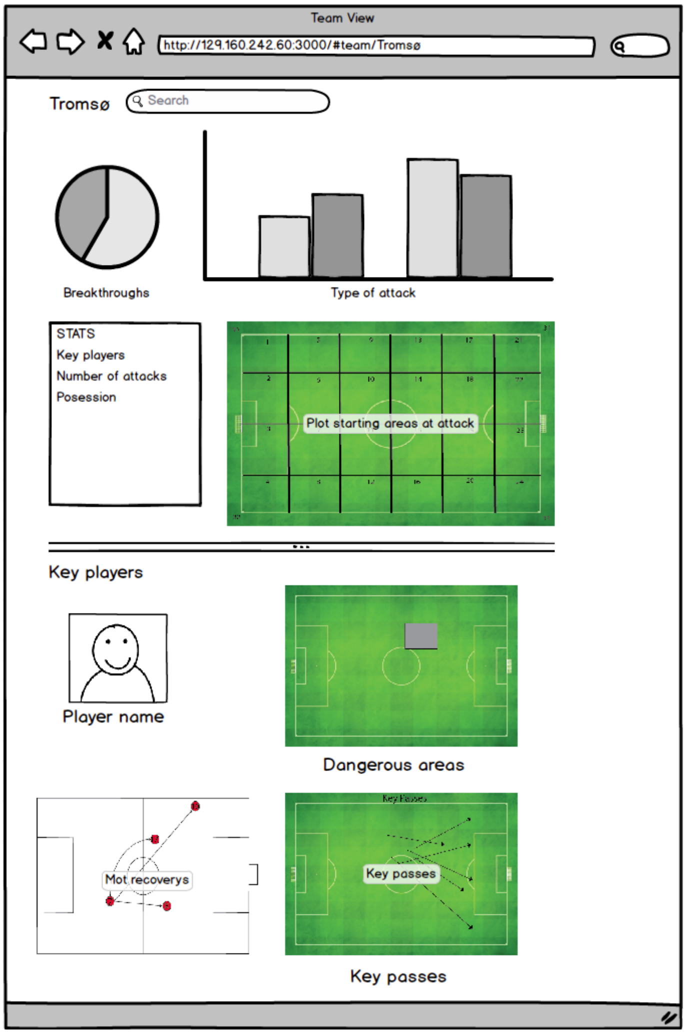


Figure 6.1: A mockup of the main analytic page

Clicking New match sends you to match register interface shown in figure 6.3. The users have to register the match he wants to capture data before anything else.

The whole web page follows a theme throughout the pages. This theme is the standard theme in the Cascaading Style Sheets (CSS) and JavaScript library Bootstrap[1]. Bootstrap makes the website by default responsive. This means the content on the site is automatically re-sized out from your browser window size.

[1]http://getbootstrap.com/

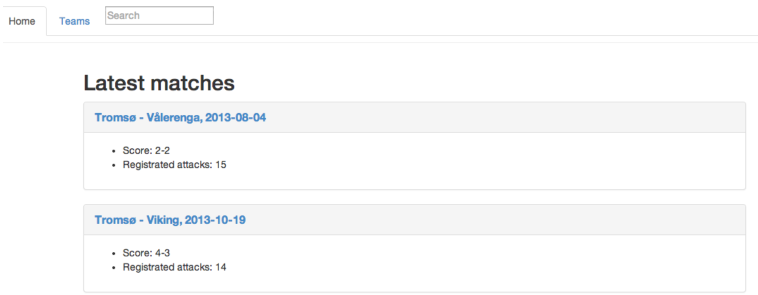


Figure 6.2: All matches registrated in the database are listed on this page. Image is cropped not listing all matches.

Figure 6.3: Interface for registering a match

6.1.2.2 Match view

Clicking on a match gives you overview of all attacks registered for that particular game shown in figure 6.4. This view is only meant for quality checking the data.

From the match page you can add new attack attempts in the interface figure 6.5 shows. The user will fill in the form for the attack. Attack start, attack end, breakthrough player, time and team needs to be filled out. The passes in the attack are added by pressing new pass. This will add a new pass to the form. Passes can be deleted by clicking the red remove pass button.

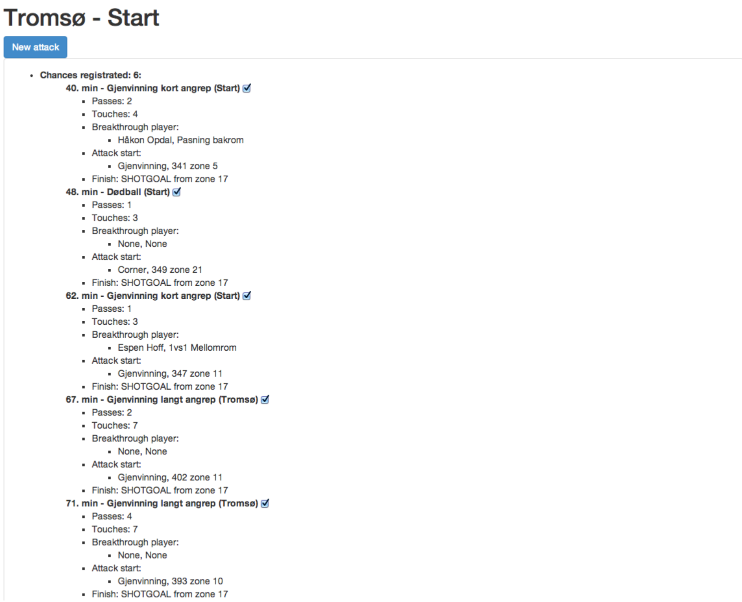


Figure 6.4: Interface listing up all attacks for a match.

6.1.2.3 Team view

Then you have the team selection page shown in figure 6.6). This is a simple layout listing all the teams in the Norwegian premier league. From here you select the team you want to analyze.

Then there is the main view used for analyzing a team, showed in figure 6.7, figure 6.8 and figure 6.9. The page design is not exactly the same as the mockup. This comes from a various things. A description of different breakthroughs has been added to enlighten users of the system. Users of the system may or may not have been involved in the process of capturing the data and therefor-extra information is needed to clarify concepts. In the mockup key players where highlighted. In the final view various statistics is presented to the user and he can then click on players he find interesting to know more about as shown in figure 6.10. The whole page is divided into 4 sections; key aspects, offensive play, defensive play and players.

6.1.2.4 Player view

Clicking on a player brings you to the player view shown in figure 6.11 and figure 6.12. Here individual statistics is highlighted.

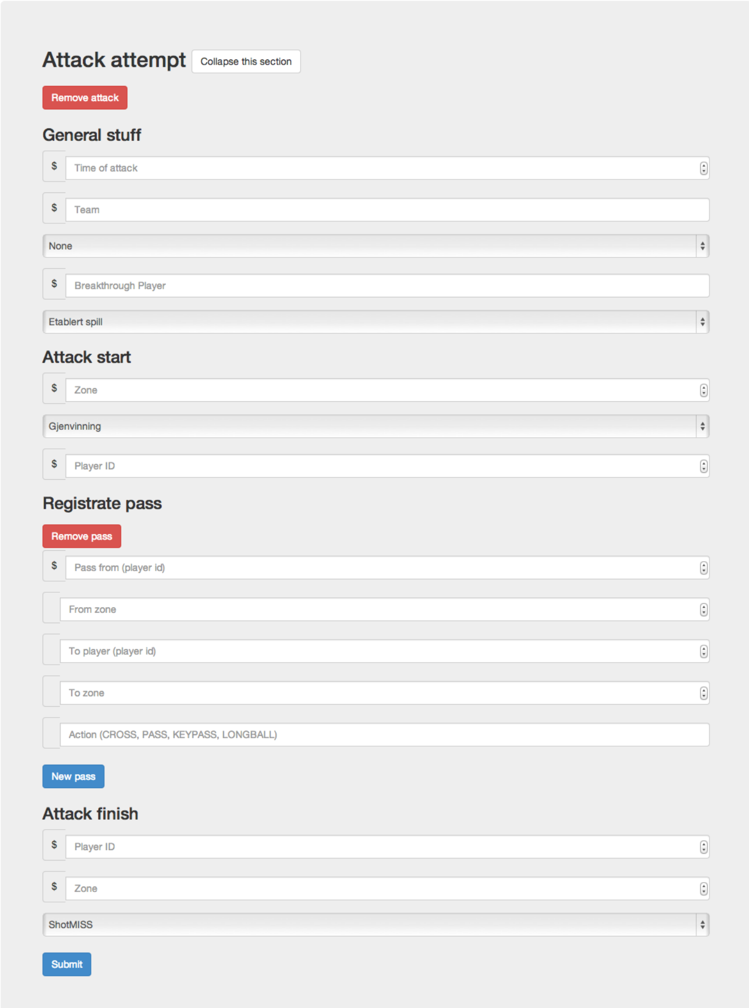


Figure 6.5: Interface for register an attack.

6.2 Capturing process

As mentioned, the process of capturing data is manually. In the beginning it took up to 1 hour to capture all attacks for a match. When you get used to the interface and is able to quickly identify players the time used went down 15-20 minutes. Where most of the time went was getting the players id by looking up in the database manually. To speed up the capturing process the starting 11 names and their corresponding ID where written down with the formation the team plays in to a piece of paper before starting. This also helps identifying players, which can be a bit of a hassle. Match videos came in the resolution 640 * 360 pixels, which is not very good. In addition, on

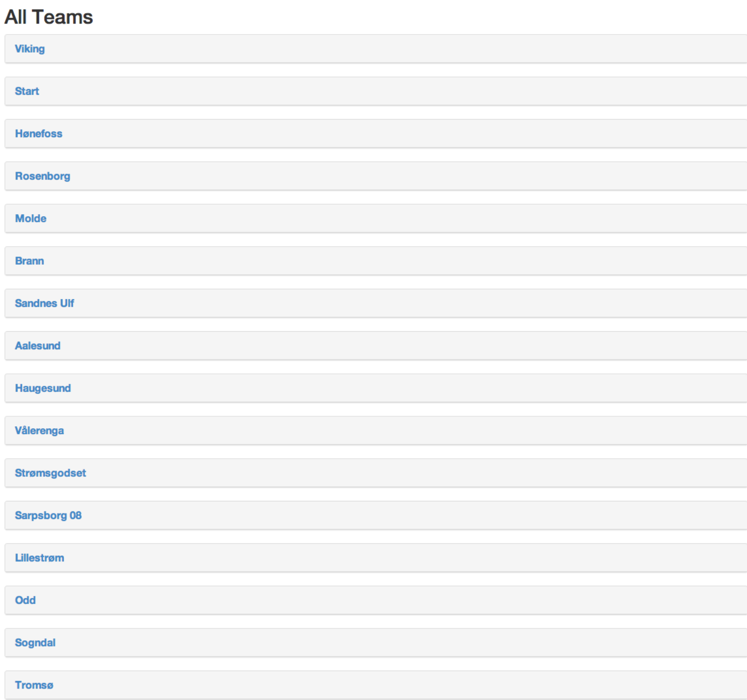


Figure 6.6: Interface listing all teams.

some stadiums there is a running track around the pitch meaning that the grand stands and the main camera is located further away from the pitch. It makes it even harder to identify players on the opposite side of the where the camera is. Having the formation of the team on a paper helps you to identify the players.

The process of capturing has been the following:

1. Download the match video.
2. Find the match in the VGLive.no archives. VGLive.no is used to quickly find all attacks ending with a finish. Figure 6.13 shows some comments from a match.
3. Register the match by filling out the form in figure 6.3.
4. Start finding possible attacks to capture by using VGLive's annotation and jumping in the video. Every event has a time in VGLive.
5. Capture every attack that leads to an attempt on goal one by one via the interface shown in figure 6.5.

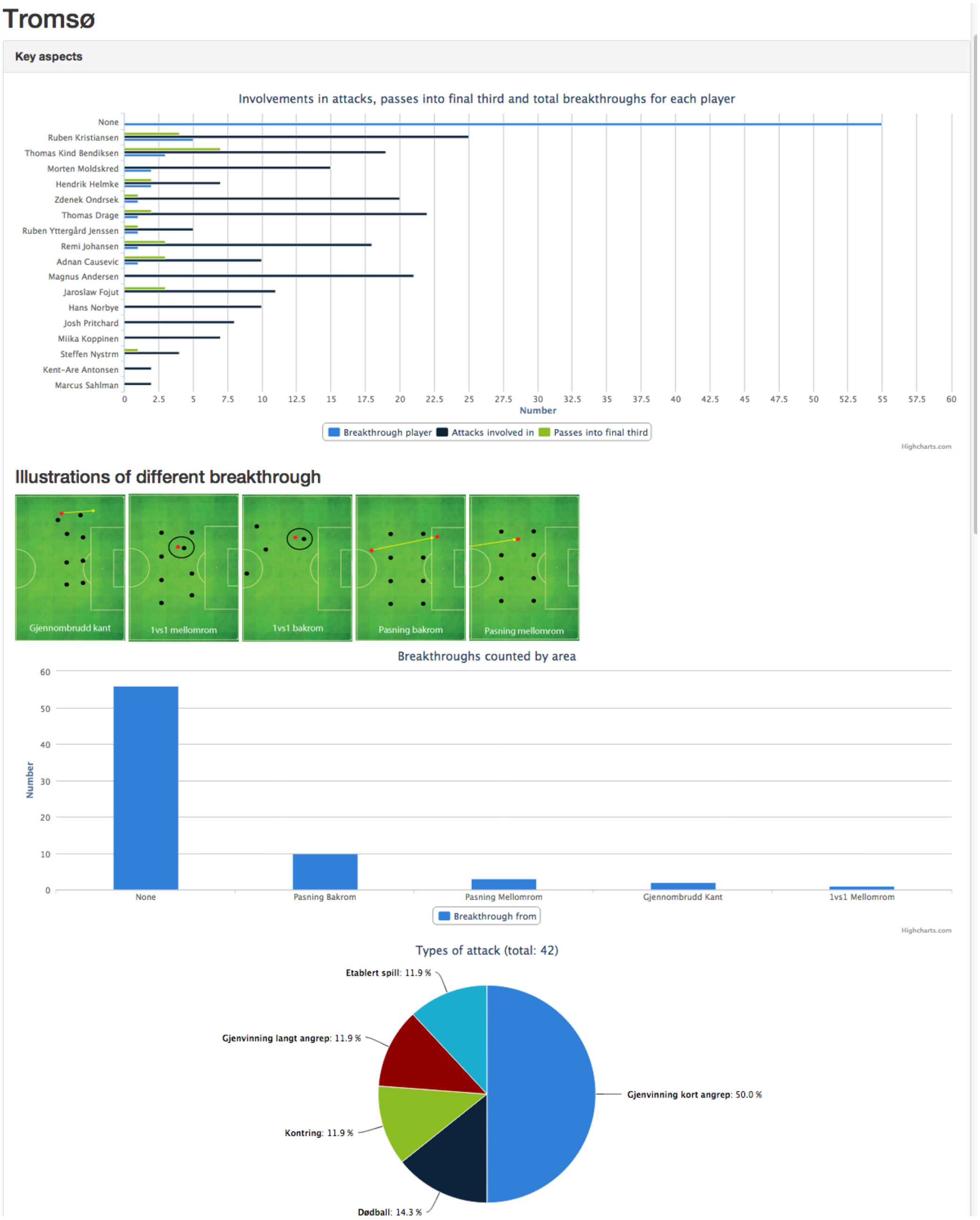


Figure 6.7: Main interface for information about opponents - key aspects.

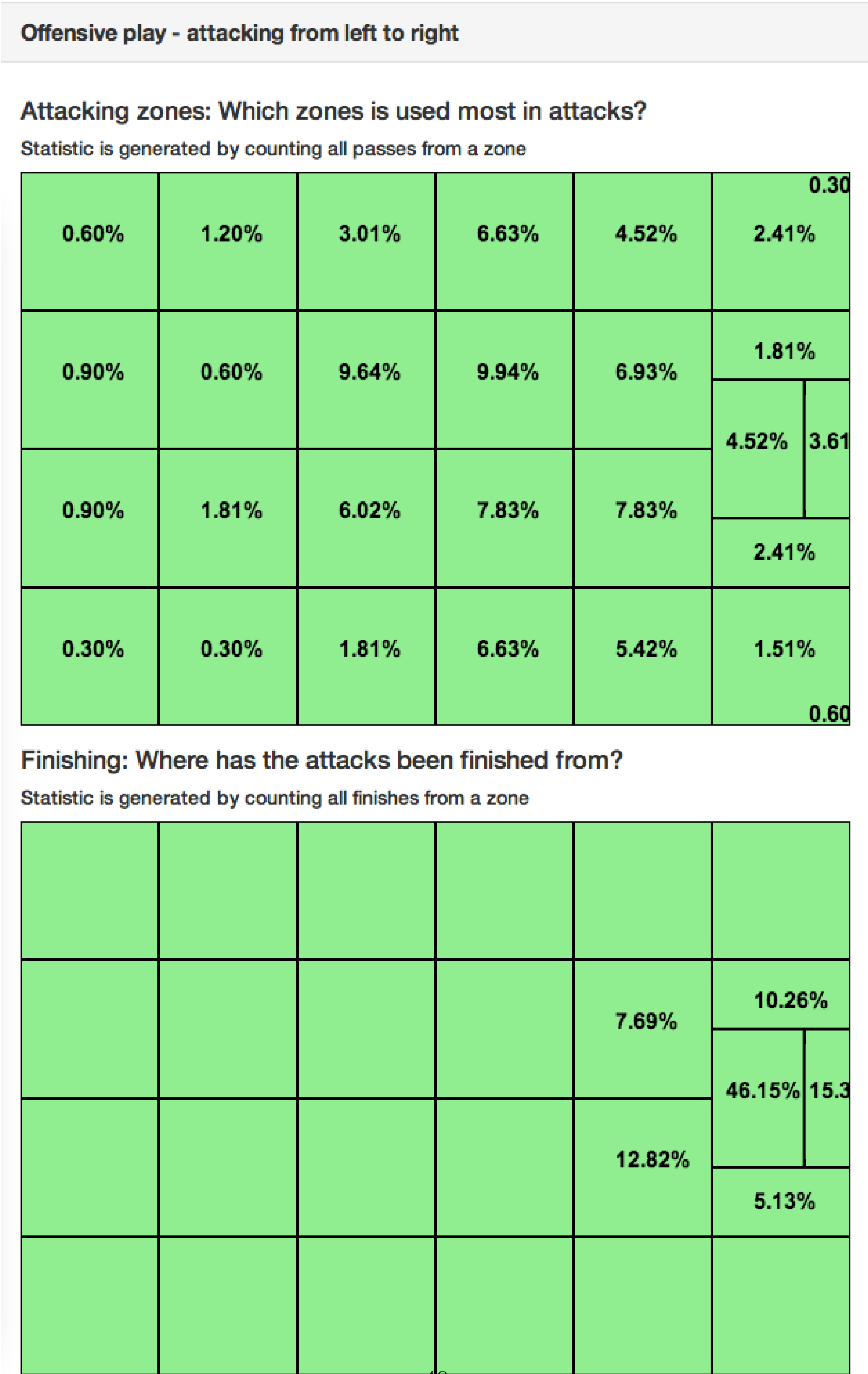


Figure 6.8: Main interface for information about opponents - offensive play.

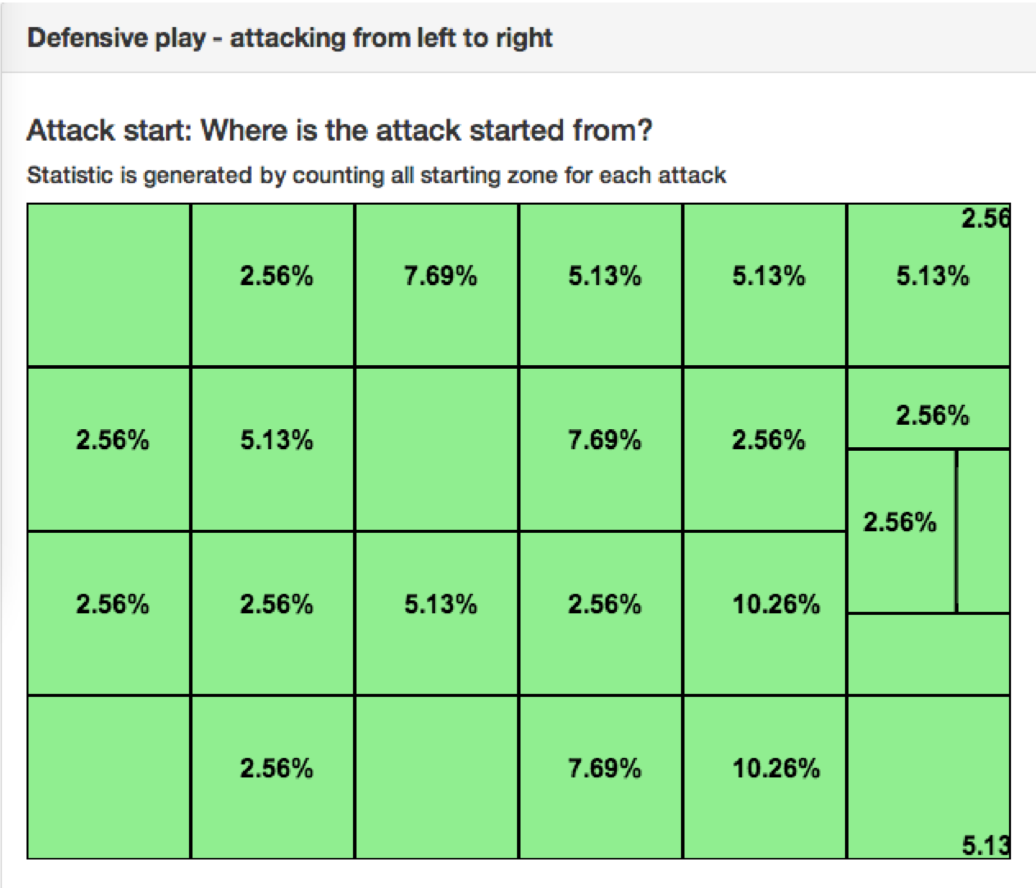


Figure 6.9: Main interface for information about opponents - defensive play.

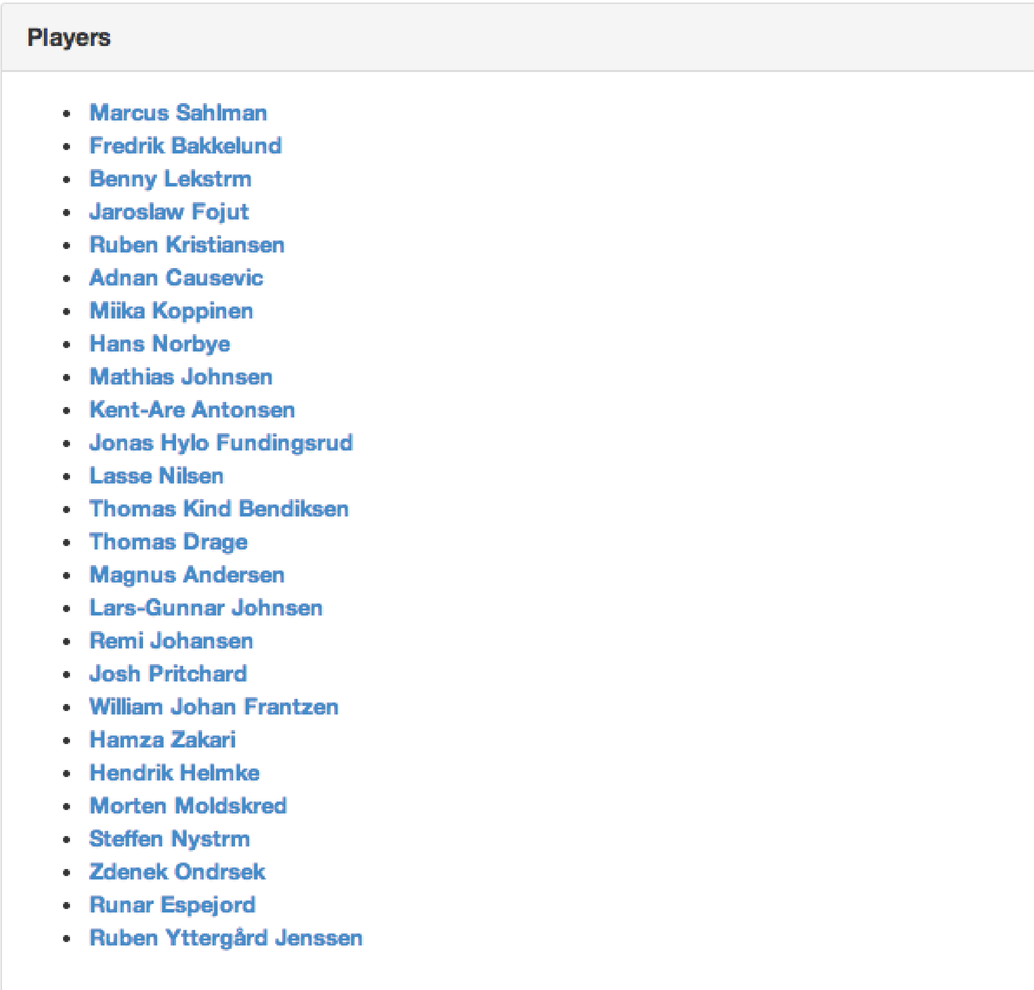


Figure 6.10: All players are listed below the team analytic page. Clicking on a player brings you to the player view.

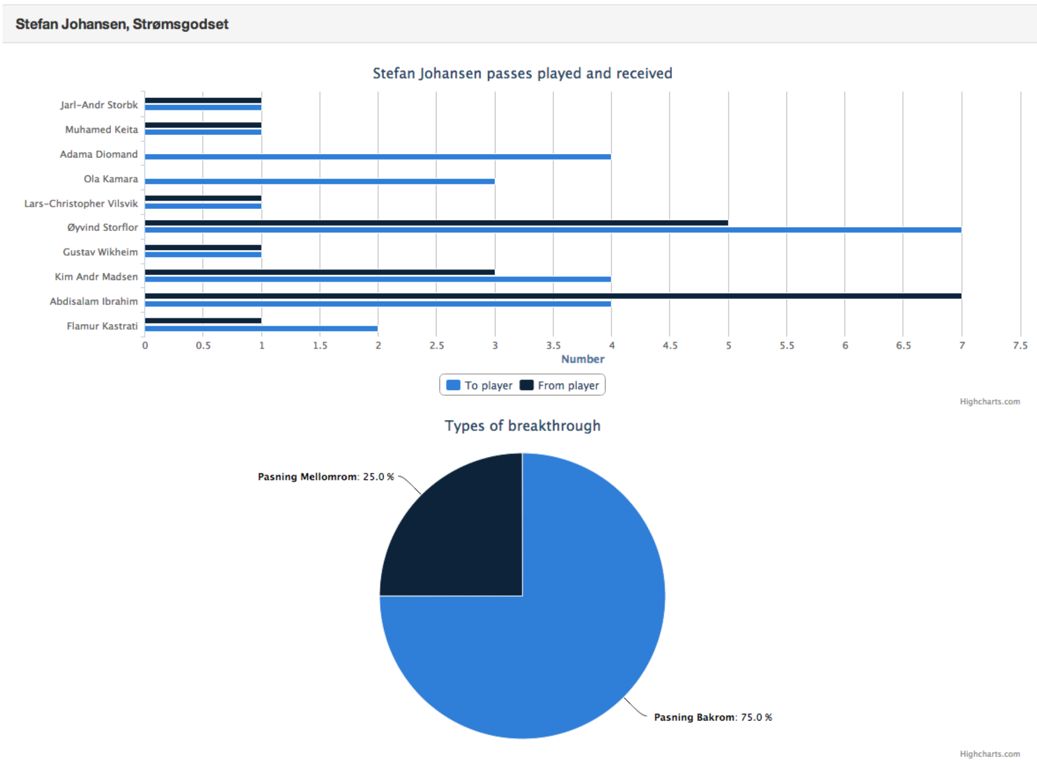


Figure 6.11: Player view highlighting individual statistics

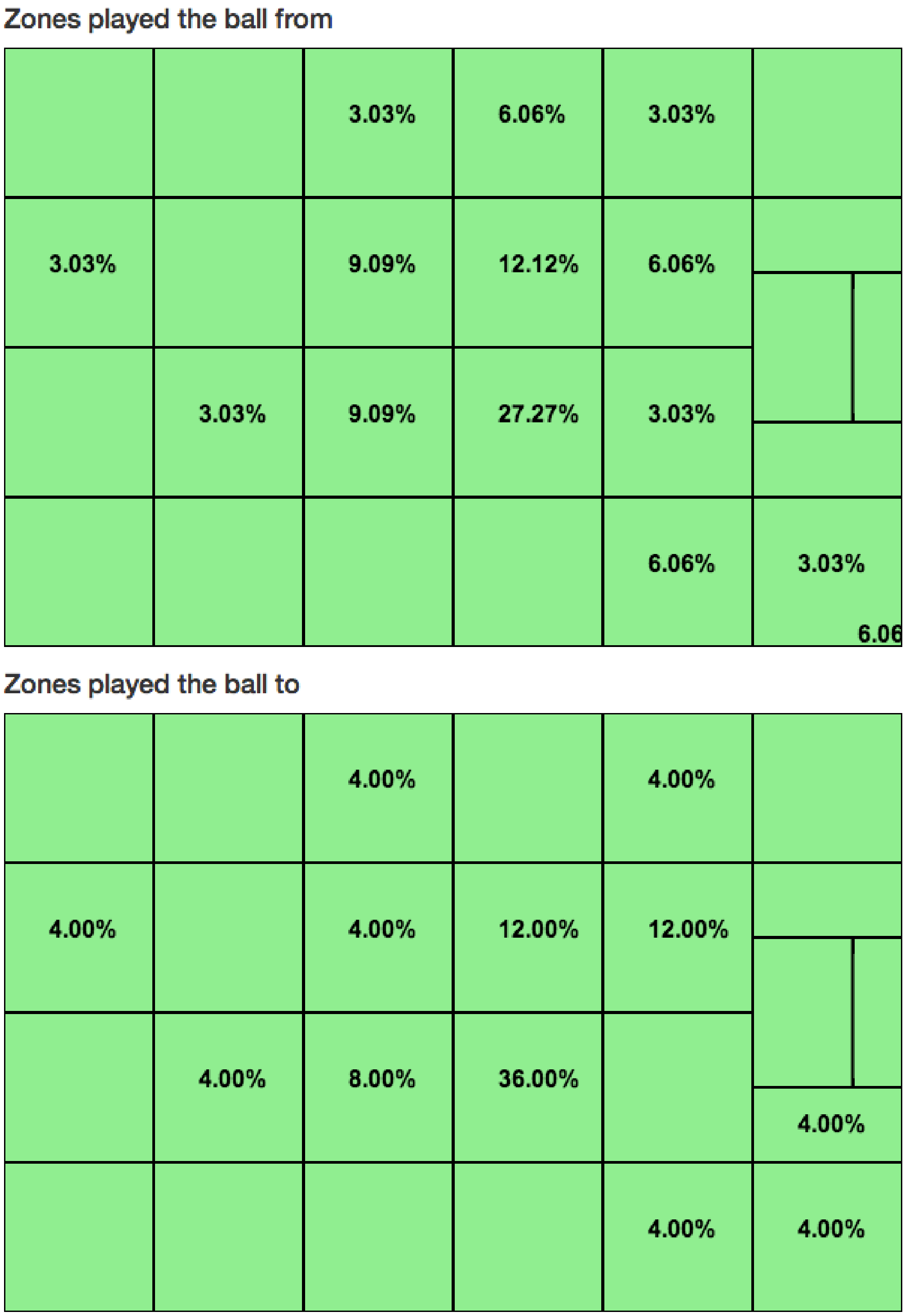
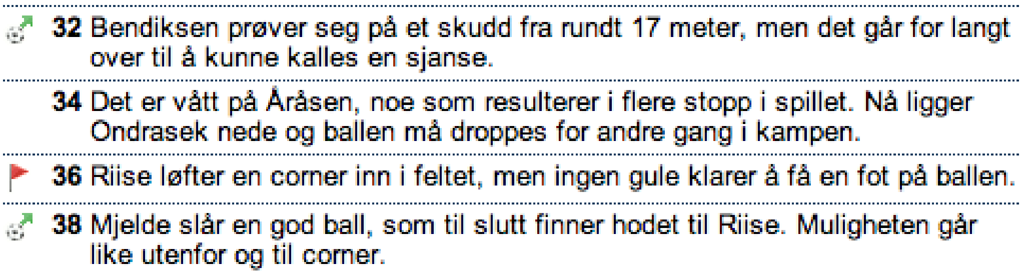


Figure 6.12: Player view highlighting individual statistics



32 Bendiksen prøver seg på et skudd fra rundt 17 meter, men det går for langt over til å kunne kalles en sjanse.

34 Det er vått på Åråsen, noe som resulterer i flere stopp i spillet. Nå ligger Ondrasek nede og ballen må droppes for andre gang i kampen.

36 Riise løfter en corner inn i feltet, men ingen gule klarer å få en fot på ballen.

38 Mjelde slår en god ball, som til slutt finner hodet til Riise. Muligheten går like utenfor og til corner.

Figure 6.13: VGLive.no writes about events in the match. Attempts on goal are tagged with an icon. In the screenshot minute 32 and minute 38 has this icon.

Chapter 7

Evaluation and Results

This chapter presents methods used to evaluate the system; results collected evaluating the system and a discussion around the system.

7.1 Methods

7.1.1 User Survey

The system is measured by doing user surveys on the end users. In our user survey we have coaches rating how much they agree with a statement on a Likert-scale. The statements compare the system against other systems in use at Alfheim today. The Likert-scale chosen is a 5 point scale from *strongly disagree* to *strongly agree.*[1]. In short it will let each individual to note how much they disagree or agree with a particular statement.

7.1.2 UI Performance

Measuring User Interface (UI) Performance is way of measuring usability of the system. These two are directly linked up to each other [6]. A slow system and unresponsive system decrease satisfaction and usability for the end-user. Foundings in [9] indicated that a response time of longer than 1000ms would decrease the satisfaction of the user. The thought-flow of the user will not

[1]http://www.simplypsychology.org/likert-scale.html

be interrupted and therefor one can argue that the UI it's fast enough. We will use this threshold to evaluate our UI performance in the system.

To measure the UI Performance we use Google Chrome DevTools[2]. A web page is only fully loaded when all requests have been fully received. In our Single Page Application (SPA) new requests will be spawned after the Document Object Model (DOM) is loaded. Therefor, we run performance tests for both fully reload of the page, when the DOM is ready and when the users navigates in already rendered page, which will be the most typical way for the user to navigate to different pages.

7.1.3 Compatibility

In the requirement specification we stated that going for a web page would ensure compatibility and accessibility. Every device now has a web browser and therefor the page can be reached from anywhere as long as you have an Internet connection.

To test this we use PowerMapper[3] website testing and site mapping tool. This test checks a wide range of things like browser support, dead links and accessibility. We use it to for compatibility testing for web browsers.

7.2 Experiments and Results

In the tests Tromsø Idrettslag (TIL) have been used to evaluate the system. In the tests we are after if we have achieved or goals listed in the requirements. To recapture the main goals; we wanted to identify key players in soccer opponent teams and we wanted to create a system that difference it from the existing systems and at the same time provides valuable information for opponent soccer analytic.

7.2.1 Database size

In figure 7.1 the size of the different indexes in the database is listed. Match related data is stored in the match, attack and pass index. The total size of

[2]https://developers.google.com/chrome-developer-tools/
[3]http://try.powermapper.com/demo/sortsite

those 3 indexes is 379.5kb and with a total of 12 matches in the database. This means that an average match takes around 31.6kb of storage.

Database size	Size (number of documents)
Match index	124.9kb (12)
Attack index	135.5kb (566)
Pass index	119.1kb (437)
Sum	**379.5kb (1015)**
Teams index	23.9kb (16)
Player index	155kb (445)
Total	558.4kb (1476)

Figure 7.1: Table with the size of each index and the total size of all indexes

7.2.2 Test data and Assessors

In the evaluation the database had been populated with data from TIL and Strømsgodset matches. Only attacks from these two teams have been used to answer the user servey. A total of 34 attacks have been captured for Strømsgodset over 5 matches. For TIL a total of 42 attacks have been captured over 9 matches. Figure 7.2 lists all matches. Some matches include data from other teams than TIL and Strømsgodset. They have not been taken into consideration in the evaluating process.

Match	Attacks registrated
Viking – Tromsø, 2013-05-26	8
Strømsgodset – Tromsø, 2013-06-29	5
Molde – Tromsø, 2013-06-10	14
Strømsgodset – Vålerenga, 2013-05-10	8
Strømsgodset – Start, 2013-10-07	8
Tromsø – Vålerenga, 2013-08-04	15
Tromsø – Start, 2013-09-29	6
Tromsø – Viking, 2013-10-19	14
Tromsø – Aalesund, 2013-08-18	12
Tromsø – Strømsgodset, 2013-11-03	15
Sarpsborg – Tromsø, 2013-09-22	5
Sogndal – Strømsgodset, 2013-09-27	13

Figure 7.2: Matches that have been captured and persisted into the database

7.2.3 SAT as a tool for opponent analytic

In this user servey the assessors where asked how SAT is as a complementary opponent analytic tool. In the requirement process we specified that the system should to be a complementing tool and not a direct replacement of the systems in use. The results shows that most of the assessors values what the system provides.

Assessors not in the Troms IL system do not necessarily use or have experience with an existing tool. Therefor the statement was re-phrased. The assessors were asked how SAT is as an opponent analytic tool, a stand-alone product. The results shows

From these results we can conclude that the end-users view the Soccer Analytic Tool as a good system to complement with for opponent analytic.

7.2.4 SAT as a tool for identifying key players

In this user servey the assessors where asked how SAT is as a tool for identifying key players in a soccer team. This was one of the main goals of the system. We can see from the results of the User Servey that the system identifies key players well. All votes for strongly agree.

7.2.5 UI-performance

UI performance was measured by testing the most content rich web page the team analytic page for TIL. All the response times were measured by doing 20 samples. Tests where run at localhost with server and database running on the same machine, meaning that a further increase in time would likely be seen if the system was in production. Figure 7.3 lists the results.

Two tests were run to see the difference of a DOM load and a fully page load (all requests to the back-end done). The difference between the two is that in the latter we also add the time for the requests for team statistics as these requests are run when the JavaScript code is executed.

The results show that we are below the 1000ms threshold in both tests Nielsen [9] suggested.

Interface	Avg response (ms)	Min (ms)	Max (ms)
Team analytic page (DOM loaded)	434,4	402	488
Team analytic page (everything loaded)	623	583	695

Figure 7.3: UI Performance test results.

7.2.6 Compatibility

Figure 7.4 shows the result after running the page in different web browsers with the PowerMapper tool. The results shows that the web page works in most browsers except from some issues in older Internet Explorer versions. We can conclude that the page has good compatibility with different web browsers making in accessible on many platforms.

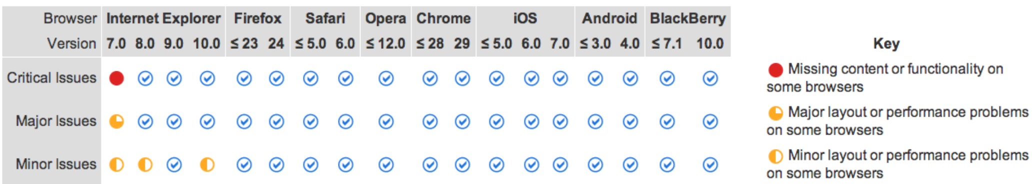


Figure 7.4: Compatibility test by running the web page in different browsers. Screenshot from powermapper.com free trial version

7.3 Discussion

7.3.1 Input

Perhaps the biggest limitation of the system is that it requires manual work to be functional. The system requires a lot of manually work to be operational as an opponent analytic tool. Up to one hour is the normal time spent capturing all data for a match with the current capture interface. On average, teams across the top four top leagues (La Liga, Premier League, Serie A and Bundesliga) took about 13 shots per match, measured in the 2010/11 season

[11]. This means you will have possible have 26 attacks to capture for each match.

In section 6.2 it is mentioned that you have to store the players by their ID. IDs are only found in the database. Instead of this a player selector interface could have been developed letting you just click on an image of the player involved and the ID will be automatic inserted in. We suggest you can save up to 20 minutes by having this feature added.

7.3.2 Accuracy

There is minimal quality checking of the input data except from the match view page that lists every attack captured, meant for external operators to verify the data. Other than that you have to trust the operator that is capturing match data. The input is to some degree subjective for some data like identifying breakthroughs. In some situations one operator may say that it was a breakthrough and another wouldn't agree. This may not be the biggest problem if you set some rules to follow for the operators. Identifying the zone for an event can also be a bit tricky. A low camera angel makes it harder to judge the pitch dividing.

An average match is around 31.6kb size large. In comparison with ZXY a match is around 500-700mb. A better comparison would be against the statistic provider Opta as they are more in the same genre as our system using only manual input. As previously mentioned, they capture every pass in the game meaning that they most likely store more data per match. Another question is do you need this extra data to get a good opponent analytic system.

7.3.3 Domain model

The domain model has been dedicated a lot of time to in the process of defining and building the system. It was crucial to have a domain model that reflected what type of opponent analytic information the system should give the end-user. What is enough data? Having a larger domain model would increase time spent in the process of capturing data.

A thing that could have been changed to get more precise analytic information is how breakthroughs are represented in the database. The current solutions store each type of breakthrough as text saying where on the pitch the breakthrough came from. Rather than this, we could have taken advantage of the already defined dividing of the pitch. This would have given use more precise data to present to the end-user.

7.3.4 UI Performance

A thing to take into consideration is the amount of matches in the database under the tests. A system in production would contain a lot more matches and would increase the response time. Evaluate how long the team query takes..

In SPA you don't necessary get all the files at once, but when you need them. On render, the web browser will first get the root HyperText Markup Language (HTML) document and then start requesting all other files that is referenced. JavaScript files can be bundled to one file saving a lot of network traffic. When doing a clean reload of the team analytic page (non 303 Not Modified status codes) Google Developer Tools reports that total 55 requests is done to the back-end, requesting various files and requests against the Representational state transfer (REST) Application Programming Interface (API).

7.3.5 Scalability

Not in requirement

database size...

Write about scalability elasticsearch and node.js.

In the system it is clear that it is the storage that is the bottleneck of the system. It is the only component that does any computation. The client only gets data from the back-end and present it. The back-end

7.3.6 Usability

One comment from an end user was about the graphical components illustrating statistics on the client. Improving the graphical components would increase the overall experience of the system. This has not been one of the main focuses during the development of the system. However, different illustration techniques are used by combining HTML5 features and JavaScript library's to illustrate analytic information.

Chapter 8

Conclusion

This chapter presents our achievements, gives some concluding remarks and outlines possible future work.

In this project, develops and evaluates a system for capturing, persisting and presenting information in the field of soccer opponent analysis. Particularly we have focused on identifying key players in the opponent team

This thesis will develop a system complementing the Muithu and Bagadus systems. Focus will be on soccer opponent analytics, where a data repository need to be developed capturing important events relevant for this type of analytics. Specially we want to identify the key players in a team. A user interface component providing the core information about the opponent should also be developed.

In the requirement specification we stated what we wanted from the system; a system that identifies key players in the offensive play of a soccer team. We also wanted to be able to see where on the field the key players contribute from.

8.0.7 Concluding Remarks

8.1 Future Work

References

[1] Valter Di Salvo Adam Collins Barry McNeill Marco Cardinale. Validation of prozone ®: A new video-based performance analysis system. *mordi*, 2006.

[2] D. E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young. Computing as a discipline. *Commun. ACM*, 32(1):9–23, January 1989.

[3] Håvard D. Johansen Svein Arne Pettersen Pål Haloversen and Dag Johansen. Combining video and player telemetry for evidence-based decisions in soccer. *Regional Centre for Sport, Exercise and Health - North*, 2013.

[4] Martin Hardy. How the science of opta and prozone's statistics are changing the premier league, Nov 2011.

[5] Jonathan Howard. Sports science and big data opportunities, October 2013.

[6] John A. Hoxmeier and Chris DiCesare. System response time and user satisfaction: An experimental study of browser-based applications. psu.edu, 1999.

[7] D. Johansen, M. Stenhaug, R.B.A. Hansen, A. Christensen, and P.-M. Hogmo. Muithu: Smaller footprint, potentially larger imprint. pages 205–214, 2012.

[8] Lori MacVittie. The impact of ajax on the network. *F5 Networks, Inc*, 2007.

[9] Jakob Nielsen. *Usability Engineering*, chapter Interactive technologies. Morgan Kaufmann Publishers, 1993.

[10] Simen Sægrov, Alexander Eichhorn, Jørgen Emerslund, Håkon Kvale Stensland, Carsten Griwodz, Dag Johansen, and Pål Halvorsen.

BAGADUS: An integrated system for soccer analysis (demo). In *Proc. of the ACM/IEEE International Conference on Distributed Smart Cameras*, November 2012.

[11] Anderson Sally. Shots and goals in the big 4: Offensive performance across leagues in 2010/11. http://www.soccerbythenumbers.com/, aug 2011.

[12] S. Tilkov and S. Vinoski. Node.js: Using javascript to build high-performance network programs. *Internet Computing, IEEE*, 14(6):80–83, 2010.

[13] Jason Turbow. Soccer embraces big data to quantify the beautiful game, June 2012.

[14] Mark Venables. Sportstech: Football, 2013. [Online; accessed 12-November-2013].